A Model-Based Approach to Formal Verification in Early Development Phases: A Desalination Plant Case Study [*]

Alarico Campetelli, Maximilian Junker, Vasileios Koutsoumpas, Xiuna Zhu
Technische Universität München
Garching bei München, 85748, Germany

Birthe Böhm, Maria Davidich, Jan Christoph Wehrstedt
Siemens AG
Otto-Hahn-Ring 6, München, 81739, Germany

Abstract: Model-based development approaches have attracted a lot of attention in the last decade due to their ability to deal with complexity in large software engineering projects. However, a natural question raises, to what extent model-based development approaches are suited to tackle engineering challenges from other application domains such as the automation domain. In order to answer this question, we apply the model based SPES development method to an industrial case study from the automation domain, focusing on the control software components of a desalination plant. In particular, we demonstrate the formalization of requirements, the elicitation of a functional and a logical specification with automatic verification of formal requirements. Compared to classical software engineering approaches, where verification phase is done after the logical implementation phase, our approach focuses on a strict integration of modelling and automatic verification of formal requirements in early development phases such as the system functionalities definition.

Keywords: embedded systems, model-based development, requirements management, formal verification, automation domain

1 Introduction

Formal methods are widely recognized as powerful engineering methods for the specification, simulation, development, and verification of embedded systems [BS01]. They follow the principle of "*correctness by construction*" and are therefore well suited for safety-critical systems [HC02]. Although the advantages of formal methods are well known [LG97], there are many limitations preventing their usage in industrial systems.

The industrial systems evolve to more and more complex structures to meet the increasing complexity of requirements, especially when combining embedded systems with physical components, termed Cyber-Physical Systems (CPSs). Moreover, the involvement of multiple engineering disciplines, which are targeting cross cutting aspects of the system under development, facilitates wider application of formal methods, which can overcome the underlying challenges. In particular, the pursuit of shorter development time and smaller project budgets requires: 1) an increased parallelization of the single development steps 2) consistent exchange of information between the involved disciplines 3) a strategy for the

[*]This work was partially funded by the German Federal Ministry of Education and Research (BMBF), grant "SPES XT, 01IS12005M". The responsibility for this article lies with the authors.

validation of functional requirements at an early stage. Thus, flexible and efficient modelling frameworks are required which should include more powerful engineering tools and methods. Such a modelling framework has been developed within the German research project SPES XT[1][PHAB12], which on the one hand facilitates reuse and allows an efficient artefacts development while on the other hand it is designed to be independent of any application domain. In doing so, we apply the SPES methodology to a case study from the automation domain, focusing on the control software components of a desalination plant.

This paper is structured as follows: Section 2 presents the related foundations w.r.t. the system model, the development method and tool support required for the case study. Section 3 illustrates the set-up of a case study of a desalination plant which is modelled according to a formal system model FOCUS and the SPES development method. Furthermore, the verification of functional requirements is addressed. Section 4, discusses the benefits of a model-based development approach. Finally, Section 5 concludes the present work and describes possible future directions.

2 Background

We perform this case study based on the formal system model FOCUS [BS01], the model-based development method SPES[BDH$^+$12], as well as the CASE tool AutoFOCUS 3 which supports both the formal system model as well as the development method.

2.1 System Model

The system model FOCUS (for details, see [BS01]) is based on a specific notion of a system. A system is represented as a model consisting of a syntactic interface, denoting the input and output channels. Through the channels the system can communicate with its environment. The behavior of the system can be observed at the interface.

Formally, we can represent a syntactic interface as a pair $I \triangleright O$, where I is a set of input channels and O is a set of output channels. We can represent the behaviour of a system as a function $\overline{I}^{\infty} \rightarrow \overline{O}^{\infty}$. Here, $\overline{C}^{\infty}$ denotes an infinite sequence of messages for every channel in the channel set C. Using this notion of a system and its behaviour, we can compose systems from subsystems. Formally, we can define the composition $S_1 \otimes S_2$ for two interface behaviours S_1 and S_2 representing the behaviour of the subsystems.

