Specifying Software Requirements for Safety-Critical Railway Systems: An Experience Report

Luciana Provenzano[1,2(✉)] and Kaj Hänninen[2]

[1] Bombardier Transportation, Västerås, Sweden
[2] Mälardalen University, Västerås, Sweden
{luciana.provenzano,kaj.hanninen}@mdh.se

Abstract. *Context and motivation*: Software safety requirements are fundamental in the definition of risk reduction measures for safety critical systems, since they are developed to satisfy the system safety constraints as identified by mandated safety analyses. It is therefore imperative that the requirements are defined clearly and precisely. *Question/Problem*: We describe our experiences in introducing a safety compliant method of writing safety software requirements for railway projects in a distributed organization. Our goal was twofold, to develop requirements specifications that comply with the EN 50128 standard and that are understandable by the persons involved in the software development. *Principal ideas/results*: We introduced methods to transform natural language requirements to functional requirements described as scenarios, sequence, use-case and state-machine diagrams. *Contribution*: Our experience shows that new ways of expressing requirements, even if proper to solve technical issues such as compliance with standards, bring other challenges to the organization like people's reluctance to changes in working routines and process updates.

Keyword: Software requirements · Safety critical system · Railway domain · Compliance with safety standards

1 Introduction

In large-scale distributed development organizations, projects are often executed by people and teams from different working-sites and countries. Teams with specific responsibilities and differences in safety cultures are cooperating and contributing with their knowledge and resources to develop parts of products that will be integrated into a final system. Assuring safety and compliance with a safety standard is often a challenge in distributed organizations.

In this paper we describe our experiences of introducing new ways of specifying safety requirements in a development organization. The aim of the work was to transform the way requirements were expressed, from a natural language, to semi-formal descriptions in the form of diagrams according to the EN 50128 railway standard [1]. This implied that a new way of working and new processes, that affected people dealing with requirements engineering, had to be introduced.

P. Grünbacher and A. Perini (Eds.): REFSQ 2017, LNCS 10153, pp. 363–369, 2017.
DOI: 10.1007/978-3-319-54045-0_25

In modern safety critical systems, the software is a vital part of the risk reduction measures in the sense that software functions are used to control or reduce the risk of hazards that may cause a system to fail with catastrophic consequences for human life, the environment and facilities. For this reason, software safety requirements and design constraints are the fundamentals in the definition of risk reduction measures for these types of systems, since they are developed to satisfy the system safety constraints as identified by mandated safety analyses. It is therefore imperative that the software safety requirements are defined clearly and precisely so that difficulties and ambiguities in interpreting them are avoided. In the railway domain, the EN 50128 standard prescribes best practice processes to be followed when developing the software, so that it achieves the necessary level of safety, called safety integrity level (SIL). With regard to the software safety requirements, the standard addresses both the requirements content, by pointing out the need to define failure modes, and the software properties that shall be considered, such as safety, robustness, maintainability, and so on. Depending on the criticality of the system, the standard also suggests techniques and measures that have to be applied when structuring requirements. This should be done so that the resulting specifications are understandable, testable, realizable, consistent and complete.

In reality, techniques and descriptions used to specify requirements shall be understandable by all the persons involved in a software life-cycle. This implies that people from different teams and with different experiences of requirements have to be able to use them in their daily work. In this paper, we describe the approach used to comply with the standard. We discuss the impact of this approach on the current process through different stakeholder's feedback and we conclude with some lessons learned.

2 Software Safety Requirements

The software safety requirements subject of this report concern the Train Control Management System (TCMS) for a high-speed train. The TCMS is a real-time on-board system in charge of the execution of the train control functions, the transmission of data inside and outside the train, and the collection of diagnostic data.

Software safety requirements for the TCMS system are derived from the vehicle safety requirements that are identified during the system hazard analysis. These requirements together with design constraints constitute the mitigation measures that have to be implemented to reduce any risks with the product to acceptable levels.

2.1 The Project Context and the Need for Change

When we began to document and assess the TCMS safety requirements, the project had been running for approximately two years. The most of the non-safety critical requirements had been written in natural language. With this situation, both designers and testers of the TCMS system expressed their issues concerning the quality of the existing requirements for the following reasons:

- Some requirements were not testable, mainly because of conditions that contained many implicit assumptions due to the fact that they were written by the most experienced persons with deep domain knowledge. These assumptions resulted in ambiguities for the testers.
- The sources of the input conditions and the destinations of the output results were not defined for all the requirements. This made the integration test very difficult to perform. The testers had to check the details of the implementation to understand the overall functionality.
- Requirements were not complete with regard to failure cases definitions. So decisions on possible alternative behaviors were taken by the designers. The testers had little or no chance to discover the alternative behaviors when performing the functional test, without checking the actual implementation.
- Many of the software requirements had been purely copied from system requirements without refinement for their actual use.

