DOI 10.1007/s00165-016-0365-3
BCS © 2016
Formal Aspects of Computing

Formal Aspects of Computing

Ready for testing: ensuring conformance to industrial standards through formal verification

Sergio Feo-Arenis[1], Bernd Westphal[1], Daniel Dietsch[1], Marco Muñiz[2], Siyar Andisha[1], and Andreas Podelski[1]

[1]Albert-Ludwigs Universität Freiburg, Georges-Köhler Allee 52, 79110 Freiburg, Germany
[2]Aalborg University, Aalborg, Denmark

Abstract. The design of distributed, safety-critical real-time systems is challenging due to their high complexity, the potentially large number of components, and complicated requirements and environment assumptions that stem from international standards. We present a case study that shows that despite those challenges, the automated formal verification of such systems is not only possible, but practicable even in the context of small to medium-sized enterprises. We considered a wireless fire alarm system, regulated by the EN 54 standard. We performed formal requirements engineering, modeling and verification and uncovered severe design flaws that would have prevented its certification. For an improved design, we provided dependable verification results which in particular ensure that certification tests for a relevant regulation standard will be passed. In general we observe that if system tests are specified by generalized test procedures, then *verifying* that a system will pass any test following those test procedures is a cost-efficient approach to improve the product quality based on formal methods. Based on our experience, we propose an approach useful to integrate the application of formal methods to product development in SME.

Keywords: Certification tests, Verification, Model Checking, SME, Dependability, Safety-critical systems.

1. Introduction

Wireless communication offers a low-cost solution for distributed sensing and actuation systems. In recent years, wireless systems have expanded their roles towards performing an increasing number of safety-critical tasks. The addition of more features inevitably increases their complexity, which in turn increases the risk of critical malfunctions. Consequently, there is a pressing need for methods and tools to verify the safety of wireless systems in critical applications. In this paper, we report on the verification of a wireless fire alarm system. Wireless fire alarm systems (WFAS) are increasingly preferred over wired ones to monitor objects where wiring is not possible, e.g. due to regulations for the protection of historic monuments, where wiring is not available and supplementary wiring is not economic, or where high spatial flexibility is needed, e.g., for exhibition stands, for overnight monitoring of vehicles at bus depots, or for monitoring planes during repair and maintenance.

Correspondence and offprint requests to: S. Feo-Arenis, E-mail: arenis@informatik.uni-freiburg.de

Published online: 04 March 2016

The main purpose of a fire alarm system is to reliably and timely notify occupants or staff about the presence of indications for fire, such as smoke or high temperature. The fulfillment of this purpose can be compromised by failures of system components, e.g. physical damage. Thus, fire alarm systems need to employ self-monitoring procedures and notify their maintainers if they are not able to fulfill their main purpose. Both false alarm notifications and false maintainer notifications should be avoided as they induce unnecessary costs.

Given the safety and liability issues associated with system failures, it is necessary to establish, with a good level of confidence, that the system design of a WFAS is correct with respect to its requirements. Main obligations can be derived from the European standard EN 54, part 25 [DIN05] by which commercially available WFAS are regulated. On the European market for WFAS, compliance with EN 54 as confirmed by independent certification authorities is effectively necessary for the economic success of WFAS products. In Part 25 of EN 54, requirements are stated by specifying test procedures to be conducted by test engineers at the certification authority. For example, Part 25 of EN 54 requires that if an alarm is triggered at a single sensor anywhere in the system by the test engineer, it must cause a fire notification within 10 s.

Given the characteristics of the system, we encounter most technical challenges known for conventional testing of real-world industrial systems. It is difficult to precisely control environment conditions such as radio interference and the timing of system inputs, not to mention the need for a prototype implementation and hardware in order to conduct tests. We propose the use of formal verification tools and techniques to overcome these difficulties. Nonetheless, state-of-the-art verification techniques also face several challenges while treating a system with such characteristics. First, the number of components and topologies is large. Second, the standard documents explicitly specify complex environment assumptions which need to be considered in the analysis. Third, the relevant properties are real-time properties. A further challenge is posed by the desire to analyze an unfinished design *a priori*, i.e. during its development and before any first implementation, thus it is necessary to efficiently handle design changes during modeling and verification.

In the work reported here, we accompanied the development of a wireless fire alarm system by SeCa GmbH[1], a small company specialized in radio technology. The main goals of our case study were to investigate whether and by which approach small to medium-sized enterprises (SMEs) can benefit from formal methods, investigate whether today's verification approaches and tools are able to handle safety-critical designs as found at SMEs (which we expected and found to be challenging but smaller than those considered today in strong domains of formal methods like avionics or automotive), and to ensure that EN 54 certification tests will not fail for the final system due to design flaws. As discussed further in Sect. 5, the literature reports low acceptance of formal methods in SMEs. It is an open question whether SMEs perceive a pay-off in terms of costs or risks compared to a conventional, testing-based development process which bears the high risk of design problems being only uncovered as late as during certification tests at the certification authority.

During our case study, we assumed the role of external consultants who complement the activities of the development team at the company. We created, validated, and verified models of the WFAS under design and the environment conditions specified by EN 54 Part 25 using several formalisms and tools. Most aspects of the protocol are modeled using timed automata [AD94], which were subsequently verified using Uppaal [BDL04]. Untimed liveness aspects of the alarm functionality of the system were verified using SPIN [Hol04]. We provided the company with sufficient evidence that the system fulfills its requirements while giving a detailed account of the assumptions and limitations of the analysis, i.e. we strove to provide *dependable* [Jac09] analysis results. As the original requirements are stated as test procedure specifications in EN 54, we generalized and formalized those test procedures and verified that the WFAS design will pass certification tests executed according to those test procedures in all possible scenarios. That is, we used formal methods and verification tools to *verify* that any *tests*, regardless of the scenario chosen by the certification authority will pass, as long as they are conducted according to the test procedures. We did so because the certification authorities for WFAS do not yet consider a supplementation of testing activities by providing formal evidence of correctness as in the avionics standard DO-333 [RTC11], yet our experience from the case study may open perspectives to improve certification processes of critical systems in other domains such as automotive or building safety by allowing the use of formal methods to yield credit towards certification. This, in turn, would stimulate the use of formal methods in the development of safety-critical systems.

[1] http://seca-online.de.

Based on our experience, we propose an approach to integrate the use of formal methods into the established development processes of SME. By complementing various development activities with formal methods through the intervention of a consulting party, an increased confidence in the product design can be achieved while constraining the risks associated with the introduction of modifications to an established and proven development process.

Related work Many case studies on the verification of safety-critical and distributed systems have been published. For conciseness, the papers cited here are those that we consider most closely related to the specific application of the present work. For a more extensive list of related case studies we refer the reader to a survey of applications of formal methods for industrial critical systems in [GM12], a survey of applications that specifically use the UPPAAL tool in the automotive industry in [KLN$^+$15], and a review of techniques and studies for the verification of wireless network protocols in [CZZ$^+$13].

Fehnker et al. [FvGH$^+$12] report on the verification of the ad hoc on-demand distance vector routing protocol (AODV) where they enumerate all possible topologies with up to five nodes. We provide verification results that are applicable for all topologies of a distributed system. Closer to our work is the verification of an AODV protocol draft [BOG02]. Their approach using HOL and SPIN considers real-time only in form of integral factors, the authors, as opposed to our work, did not perform validation with the development engineers. The works [vOS01, K$^+$10] report on verification of the CAN bus protocol and a self-recovery algorithm without considering real-time properties.

Madl et al. [MAS06] use UPPAAL as part of a model-based verification framework that performs limited compositional analysis where only isolated aspects of the considered protocol are modeled using timed automata. Semi-automatic verification of time-triggered architectures has also been carried out in the works of Kopetz et al. [KB03] and Tripakis et al. [TPB$^+$08]. In contrast, our verification strategy is based on the use of fully automatic tools to discharge the main requirements after decomposition and optimization.

In addition, verification of real-time properties of wireless sensor networks found in the literature is typically applied *a posteriori*, e.g., for LUNAR [WPP04], an implementation already existed, and they consider to “employ a formal construction method rather than a post-construction verification” as future work. Similarly, Dong et al. [DSSW99] verified E-2C when it was already “in test” and Gebremichael et al. [GVZ06] verifies the then well-known Zeroconf protocol. Gerke et al. [GEFP10] verified the FlexRay protocol, and pose the explicit challenge to “carry out the analysis a priori, exploring the design space of an as yet unfinished protocol”. The work reported here took up that challenge and performed verification activities *a priori*, demonstrating how the use of formal methods influences the final product design.

An additional discussion of works related to the use of formal methods in the context of small-to-medium enterprises (SME) can be found in Sect. 5, where further references are provided.

The following section describes the requirements analyzed in our case study together with our formalization. Thereafter, Sects. 3 and 4, describe the verification of a concrete system design which aims at implementing those requirements.

2. Formalization of EN 54–25 requirements

Fire alarm systems are regulated in the European Union by the standard EN 54 [DIN97b]. Its requirements are expressed with respect to *components*, e.g. sensors, and a *central unit*. The central unit is a special device which is the interface of the system with its users. It, in particular, *displays* events such as alarms and sensor failures. Indications of fire detected by the *sensors* raise alarm events. Sensors communicate using radio channels to cause the central unit to display an alarm. Sensors need to be monitored constantly in order to ensure that their communication path towards the central unit is functioning correctly. The detection of a sensor failure is also required to be displayed at the central unit.

For each requirement, Part 25 of the EN 54 standard [DIN05] contains specifications of *test procedures*. Each test procedure is described in natural language and includes environment conditions, test engineer interactions, and expected behaviour of the system under test. Both test engineer interactions and expected system behaviour have to be observed according to real-time deadlines. All test procedures assume a *ready-for-use system* as specified by the system manufacturer. A system is ready-for-use when the alarm and monitoring functions are fully operational and events are expected to be detected and displayed. During a test, one can assume that the test engineer conducts exactly those interactions with the system that are prescribed by a particular test procedure. Additionally, the standard specifies the number of system components that should be used for each test. There are certification authorities which perform tests based on the procedures specified by the standard and issue compliance certificates if those tests are passed. Thus we consider a design *correct* if it is guaranteed to pass all tests following those test procedures.

The standard provides test procedures for the following requirements which were selected by the company as the ones with the highest risk value:

(R1) The loss of the ability of the system to transmit a signal from a component to the central unit is detected in less than 300 s and displayed at the central unit within 100 s thereafter.

(R2) A single alarm event is displayed at the central unit within 10 s.

(R3) Two alarm events occurring within 2 s of each other are both displayed at the central unit within 10 s after their occurrence.

(R4) Out of exactly ten alarms occurring simultaneously, the first should be displayed at the central unit within 10 s and all others within 100 s.

(R5) There must be no spurious displays of events at the central unit.

(R6) Requirements R1 to R5 must hold as well in the presence of radio interference by other users of the frequency band. Radio interference by other users of the frequency band is simulated by a jamming device specified in the standard.

Note that the testing guidelines from the standard allow a certain freedom of the certification authority about how to instantiate each test procedure. For example, in order to test R2, a test engineer may choose any component of the system and trigger an alarm while measuring the time elapsed between triggering that alarm and the alarm being displayed at the central unit. The test may be repeated for other system components at the request of the test engineer. The test is considered passed if the time measured is 10 s or less. The certification is awarded only if all tests performed for all requirements are passed. Thus, we illustrate the importance for the manufacturer to ensure that the system is able to pass all certification tests, regardless of the multitude of choices that can be made by test engineers during certification.

2.1. Requirements analysis

The basis for the formalization of the requirements was set using standard activities of requirements analysis. Based on workshops with the company, we created a *dictionary* which represents and clarifies the naming conventions used at the company and maps the names used at the company to terms introduced in the EN 54 standard. The company, for example, distinguishes central unit, repeaters (cf. Sect. 3), and terminal participants like smoke or heat sensors (here called sensor for short). The standard document only distinguishes central unit and components, so repeaters and sensors are both components in the sense of EN 54.

To resolve ambiguities, conflicting requirements, and questions regarding the scope of the standard, we developed a preliminary formalization of the company's understanding of the requirements using simple predicate logic interpreted over timed evolutions of input and output observables. The focus of this approach was to make ambiguities and conflicts well visible in the preliminary formalization in order to support their resolution. In a workshop with representatives of the certification authority and of the company, we successfully used visual narrations [DFAWP11] based on the preliminary formalization to illustrate ambiguities, conflicting requirements, and questions.

The most prominent ambiguity was found in Requirement R1. The EN 54 standard does not clearly define *detect* and *display*. In the architecture envisioned by the company, considering detection to be located at the central unit (instead of at a repeater) would effectively define a quite restrictive deadline of 300 s since the step from detection to display *inside* the central unit does not require significant time. We came to an agreement with the certificate authority that detection denotes the point in time where the overall system, that is, in particular a repeater recognizes a sensor failure. The time up to displaying the sensor failure is measured from this point in time on. The more elaborate formalization presented in the following Sect. 2.2 reflects this agreed distinction between detection and display.

A contradiction existed between the newer Part 25 of the EN 54, which addresses wireless fire alarm systems, and the older regulation of wired fire alarm systems in Part 2 [DIN97a]. Part 2 requires that alarm events are displayed within 10 s, independent of the number of active alarm events. This requirement was considered unrealizable with wireless technology at acceptable costs, the certificate authority agreed to interpret Part 2 to only apply to one alarm at a time. In this interpretation, the corresponding requirement from Part 2 is subsumed by the requirements of Part 25 and was thus not considered further. In [DFAWP11], we present a more detailed account of the lessons learned from formalizing the EN 25 documents.