2.2 Development Method

In the research project SPES, the SPES Modelling Framework (SPES MF) is developed to enable seamless model based development for embedded systems [BDH$^+$12]. The SPES MF focuses on artefacts that are created during the development of embedded systems and structures artefacts according to *viewpoints*. The SPES MF differentiates between the following four viewpoints:

[1]http://spes2020.informatik.tu-muenchen.de/spes_xt-home.html

- **Requirements Viewpoint**: structured documentation and analysis of requirements
- **Functional Viewpoint**: structured documentation and analysis of system functions and their behaviour (Functional Architecture)
- **Logical Viewpoint**: structured documentation and analysis of the logical solution (Logical Architecture)
- **Technical Viewpoint**: structured documentation and analysis of the technical solution (Technical Architecture).

To further reduce the complexity of the engineering process, the SPES MF introduces layers of granularity, where a coarse-grained engineering problem is decomposed into a number of more fine-grained engineering problems following the principle "divide and conquer", i.e. the composition of the fine-grained solutions is a solution for the coarse-grained engineering problem. Whenever a coarse-grained engineering subject is decomposed into a number of fine-grained engineering subjects, a new layer of granularity is constructed. Hereby, the system is decomposed into smaller and less complex parts. Since the number of such layers depends on the properties of the individual engineering context of an embedded system, the SPES MF does not define a fixed number of granularity layers. Most artefacts that are included within the SPES MF can formally be described using the formal system model outlined above.

2.3 Tooling

AutoFOCUS 3 [HF11] is an open source research tool supporting model-based development, formal specification and analysis, as well as design space exploration techniques[2]. It provides a broad range of modelling concepts at different levels of formalization for different phases in the development ranging from requirements (using the AutoFOCUS 3 plugin MIRA [TMR13]) to platform architecture and deployment. Specifically, it provides support for most of the modelling artefacts described in the SPES MF.

3 Case Study

Yet today, 780 million people live in lands where there is a lack access to potable water. The production of drinking water by desalination of sea water represents a way to reduce the lack of clean water. In the following we consider a typical desalination plant with reverse osmosis (See Figure 1). The general principle of desalination plant with reverse osmosis is the following.

Water is collected by four beach wells along the coast. The salt water is then pumped into the seawater tank, where it is collected. After pretreating with various chemicals for stabilization and biochemical cleaning the actual desalination process can take place. Therefore the water passes several filters before separating the salt in the high pressure section. After post treatment new drinking water can be delivered into the water net.

[2]http://af3.fortiss.org

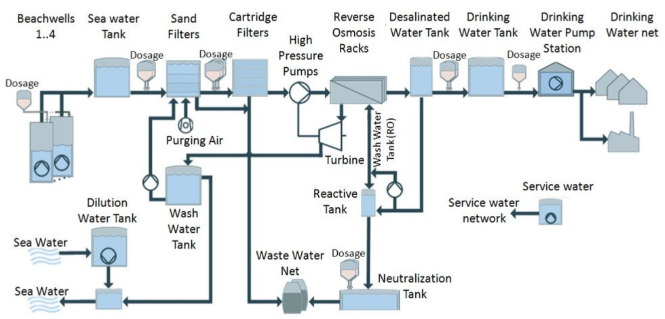


Figure 1: Configuration of a desalination plant [3].

