Disambiguation of Industrial Standards Through Formalization and Graphical Languages

Daniel Dietsch, Sergio Feo Arenis, Bernd Westphal, Andreas Podelski
Albert-Ludwigs-Universität Freiburg, Freiburg, Germany
{dietsch, arenis, westphal, podelski}@informatik.uni-freiburg.de

***Abstract*—Natural language safety requirements in industrial standards pose risks for ambiguities which need to be resolved by the system manufacturer in concertation with the certificate authority. This is especially challenging for small and medium-sized enterprises (SME). In this paper we report on our experiences with applying traditional requirements engineering techniques, formal methods, and visual narratives in an exploratory case-study in an SME.**

I. Introduction

Technical systems may be governed by international standards. Independent certificate authorities (CA) certify conformance to the standard. In certain domains it is practically necessary that newly developed systems are certified, e.g. by market conditions. Thus when developing such a new system, there is a need to know the CA's understanding of the standard. As consulting with a CA usually involves fees, it is also important to establish this understanding in a most efficient manner. Postponing the detection of misunderstandings of the standard to the certification procedure puts the whole development effort at risk.

A. Research Task

In requirements engineering terms, we are facing a validation problem. Given a requirements document (in this case the standard document), a company developing a new system needs to precisely understand the specific interpretation of the document by the CA. To this end, an interpretation (possibly including variation points) of the requirements in the standard document should be developed and validated with the CA. To reduce the risk of misunderstandings, the company's interpretation of the standard document should be formalised.

In this qualitative, explorative study, we investigate how gathering this interpretation can be supported by outsourcing structured requirements engineering including formal methods if the considered company is a SME. SMEs today often don't practice structured requirements engineering [1], [2] and in particular often cannot afford the high entry costs – in terms of training as well as tool licenses – for the introduction of formal methods [3], [4], From discussions with the company we worked with, we conjecture as a reason, that the requirements are only in few cases given as an unchangeable document which is meant to be complete. In cooperation with other SMEs, the considered company typically develops the requirement documents together with all stakeholders. Our approach reduces the entry costs while, at the same time, it allows for a gradual introduction of requirements engineering techniques and formal methods to development processes in SME according to the needs of SME [1] (cf. Sec. VI-A).

B. Contributions

Regarding SMEs, our work shows that requirements elicitation and formalisation can practically be outsourced to a consulting party (in this case us). Thus if requirements engineering knowledge is limited in-house, an SME can benefit from formal methods with limited costs. Note that the situation of fixed requirements documents not only occurs with standards but also between huge enterprises (possibly already employing formal methods) and SME component suppliers.

Regarding requirements engineering in general, we propose that consulting parties employ visual narratives as a means to make the benefits of formal methods accessible to stakeholders lacking the corresponding educational background. Feedback from the company indicates that the introduction of structured requirements engineering principles and techniques is perceived as an improvement over the situation before. Discussions related to the approach to requirements elicitation and validation (cf. Sec. IV and VI-A) confirms the finding of [1], that SMEs perceive the risk through imposing large changes to their requirements engineering process as very large. We basically followed the authors' suggestion that, to advocate changes, one should "provide evidence that a requirements technique is beneficial" or at least "provide incremental improvements".

Furthermore, our report provides another instance where formal methods sufficiently scale to treat a real-world system under development. We have found today's pattern catalogues (cf. Sec. V-D) to be inadequate for the real-time requirements in EN 54 Parts 2 [5] and 25 [6].

Regarding the formulation of standard documents, our consideration of the two considered Parts 2 and 25 of EN 54 finds the quality of the glossary decreased towards Part 25, while detailing the test procedures in Part 25 is an improvement over Part 2. For the standardisation of technical systems, our feeling is that providing test procedures significantly improves understanding. When providing test procedures, one should – in our opinion – strive for a formalisation using an appropriate

978-1-4577-0924-1/11/$26.00 © 2011 IEEE

test description language and detail implicit, hidden environmental assumptions. The test description language should refer to terms from the domain of the test engineer, i.e. interactions with and observables of the system.