2.2. Duration calculus formalization

From the preliminary formalization presented in [DFAWP11], we derived a more elaborate set of *testable* Duration Calculus (DC) [CHR91] properties.

We call a component responsible for monitoring another component a *master*. The monitored component is called a *slave*. In the system architecture envisioned by the company, the role of a master is assumed by the central unit or a repeater. The role of a slave is assumed by sensors and repeaters because repeaters need to be monitored as well (cf. Requirement R1). The master-slave relation of a wireless fire alarm system (WFAS) is called its *WFAS topology*.

Let T be a WFAS topology over the finite set $C = \{c_0, c_1, \ldots, c_n\}$ consisting of the system components $c_1, \ldots, c_n$ and the central unit c_0. Let $F = \{f_1, \ldots, f_m\}$ be a finite set of frequency bands (or radio channels) used by the WFAS. We assume the following observables for T. For $i \in C, j \in F$:

- $RDY : \mathbb{B}$ — $RDY = true$ iff the system has been declared ready for use.
- $FAIL : \{\bot, 1, \ldots, n\}$ — $FAIL = i$ iff component c_i is unable to transmit to the central unit, $\bot$ otherwise.
- $DET_i : \mathbb{B}$ — $DET_i = true$ iff the master of component c_i has detected a failure at c_i.
- $DISP_i : \mathbb{B}$ — $DISP_i = true$ iff the central unit has displayed an event at component c_i.
- $AL_i : \mathbb{B}$ — $AL_i = true$ iff component c_i has detected an alarm event.
- $JAM_j : \mathbb{B}$ — $JAM_j = true$ iff radio channel f_j is being jammed.

Note that it is sufficient to model component failure by the single observable *FAIL* because EN 25 only tests failures of exactly one component.

2.2.1. Environment assumptions

The environment conditions for the tests can be formalized as follows. The standard specifies that at most one component may be disabled during the test, and that disabled components are never re-enabled. This gives rise to the environment assumption $\mathsf{FailPers}_T$ for WFAS topology T:

$$\bigwedge_{i \in C} \neg\Diamond(\lceil FAIL = i \rceil \,;\, \lceil FAIL \neq i \rceil) \qquad (\mathsf{FailPers}_T)$$

That is, for each component, there is no interval which can be chopped into one phase where the component is disabled followed by a second phase where the component is not disabled. For Requirements R2 to R4 on alarm events, the standard specifies that all components must be enabled. We formalized this assumption by NoFail_T.

$$\lceil\rceil \vee \lceil FAIL = \bot \rceil \qquad (\mathsf{NoFail}_T)$$

That is, during all non-empty initial intervals, i.e. intervals starting at time 0, there should not be any component failure.

Similarly, the standard states that during tests of the monitoring functionality, the system is considered to be free of alarms. We formalized this assumption by NoAl_T.

$$\lceil\rceil \vee \bigwedge_{i \in C} \lceil \neg AL_i \rceil \qquad (\mathsf{NoAl}_T)$$

During the certification tests, a jamming device is used to simulate radio interference by other users of the frequency band. The behaviour of the jamming device is specified as follows:

1. Only the radio channels used by the system under test are jammed,
2. Only one, non-deterministically selected radio channel is jammed at a time,

3. Channels are continuously jammed for at least 1 s, and
4. During channel changes, all radio channels are free for at most 1 s.

More formally, we obtain Jam_T:

$$\Box \left[\neg\Big(\bigvee_{j,k\in F, j\neq k} \lceil JAM_j \wedge JAM_k \rceil\Big) \wedge \bigwedge_{j\in F} \big(\lceil \neg JAM_j\rceil \,;\, \lceil JAM_j\rceil \,;\, \lceil\neg JAM_j\rceil \Rightarrow \ell \geq 1s\big)\right] \wedge \Big(\lceil \bigwedge_{j\in F} \neg JAM_j \rceil \Rightarrow \ell \leq 1s\Big) \qquad (\mathsf{Jam}_T)$$

Note that it is especially hard for conventional testing approaches during system development to cover all non-deterministic behaviour of the jamming device in order to gain confidence that the system will pass the tests at the certification authority.

2.2.2. *System requirements: monitoring*

The two deadlines of Requirement R1 for detection and display (cf. Sect. 2.1) can be formalized as follows. Firstly, Detect_T states that phases during which component c_i is continuously unable to transmit to the central unit and where the responsible master did not yet detect this failure last for at most 300 s.

$$\bigwedge_{i\in C} \Box\,(\lceil FAIL = i \wedge \neg DET_i \rceil \Rightarrow \ell \leq 300\text{s}) \qquad (\mathsf{Detect}_T)$$

Similarly, $\mathsf{Display}_T$ constrains phases where a failure has been detected by the responsible master and where the central unit does not yet display this incidence to last for at most 100 s.

$$\bigwedge_{i\in C} \Box\,(\lceil DET_i \wedge \neg DISP_i \rceil \Rightarrow \ell \leq 100\text{s}) \qquad (\mathsf{Display}_T)$$

Requirement R5, that the central unit does not display spurious component failures, can be formalized as follows.

$$\bigwedge_{i\in C} \Box\,(\lceil DISP_i \rceil \Rightarrow \lceil FAIL = i \rceil) \qquad (\mathsf{NoSpur}_T)$$

That is, in each phase during which the central unit displays an event, the corresponding component must be disabled.

A complete characterization of the EN 54 testing procedures for Requirements R1 and R5 under the corresponding environment conditions is obtained as follows.

$$\mathsf{FailPers}_T \wedge \mathsf{Jam}_T \wedge \mathsf{NoAl}_T \Rightarrow \Box\,(\lceil RDY \rceil \Rightarrow \mathsf{Detect}_T \wedge \mathsf{Display}_T \wedge \mathsf{NoSpur}_T) \qquad (\mathsf{TestMon}_T)$$

Formula $\mathsf{TestMon}_T$ covers all possible instances of testing according to EN 54 Part 25 because observable $FAIL$ is only partly constrained by $\mathsf{FailPers}_T$. The test engineer may still disable any component at any time and the system is supposed to satisfy Detect_T, $\mathsf{Display}_T$, and NoSpur_T. Sub-formula NoSpur_T in particular needs to be satisfied if the test engineer chooses not to disable any component, so Requirement R5 is covered.

Note that $\mathsf{TestMon}_T$ includes Jam_T to express Requirement R6. Requirements R1 and R4 without Requirement R6 can be formulated by redefining Jam_T to $\lceil\rceil \vee \lceil \bigwedge_{j\in F} \neg JAM_j \rceil$.

2.2.3. *System requirements: alarm*

Similarly, we formalized Requirements R2, R3, and R4 as $\mathsf{Alarm1}_T$, $\mathsf{Alarm2}_T$, and $\mathsf{Alarm10}_T$.

$$\bigwedge_{i\in C} \lceil \overline{AL_{\{i\}}} \rceil \Rightarrow \Box\,(\lceil AL_i \wedge \neg DISP_i \rceil \Rightarrow \ell \leq 10\text{s})\,, \qquad (\mathsf{Alarm1}_T)$$

Here, $\overline{AL_I}$ is an abbreviation for $\bigwedge_{i\in C\setminus I} \lceil \neg AL_i \rceil$, i.e., $\overline{AL_I}$ requires that at most the components in the set I may have detected an event. The idea of the formalization is similar to Detect_T and $\mathsf{Display}_T$: it constrains the length of phases where component c_i has detected an event while the central unit does not yet display it.

The most involved formalization cares for Requirement R3. In the following formula $\mathsf{Alarm2}_T$, we use the logical variable x to fix the duration during which only the first of the two allowed events is detected. In the final conclusion, we use DC's $\int$-operator to measure the accumulated duration during which events i and k are detected but not yet displayed. This accumulated duration must not exceed 10 s by Requirement R3.

$$\bigwedge_{i,k\in C} \lceil \overline{AL_{\{i,k\}}} \rceil \Rightarrow \Box[\forall x \bullet (\lceil\rceil \vee (\lceil AL_i \wedge \neg AL_k\rceil \wedge \ell = x\,;\lceil AL_i \wedge AL_k\rceil) \wedge \ell \leq 2\mathrm{s}\,;\mathit{true}$$
$$\Rightarrow \int(AL_i \wedge \neg DISP_i) \leq 10\mathrm{s} \wedge (\ell = x\,;\int(AL_k \wedge \neg DISP_k) \leq 10\mathrm{s}))] \qquad (\mathsf{Alarm2}_T)$$

Formalizing Requirement R4 is possible with a simpler DC formula again. We only need to constrain the phases where none of the 10 alarms has been displayed (may last at most 10 s) and where not all 10 alarms have yet been displayed (at most 100 s).

$$\bigwedge_{i_1,\ldots,i_{10}\in C} \lceil \overline{AL_{\{i_1,\ldots,i_{10}\}}} \rceil \Rightarrow \Box\Big((\lceil AL_i \wedge \neg \bigvee\nolimits_{i_1,\ldots,i_{10}} DISP_i\rceil \Rightarrow \ell > 10\mathrm{s}\Big)$$
$$\wedge \Box\Big(\lceil \bigwedge\nolimits_{i_1,\ldots,i_{10}\in C} AL_i \wedge \neg \bigwedge\nolimits_{i_1,\ldots,i_{10}} DISP_i\rceil \Rightarrow \ell > 100\mathrm{s}\Big)\Big) \qquad (\mathsf{Alarm10}_T)$$

Overall, we obtain TestAl_T as a formalization of the EN 54 testing procedures for alarm functionality.

$$\mathsf{Jam}_T \wedge \mathsf{NoFail}_T \Rightarrow \Box(\lceil RDY\rceil \Rightarrow \mathsf{Alarm1}_T \wedge \mathsf{Alarm2}_T \wedge \mathsf{Alarm10}_T) \qquad (\mathsf{TestAl}_T)$$

Note that the observables AL_i are controlled by the test engineer. The premises of $\mathsf{Alarm1}_T$ to $\mathsf{Alarm10}_T$ ensure that the exact number of alarms prescribed by EN 54 are raised. In practice, the test engineer will expose the corresponding sensors to smoke or heat. For the case of 10 alarms, the company is supposed to provide additional equipment to raise 10 alarms simultaneously (cf. Fig. 6 on page 15).

We have thus formalized all requirements necessary to formally verify designs of wireless communication protocols against the testing procedures of Part 25 of the EN 54 standard.

2.3. Choosing a modeling approach

The choice of a suitable modeling approach was initially motivated by the nature of the requirements, UPPAAL was selected as an appropriate modeling and checking tool. Mainly because it is well established in handling timed properties, according to the literature [CZZ+13, KLN+15]. Additionally, its ease of use and the familiarity of the consultants with it offered a low initial investment.

As the results of the following sections will show (cf. Sects. 3.2 and 4.1.2), the use of UPPAAL was not enough to show an important liveness property of the protocol: that all components involved in a message collision will eventually be able to transmit their messages. We thus resorted to derive—manually—from the original model, an untimed model of the collision resolution aspect of the protocol (cf. Sect. 4.2). We chose to use the tool SPIN, which is also well established for protocol modeling and checking [GM12, CZZ+13].

Deriving an untimed model from the detailed UPPAAL model was straightforward. Nonetheless, the need for validation of the derived models remains in place. Thus, additional to the effort of derivation, validation had to be performed to ensure the consistency of the two models (cf. Sect. 4.2.3).

As expected, the need for an additional modeling and verification increased the overall cost of the project. In our case, the secondary modeling phase using SPIN was managed as a project on its own. We believe this constitutes a positive example of risk management; this project could have been made optional due to budget considerations or could have been performed as a best-effort project given a fixed budget.