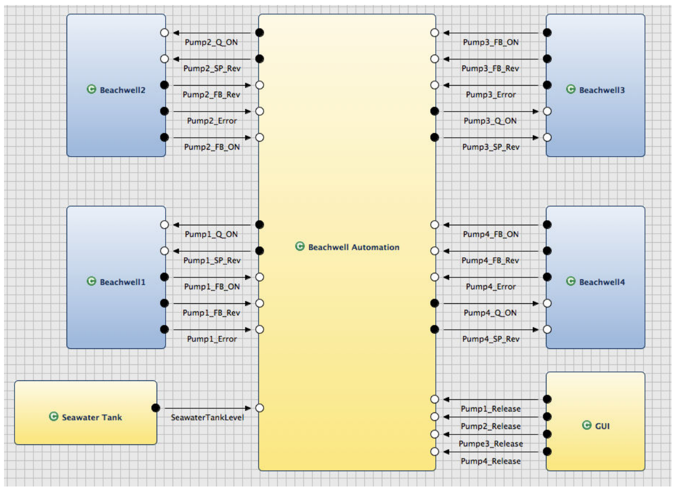


Figure 2: Determining the system boundary.

This type of plant usually is of middle to high automation complexity. As the entire desalination plant is very complex, therefore for plausibility we consider only a small part of the Beach Well-section (BWS).

3.1 System Overview

The main function of BWS in a desalination plant is pumping seawater from the sea to the sea water tank. This water is filtered by natural layers of sand. In order to guarantee sufficient water to operate this desalination plant, the level of the seawater tank should be monitored and controlled by switching pump valves.

[3]The technical properties of this running example were taken from a free instructional DVD available at http://goo.gl/ppsNNo

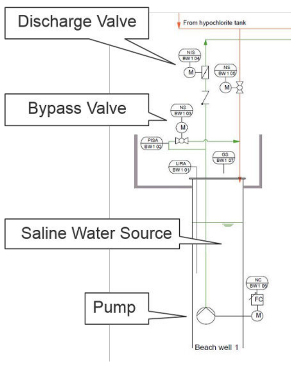


Figure 3: Beach well pump instrumentation.

Our work focused on formal modelling of functional requirements of the BWS. The communicating signals and data types of signals are provided by requirements documents. The target BWS consists of the following main components: four Beach Wells (BW), GUI, Sea Water Tank, and Beach Well Automation (BA). Fig. 2 illustrates the system boundary and interface descriptions of the components modelled in AutoFOCUS 3. Beach Well Automation is a central component which coordinates the other components. GUI describes the interaction between the plant operator (i.e. the user) and the automation software during plant operation. The sea water tank has a level sensor which monitors the water tank level, whereas each Beach Well has the same facilities and functionalities. Fig. 3 depicts the process and instrumentation diagram of Beach Well pump 1. Each Beach Well has a pump, a saline water source, a bypass control and a discharge valve. The sea water tank flow is controlled by pumps, whereas the Bypass control guarantees a minimal load for pumps. The Discharge valve protects the pump and guarantees smooth starting and stopping of the pumps. The pump is driven by an electrical motor which is controlled by the automation software as well.

3.2 Requirement Modelling

We start our proposed seamless and continuous design approach by system requirement modelling of BWS. To provide artefacts of the system like glossary, use cases description, architectural constraints, and external environment of the system, MIRA is used in this phase to elicit, specify and analyse requirements. For instance, to avoid wrong interpretation we defined all relevant technical terms by building a glossary.

To formalize the textual requirements we propose to categorize requirement. In this example, all requirements are categorized into two categories according to the component to which they deploy: Beach Well (BW) and Beach Well Automation (BA).

Then each requirement is specified in text form and analysed as an use case. Take a requirement on controlling *Discharge Valve* and *Motor Pump* (BW4) as an example. As shown in Table 2, each single requirement in AutoFOCUS 3 is formalized based on a pattern with a name, type, description, status, priority and tagged if it belongs to safety requirements.

BW1	Every pump delivers between 400 and 750 m^3/h
BW2	Pumps are only allowed to run if filling level of beach well is sufficient
BW3	Bypass control guaranties minimal load for pump
BW4a	Discharge valve has to be closed before pump starts
BW4b	Discharge valve has to open after pump starts
BW4c	Discharge valve has to close after pump stops
BA1	One pump has to be in standby
BA2	Adapt pump load to achieve 80% filling level in sea water
BA3	Pumps start and stop cascaded
BA4	In case of failure standby pump has to take over

Table 1: Textual requirements.