C. Related Work

Concerns of SME regarding requirements engineering are relatively neglected by the research community. Following [1], there is nearly no transfer of academic concepts to this part of the industry, although the companies seem to manage without it: "[...] the evidence rejects the simplistic view of a current 'software crisis', as requirements errors for these companies, though problematic, are rarely catastrophic".

The disambiguation of standards for safety-critical systems is similarly understudied.

Project EDEMOI [7], [8] uses UML diagrams and B [9] with various tools to inspect international standards in the domain of civil aviation for enforcement of security properties. In contrast to our work, EDEMOI doesn't address SMEs, directly communicates with the issuers of the standard instead of with a CA, and they do not discuss real-time requirements. In [10], the certifyability of medical systems, namely pacemakers, is investigated. They use Event-B [11] and the RODIN platform [12], which offers animation tools that visualize concrete scenarios in refined requirements. Their stakeholders are representatives of CAs, which are interested in increasing the reliability of their certificates. The overall approach is comparably heavyweight.

In [13], the ISO 15622 standard for adaptive cruise control systems is employed as a case study for an approach to improve the efficiency of ontology creation by using natural language processing. A baseline domain ontology is generated from technical documents and can then be refined by requirements analysts and domain experts based on conflict class check. We did not face efficiency problems in creating our glossary.

D. Structure

In Sections II-A, II-B and II-C we briefly characterise the expertise of the small company and the certificate authority we considered, and describe the structure of the considered standard. In Section III we outline our approach and in Sections IV, V, and VI present the activities in our approach. Section VII discusses threats to validity and Section VIII concludes and identifies further work.

II. Setting

A. Company

We consider a medium-sized company specialized in developing high-frequency radio-based fire alarm systems (FAS). The company employs about 20 people, of which three are dedicated software developers. The highest education among the software developers is a degree in electrical engineering, that is, the software developers don't have the common curriculum of computer science as a background. In an informal interview with the developers of the company about the availability of knowledge regarding formal methods, only flowcharts, class diagrams, decision trees, and propositional logic were named.

B. Certificate Authority

The certificate authority is not a public institute but a limited liability company which is charging fees for the certification procedure and for consulting regarding issues with the standard documents. It employs about 300 persons overall, of which only a subset is working on the certification of FAS.

C. EN 54: Fire Detection and Fire Alarm Systems

The European Norm 54 currently consists of 25 parts, of which most are concerned with the different devices a fire alarm system may consist of. For the radio-based FAS under development, Part 25, which is about *components using radio links* [6], and Part 2 about *control and indicating equipment* [5] are most relevant.

Both documents are similarly structured. After two introductory chapters, the third chapter details terms, definitions, and abbreviation used in the document. The subsequent chapters contain requirements in natural language, often rather unspecific, for instance, Section 4.2.2 of [6] requires that

> "the components of the system have to employ a transmission protocol on the medium in order (*sic*!) to ensure that no alarm message is lost".[1]

After chapters addressing, e.g., documentation of the software, the last chapter addresses test procedures carried out by CAs to verify compliance with the norm. Regarding system requirements involving software, such as displaying alarm conditions in time (cf. Fig. 2a), Part 2 only describes properties regarding the allowed ranges of environment variables like temperature, currents, positioning, etc. that can be assumed to hold during tests, but actual test procedures (that is, the actions a test engineer has to execute, like absence or presence of events, the inputs and the expected outputs and/or behaviors) are not described. Thus each CA may test differently.

EN 54-25 [6] also states the explicit test procedures for each functional requirement. For each requirement from Chapters 4 and 5, there is a subsection in Chapter 8 with the following three parts (see Fig. 1 for an example):

- *Purpose*: Essentially briefly restating the functional requirement for which the test case is intended.
- *Test procedure*: The actual sequence of actions a test engineer has to execute in order to test the requirement.
- *Requirements*: All quantitative and qualitative observations that have to be made during and after the execution of the test procedure to determine whether the system has passed this particular test.