3. Modeling and verification of the monitoring function

The WFAS under design is expected to work in a broad variety of buildings with possibly sub-optimal conditions for radio signals. Therefore, the developers employ *repeaters* capable of relaying messages in addition to the mandatory central unit and the sensors. In a WFAS topology, each component is assigned a unique master. The master-slave relation forms a tree with the central unit as root. Figure 1a depicts an instance of a WFAS topology with sensors $S_1, \ldots, S_6$, repeaters $R_1, \ldots, R_3$, and the central unit CU. Repeaters and the central unit function as master i. For the task, they are equipped with two radio transceivers each, Tr_1^i and Tr_2^i, that enable repeaters to send and receive messages simultaneously on two different frequencies. For displaying an event, a repeater

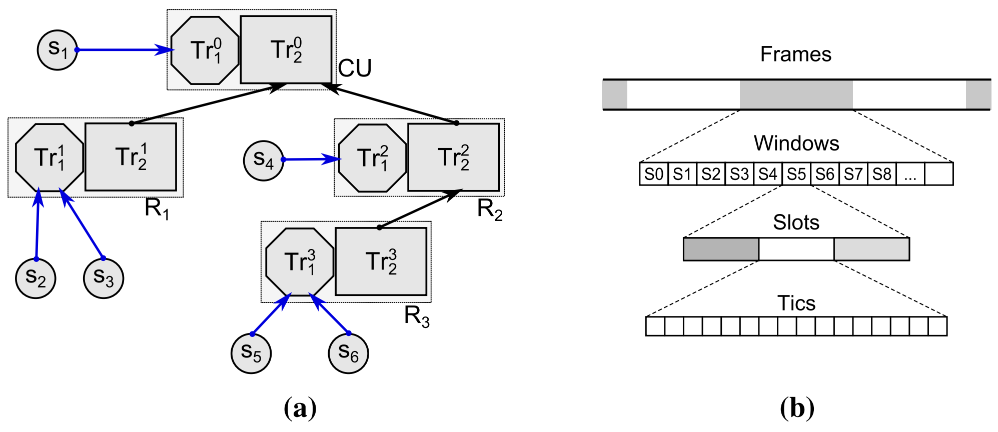


Fig. 1. Overview of the system architecture

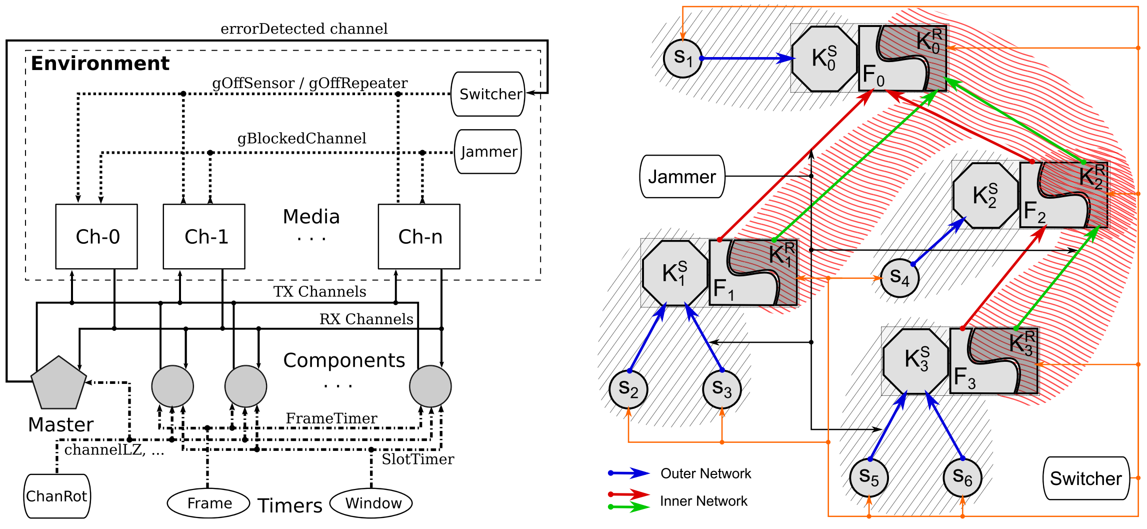
notifies its master of the incidence, which in turn notifies its master until the notification reaches the central unit. Functions of the protocol are distributed among the two transceivers in the masters. Transceiver Tr_1^i is only used for sensor monitoring, that is, this transceiver realizes the master-role towards sensors: it expects keep-alive messages periodically from the sensors, replying them with time synchronization information. Transceiver Tr_2^i realizes three functions: the slave-role towards another repeater or the central unit, the master-role towards other repeaters, and the forwarding of events. The protocol designed employs a variant of time division multiple access (TDMA) as shown in Fig. 1b. Time is partitioned into frames and frames are divided into fixed-width windows. Windows are in turn subdivided into slots, which are assigned to different protocol functions. The window length is specified in *tics*. Every sensor and repeater is assigned a unique window.

To perform failure detection, repeaters in the slave-role and sensors use the same functionality. Slaves periodically send a keep-alive message to their master in the corresponding slot of their assigned window. If no acknowledge message is received from the master, a second and third keep-alive are transmitted in the subsequent slots using a different channel. Masters listen on the corresponding channel during the slots of their assigned slaves. A master enters its error detection status when a specified number of keep-alive messages from one slave have consecutively been missed. The master then initiates the forwarding of the failure detection event. Event forwarding takes place without regarding slot assignments, using the transceivers Tr_2^i.

To compensate for unavoidable clock drift, i.e., slight deviations between clock speed in different components, a correction mechanism is used. Acknowledgements for keep-alive messages come with a time stamp which allows the slave to synchronize its clock with the master's clock. Additional time intervals added at the beginning and at the end of slots (*guard times*) ensure that transmissions of keep-alive messages do not overlap and are not lost. In the design, sensors stop sending keep-alive messages after a determined number of consecutive non-acknowledged keep-alive messages because they are then missing a sufficiently recent time stamp. This mechanism prevents a malfunctioning sensor from causing message collisions.

3.1. Modeling

In the following, we describe our timed automata model of the protocol design provided by the company. As the requirements (cf. Sect. 2) indicate a clear separation between environment assumptions and protocol components, our model of the protocol design is clearly separated from our environment model. Environment model and protocol model are coupled by a clearly defined interface (cf. Sect. 3.1.1). This modular approach allowed us to accommodate changing design ideas during development while maintaining fixed environment assumptions. Note that the protocol model presented here represents the final design of the monitoring function.



(a) **(b)**

Fig. 2. Model architecture and exemplary model instance for the monitoring function

3.1.1. Environment model

The environment model (cf. Fig. 2a) consists of modules that represent different radio channels ('Media', Ch-0 to Ch-n), non-deterministic component failures (Switcher), and radio interference by a jamming device (Jammer). The model of the communication medium consists of one timed automaton for each channel used by the system and thus allows parallel communication over different radio channels. Models of system components send messages to the media using a synchronization channel array `TX` which is indexed with the message type and the channel used. A medium then broadcasts the message using the `RX` channel array with the same indexing conventions. Note that a pair of synchronizations on `TX` and `RX` for a certain message type like `LZ` in the timed automata corresponds to the point in time where the actual `LZ` message is completely transmitted and decoded at the receiver side. That is, our models abstract from the bit-level encoding of messages.

In the model, the "loss of the ability of the system to transmit a signal from a component to the central unit" (cf. Requirement R1) and radio interference as simulated by the jamming device (cf. Requirement R6) are both realized at the media. When a component is deactivated by the Switcher automaton, any message sent by the component is discarded at the media without being relayed. Likewise, messages are discarded at the radio channel blocked by Jammer.

Furthermore, our environment model allowed us to investigate an additional aspect which is not required by EN 54 but a fundamental design decision. The protocol design for monitoring sensors is supposed to be free of message collisions. To verify this property, media models were designed to accept only one message at a time. If (and only if) two components send messages simultaneously, a *deadlock* occurs. Verifying the absence of collisions thus amounts to checking whether the complete model has deadlocks.

Radio interference is modeled by the Jammer automaton (see Fig. 3a). The currently blocked radio channel is indicated by the global variable `gBlockedChannel`. If all radio channels are free, its value is -1. As the radio jammer may have been switched on before the system is ready for use, the ready-for-use system may initially encounter a situation where a radio channel is blocked for *less* than 1 s (cf. Jam_T on page 6). This situation is explicitly modeled by location *INIT_BLOCKED* which is only reachable from the initial location.

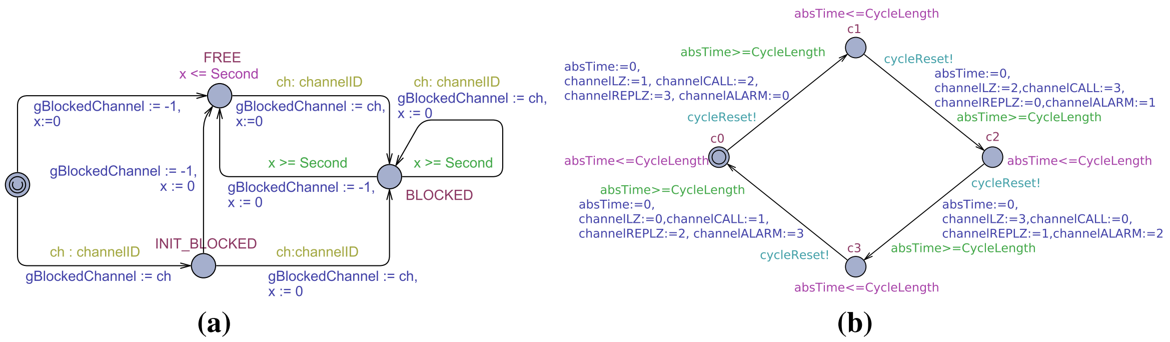


Fig. 3. Components of the environment and protocol model

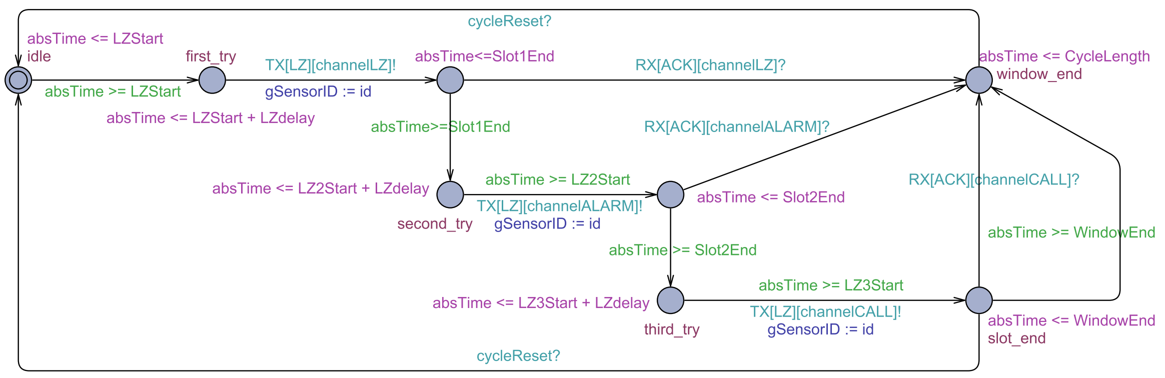


Fig. 4. Model of the slave-role of sensors

3.1.2. Model of the monitoring protocol

The protocol design is modeled by the different system components that perform the functions of masters and slaves. Figure 4 shows the timed automaton modeling the slave-role of sensors. Note that the keep-alive message of a sensor is potentially sent once per slot, i.e. three times in a window: firstly on channel `channelLZ`, then on channel `channelALARM`, and then on channel `channelCALL`. The corresponding master is aware of this scheme and will listen on the corresponding channel.

A repeater is basically modeled by three sub-models: the master-role towards sensors, the master-role towards repeaters, and the slave-role of repeaters. A master-role is further subdivided into two sub-models: the first one receives keep-alives, i.e. `LZ` messages, and replies with acknowledgements, the second one keeps track of missing `LZ` messages to determine the status of its slaves. In order to observe DET_i in the model, the second master-role timed automaton synchronizes with the Switcher automaton once a component failure is detected. A central unit is similar to a repeater, just without the model for the slave-role.

In order to maintain simple and readable models, assumptions of the main requirement $\mathsf{TestMon}_T$ are integrated directly into the model. For instance, the fact that *RDY* holds permanently is modeled by the fact that no operation modes outside of a ready-for-use system are modeled. The system model in particular starts immediately in operation mode ready-for-use. Likewise, no alarms or their corresponding features are present in the model of the monitoring protocol in order to satisfy NoAl_T. The persistence of failures, $\mathsf{FailPers}_T$, is modeled directly within the Switcher automaton. It does not return to normal operation once a component has been deactivated. Additionally, the Switcher automaton allows for only one component to be deactivated during a system run (cf. Sect. 2).

A sample topology including all necessary sub-models is shown in Fig. 2b. For example, sensors S_2 and S_3 (left side, bottom) correspond to 'Components' in Fig. 2a. K_1^S corresponds to 'Master', it models the functionality of the first transceiver in the repeaters. The lines connecting different models represent the usage of the media consisting of the four channels. The color of the line indicates the type of message transmitted: blue for keep-alive messages from sensors, green for keep-alive messages from repeaters and red for forwarded alarm and failure messages. Similarly, K_1^R and K_2^R correspond to 'Components' in Fig. 2a, this time modeling the monitoring functionality for repeaters which technically uses the second transceiver. Here the central unit K_0^R corresponds to 'Master'. The sub-models F_i are only used for forwarding events and discussed in Sect. 4.

To be more robust against effects of the physical environment on radio communication, and in particular radio interference by other users of the frequency band, the system design uses multiple radio channels. The current system design uses four channels for the following four purposes:

- One channel for messages related to monitoring of the sensors (`channelLZ`, drawn in blue in Fig. 2b) and repeaters (`channelREPLZ`, drawn in green),
- one channel `channelALARM` (drawn in red) for messages related to monitoring of the repeaters, for alarm messages from sensors to masters, and for forwarding component failure or alarm events towards the central unit, and
- one so-called CALL-channel for other communication, e.g., for querying the battery status of a component (`channelCALL`, not in Fig. 2b).

At each point in time, at most two of the four channels are used simultaneously. To maximize robustness, the assignment of channels to purposes is changed after each frame following a fixed channel rotation scheme which is known to all components and the central unit. Figure 3b shows the timed automaton ChanRot modeling the channel rotation. The current channel assignment is available to all model components via global variables `channelLZ`, `channelREPLZ`, `channelALARM`, and `channelCALL`.

We have validated the protocol model with engineers from the company. Using the animation facilities of the UPPAAL tool [BDL04], we presented different scenarios to company representatives and asked them to mentally match the shown animation against their understanding of the protocol. The engineers participating in the walk-through sessions did not obtain prior formal methods training, we translated the timed automata semantics to concrete scenarios. Here, we in particular focused on timing aspects. In our perception, the walk-through sessions significantly benefited from our aim to keep the models close to the protocol design and to terms from the dictionary rather than aiming at highly-abstract models which are optimized for efficient verification.

Note that our protocol as shown in Fig. 4 is a slight over-approximation of the protocol design. We agreed with the company representatives that we need not model *lower bounds* for the duration of actual message exchange (cf. Sect. 3.1.1). Thus, we can in particular observe message exchange with a duration of 0 s in our model which does not occur in the real system.