Name	BW4
Type	Requirement
Description	Discharge valve has to be closed before pump starts. Discharge valve has to open after pump starts. Discharge valve has to close after pump stops.
Status	identified
Priority	normal
Safety requirement	no

Table 2: Formalization of requirement BW4 in a pattern.

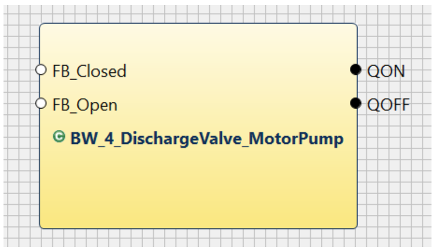
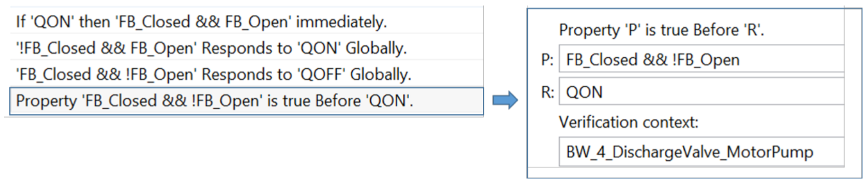


Figure 4: Interface definition based on requirement analyses.

The requirement analysis is integrated with architecture design, since the component interface is derived by the relevant use cases specification which describes the components and the interactions between them. As shown in Fig.4, the partial interface of a component can be derived from the *BW4* use case description of the component Beach Well. Requirement BW4 has two input ports (FB_Closed and FB_Open, which indicate the feedback of pump to mark if discharge valve is closed or open) and two output ports (QON and QOFF which determine pump motor's work state is on or off). In this example, all i/o ports are Boolean-valued.

To avoid the consistency and ambiguity problems, the informal requirements should be modelled formally with structured text templates but still in a use-friendly textual manner. Fig.5 illustrates the Assumption/Guarantee (A/G) specification of requirement BW4. Thus, requirements for completeness, correctness, and consistency can be later analysed


If 'QON' then 'FB_Closed && FB_Open' immediately.
'!FB_Closed && FB_Open' Responds to 'QON' Globally.
'FB_Closed && !FB_Open' Responds to 'QOFF' Globally.
Property 'FB_Closed && !FB_Open' is true Before 'QON'.

Property 'P' is true Before 'R'.
P: FB_Closed && !FB_Open
R: QON
Verification context:
BW_4_DischargeValve_MotorPump

Figure 5: Assumption/Guarantee specification. The informal requirement BW4a is formalized as A/G specification interpreted as property that 'FB_Closed && !FB_Open' is true before 'QON '.