[1]Translation by the authors. It was tried to preserve any ambiguity we faced; the same ambiguity may not be present in the official English or French translations of EN 54.

4.2.6 Loss of communication
The loss of the ability of the system to transmit a signal from an HF-connected component to the central unit within the in EN-54 specified time bounds has to be detected in less than 300s and has to be displayed in less than 100s.

8.2.8 Test to detect loss of communication on a connection
8.2.8.1 Purpose
Proof of the receiver's ability to recognize the loss of communication with a transmitter in the system. The test must demonstrate the basic function of the system.
8.2.8.2 Test procedure
The manufacturer must provide an appropriate testing instrument and sufficient details of the measures for ensuring the correct and proper operation of the HF-connection. [...]
The transmission of monitoring signals of a randomly selected component has then to be prevented for at least 300s, for example by disrupting the power supply of the transmitter.
During the test the maximum number of components as specified by the manufacturer has to be connected to the base station. [...] The test has to be conducted on a randomly selected part and repeated twice.
8.2.8.3 Requirements
The central unit has to change its state to the fault state after the loss of communication within the in 4.2.6 specified times.

Fig. 1. Example for a requirement from EN 54-25 [6].

III. Approach

We approached the task by the following activities:

1) *Initial requirements elicitation* comprising grounding with the aim to match the nomenclature of the company with the terms from the standard documents.
   The (natural language) requirements gathered in this activity constitute the interpretation of the standard document by the company in terms of the nomenclature at the company. One requirement in the standard document may give rise to multiple different interpretations in case of ambiguities. As preparation for conflict analysis, requirements are formalised.
2) *Conflict Analysis and resolution with the company*. Each possible task conflict is presented by visual narratives (cf. Sec. VI), that is, viewed from the company, external experts serve as mediator between the formalisation and the domain experts. Task conflicts that cannot be resolved without knowing the intention of the CA are collected.
3) *Conflict resolution with the CA*. Again, visual narratives served as means for presentation, here, we mediated between company and CA.

IV. Requirements Elicitation

A. Grounding

In order to be able to communicate efficiently with the company, we needed to become familiar with the domain and its specific terms and notions. For elicitation and documentation, we followed the approaches proposed in [14]. From documentation of preceding systems from the company, we created a glossary for the company's nomenclature which we validated in meetings with the developers.

In a review we collected the terms and notions used in the standard documents and established the relation to the nomenclature of the company by additional workshops with representatives of the company. In total, we identified 48 domain-specific terms (e.g. *master*, *participant*, *HF-connection*, *display*, etc.) and their relation among each other.

B. Capturing

We created an own document and a structure to capture the requirements and the domain assumptions from an own review of the standard. For requirements management reasons, each requirement was captured with the following attributes:

- *Title*, unique *ID*, *priority*, and *version*,
- *category* ranging over environmental assumption, non-functional requirement, and functional requirement,
- *source*, e.g., reference into a normative document, or stakeholder formulating the requirement,
- *status* changing from *draft* to *discussion* on entering and to *stable* on successfully completing validation,
- a natural language *description* of the requirement,
- the *test procedure*,
- a list of IDs of requirements in *conflict*, together with detailed descriptions of the conflict,
- the (possibly empty) *formalization* of the requirement,
- a list of *comments* comprising open questions or remarks for that requirement,

Initially, we captured 20 system level requirements and 8 environmental assumptions in total.

During internal validation workshops (cf. Sec. VI-A) it became clear, that nearly all of the requirements depend on assumptions about the environment (such as radio disturbances from external systems are within normal bounds), the system state (such as "the system has to be in a operational state"), or the principle to conduct tests strictly sequentially that are not explicitly stated in the original source [6]. Although representatives of the company regarded the larger part of those assumptions as obvious, we asked for confirmation of some of them during external validation,

Overall it took six workshops and additionally a number of phone calls to clarify the requirements and incorporate all necessary environmental assumptions.