3.1.3. Clock drift and clock reduction

Recall that in the real distributed WFAS, each component is equipped with a local clock chip and that a clock synchronization mechanism coupled with the `LZ` messages is used to compensate for unavoidable clock drift. In [JW13], the authors provide an analytical investigation of the effectiveness of clock drift compensation in TDMA protocols. For a given specified range of quartz oscillators and parameters of tree-like topology (like nesting depth and overall size), the results of [JW13] allowed us to derive lower bounds for guard times such that collision and message loss due to clock drift is avoided. The concrete parameters of the WFAS considered here are investigated in [JW14]. It is in particular verified that the guard times used in the WFAS design discussed here satisfy the conditions of [JW13] thus we can assume absence of message loss or collision due to clock drift and make the simplifying assumption that clocks are perfectly synchronized. Note that a safe guard time only guarantees the absence of message collision *due to clock drift*. If *any* message collision should be avoided, the communication protocol needs to be designed accordingly. In the protocol design considered here, we have both situations. The outer network is designed to be free of collisions [we checked this property by query Q2 (cf. Sects. 3.2 and 3.1.1)]. In the inner network, messages for forwarding an event to the central unit may collide with each other; `LZ` messages on the inner network obtain lowest priority in the collision resolution protocol (cf. Sect. 4) thus are effectively lost in the presence of event forwarding messages.

From [JW14] we can in particular conclude that the maximum distance between the master clock in the central unit and any component's clock is the employed guard time (in the model: 8 tics á $\frac{1}{64}$ s, i.e. 125 ms). So assuming that the central unit, which exhibits very low amounts of clock drift as it is running on wall power, provides the reference time, the timing results of our analysis need to take this offset into account. The tolerance value named above was considered acceptable by the company because drift values observed in the real test environment are expected to be much lower due to non-extreme climate conditions and because the guard time used in the system provides about 60 % safety margin compared to the minimal guard time computed in [JW14].

For Requirement R1, EN 54 requires the use of a maximal topology according to the system design. So in the design considered here, in particular configurations with 126 sensors assigned to the central unit need to be verified. Representing each component's clock chip by a clock variable in the corresponding timed automaton in the model effectively inhibits this verification task for UPPAAL. To overcome this problem, we used the observation that many clocks in the system are *quasi-equal* [HWA+12], that is, they are equal except for isolated points in time where they are reset. Note that at the end of each frame, each component would reset its clock, so in the timed automaton model, there is an interleaving of clock resets without time passing. By [HWA+12, HWP14], there is a property-reflecting, syntactic transformation of timed automata models which replaces all clocks of one equivalence class of quasi equal clocks by a single, centralized clock This leads to significant decreases of verification complexity. In our case, the environment model includes the central clock sources Frame and Window (cf. Fig. 2a). They provide a global clock variable which is reset at the end of each window and use a broadcast channel to notify components whenever the clock is reset. The automaton Window also handles keeping track of the number of windows passed since the beginning of the frame.

Note that symmetry reduction can not be applied in this case because the assignment of slots to components is based on the components' identity.

3.2. Verification

We realized the environment and protocol model including assumption treatment in UPPAAL. In the models, the observables from Sect. 2 are modeled either as locations in the timed automata, or as mappings to (both continuous and discrete) variable values. For example, we have the following mapping for JAM_j, $FAIL_i$, and DET_i:

$$\begin{aligned} JAM_j &\iff \texttt{gBlockedChannel} = j \\ FAIL_i &\iff \texttt{gOffSensor} = i \\ DET_i &\iff \text{Switcher}.\texttt{ERR} \wedge \texttt{gOffSensor} = i \end{aligned}$$

UPPAAL queries can be derived from the *testable* [OD08] DC formulae in Sect. 2 to check the satisfaction of the requirements. The queries are reported in Table 1 on page 12.

3.2.1. Decomposition into inner and outer network

At each point in time, at most two of the four channels used by the system are used for communication simultaneously (cf. Sect. 3.1.2). The channel assignment provided by ChanRot (cf. Fig. 3b) and the design of modules responsible for sending LZ messages (slave role) and the modules responsible for receiving and acknowledging LZ messages (master role) for sensors and repeaters ensure that the monitoring of sensors and the monitoring of repeaters always uses disjoint channels. Therefore any WFAS topology can be seen as consisting of two independent networks with the repeaters as gateways. Recall that repeaters comprise two transceivers which operate on different radio channels at each point in time (at the time point where the rotation scheme is changed, at the end of each frame, no communication takes place due to guard time). We call the network used for communication between repeaters and the central unit the *inner* network and the network used to communicate between sensors and repeaters the *outer* network. In Fig. 2b, instances of inner and outer networks are highlighted for the depicted topology. So the detection aspect of the monitoring functionality for sensors and repeaters, Detect_T, is *local* to the inner and outer network.

The repeaters work as gateways when forwarding sensor failure events towards the central unit. The transceiver which detects a failure of one of its sensor slaves on the outer network requests the second transceiver to forward the event on the inner network. As we have verified separately, the messages for monitoring do not communicate or interfere with each other inside sub-networks because of the TDMA scheduling. So for the verification of the detection aspect for sensors, we can assume that there is no interference from the inner network. For the detection of repeater failures, the argument is more intricate. There are at most 10 repeaters in a system and their slots

Table 1. Verification of the final design (Opteron 6174 2.2Ghz, 64GB, UPPAAL 4.1.3 (64-bit), options `-s -t0 -u`).

		Sensors as slaves, $N = 126$.			Repeaters as slaves, $N = 10$.		
	Query	Seconds	MB	States explored	Seconds	MB	States explored
Q1	Detection possible `E<> switcher.DETECTION`	10,205.13	557.00	26,445,788	38.21	55.67	1,250,596
Q2	No message collision `A[] not deadlock`	12,895.17	2343.00	68,022,052	368.58	250.91	9,600,062
Q3	Detect_T `A[] (switcher.DETECTION imply switcher.timer <= 300*Second)`	36,070.78	3419.00	190,582,600	231.84	230.59	6,009,120
Q4	NoSpur_T `A[] !center.ERROR`	97.44	44.29	640,943	3.94	10.14	144,613

for monitoring messages are evenly distributed over the frame so there is about 15 s time between two repeater LZs. Forwarding of events starts immediately after detection and is completed in less than 2.08 s (cf. Sect. 4). So we can assume that forwarding the detection of a repeater failure does not interfere with the monitoring of the other repeaters under the conditions specified by EN 54, i.e. that at most one component is disabled during a certification test.

We thus verified separate models for the monitoring function of sensors (outer network) and of repeaters (inner network), while abstracting away the networks and components that do not participate in the detection function.

3.2.2. *Topology coverage*

In order to verify that the protocol design will pass all certification test instances allowed by the EN 54 standard, all possible topologies need to be covered. The EN 54 standard requires that a maximal extension of the system under test is used, but how sensors and repeaters are arranged in the topology, i.e., which master monitors how many slaves can be chosen by the test engineer.

As discussed in Sect. 3.2.1 above, we may consider inner and outer network separately. Still, the resulting number of topologies is intractable for an exhaustive verification. As also discussed in Sect. 3.2.1, we can assume that components in one network do not interfere. So to verify the detection functionality, it is sufficient to verify the book-keeping at the masters. As masters don't interfere, we may consider one master at a time. Here the following symmetry argument applies. While the behaviour of the slaves is not symmetric in their identity because the slot in which they are active, a master is listening during the whole frame so its behaviour is independent from *its* identity.

We performed verification on two models, one for the outer and one for the inner network. Each model consists of exactly one master with the maximum number of slaves allowed by the design as required by EN 54-25. The model with sensors as slaves comprises a master (representing both a repeater and the central unit because both use the same functionality) and 126 sensors, plus the central clocks, channel rotation, and the environment model with Jammer, Switcher, and 'Media'. The model of the master role with repeaters as slaves is composed similarly, here the maximum number of repeaters is 10.

Verifying a sub-topology with the maximum number of connected sensors subsumes all other sub-topologies with a smaller number of sensors because a functioning sensor over-approximates, in particular, the behavior of an absent sensor. Together with the observation that sub-networks are isolated in their detection function, verifying detection on the models provided is sufficient to prove the satisfaction of Detect_T and NoSpur_T in all topologies T, because thereby a projection of the actual topology on inner/outer network and on one particular (representative) master is considered.

3.2.3. *Results*

The verification of a model of the initial design of the monitoring protocol resulted in the discovery of two design flaws: A corner case in which the detection deadline was exceeded by one tic and a violation of NoSpur_T in the presence of a jamming signal. Given the timing constraints specified for the jamming signal, it was possible for the jamming signal to continuously block the keep-alive messages of a sensor, thus causing spurious failures to

Table 2. Models and their corresponding sizes

Model	Templates	Instances	Total locations	Clocks
Monitoring				
Sensors as slaves	9	137	1040	6
Repeaters as slaves	9	21	82	6
Alarm				
One alarm (Q5)	6	16	101	16
Two alarms in 2 s (Q6)	5	16	108	12
Ten simultaneous alarms (Q7)	6	25	200	15

be detected at its master. The design was consequently adapted to shorten the length of the windows assigned to sensors and to include an additional retry (to make three in total) for the transmission of the keep-alive packets, together with a faster channel rotation scheme. Verification of the adapted design revealed no further vulnerabilities.

As discussed in Sect. 3.2.2, we considered two models for verification, one for sensors as slaves and one for repeaters as slaves. In order to further validate our design model and in order to ensure that requirement Detect_T is not trivially satisfied, i.e., that it is not the case that the premise of Detect_T is never satisfied in the model, we used UPPAAL with Query Q1 (cf. Table 1) to produce a diagnostic trace which serves as a witness that the model is at all able to detect component failures for each of the two models. Query Q2 confirms that there is no message collision within the considered inner or outer network (cf. Sect. 3.2.1). Query Q3 directly corresponds to Detect_T. Likewise, we verified NoSpur_T in the final model (without the Switcher automaton to leave all components enabled) using Query Q4. The significantly smaller number of states explored can be explained by not considering the switcher which leaves a significantly lower number of scenarios for exploration. In addition, we verified that the automaton Jammer satisfies formula Jam_T. Note that the verification of Q3 proved challenging for UPPAAL (about 10h verification time, about 3.5 GByte memory) but is feasible without any further abstractions or optimizations. Table 2 gives some figures regarding the size of the verification models. The column 'total locations' gives the number of locations faced by the model-checking tool *after instantiating* the templates [BDL04].

Having successfully verified the detection phase of the protocol, we still need to verify $\mathsf{Display}_T$ in order to discharge requirement $\mathsf{TestMon}_T$. As forwarded failures and alarms are treated equally according to the design, we condition the satisfaction of $\mathsf{Display}_T$ to the satisfaction of the alarm deadline requirements. Given that the deadline for displaying failures is much larger (100 s as opposed to 10 s for alarms), we deduce that satisfying alarm deadlines subsumes satisfying failure display deadlines. Thus with the results from the following Sect. 4, the reasoning of Sects. 3.2.1 and 3.2.2, and the verification results shown in Table 1 we have discharged Requirements (R1), (R5), and (R6) for the monitoring functionality.

4. Modeling and verification of the alarm function

Whenever an event is detected, it is forwarded towards the central unit to be displayed. Forwarding is performed by the second transceiver Tr_2, which always uses a different radio channel than Tr_1. An event is transmitted in the slot immediately following detection or reception for forwarding, thus ignoring the window assignments. This design choice speeds up transmission, but allows for *collisions*, i.e. simultaneous transmission of two or more messages. When a collision occurs, the information of the participating messages may be distorted or destroyed.

A resolution protocol was devised in order to accommodate for the possibility of collisions. The protocol follows a tree-splitting [GGLA98] approach based on the system-wide unique ID numbers of the components. The protocol assigns a tic for the start point of the transmission in the next slot. At the start point, a component listens shortly to determine whether the channel is free, in which case the transmission is started (carrier sensing or "listen before talk"). If the channel is in use, the component waits until the next slot to retry transmitting using the same starting point. After a transmission, if no acknowledgment is received, a collision is assumed and, based on the binary representation of the components' ID and the respective collision counter, the colliding components deterministically modify their starting point for retrying the transmission in the next slot. The process is repeated as long as an event is not successfully transmitted and should guarantee that every waiting event eventually reaches the central unit.

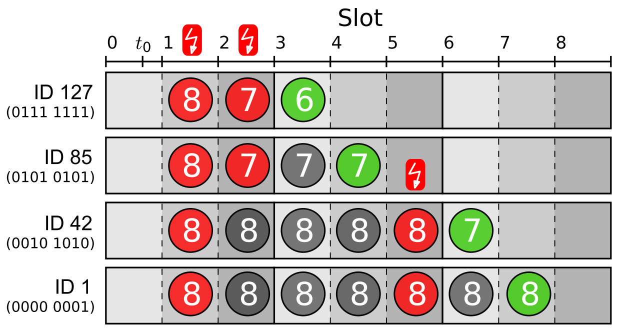


Fig. 5. Collision resolution. The transmission start tics are shown inside the *circles*. Colliding components are shown in *red*, waiting ones in *gray*, and successful transmissions in *green* (color figure online)

In Fig. 5, the initial steps of the collision resolution are depicted for an example scenario. Assume, components with IDs 127, 85, 42 and 1 detect an event at time t_0. In the slot directly following t_0, all components start transmitting their events at initial transmission start tic 8, which results in a collision of all 4 messages. All components detect the situation (they did not receive an acknowledgement), advance their collision counter, and choose based on their ID and the new value of the collision counter a new starting point. In slot 2, components 127 and 85 start their transmission at tic 7 while components 42 and 1 continue to start at tic 8. The messages of components 127 and 85 collide once again, while 42 and 1 find the channel in use and wait for the next slot to retry. In slot 3, all colliding components advance their counter and now 127 chooses an earlier starting point while 85's starting point does not change. This allows 127 to transmit successfully while the remaining components detect the transmission and wait. The process repeats and 85, 42 and 1 finally deliver their events to their masters.