2.2 The Approach Towards Safety Compliance

Based on the above observations, our choice was to introduce a safety compliant method of writing requirements that aimed at:

- Improving the requirements' content to obtain clear, precise, unambiguous, testable, and feasible requirements;
- Including in the requirements the description of the required failure modes according to the EN 50128 standard;
- Describing ways to express the requirements' properties required by the standard, such as safety, robustness, maintainability, performance, efficiency;
- Identifying and documenting the internal and external interfaces of the TCMS.

Our methodology to write and structure the TCMS software safety requirements was driven by the EN 50128 standard. To evaluate the applicability and feasibility of the new method within our organization, this approach was submitted for approval to the project leads and line managers.

Clear, Precise, Unambiguous, Testable and Feasible Requirements through Scenarios. We applied use cases to identify the functional safety requirements. Each use case was described by a success scenario (basic scenario) *"and a set of scenario fragments as extensions of it"* [5]. Even if scenarios are not directly suggested by the EN 50128 standard, we decided to use them due to the following reasons:

- The requirements expressed as scenarios were more precise and clearer than the ones written in natural language. In particular, each step of the scenario was specified according to a well-defined style. This contributed to reduce possible ambiguities caused by natural language sentences while keeping requirements easy to understand by the persons using them.
- The resulting requirements described the failure cases, due to the possibility of defining alternatives and exceptions for each step in scenarios.
- We were able to check the consistency and the completeness of the input requirements, by examining input requirements with the aim of writing scenarios.

In fact, by searching for use cases out of a set of input requirements we could discover the overall function/s, i.e. understand what the TCMS was supposed to offer with regard to a specific set of requirements. Then by building the steps in the scenario for a given use case, we could check if the input requirements were consistent according to the overall goal, and/or if some requirements were missing or incorrect.

Interfaces Through Sequence Diagrams. We described the interfaces of the safety functions by a sequence diagram for each basic scenario. The description of the interfaces covered both internal and external input/output to perform a particular function. Sequence diagrams are a "highly recommended" technique suggested by the EN 50128 standard when modeling is chosen to specify safety requirements. We used the sequence diagrams for interfaces description because we believed that a graphical representation was a more intuitive and concise way to show the interactions of a safety function, and particularly useful when performing the integration test. We intentionally kept the sequence diagrams simple, i.e. they were not used to design the function logic. We did it to reduce the need of extensive training of those using the requirements.

Non-functional Requirements. The EN 50128 standard requires non-functional requirements to be included in the safety requirements specification, but it does not suggest how they should be specified. Since functional safety requirements were written as scenarios, we tried to figure out how these non-functional properties (i.e. robustness, efficiency, etc.) could be specified by stating the following questions:

- Can scenarios also be used to describe some of the non-functional properties?
- If non-functional requirements cannot be specified by scenarios, what is the typical content of a non-functional requirement?
- Are non-functional requirements only applicable to specific functional requirements or generic for all requirements? Which of the non-functional properties can be considered generic for all requirements?

By doing a literature review of current state of art, see for example [2–4], we discovered that non-functional requirements generally consist of a requirement identifier, a title, a description, and a list of sources and standards for the traceability.

So we decided to create a specific section for the non-functional requirements, to include all software properties (such as performance) that were not specific for any particular safety function. These non-functional requirements were specified using the above-mentioned format that was based on state of art. Non-functional requirements that concerned a specific safety function were described as part of the functional requirement by extending the scenario of that function.

3 Outcomes and Impact on Users and Process

Based on our new approach, we reviewed and accepted approximately 140 system safety requirements, which corresponded to the 10% of the whole set of system requirements allocated to the TCMS. From the system safety requirements, we

identified about 70 use cases and we described each use case through scenarios. The safety software requirements specification was assessed by the safety assessor.

An interesting question out of this work was to understand if this method could be employed to manage safety requirements within other projects and to which extent so as to establish a common process issued from this experience to be used in the organization. To address this question, we collected data by informal interviews with the different stakeholders, and discussed with the test, design, change management and quality assurance leads within the project. Informal interviews with team members working with this new method were performed throughout the duration of the project to adapt the approach to the users' feedback. We therefore discuss in this section the impact of this method on the current process and the feedback from the persons who experienced it.

3.1 Impact on the Current Process

To identify use cases and scenarios, we needed to review the input requirements in-depth to grasp the overall functional behavior for each set of safety requirements. The review process was an opportunity to discuss and clarify the safety functions with the customers at a very early stage. This resulted in a better quality of the input requirements and in more involvement of the customer in the software development.

However, a need for a well-defined acceptance process of the input requirements became fundamental as well as the definition of a new role of the requirements manager. Managing requirements with this new approach required more activities than the ones performed to manage the non-safety requirements. This implied that the project needed to invest more time and resources into the requirements phase, and new skills, especially in software engineering, became necessary.