C. Lessons Learned About EN 54

1) Ambiguous Requirements in EN 54-25: Figure 1 gives an example for an ambiguous requirement. The ambiguity here lies in the meaning of the words *"detect"* and *"display"*, i.e. it is a semantic conflict [15]). In the prospective system design, the failure of a components such as a smoke sensor is detected by a unique other component, called master, which notifies the system's central unit which in turn displays the failure.

The central unit is a special case of a master, but masters in general don't comprise a display to indicate the point in time when the rather technical *detection* takes place. A practical interpretation of this requirement could thus be to admit at most 400s between failure of a component and the corresponding display. An alternative interpretation assumes

7.1.3 Except for [...], the time required for the extraction process or the processing of signals of detectors [...] may not delay the display of a fire alarm state [...] by more than 10s.
7.1.4 The central unit has to change to the fire alarm state within 10s after the activation of a non-automatic fire detector.

(a) EN 54-2, 7.1.3–4

8.2.3.2 Test procedure
10 components have to be triggered simultaneously by the manufacturer-supplied means in order to send or receive an alarm signal. [...]
8.2.3.3 Requirements
The first alarm signal has to be displayed within 10s and the last alarm message within 100s. No alarm signal may be lost. [...]

(b) EN 54-25, 8.2.3.2–3

Fig. 2. Conflicting requirements.

12.5.2: To minimize the effects of disturbances (short circuit or interruption) of transmission paths, measures must be stated and made available to ensure that the function of the remaining devices is recovered within 300s after the fault occurs.

Fig. 3. Requirement from EN 54-2 A/1 prohibiting signal-extending devices.

that the company provides means to make the detection time observable (“*“appropriate testing instrument”*”) and then admits at most 300s up to detection, and at most 100s between detection and display at the central unit. Negotiation with the CA leading to a conclusion is detailed in Section VI-B.

2) Conflicting Requirements: Figure 2 shows requirements from Part 2 and 25 dealing with the time an alarm signal may take from its generation in one of the sensors until it has to be displayed at the central unit. The conflict here is an inconsistency in the description. The requirement from Part 2 demands a time bound of 10s for *all* alarms under any circumstances which is well realisable in wired but not wireless systems. The requirement from Part 25 allows, for each additional alarm, a time of up to 100s. We proposed to interpret the requirement from EN 54-2 to only apply to one alarm at a time, which was accepted by the CA.

3) Beyond the Scope of the Standard: The range of wireless networks can be extended by so-called repeaters, in 802.11 networks as well as in radio-based FAS available on the market. Such components are not considered in Part 25 of the standard. A problem arises, e.g., from Table 3 in Section 8.2.1 (not shown) where the number of components for each test is detailed. For instance, test 8.2.3 (cf. Fig. 2b) shall according to Table 3 be conducted with exactly 10 sensors and one central unit. Repeaters being excluded may be an unintentional weakening of the test case, as repeaters may increase the time it takes between detection and display of alarm. The CA considers this an issue in the standard, a revised version is expected in the future. Currently there is no known solution.

A similar issue arises from the requirement from Part 2 which is shown in Figure 3. As repeaters may concentrate multiple transmission paths, they provide single points of failure possibly affecting a huge number of remaining devices.

V. Formalisation

A. An Abstract Model of Real-Time Systems

We model real-time systems as considered in [6] as follows. Let $\mathcal{I}$ and $\mathcal{O}$ be finite disjoint sets of *input* an *output events*. $\mathcal{I}$ models the possible interactions of the test engineer with a considered system and $\mathcal{O}$ models the possible reactions of the system which are observed by the test engineer.

An *evolution over time* over $\mathcal{I}$ and $\mathcal{O}$ is a function

$$\pi : \mathsf{Time} \to 2^{\mathcal{I} \cup \mathcal{O}}$$

which maps a point in dense time $t \in \mathsf{Time} = \mathbb{R}_0^+$ to the set of input and output events $\pi(t) \subseteq \mathcal{I} \cup \mathcal{O}$ occurring at time t. A *real-time system* over $\mathcal{I}$ and $\mathcal{O}$ is a set $S \subseteq (\mathsf{Time} \to 2^{\mathcal{I} \cup \mathcal{O}})$ of evolutions over time over $\mathcal{I}$ and $\mathcal{O}$.