4.1. Alarm deadlines

4.1.1. Environment and collision resolution model

For the environment, we employed an architecture similar to the one described in Sect. 3. The media models were extended to explicitly accept and observe message collisions. A newly added parameter models the range of radio communications, which limits the number of components whose messages may collide. With a value of 1, only messages from components connected to the same master and messages from their master can collide, higher values allow collisions with components connected to masters further away.

Additionally, a simpler radio jammer was used. The non-deterministic jammer as shown in Fig. 3a proved too complex in its behavior, causing the verification to timeout. After consulting with the company and the certification authority, we elicited additional assumptions about the jamming device used for EN 54 certification. Those assumptions allowed us to model and use a *sequential* jammer that deterministically chooses the channel to be jammed (but not the time at which the sequence is started). Again, we used quasi-equal clock reduction and the assumption that clocks are perfectly synchronized. Note that, due to carrier sensing, message collisions only occur if transmissions start at the exact same time, hence perfectly synchronized clocks present a more difficult scenario for collisions.

We modeled the tree splitting collision resolution algorithm for the alarm behavior of the sensors and the forwarding component of repeaters. Repeaters employ an event queue that was implemented as an array and a counter variable indicating the length of the queue. For the verification of a single alarm and ten simultaneous alarms, i.e. $\mathsf{Alarm1}_T$ and $\mathsf{Alarm10}_T$, all sensors start in the alarmed state without loss of generality. By this modeling decision, we cover all possible detection time points in the previous slot and all sensors in the alarmed state will enter the collision resolution protocol at the beginning of the subsequent slot. In the model for two alarms, i.e. $\mathsf{Alarm2}_T$, non-determinism is introduced to allow for the alarms to occur at all possible times inside an interval of 2 s length, in particular simultaneously.

Table 3. Resource consumption for the verification of the alarm functionality.

			$T = T_1$ (palm tree, full collision)			$T = T_2$ (palm tree, limited collision)		
	Query	ids	Seconds	MB	States expl.	Seconds	MB	States expl.
Q5	Alarm1$_T$	–	3.6 ± 1	43.1 ± 1	$59k \pm 15k$	1.4 ± 1	38.3 ± 1	$36k \pm 14k$
	`A[] !Center.ALARMED imply time < 10*Second`							
Q6	Alarm2$_T$	sequential	4.7	67.1	110,207	0.5	24.1	19,528
	`A[] (!Sensor0.DONE || !Sensor1.DONE) imply time <= 10*Second`							
Q7	Alarm10$_T$	sequential	44.6 ± 11	311.4 ± 102	$641k \pm 159k$	17.3 ± 6	179.1 ± 61	$419k \pm 124k$
		optimized	41.8 ± 10	306.6 ± 80	$600k \pm 140k$	17.1 ± 6	182.2 ± 64	$412k \pm 124k$
	`A[] (!Sensor0.DONE || !Sensor1.DONE || ... || !Sensor9.DONE) imply time <= 100*Second`							

4.1.2. *Verification and results*

The event forwarding mechanism of the proposed design posed a challenge for verification for two main reasons: First, the forwarding times strongly depend on the topology. That is, the time required for forwarding a packet towards the central unit depends directly on the number of nodes along the route, which is determined by the specific system topology. Second, the algorithm employs complex discrete data structures, such as circular buffers to determine retry timings and queues to contain packets waiting to be forwarded.

The EN 54 standard requires that an "especially difficult" topology is used in certification tests. The developers agreed with the certificate authority on one topology which involves the maximum number of chained repeaters allowed by the design and is expected to cause many collisions based on the IDs of the involved components. This topology resembles a "palm tree", with the central unit as root, 5 chained repeaters, and 10 sensors connected to the farthest repeater.

We realized the chosen topology using UPPAAL, but had to use the convex-hull over-approximation [WT95] to successfully verify all properties. In this approach, verification is accelerated by abstracting the reachable states of the system by convex polyhedra, the convex hull of a set of *zones* that represent the exact system states reached after a transition, with the added risk of reporting false positives. Without this abstraction, verification terminated prematurely due to memory usage. To better understand the effect of the IDs of the involved components on the collision resolution protocol, we considered different scenarios for the "palm tree" topology along several dimensions for the verification runs:

1. Two different ranges of collisions: "full" for all components being in range of each other, and "limited" for only messages from neighboring components colliding, i.e. range 1.
2. The assignment of IDs for the colliding sensors: "optimized" for IDs with large edit distances and "sequential" for components with sequentially assigned IDs.
3. Which receiver is influenced by the radio jammer (averaged results are shown).

Average verification time, memory and states explored are shown in Table 3.

Additionally, we were requested by the company to extract the expected worst-case times for alarm delivery in the presence of a channel blocker, i.e. the duration from a sensor entering the alarm state to the central unit displaying the alarm. We employed the `inf` and `sup` functions provided by UPPAAL and obtained upper bounds for the time needed to deliver 10 simultaneous alarms with the sets of IDs we considered. In topology T_1 with maximal collision range, the first alarm is displayed after at most 4.32 s and the tenth alarm after at most 44.4 s. In topology T_2 with limited collision range, the first alarm is displayed after at most 5.89 s and the tenth alarm after at most 33.45 s. The difference between the transmission times is explained by the collision range. When it is limited, messages on their way to the central unit are transmitted without further delay once they have surmounted the repeater affected by the radio jammer. In the case of maximal collision range, messages being forwarded close to the central unit can still be delayed by messages further away, causing additional iterations of the collision resolution mechanism.

As soon as prototype hardware and software were available, the developers at the company measured the response times over several trials for a test scenario without a radio jammer. On Table 4, the predicted transmission times for models of T_1 with sequential IDs, T_2 with optimized IDs, and a model of the test scenario are shown together with the average transmission times measured by the company. The test set-up employed to measure

Table 4. Predicted alarm transmission times vs. Measurements on real hardware

	Model sequential (s)	Model optimized (s)	Model test scenario (s)	Measured Avg.(s)
First alarm	3.26	2.14	3.31	2.79 ± 0.53
All 10 alarms	29.03	27.08	29.81	29.65 ± 3.26

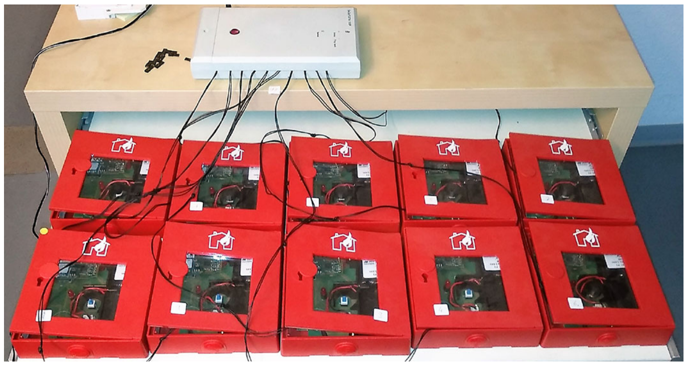


Fig. 6. Prototype test bench employed for model validation

transmission times attempted to replicate the assumptions made during modeling as faithfully as possible. For that, a programmable trigger was wired to ten sensors that were arranged closely. Thus, the engineers achieved almost simultaneous alarms and ensured mutual interference between sensors. A photograph of the prototype test bench is shown on Fig. 6.

From the measurements, we were able to conclude that our predictions were accurate. This validation step enhanced the confidence of the developers in their design and our confidence in the modeling effort.

Thus with the verification results shown in Table 1 we have discharged Requirements (R2), (R3), (R4), and (R6) for the alarm functionality under the additional topology assumption as discussed above and as negotiated with the certificate authority.

4.2. Non-starvation of the collision resolution

Although valuable for certification, only limited topology and scenario coverages are achieved by the results reported on in Sect. 4.1. Thus, not enough evidence was provided that the algorithm has the no-starvation property. That is, that the protocol satisfies the liveness property that a message delivery request is *eventually* served.

Given the complexity of the collision resolution scheme employed by the system, aiming at full, parameterized verification of the no-starvation property was well out of scope given the resource bounds of the project. Still, to increase confidence on the effectiveness of the collision resolution protocol, we set out to produce sufficient evidence that delivery of messages is ensured by creating a model abstract enough to cover an acceptable number of topologies and scenarios.

4.2.1. Untimed collision resolution model

For more general scenarios than the ones considered in Sect. 4.1, UPPAAL proved unwieldy. Thus we provided a Promela [Hol04] model for SPIN, which is a state-of-the-art tool for checking models with bounded integer data. In our model, a single process non-deterministically selects, from a given set I of component IDs, $N \leq | I |$ component IDs which will detect an alarm. The protocol state is encoded by an array indexed by the component IDs allowed in the system (256 in our case). At each array position, there is a collision counter and binary flags indicating whether the component is in its initial state, is operational (online), has detected an alarm, or has delivered its message (the final state).

Initially, all participants are offline, i.e. not operational, and also not in the alarmed state. Hereafter, a subset of participants is selected and their state is set to online but not yet to the alarmed state. The size of the selected participant subset is determined by parameter N. For the selected participants, the model enters a loop. One step of the loop represents the evolution of the system during one slot. To accelerate verification, this evolution is implemented in an atomic block, that is, the loop body is a single indivisible statement. The time, i.e. the slot, *when* the alarm is detected is also chosen non-deterministically for each selected participant. Each participant then evolves according to its scheduled transmission start. Each alarmed participant then tries to send its alarm message and, if it was not successful, executes the rescheduling algorithm. The loop terminates if all participants could send their alarm message successfully. In case of termination, the model also terminates and the non-starvation property holds for the specific selection, otherwise the property is violated.

The model covers all possible selections of N participants from the set I. To illustrate the model's function, consider e.g., a given set $I = \{i_1, \ldots, i_{10}\}$ of 10 IDs. The case $N = 3$ analyses all possible collisions of size 1, 2, and 3 of IDs from I with any possible overlaps in time. The case that only i_1, i_2 detect an alarm at the same time is covered by the case where i_3 detects an alarm earlier and immediately transmits its message successfully due to absence of collisions.

4.2.2. *Topology coverage*

There is an issue inherent to all carrier sensing protocols: The hidden terminal problem [MM04]. When two components are unable to detect the transmissions of each other and repeatedly transmit simultaneously, a common receiver effectively loses all information. The problem was deemed, however, acceptable by the developers, since it rarely occurs in practice and can be easily avoided by slightly adapting the physical distribution of sensors. Therefore we only considered scenarios without the hidden terminal problem for verification.

The topologies our model can represent are those which can be formed by a fixed number of participants, i.e. sensors and repeaters, which can all receive messages from (and thus collide with) each other. A choice of N IDs from I in our model covers all topologies where those IDs are logically and physically distributed such that their messages may collide. Thus, verifying the model for a given value of N is equivalent to checking no-starvation for the protocol in all topologies where up to N messages may collide. The coverage argument for the untimed model is similar to the one employed for the models used in the verification of the monitoring mechanism (cf. Sect. 3.1.2).

To make verification more tractable, we made the following assumptions:

- No jamming device is present. We show no starvation for scenarios where collisions are intrinsic to the system. I.e., caused specifically by the interplay of system components. This assumption is justified because here we investigate the collision resolution protocol for inherent problems. The performance in the artificial setting of EN 54 certification tests was investigated using the timed models.
- A component that has transmitted its alarm successfully remains silent afterwards. No more alarms from the same component are allowed during a verification run. This assumption is justified by the operation principle of real WFAS: an alarm needs to be manually confirmed to reset a sensor, and sensors stay silent for a certain time after having been reset to avoid reporting of the same fire indication again.
- The set of selected components does not exhibit the *hidden terminal* problem because, according to the developers, this problem can be addressed by careful installation of the system as discussed above. This follows directly from the model's main characteristic that all selected components can receive messages from each other and from their master.

4.2.3. *Model validation*

In order to provide an acceptable level of confidence that the untimed model is consistent with the timed model, additional validation was necessary.

For this purpose, two kinds of validation were performed. Initially, model consistency tests were performed, where the behavior of the model for known inputs was checked against known protocol outcomes. These tests were also used as a sort of “regression tests” during model development.

Table 5. Resource consumption for the verification with SPIN 6.2.3.

N	$\mid I \mid$	Seconds	MB	States expl.
2	255	49	1610	1,235,970
10	10 (high Hamming similarity)	3393	6390	6,242,610
10	10 (low Hamming similarity)	4271	10,685	10,439,545
10	10 (random)	4465	11,534	11,268,368
Average for $N = 10$:		4138	9994	9,763,809

Furthermore, consistency checks with timed models were performed. Simulation traces were exported from both UPPAAL and SPIN models representing the same topology. The traces were compared by abstracting away from clock valuations (in the case of the timed model) to check that both models showed an identical behavior, and thus, transitively, represent the protocol design faithfully.

In all, the cost of performing additional verification using untimed models, when seen as a separate project, was estimated a posteriori to be around 10 % of the cost of verification using UPPAAL.