3.2 How Did People Accept This Approach?

Eleven team members who worked at the same site adapted the new method in their daily work. The team members consisted of: one safety manager and a safety engineer, two requirements engineers, three designers, one test lead and two testers, and a software quality manager. Four internal customers from different sites of the company collaborated with the safety manager and the requirements engineer to clarify the input safety requirements using this method as basis for discussions. An independent safety assessor was in charge of the assessment of the safety requirements specified according to this new approach.

Independent Safety Assessor (ISA). The ISA found the safety software requirements very easy to assess since all the EN 50128 standard recommendations had been taken into account (see Sect. 2.2). We were able to provide the assessor with a clear explanation of how each clause had been fulfilled and where in the safety software requirements specification the corresponding information could be found.

Management. The management appreciated that the safety requirements were assessed, which resulted in time-saving and reduced cost for any reworking activities. However, they judged this new method expensive due to the need for additional training of the personnel involved.

Designers. According to the designers, the new way of expressing requirements was too much detailed and overworked. Moreover, they argued that the precise description of the function behavior through scenarios constrained their possible interpretation of the requirements. We think that this was due to the fact that designers were the most experienced engineers in the project (most of them had been working on the TCMS for more than 10 years). They stated that the new way of expressing the requirements constrained them from using their skills and domain knowledge in their daily work. However, designers appreciated the description of the failure cases.

Testers. Testers needed extensive discussions with the requirements engineers to understand how to use the new requirements in order to build the test cases. They were used to work with non-safety requirements which were written in natural language, i.e. they were not familiar with requirements specified as scenario. Initially they claimed that they did not derive any tangible benefit from the sequence diagrams to perform the integration test. We observed that they had difficulties in understanding the relationships among the different use cases and the sequence diagrams. A possible explanation to this may be the way in which the software safety requirements were structured in DOORS [7]. In fact, sequence diagrams were described through DOORS objects tagged as "Information". As a result, sequence diagrams were not considered as actual requirements but as descriptions and, as such, discarded. They also thought that the number of test cases was considerably increased since they were obliged to test all the alternatives and exceptions for each scenario.

Safety Manager. The safety manager found the modeling very useful in discovering potential errors, oversights and inconsistencies in the input requirements. The manager also observed that the number of undefined behaviors identified when performing a Failure Mode and Effects Analysis (FMEA) [8] was drastically reduced due to the failure modes described in the alternative and/or exceptions sections of the scenarios.

Stakeholders. The stakeholders from different sites in the distributed organization appreciated the use of semi-formal modeling with a clear and precise semantics. This provided the stakeholders with a common formalism for discussions. Modeling therefore became the primary means of communication and understanding of the safety requirements.

4 Conclusions and Lessons Learned

In this paper, we introduced a safety compliant method of writing software safety requirements for railway projects in a distributed organization. Our experience shows that dealing with safety requirements was a great challenge that went beyond the technical aspects of producing a requirements specification that complied with the EN 50128 standard. We observed that most of the time and effort was devoted to make this

new approach accepted by the persons involved in the software development, rather than to interpret the standard and propose a suitable solution. The reasons behind people's reluctance to change the working routines are many. In the organization, the use of semi-formal models to specify requirements was the most difficult and perplexing change. Models were not understood as being part of the requirements. However, in the long term the sequence diagrams and the scenarios were used by the testers and the designers to reason about functions. This resulted in constructive discussions, especially during the review meetings, that contributed to a deeper and better understanding of the safety functions.

We believe that the introduction of new methods must be enforced by the top management to be effective, especially in large-scale organizations. Moreover, the working processes have to be updated accordingly for the new techniques to be efficiently adopted. In fact, changes in the way of writing requirements impact the project management in terms of new review processes, new change management routines, new roles and broadened skills, new tools set-up, etc.

This approach pushed the organization to further realize that requirements have *"a crucial importance ... in critical software systems engineering"* [6], and efforts are now made to further improve the requirements management.

References

1. CENELEC EN 50128 Railway applications – Communication, signaling and processing systems – Software for railway control and protection systems (2011)
2. Shahrokni, A., Feldt, R.: Towards a framework for specifying software robustness requirements based on patterns. In: Wieringa, R., Persson, A. (eds.) REFSQ 2010. LNCS, vol. 6182, pp. 79–84. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14192-8_9
3. Gustavsson, J., Österlund, M.: Requirements on maintainability of software systems – an investigation of the state of the practice. In: SERPS 2005 5th Conference on Software Engineering and Practice in Sweden (2005)
4. Bondi, A.B.: Best practices for writing and managing performance requirements: A tutorial. In: ICPE 2012 Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, pp. 1–8 (2012)
5. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, Boston (2011)
6. Larrucea, X., Combelles, A., Favaro, J.: Safety-critical software [Guest editors' introduction]. IEEE Softw. **30**(3), 25–27 (2013)
7. IBM Rational DOORS. http://www-03.ibm.com/software/products/en/ratidoor
8. FMEA. https://en.wikipedia.org/wiki/Failure_mode_and_effects_analysis