The syntax of first-order logic formulae over the logical variables in $\mathcal{V}$ and $\mathcal{I} \cup \mathcal{O}$ is given by the following grammar:

$$\varphi ::= E(t_1) \mid t_1 \leq t_2 + c \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \forall t \bullet \varphi_1$$

where $E \in \mathcal{I} \cup \mathcal{O}$ is an event, $t_1, t_2 \in \mathcal{V}$ are logical variables, c is a constant, and φ_1 and φ_2 are formulae. The satisfaction relation between system evolutions π and formulae φ under a variables valuation $\beta : \mathcal{V} \to \mathsf{Time}$, denoted by $\pi, \beta \models \varphi$, is inductively defined as usual with the following base cases:

- $\pi, \beta \models E(t_1)$ if and only if $E \in \pi(\beta(t_1))$
- $\pi, \beta \models t_0 \leq t_1$ if and only if $\beta(t_0) \leq \beta(t_1) + \hat{c}$, $\hat{c} \in \mathsf{Time}$.

In the following, we may use $\emptyset(t)$ as an abbreviation for $\neg \bigvee_{E \in \mathcal{I} \cup \mathcal{O}} E(t)$, which characterises the absence of any events at time t, and the common abbreviations “$\wedge$”, “$\implies$”, “$\exists$”, etc., and “$<$”, “$\geq$”, etc.

B. Observable Events for the Fire Alarm System

For example, when testing requirement 4.2.6 (cf. Sec. IV), the only desired interaction of a test engineer with the system is to disable the radio based communication of a component. We model this by event $Disab_S \in \mathcal{I}$.

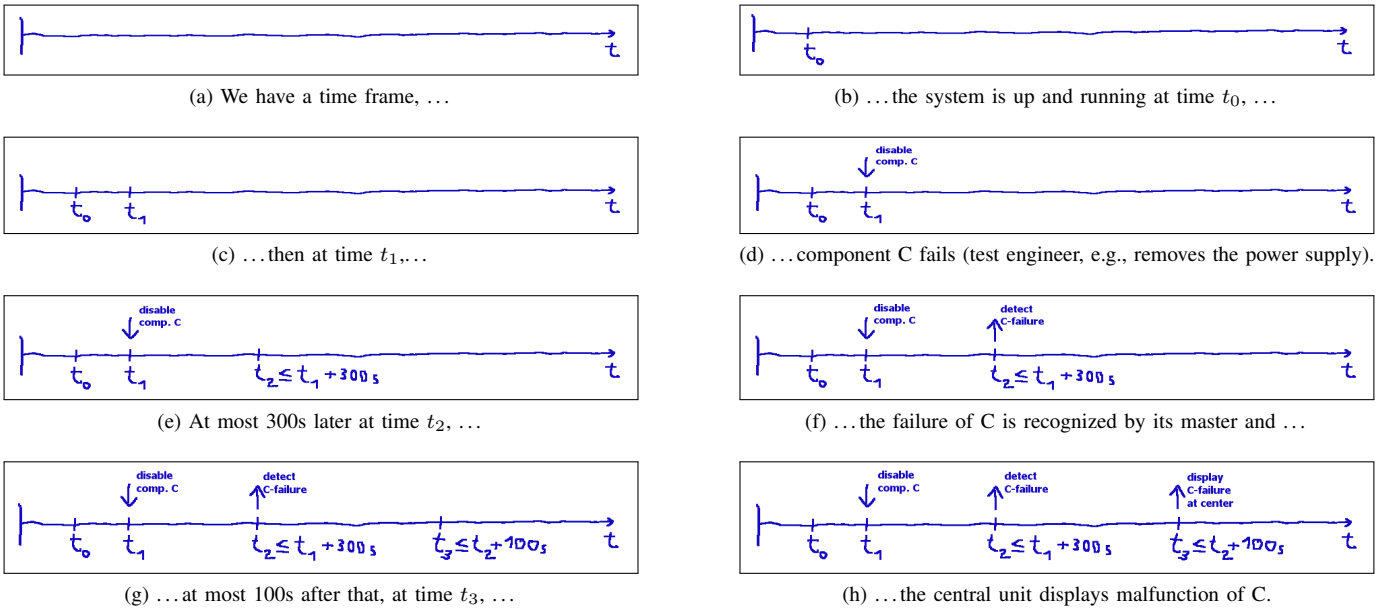
The test engineer may observe, depending on the interpretation of the requirement, three or four of the following events:

- FS: FAS switches from setup to *full surveillance mode*.
- Det_S: FAS just detected radio failure at component S.
- $Disp_S$: the central unit has just started to display the radio failure of component S.

C. Formal Requirements

We formalise, e.g., requirement 4.2.6 by providing a formula φ over $\mathcal{I}$ and $\mathcal{O}$ which characterises those system evolutions which pass the corresponding test. That is, we say π *passes* the test case of 4.2.6 if and only if $\pi, \emptyset \models \varphi$.

Recall from Section IV, that 4.2.6 has two reasonable interpretations, one of them distinguishing detection and displaying of the failure. In this case, a system evolution passes the test if, after the system is fully configured, as indicated by the FS event at time t_0, the test engineer *only* cuts the power of the chosen component, say sensor S, at time t_1 as indicated by event $Disab_S$, and the system detects the failure at time t_2 which occurs at $t_1 + 300$ the latest, and after $t_2 + 100$ the



(a) We have a time frame, ...

(b) ...the system is up and running at time t_0, ...

(c) ...then at time t_1,...

(d) ...component C fails (test engineer, e.g., removes the power supply).

(e) At most 300s later at time t_2, ...

(f) ...the failure of C is recognized by its master and ...

(g) ...at most 100s after that, at time t_3, ...

(h) ...the central unit displays malfunction of C.

Fig. 4. A visual narration for requirement 4.2.6 from EN 54-25 (see Fig.1).

latest shows the failure at the central unit. If the system fails to detect or show the failure in time, or shows other than the desired behaviour, it doesn't pass the test. Formally, a system evolution is passing the test in the first interpretation if and only if it satisfies formula (1).

$$\begin{aligned} &\exists\, t_0, t_1, t_2, t_3 \bullet t_0 \le t_1 \le t_2 \le t_3 \\ &\quad \wedge\, FS(t_0) \wedge Disab_S(t_1) \wedge Det_S(t_2) \wedge Disp_S(t_3) \\ &\quad \wedge\, t_2 \le t_1 + 300 \wedge t_3 \le t_2 + 100 \\ &\quad \wedge\, \forall\, t \bullet t \ge t_0 \wedge t \ne t_1 \wedge t \ne t_2 \wedge t \ne t_3 \implies \emptyset(t) \end{aligned} \tag{1}$$

The second interpretation assumes that detection of the failure is not visible to the test engineer. This can be formalised by changing the third line in (1) to

$$t_3 \le t_1 + 400 \tag{2}$$

and omitting the conjunct $Det_S(t_2)$.

Note that formulae (1) and (2) already differ in the used observables. If the CA accepts that detection is not explicitly shown, basically only the second interpretation remains.

D. Discussion

As pattern catalogues [16], [17] are advocated as an accessible way towards formalisation instead of teaching, e.g., full temporal logic, we initially tried to formalise the EN 54-25 requirements as instances of patterns. It turned out quickly that the considered pattern catalogues don't provide a good match to the EN 54-25 requirements given in form of test cases or scenarios. In particular for the test procedure 8.2.3 corresponding to Requirement 4.2.2 (cf. Fig. 2), we didn't find an adequate pattern instantiation. Furthermore, patterns typically refer only to a small number of observables to be instantiated which is easily exceeded by the number of events relevant for EN 54-25 requirements

In addition, it turned out to be crucial to capture environment and state conditions of the EN 54-25 requirements. For instance that the interactions with the system before being fully configured do not contribute to the outcome of a test while additional unexpected outputs of the system during a test may inhibit passing the test. Integrating such assumptions into the patterns easily clutters the pattern instances and distracts from the actual requirements, namely the relevant input and output events and the corresponding time-bounds. A global, orthogonal integration of these assumptions ("and nothing else shall be observed") turned out not to be adequate.