4.2.4. Verification and results

The analysis of the untimed model uncovered several vulnerabilities of the protocol that didn't show up in the limited setup for investigation of the EN 54 requirements as reported in Sect. 4.1. For component selections with $N = 2$ (two colliding components), a problem was uncovered: Due to the limited number of starting points for transmission (7), whenever components with IDs 0 and 128 entered into collision resolution with a third component causing them to collide at the same tic, the algorithm caused the collision to repeat in an endless loop. Due to the similar binary representations of the IDs (their 7 least significant bits are the same), identical and repeating start point selections were made by both components. The company adapted the configuration tools for the WFAS in a way that avoids assigning those IDs to components in close physical proximity.

For scenarios where more than two components collide, i.e., for higher values of N, the memory and time available were insufficient to verify all possible ID selections. Despite this, in order to investigate our target requirement, i.e., no-starvation for collisions of up to 10 sensors, we resorted to perform a random selection of ID assignments according to the similarity of their binary representation. This approach was motivated by the observation that the number of steps required by the collision resolution algorithm correlates with the similarity between the binary strings of the IDs.

The selection of scenarios was refined in order to provide a better coverage of the space of possible topologies. Thus, the distribution of the samples chosen was intentionally biased to explore the effect of ID similarity using the *Hamming distance* of the component IDs within a sample. The Hamming distance of two binary strings is the minimum number of bit replacements necessary to transform one string into the other.

Three different categories were chosen according to the average value of pairwise Hamming distances between the component IDs of a sample: low similarity (lowSim), high similarity (highSim), and pure random samples (Rnd). In total, we sampled 31,744 different 10-component selections, each covering all system topologies where the messages of the selected components may trigger the collision resolution algorithm.

In this way, we performed what we call *testing-like* verification to overcome the difficulty of achieving full formal verification in this particular case. Testing-like because we only sample the space of scenarios just as testing approaches select a number of test cases or generate random test cases. Verification because for each considered set of scenarios, the system behaviour is exhaustively checked and thus yields results for a family of topologies.

The average resource consumption figures for our verification effort with SPIN are shown in Table 5. As seen in the table, similarity appears to have an inverse correlation with the size of the explored state space. We can thus assert that an acceptable coverage over the space of ID selections was achieved. With it, the number of topologies for which we provide verification results is significantly larger than in the case of the timed models. Thanks to the use of our untimed models, we provide an increased level of confidence in the design, enough for the developers to accept it as base for the system implementation.

5. SME and formal methods

Small projects carried out by small companies can be highly safety-critical, a malfunction of such systems can have catastrophic consequences. As we have seen, the development of a radio-based fire alarm system with about 100 sensors is well within the scope of a small company in the high-tech branch. Despite the fact that small enterprises represent a large portion of the software industry, one can yet observe that employing formal methods to improve the quality and overall dependability of their products is not widespread and is found more usually in large companies or in projects for expensive software of very high criticality [WLBF09]. We acknowledge that adoption of formal methods is not directly related to the size of the company. A more decisive factor are the problems that the company faces, and the change aversion of the relevant leaders.

Little work has been done towards documenting and understanding the specific characteristics of software development processes in small companies [AEW07, Ric02, NL03]. Some have investigated the application of formal methods [N+00, DFAW+10] and found none to very few instances of companies applying them regularly.

The reasons for the low acceptance of formal methods in small to medium sized enterprises (SME) are diverse. From experience of our case study, we can confirm the following results from the literature:

- The state-of-the-art of formal methods is simply unknown to many SMEs even in the high-tech branch due to the specific education of the engineers [DCC+13]. In our case, the team of developers consisted of about 5 people whose educational background are classical communication or electrical engineering.
- Despite all progress in usability, additional training is required in logic, complex formal languages, or proof strategies. Thus, the sensible application of formal methods and tools imposes significant requirements on the skills of their prospective users. In order to achieve any significant successes in employing formal methods on the level as reported in this paper, a substantial amount of training is necessary at once and beforehand. For many SME, the costs for this initial adequate training (the "entry costs") are too high and the amount of the initial investment necessary for the training is not affordable. According to [LAO08], "[Very small enterprises] are economically vulnerable as they are driven by cash-flow and depend on project profits, [...]." Therefore, it is in particular not considered an option to try introducing formal methods into an established development process within a live project. The increased workload would pose an unacceptable risk of increased delivery times and thus decreased market competitiveness.
- According to, e.g., [AEW07] for the particular case of requirements engineering, small companies are very skeptical regarding radical changes in their development process. Changes must provide tangible evidence of their potential benefits and offer means for precisely quantifying the risks involved (i.e., they should be accountable). We conjecture that this skepticism is another effect of the elevated risks posed by changes as discussed above.

A complete risk assessment for the introduction of formal methods is hindered by the fact that the cost and the benefit of the utilization of such methods in software development are not easily foreseeable, in particular due to the limited coverage in the literature.

It was one aim of the work reported here to contribute experimental results to the literature. We explicitly did not aim at just another application of formal methods to an industrial system design, but investigated how in particular small companies can possibly benefit from formal methods in their development given the situation discussed above. To care for the need for accountability as discussed above, our goal was also not to strive for complete verification of the system under design disregarding all costs. Our aim was to investigate to what extent the most relevant functionality (as discussed with the company) can be formalized and analyzed in given time budgets.

In the following, we present and discuss our methodical approach. Based on our experience, we believe that the approach used in this case study is applicable to other projects, this needs to be subject of further research. In the following sections, we analyze the process and method aspects of our case study for circumstances that lead to successfully applying formal methods in this particular project and strive to place it in the broader context of introducing formal methods in small-to-medium enterprises (SME).

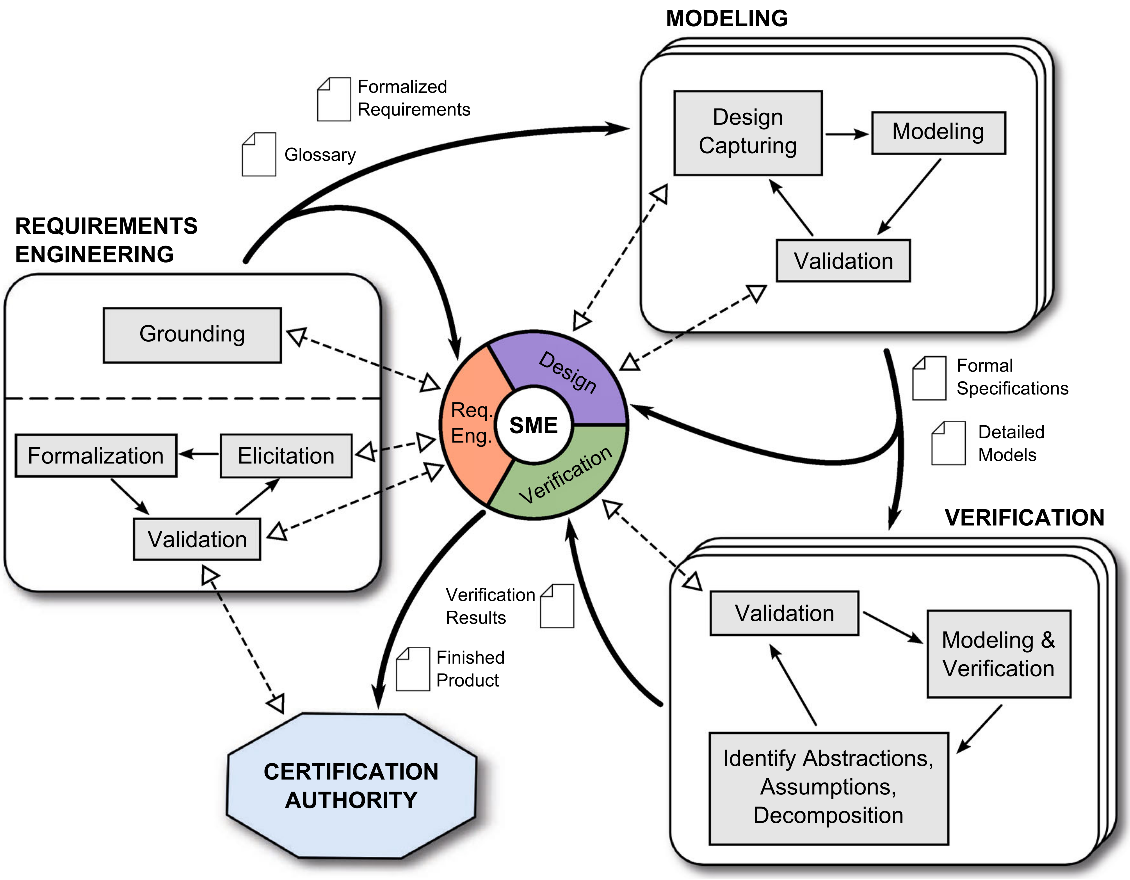


Fig. 7. Process model for outsourcing the application of formal methods

5.1. Approach

As discussed above, training costs are one of the main obstacles for the introduction of formal methods in the software processes of small organizations. In discussions with our project partner it was confirmed that corresponding training of in-house personnel is not an economic option with today's innovation cycle where design phases are followed by implementation, testing, and maintenance phases for the product. To turn a new design into a marketable product can take up to a few years during which the effort of acquiring special education in formal methods would not be directly useful. The application of formal methods focuses, and is more intense on the initial design phase and in the final validation phases. Therefore we proposed to outsource formal activities during the design phase to specialized consulting parties. External consultants in formal methods have the knowledge necessary to recommend formalisms and to apply them towards ensuring improved results from the development activities they accompany.

External consultants, however, usually lack the domain-specific knowledge necessary to perform in the specific branch of the development company. It was therefore necessary to devise efficient and effective communication methods that allow undertaking joint software engineering efforts with satisfactory results. Developers needed means to comprehend the artifacts that result from formalization in order to support validation, i.e., that the formalization properly models the design, and to integrate the information provided by the external consultants into the product under development. Conversely, consultants needed to understand the concepts of the specific domain and their connections to enable capturing the design ideas of the developers and transforming them into formal representations that can subsequently be formally analyzed.

The approach to outsourcing formal methods used in our case study is depicted in Fig. 7. The SME is put at the center. From the company's established process, we focus on the activities requirements engineering, design, and design verification, which are following a conventional, informal and testing-based approach today. We saw the opportunity to complement these activities with formal methods to in particular improve upfront discussion with the certification authority and increase confidence in passing the certification tests.

In Fig. 7, these activities are represented by the split circle in the middle of the diagram. The certification authority (represented by the octagon at the bottom of the diagram) participated in this particular development project as source for the clarification of requirements from the EN 54 standard and as receiver of the finished product for certification. This coarse process structure is the basis of our approach: we strove not to change the company's established process.

Each activity at the company is complemented by an external activity conducted by external consultants (represented by rounded rectangles). Each external activity is structured into sub-activities (represented by gray boxes) as discussed in more detail below. Interaction points between consultants and developers are depicted as dashed arrows between external sub-activities and activities at the company in Fig. 7. These interactions serve two main purposes: First, to obtain domain knowledge from the company, e.g., to establish a common vocabulary and to understand the ideas for the protocol design. Second, to validate formal artifacts and to communicate and negotiate the scope of the formal analyses as limited, e.g., by simplifying assumptions in order to achieve dependability [Jac09]. At each stage of the process, it is the responsibility of the consultants to maintain validity and dependability of their results. That is, that the formalizations produced correspond faithfully to the product being developed (validity) while accurately documenting all assumptions made in the process (dependability). The developing side can not be expected to judge validity and dependability of just the formal documents due to its lack of expertise in formal methods.

Note that we obtain three kinds of (sub-)activities: those executed completely at the company (like capturing the additional requirements next to the ones from EN 54), those executed completely by the external consultants (like formalization, modeling, and verification), and those with interaction points like validation and dependability. Further note that the number of interaction points *decreases* with an increased level of formalization. The initial formalization of the requirements needed three kinds of interactions (for grounding, requirements elicitation, and validation), while verification only needs interaction for validation and dependability because verification is based on the (already validated) formalization of the relevant requirements and on the (already validated) formal model of the protocol design.

The flow of artifacts resulting from the consulting activities is shown as thick, curved arrows annotated with document icons. We distinguish two kinds of flows. Firstly, flows between external activities, e.g., the formal protocol design model is input to the external verification activity, and secondly, flows of artifacts from external activities to the company. The latter are annotated versions of the formal artifacts together with dependability information.

Note that Fig. 7 shows formal methods activities in a waterfall-like fashion. The main reason for that was to match the structure of the development process already established at the company. The inherent risk of this form of integration is actually a construction principle of our proposal: allocate a fixed amount of resources, and make it a responsibility of the consultants to deliver something useful within that frame. We thus, in particular, do not aim for completeness.

Nonetheless, during the later stages of the project, minor fixes in previous phases may be necessary (e.g., fixing requirements during the design or implementation phases). We consider in this case that merging back results from later stages into earlier, finished stages is a constituent part of every stage, instead of a reiteration of the early stage.

A further construction principle of our proposal is the explicit non-dependence of the development activities on the consulting activities. We present this as an instance of risk management to avoid impairing the progress of the project: after a phase is concluded, it is made explicit that using the results of the consultants is optional, the project can benefit from but does not strongly *depend* on the outcome of applying formal methods.

For instance, the project may progress into the implementation phase even if the formal analysis of the design is not concluded. In this particular case, there is an inherent risk that model analysis may still find serious flaws. We consider this risk to be mitigated by the fact that by modeling the design, the developers have already gained a better understanding and, since preliminary analyses have not yet uncovered any flaws, they are less likely to be found by more detailed analyses. Other flaws in the design are already accounted for and considered manageable in the classical development process.

One advantage of our approach is that it can also be used in a spiral-like development process, the different activities can be restarted or interleaved as necessary to follow the development process of the clients correspondingly.

In the following subsections, we discuss the external activities in more detail. Each external activity is structured into an iterative cycle that performs its sub-activities until the corresponding activity results are accepted by the developing company. All external activities consist of an information gathering phase, where input from the developers and previous development stages is consumed, followed by a formalization or modeling phase that is performed by the expert consultants, and a validation of the results with the developers from the company. The results of the validation are used to refine the formalizations in a further iteration or are compiled into final activity results.

We highlight the fact that the external activities in the proposed approach intentionally stopped before the actual implementation phase since the considered company has their core value in the know-how of the target hardware used in their specific branch and therefore refrained from outsourcing implementation activities. We in particular did not consider approaches which aim at code generation from models, such as, e.g., SCADE [HRR91] or Statemate [HLN+90], or larger methodologies based on step-wise refinement [K+81] such as, e.g., the B-Method [Lan96] or RAISE [GH+95].

5.1.1. Requirements engineering

The external requirements engineering activity consists of four sub-activities. Elicitation, formalization, and validation were conducted iteratively, grounding was conducted in parallel. The goal of the external requirements engineering activity was to ensure that product requirements are consistent and precise enough to avoid ambiguities and misunderstandings that may result in not passing the certification tests or general product defects. The input of this activity included existing natural language requirements documents provided by the company, the documentation of existing systems, and external documents such as the EN 54 standard.

During the external requirements engineering activity, we maintained two documents: A dictionary (or glossary), which streamlined the communication between consultants and the development team for other phases of the project, and a requirements specification. After first, coarse prioritization, the requirements specification focused on all relevant parts of EN 54 (in addition to part 25 for wireless WFAS, in part 2, which applies to all fire alarm system, needed to be considered). The level of formality differed, some requirements were only described in natural language. Non-functional requirements were identified and excluded from the formalization activity, for example EN 54 requirements on battery lifetime.

A crucial step in this activity is the choice of the formalism to precisely capture the requirements by the external consultants. After a second prioritization, which also took into account time budget considerations, we developed a preliminary formalization and visual narratives to support the discussion with the certification authority regarding conflicts, ambiguities, and scope of the standard (cf. Sect. 2). The preliminary formalization was analyzed for conflicts and ambiguities by the external consultants. During validation, some of these issues could be resolved based on the domain knowledge of the company. In order to communicate the formalized requirements effectively and efficiently, we used scenarios that illustrate both positive and negative cases in terms of the common glossary. This proved key to establishing an effective communication channel between consultants and developers, and later with the certification authority. As detailed in [DFAWP11], the preliminary formalization and the corresponding visual narratives worked exceptionally well in this particular project. One half-day workshop at the certificate authority was sufficient to enable the engineers at the certification authority to understand the conflicts and ambiguities without any prior training in formal methods. [2] This is in particular notable because no WFAS had previously been certified to be compliant with the new part 25 of EN 54.

After a final prioritization, we developed the thorough formalization in (testable) Duration Calculus formulae (cf. Sect. 2.2). Duration Calculus was chosen to optimize readability for the external consultants and because of its suitability for the following verification tasks. In hindsight, the preliminary formalization could have been skipped in favor of consistently using Duration Calculus. Testable Duration Calculus formulae can be translated to natural language as well and, in particular, to positive and negative examples in form of visual narrations.

The results of the requirements formalization activity may also be integrated into the text based requirements documents maintained at the company for all requirements including those which were not the subject of formalization.

Although the engineers were at times a bit overstrained by our quest for precision, we received positive feedback. The company in particular valued the detection and clear and comprehensible documentation of conflicts and ambiguities.

[2] The educational background of the engineers we met at the certification authority was similar to the engineers at the company.

5.1.2. Modeling

The external modeling activity consists of three sub-activities. The input of this activity are the glossary and the formal requirements from the previous activity, together with a description of the proposed design(s). Existing design documents were complemented by an oral presentation of the design by the company's engineers.

Consultants used the input to build an understanding of the design idea(s) described using common terminology. A proposal for the prioritization of design aspects was developed based on time budget considerations. The most critical aspects were chosen as candidates for modeling and verification.

The consultants' understanding was refined based on joint discussion until an agreement was reached. The consultants chose a modeling language capable of expressing the relevant aspects of the design, in our case timed automata (cf. Sects. 3 and 4), and determined a suitable level of abstraction. They proceeded to create models for each aspect, which was later used to communicate with the developers.

In our experience, models created in this activity should reflect the design in the most faithful way, even at the expense of generating large, complex models not entirely suitable for verification tools. By recreating the developers' ideas in the most explicit way, the communication between the teams for validation was facilitated. The models were used to generate positive and negative scenarios for the requirements analyzed. It turned out valuable to select a modeling tool that supports simulation of the model and allows to observe the evolution of the model variables directly in order to communicate the scenarios to the system developers. If the models are made such that the concepts fixed during the requirements engineering activity are easy to observe, validation can be carried out by replaying scenarios. We asked the developers to build a correspondence between the evolution over time of domain observables they are familiar with and their representation by model observables. Witnesses for such scenarios are at best generated using formal verification techniques, an example is query Q1 discussed in Sect. 3.2.3 which generates a witness for successful sensor failure detection. Through validation in this form, it can be confirmed whether the model resembles the design faithfully and whether the scenarios simulated are valid in order to refine the models. The models in turn enable assessing whether the design idea leads to satisfying the requirements.

By mapping system observables to model elements, queries were derived from the formalized requirements (cf. Sects. 3.2.3 and 4.1.2). Different design models were confronted with the specifications to evaluate their compliance and compared to determine which alternatives provide the best solution to fulfill the requirements. Thus modeling and verification techniques enabled an informed search of the design space for the product. Once a design is fixed, further analyses that are deemed important by the developers can be carried out (for example the non-starvation analysis reported in Sect. 4.2 which is not required by EN 54 but necessary for a reliable WFAS product).

Resulting from this activity were models that documented the protocol design in a formal fashion together with formal specifications (or queries) that concretized the requirements and analysis information useful for the implementation of the design.

5.1.3. Verification

The external verification activity consists of three sub-activities, two of them independent from the company and only one with interaction points (cf. Fig. 7). This activity was expectedly the most distant from the skills of the developers.

The goal of the external verification activity was to verify that the chosen design will pass all certification tests allowed by the standard EN 54 Part 25 for the considered requirements. The design verification activity requires mainly formal input from the previous phases. It relies on model specifications and on the existence of models that precisely capture the design ideas.

As a first task, the consultants assessed the costs of verifying the supplied models and proposed concrete verification goals in view of the resources allotted and the capabilities of the tools available. Verification assumptions (such as limitations on the use cases considered) and a subset of design aspects and properties for which verification appeared feasible given the resource constraints, were presented to the developers to validate their acceptability regarding the confidence level desired by the company. Once the scope of verification was established, the consulting team built specific models, including the "palm tree" topologies with different collision ranges as described in Sect. 4.1.2 and the untimed Promela model of the collision resolution protocol described in Sect. 4.2. While the modeling activity aimed at successful model validation, and thus did not yet optimize models for verification, we observed that verification can make use of any proof techniques such as (manual) abstraction, as long as these techniques can be proved to be sound approximations of the design model. Other techniques such as compositional verification can also be used. Time budget considerations in particular led to the choice of cases for the verification of the untimed model.

Verifying the models produced information on the compliance of the design with the requirements from the standard, violations of the requirements were validated with the developers to determine whether the models needed to be refined or if the design required to be corrected.

After successful validation of the results, the verification activity delivers as output a dependable documentation of the assumptions made on the product environment, the use cases analyzed and the verification results. We presented the verification results in a tabular format that lists the properties verified specifying sub-properties that may have contributed to their verification. For each sub-property the models that were used to perform the checking are listed and the environmental or limiting assumptions that affect each model are added as annotations. Tables 1 and 3 are excerpts of the presented verification results.

5.2. Discussion

Influence in project progress Figure 7 shows external activities in a waterfall-like form. In reality (as usual for development processes), external activities are not executed in strictly sequential order. In the project we report about here, modeling and verification activities overlapped. Verification of a first design candidate uncovered two design flaws, one known and one unknown to the developers at the company. The known design flaw was a small timing problem: The parameters of the TDMA scheme were chosen in a way that allowed a component failure to go undetected for longer than permitted by the standard. Uncovering this known design flaw with modeling and verification strongly supported the validity of the model because this known design flaw was not communicated beforehand to the external consultants. The lengths of the frames and slots were adapted in response. The unknown design flaw was a vulnerability to radio interference from other frequency band users. The jamming device as specified by EN 54 was able to provoke spurious detections of component failures. Design flaws of this kind are very likely to remain undetected by conventional, testing-based approaches: Not only are prototype implementations and hardware needed as well as a non-deterministic jamming device, but experiments need to exhibit the (yet unknown) switching sequence which provokes spurious detection. Therefore, detection of this unknown design flaw was highly appreciated by the developers at the company. The failure scenarios (or counter-examples) found by the model checker provided sufficient insight to inspire an improved version of the protocol. Here the modular design of the overall model with separate environment and protocol model proved useful. We only needed to modify the protocol models. The new, modified models could then be verified successfully. Note that Sects. 3 and 4 present the final design. As described in Sect. 4.1.2, the developers at the company were able to confirm the model's timing predictions for alarm resolution using prototype hardware and software. This validation step further enhanced the confidence in the validity of the models.

In our opinion, a crucial aspect for the success of the case study was to keep the company involved through the interaction points shown in Fig. 7. As one cannot assume that the engineers at an SME are trained in formal methods, the artifacts produced by the consulting parties are not directly understandable for the company's engineers. Thus, extensive and interactive validation was necessary. Our approach was to let the external consultants serve as mediators between the two worlds of formal methods and classical engineering by translating the formal artifacts to terms and concepts known to the engineers. In particular visual narrations, model simulation, and counter-examples produced by the model checker served well for this purpose: The external consultants outlined the concrete cases represented by the formal representations and asked the developers to match this against their known views and expectations of the behaviour of protocols when implemented and executed on the target hardware. Re-discovering the known design flaw as discussed above and timing measurements on the prototype implementation further increased confidence in the validity of the models in our case study. Only the verification activity required less intensive interaction with the developers, here the focus was on dependability. Following [Jac09], one should avoid the false impression that the system is "completely verified" and has been proven not to fail under any circumstances by clearly pointing out all assumptions used during verification.

In summary, the progress of the project was not negatively affected by the consulting activities. The explicit non-dependence of the development activities ensured that the project would continue even if the consulting activities failed to provide any useful results.

The cost of formal methods According to the discussion at the beginning of this section, economic vulnerability and limited accountability are today seen as the main obstacles for using formal methods even in high-tech SMEs. Employing formal methods is perceived as a hardly assessable risk by SMEs. Our approach explicitly addressed the aspect of risks. First of all, the established development approach of the company was left unchanged and was merely *complemented* by external activities. The overhead caused by the addition of external activities was small, in particular when compared with their potential benefits. The company did not strictly depend on the results of the consulting party, they could have reduced their effort at any time. Feedback from the company confirmed that in all phases, interaction with a third party lead to a deeper understanding of the requirements, the environment assumptions, and the own design decisions. For instance, the sole activity of communicating a design to the consultants often helped to avoid overlooking crucial cases. Even if one of the later outsourced activities had failed, the engineers would have obtained benefits from the intensive interaction.

The impact of outsourcing formal methods is also accountable. A fixed amount of (financial or human) resources can be allocated to the external consulting service. It is the responsibility of the consulting party to provide guidance in identifying a set of requirements or designs which is most relevant for the project and feasible within the given budget. By explicitly managing the resource constraints while striving to maximize the information delivered to the company, the expected costs can be more precisely quantified.

The experience gained in the application of formal methods, especially by being more strict when producing a glossary, may motivate improvements in the software development process that go beyond a specific project and contribute toward increasing the overall product quality. The returns on the investments in the outsourcing of formal methods are thus not only local to the projects which are complemented by external formal activities but may also be noticeable in future undertakings. Having evidence originating in formal verification provides highly increased confidence in the design prior to starting the implementation phase when compared to classical design validation approaches. Additional effects on the implementation activities at the developing company are, for example, that the results of additional analyses performed on models can be utilized as means of validating the code, by direct comparison of the behavior of the implementation or as basis for the derivation of test cases. Feedback from the company reported a more streamlined implementation by using the high-level design description and significantly fewer issues during testing of the prototype implementation and hardware compared to previous, similar projects. Furthermore, the untimed analysis of the collision resolution protocol (cf. Sect. 4.2) provided valuable information for installation and operation of the WFAS.

Given that in this specific study, the cost of consulting was absorbed by our research project, we questioned the company's management about the perceived monetary value of the consulting activities and the perspective of applying formal methods at their own expense.

The foremost requirement to considering formal methods is the complexity of the project. In the context of a small company, not all development projects have a magnitude that justifies engaging a third party. Simple projects can, with relatively high confidence, rely on the experience of the company's developers alone. Additionally, the fixed costs associated with a business-to-business interaction would be disproportionate to the total costs and expected revenue of the project.

For projects where the magnitude and complexity justify hiring consultants, the economic reasoning follows a risk hedging strategy. From the potential savings expected from the use of formal methods compared to the expected cost of the project without consultants (e.g. because the testing effort is reduced as in our particular case), the company is willing to invest half. For instance, if the use of formal methods expectedly saves 50 % of the total projected costs without consulting, a risk-covering investment sum would amount to 25 % of those costs. Thus, the risk of budget overrun in the case that the expected savings are not attained at all is limited to 25 %. All this under the assumption that the developers at the company accept and follow the consultants' advice in a disciplined fashion.