VI. Validation

A. Internal Validation

By internal validation we denote communication with representatives from the company in order to ensure that our (formalised) interpretation of the norm requirements matches the interpretation of the domain experts at the company. To this end, we usually compiled a list of conflicts, uncertainties, and ambiguities per requirement and explained them.

It turned out, that spoken language was not sufficient to efficiently convey the often subtle issues. On the other hand, we got the impression that teaching our simplistic model and logic (cf. Sec. V), or established models such as timed Automata [18], or visual formalisms like Live Sequence Charts [19], [20] would be a too large change in the processes at the company and in addition would leave us with an additional validation problem, namely, to ensure that we successfully conveyed the meaning of the formalism.

Following [1], we decided to switch to a more pragmatic and incremental approach which consists of using drawings that, as a generic language, don't have an explicit, formal, written semantics but where each particular instances used by us is

backed up by formulae (cf. Sec. V). The idea was, that because of the thorough grounding process, the shared terminology already carried enough semantic information to support simple drawings. If we could support the actual drawing process with the already-known terms, we should transport the semantics of the to-be-discussed scenarios through them. This lead to a technique we call visual narratives (see Fig. 4).

Essentially, the idea is to explain – while drawing – what the drawing means, carefully using domain-specific terms. After the meetings we photographed the drawings and integrated them into the requirement document for later use.

Especially for the timing relations prevalent in the considered requirements, the drawing and the documentation proved to be very effective. We could convey differences between interpretations with it and later it was also possible to refer to the drawings to compare different scenarios among each other.

B. External Validation

By external validation we denote communication with representatives from the CA. To minimize duration (and thus cost) of the meeting with the CA, we first classified requirements according to the criteria from [14] and [21]. We found, that only for structural conflicts (e.g., the repeater issues from Sec. IV-C3) and certain task conflicts (i.e. requirements with multiple reasonable interpretations), the CA's understanding is needed.

On site, we discussed the different issues and again used the concept of visual narratives. Although the representatives of the CA took more time to check the standard against our depiction than those from the company, they confirmed our conclusions. Notably, the CA told us that up to this point no successful certificate was issued and that most of the test procedures had yet to be developed. Nevertheless, we could reach agreements for the task conflicts. The structural conflicts could not be resolved by the CA alone. The CA itself needs to discuss them with other CAs and the certificate issuers.

VII. Threats to Validity

As our work is a qualitative, explorative study, we discuss internal and external validity only briefly. Regarding internal validity, the developers at the company didn't change over time of the case study, did not receive other training regarding formal methods than ours, and did not conduct other direct validation with the CA outside the reported meeting.

To the best of our knowledge, the company we worked with can be considered representative for SMEs active in the domain of embedded systems in Germany. Rosenthal effects cannot be excluded as we participated as the consulting party in the study.

VIII. Conclusion and Further Work

We reported on successfully supporting a medium-sized enterprise in disambiguation of a given standard document by outsourcing formal requirements engineering. Given the high risks and the volume of the product under development, the necessary effort is not prohibitively expensive for the considered SME. This claim is further supported by [22], where requirements on a real-time bus arbitration protocol were successfully formalised as a part of Master's thesis. A crucial aspect of outsourcing requirements engineering, is validation of the formalised requirements against stakeholders with only limited education in formal methods. To this end we propose to employ visual narratives.