We additionally conjecture that by leveraging the experience of the consulting parties and amortizing the costs of training over multiple projects, the costs incurred by the development companies are economical for safety-critical projects. The case study shows that by selectively investing in formal methods, a substantial quality improvement can be achieved at acceptable costs.

6. Conclusion

In this work, we investigated whether and by which approach small to medium-sized enterprises (SMEs) can benefit from formal methods, investigate whether today's verification approaches and tools are able to handle safety-critical designs as found at SMEs, and to ensure that certification tests for an international standard (EN 54) will not fail for the final system due to design flaws.

Our verification effort proved valuable for the development process of the company involved. We discovered previously unknown flaws that triggered significant revisions of the design. For the final design, we delivered concrete information about the operational circumstances for which our verification results apply. According to the testimony of the company, the project was accelerated compared to previous developments without the use of formal methods:

- the joint preparation of a dictionary was reported useful,
- the formalization of requirements including scenarios for conflicts and ambiguities in EN 54 helped clarification at the certification authority,
- the validation of detailed models helped to gain a deeper understanding of the design, while techniques like automatic reduction of quasi-equal clocks helped us to obtain more abstract, verifiable models,
- the simulation of the formalized design was reported useful to reproducibly inspect timing corner-cases at an early stage of development,
- compared to previous, similar projects, the formal model of the protocol design was reported to lead to a more streamlined implementation and significantly fewer issues were discovered during testing of the prototype implementation and hardware.

The first prototype implementation already passed all initial in-house tests, thus the test phase was substantially shortened and the effort of bug-fixing ameliorated. The certification tests at the certificate authority for the aspects related to the communication protocol were all passed successfully and without incident as indicated by our verification effort.

The formal modeling and verification of the new wireless fire alarm system proved challenging due to the following characteristics which are, to the best of our knowledge, unique in this combination:

- we covered a large number of components and topologies,
- we modeled complex environment assumptions,
- the majority of the relevant requirements are inherently real-time properties,
- modeling and verification were conducted truly *a priori*, complementing the development of an industrial safety-critical system at a small company,
- we needed to emphasize validation and dependability due to the lack of prior training in formal methods at the company.

Employing different state-of-the-art techniques and tools such as property decomposition, internal assumption treatment, meta-reasoning about topology coverage, and timed and untimed verification support of UPPAAL and SPIN enabled us to dependably provide sufficient evidence that EN 54 certification tests will not fail due to design flaws. We can conclude that formal methods and tools available today are directly suitable for the designs found in the SME we worked with. All models are available for download. [3]

What can be learned from our effort? We feel that technically, the key for providing valuable results in limited time was to generalize and formally specify relevant and involved *test procedures*, and *verify* that tests following those procedures will be passed. Because most companies specify tests and are used to this activity, formalizing test procedures seems to be a cost-efficient way of obtaining precise specifications for formal verification. From our experience, verifying that a design will pass all tests according to the given generalized test procedure can avoid huge test efforts; in addition, verification does not require initial implementations and hardware prototypes as opposed to conventional testing. Of course, the development goal here was not a system which only passes the certification tests, but a *good system*, one that fulfills all functions it was designed for. The WFAS, for example, should properly handle the failure of more than one sensor at a time. Checking such scenarios is possible with our techniques and would further increase confidence, but incurs additional costs for specification and modeling.

[3] http://swt.informatik.uni-freiburg.de/projects/CaseStudyRepository/WFAS.

Here, conventional testing is appropriate: From knowledge about the models, we expect the given scenarios to pass.

From the process point of view, we feel that the key for success was to *complement* and not to change the development process at the company. Careful selection of interaction points and the form of the interaction allowed us to achieve a high degree of validity and dependability. For example, to relate the models to the knowledge and experience of the designers by simulating different scenarios directly in UPPAAL. Discussing these scenarios facilitated the assessments of whether the models faithfully represent the design under development, i.e. model validation. In our case, a further indication for validity is that time bounds predicted by the model could be confirmed by the developers by measuring the implemented system. Feedback from the company also indicates that external consulting would be considered even if not backed up by a research project but offered commercially. As the substantial price of the certification procedure induces a high economical risk, approaches to mitigate this risk are welcome. Our approach outlined in Sect. 5.1, which is based on complementing the development, prioritizing analyses, and adjusting the goals of analysis according to a fixed time and resource budget instead of aiming at more complete verification, i.e. using *testing-like verification*, could serve as a starting point to establish external consulting in formal methods for SME. In our experience, training the developers at the company to conduct modeling and verification in-house is not economic for SMEs where the innovation cycle is dominated by implementation, maintenance, and evolution of existing systems while radically new designs are developed only every few years.

We see that formal methods and tools available today are capable of treating problems of SMEs while adding value to their development process. *A priori* design verification as conducted in our case study facilitates finding design errors early and potentially saves efforts and costs. For the certification of critical systems, verification of design models could also improve certification processes, in addition to complementing testing by providing formal evidence of correctness as in DO-333 [RTC11].

References

[AD94] Alur R, Dill DL (1994) A theory of timed automata. Theor Comput Sci 126(2):183–235

[AEW07] Aranda J, Easterbrook SM, Wilson G (2007) Requirements in the wild: how small companies do it. In: RE, IEEE, pp 39–48

[BDL04] Behrmann G, David A, Larsen KG (2004) A tutorial on Uppaal. In: SFM-RT 2004, number 3185 in LNCS, Springer, pp 200–236

[BOG02] Bhargavan K, Obradovic D, Gunter CA (2002) Formal verification of standards for distance vector routing protocols. J. ACM 49(4):538–576

[CHR91] Chaochen Z, Richard Hoare CA, Ravn AP (1991) A calculus of durations. Inf Process Lett 40(5):269–276

[CZZ$^+$13] Chen Z, Zhang D, Zhu R, Ma Y, Yin P, Xie F (2013) A review of automated formal verification of ad hoc routing protocols for wireless sensor networks. Sensor Lett 11(5):752–764

[DCC$^+$13] Davis JA, Clark MA, Cofer DD, Fifarek A, Hinchman J, Hoffman JA, Hulbert B, Miller SP, Wagner L (2013) Study on the barriers to the industrial adoption of formal methods. In: Pecheur C, Dierkes M (eds) Formal methods for industrial critical systems (FMICS) 2013, volume 8187 of LNCS, Springer, pp 63–77

[DFAW$^+$10] Dietsch D, Feo-Arenis S, Westphal B (2010) Abwicklung von Sofwareentwicklungsaufträgen in kleinen und mittleren Unternehmen: Analyse. Technical report, Universities of Freiburg and Mannheim, Project SALOMO

[DFAWP11] Dietsch D, Feo-Arenis S, Westphal B, Podelski A (2011) Disambiguation of industrial standards through formalization and graphical languages. In: IEEE 19th international requirements engineering conference, pp 265–270

[DIN97a] DIN eV (1997) Fire detection and fire alarm systems—Part 2: Control and indicating equipment; German version EN 54-2

[DIN97b] DIN eV (1997) Fire detection and fire alarm systems; German version EN 54

[DIN05] DIN eV (2005) Fire detection and fire alarm systems—Part 25: Components using radio links and system requirements, German version EN 54-25

[DSSW99] Dong Y, Smolka SA, Stark EW, White SM (1999) Practical considerations in protocol verification: the e-2c case study. In: Engineering of complex computer systems, 1999. ICECCS'99. Fifth IEEE international conference on, IEEE, pp 153–160

[FvGH$^+$12] Fehnker A, van Glabbeek RJ, Höfner P, McIver A, Portmann M, Tan WL (2012) Automated analysis of AODV using UPPAAL. In: Flanagan C, König B (eds) Tools and algorithms for the construction and analysis of systems—18th international conference, TACAS, volume 7214 of LNCS, Springer, pp 173–187

[GEFP10] Gerke M, Ehlers R, Finkbeiner B, Peter H (2010) Model checking the flexray physical layer protocol. In: Kowalewski S, Roveri M (eds) Formal methods for industrial critical systems - 15th international workshop, FMICS, volume 6371 of LNCS, Springer, pp 132–147

[GGLA98] Garcés R, Garcia-Luna-Aceves JJ (1998) Collision avoidance and resolution multiple access (CARMA). Clust Comput 1(2):197–212

[GH$^+$95] George C, Haxthausen AE (1995) The RAISE Development Method. The BCS Practitioners Series. Prentice Hall Int, UK

[GM12] Gnesi S, Margaria T (2012) Formal methods for industrial critical systems: a survey of applications. Wiley, New York

[GVZ06] Gebremichael B, Vaandrager FW, Zhang M (2006) Analysis of the zeroconf protocol using UPPAAL. In: Sang Lyul M, Wang Y (eds) Proceedings of the 6th ACM and IEEE International conference on Embedded software, EMSOFT 2006, October 22–25, 2006, Seoul, Korea, ACM, pp 242–251

[HLN+90] Harel D, Lachover H, Naamad A, Pnueli A, Politi M, Sherman R, Shtull-Trauring A, Trakhtenbrot MB (1990) STATEMATE: A working environment for the development of complex reactive systems. IEEE Trans Software Eng 16(4):403–414

[Hol04] Holzmann GJ (2004) The SPIN Model Checker—primer and reference manual. Addison-Wesley, USA

[HRR91] Halbwachs N, Raymond P, Ratel C (1991) Generating efficient code from data-flow programs. In: PLILP, number 528 in LNCS, Springer, pp 207–218

[HWA+12] Herrera C, Westphal B, Feo-Arenis S, Muñiz M, Podelski A (2012) Reducing quasi-equal clocks in networks of timed automata. In: Jurdzinski M, Nickovic D (eds) FORMATS, volume 7595 of Lecture notes in computer science, Springer, pp 155–170

[HWP14] Herrera C, Westphal B, Podelski A (2014) Quasi-equal clock reduction: more networks, more queries. In: Ábrahams E, Havelund K (eds) TACAS, volume 8413 of Lecture Notes in Computer Science, Springer, pp 295–309

[Jac09] Jackson D (2009) A direct path to dependable software. CACM 52(4):78–88

[JW13] Jubran O, Westphal B (2013) Formal approach to guard time optimization for TDMA. In: RTNS, ACM, pp 223–233

[JW14] Jubran O, Westphal B (2014) Optimizing guard time for TDMA in a wireless sensor network—case study. In: LCN, IEEE, pp 597–601

[K+81] Kant E (1981) The refinement paradigm: the interaction of coding and efficiency knowledge in program synthesis. IEEE Trans. Softw Eng 7(5):458–471

[K+10] Kamali M (2010) Self-recovering sensor-actor networks. In: Mousavi MR, Salaün G (eds) FOCLASA, volume 30 of EPTCS, pp 47–61

[KB03] Kopetz H, Bauer G (2003) The time-triggered architecture. Proc IEEE 91(1):112–126

[KLN+15] Kim JH, Larsen KG, Nielsen B, Mikucionis M, Olsen P (2015) Formal analysis and testing of real-time automotive systems using UPPAAL tools. In: Núñez M, Güdemann M (eds) Formal methods for industrial critical systems (FMICS) 2015, volume 9128 of LNCS, Springer, pp 47–61

[Lan96] Lano K (1996) The B Language and Method. Springer, New York

[LAO08] Laporte CY, Alexandre S, O'Connor R (2008) A software engineering lifecycle standard for very small enterprises. In: Software process improvement, 15th European Conference, EuroSPI, volume 16 of communications in computer and information science, Springer, pp 129–141

[MAS06] Madl G, Abdelwahed S, Schmidt DC (2006) Verifying distributed real-time properties of embedded systems via graph transformations and model checking. Real-Time Syst 33(1-3):77–100

[MM04] Siva Ram Murthy C, Manoj BS (2004) Ad hoc wireless networks: architectures and protocols. Prentice Hall PTR, UK

[N+00] Nokula U (2000) A state-of-the-practice survey on req. engineering in small- and medium-sized enterprises. Technical report, Lappeenranta Univ. of Tech

[NL03] Neill CJ, Laplante PA (2003) Requirements engineering: the state of the practice. IEEE Softw 20(6):40–45

[OD08] Olderog E-R, Dierks H (2008) *Real-time systems*. Cambridge University Press, Cambridge

[Ric02] Richardson I (2002) SPI models: what characteristics are required for small software development companies? Softw Quality J 10(2):101–114

[RTC11] RTCA (2011) DO-333 Formal methods supplement to DO-178C and DO-278A

[TPB+08] Tripakis S, Pinello C, Benveniste A, Sangiovanni-Vincentelli AL, Caspi P, Natale MD (2008) Implementing synchronous models on loosely time triggered architectures. IEEE Trans Comput 57(10):1300–1314

[vOS01] van Osch MJP, Smolka SA (2001) Finite-state analysis of the CAN bus protocol. In: 6th IEEE international symposium on high-assurance systems engineering (HASE 2001), Special topic: impact of networking, IEEE Computer Society, pp 42–54

[WLBF09] Woodcock J, Larsen PG, Bicarregui J, Fitzgerald J (2009) Formal methods: practice and experience. ACM Comput Surv 41:19:1–19:36

[WPP04] Wibling O, Parrow J, Pears AN (2004) Automatized verification of ad hoc routing protocols. In: de Frutos-Escrig D, Núñez M (eds) Formal techniques for networked and distributed systems (FORTE), volume 3235 of LNCS, Springer, pp 343–358

[WT95] Wong-Toi H (1995) Symbolic approximations for verifying real-time systems. PhD thesis, Stanford University

Received 31 May 2015

Accepted in revised form 2 February 2016 by Stephan Merz, Jun Pang, and Jin Song Dong