Further work consists of supporting our claims by quantitative studies. We're currently working on an extension of visual narratives into a visual formalism to describe real-time test cases and investigate scalability. Preliminary results find visual narratives to be closely related to sequence diagrams such as [19] yet significantly more concise. Shall the need for conditional branching or loops arise, concepts from, e.g., Live Sequence Charts should easily carry over.

References

[1] J. Aranda, S. M. Easterbrook, and G. Wilson, "Requirements in the wild: How small companies do it," in *RE*. IEEE, 2007, pp. 39–48.

[2] D. Dietsch *et al.*, "Abwicklung von Sofwareentwicklungsaufträgen in KMU – Analyse," 2010, http://www.salomo-projekt.de/survey-results/.

[3] A. Hall, "Realising the benefits of formal methods," *J. UCS*, vol. 13, no. 5, pp. 669–678, 2007.

[4] C. F. Snook and R. Harrison, "Practitioners' views on the use of formal methods," *IST*, vol. 43, no. 4, pp. 275–283, 2001.

[5] DIN Deutsches Institut für Normung e.V., "Fire detection and fire alarm systems - Part 2: Control and indicating equipment; German version EN 54-2:1997," 1997.

[6] ——, "Fire detection and fire alarm systems - Part 25: Components using radio links and system requirements, German version EN 54-25:2005," 2005.

[7] D. Bert *et al.*, "Validation of Regulation Documents by Automated Analysis of Formal Models," in *ReMo2V*, ser. CEUR Workshop Proceedings, R. Laleau and M. Lemoine, Eds., vol. 241, 2006.

[8] R. Laleau *et al.*, "Adopting a situational requirements engineering approach for the analysis of civil aviation security standards," *SPIP*, vol. 11, no. 5, pp. 487–503, 2006.

[9] K. Lano, *The B Language and Method: A Guide to Practical Formal Development*. Springer, 1996.

[10] D. Méry and N. K. Singh, "Trustable formal specification for software certification," in *ISoLA (2)*, ser. LNCS, T. Margaria and B. Steffen, Eds., vol. 6416. Springer, 2010, pp. 312–326.

[11] J.-R. Abrial, *Modeling in Event-B: System and Software Engineering*, 1st ed. Cambridge University Press, Jun. 2010.

[12] J.-R. Abrial *et al.*, "Rodin: an open toolset for modelling and reasoning in Event-B," *STTT*, vol. 12, no. 6, pp. 447–466, 2010.

[13] I. Omoronyia *et al.*, "A domain ontology building process for guiding requirements elicitation," in *REFSQ*, ser. LNCS, R. Wieringa and A. Persson, Eds., vol. 6182. Springer, 2010, pp. 188–202.

[14] K. Pohl, *Requirements Engineering: Grundlagen, Prinzipien,Techniken*, 2nd ed. dpunkt.Verlag GmbH, 2008.

[15] D. M. Berry *et al.*, "From contract drafting to software specification: Linguistic sources of ambiguity - a handbook version 1.0," last visited 6/1/2011. [Online]. Available: http://se.uwaterloo.ca/~dberry/handbook/

[16] M. B. Dwyer *et al.*, "Patterns in property specifications for finite-state verification," in *ICSE*. ACM, 1999, pp. 411–420.

[17] F. Bitsch, "Safety patterns," in *SAFECOMP*, ser. LNCS, U. Voges, Ed., vol. 2187. Springer, 2001, pp. 176–190.

[18] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[19] W. Damm and D. Harel, "LSCs: Breathing life into Message Sequence Charts," *FMSD*, vol. 19, no. 1, pp. 45–80, Jul. 2001.

[20] J. Klose *et al.*, "An autom. based interpretation of LSCs," in *TACAS*, ser. LNCS, T. Margaria *et al.*, Eds., no. 2031. Springer, 2001, pp. 512–527.

[21] C. Moore, *The Mediation Process: Practical Strategies for Resolving Conflicts*. Jossey-Bass, 2003.

[22] D. Butt, "Towards risk mitigation in critical software development: Introducing model based development in smes," Master's thesis, Albert-Ludwigs-Universität Freiburg, Feb. 2011.