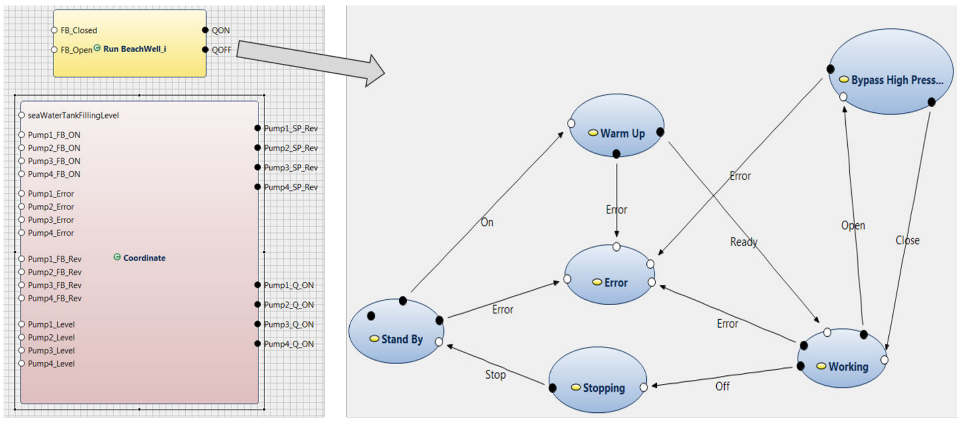


Figure 6: The functions of the functional architecture root (left) and the state machine implementing the requirement BW4 where each state represent a mode of a beach well (right).

with a model checker (cf. Sec. 3.4). Thereby, verification is possible in the early stage of requirement modelling process.

3.3 Functional Architecture

A functional architecture of the system is derived from the requirement artefacts according to the presented SPES development method. This view is focused on the behaviour of the system, i.e. on functionalities and features. Moreover, it represents also a sort of documentation of the system to be developed. Functions can be subdivided in sub-functions and relations can be structured between them.

We constructed a functional representation of the system in AutoFocus 3 using component architectures. Each component has a formal defined interface through i/o ports and an internal behaviour. We defined a root component that represents the whole functional architecture. We subdivided the requirements in two categories: requirements relative to the control of the whole plant and to a single beach well element. Therefore, we model each requirement category as a subcomponent, which is contained in the root component of the functional architecture. The i/o ports of these components are derived directly from the relative requirement artefacts. In fact, the functional specifications of the requirements permit to define the typed i/o interface of the functional architecture components.

We decided to implement the internal behaviour of each component with a finite state machine. There are also other alternatives, as for instance functional and code specifications. In Figure 6 the components corresponding to the requirement categories are depicted and the state machine implementing the single beach well functionalities is represented. This implementation has only the states and transitions relevant for the presented BW4 requirement. State machines can be simulated with desired environment conditions and can be formally verified as explained in the next section.

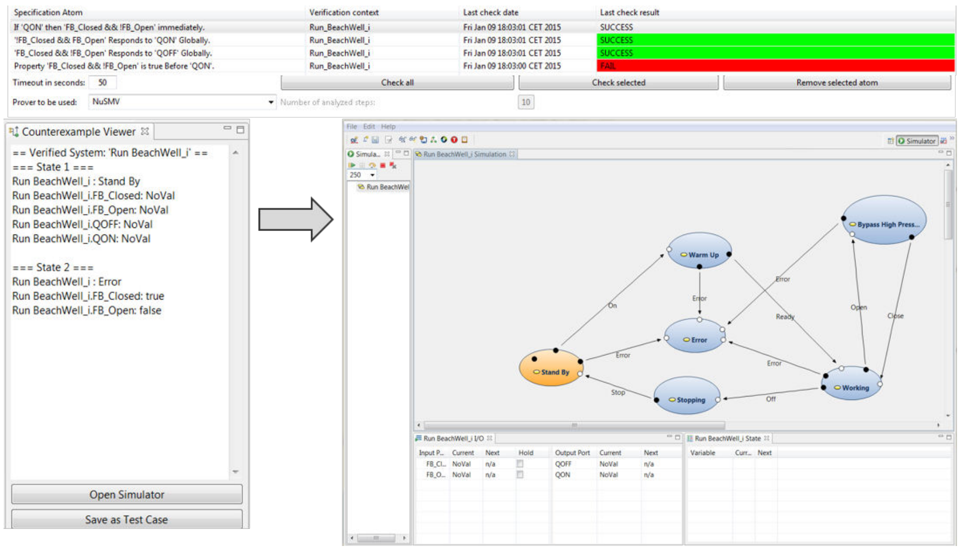
3.4 Verification

The definition of a functional architecture of the system permits to detect inconsistent and incomplete requirements using simulation and verification techniques. That is possible with enough detailed definition of the requirements with informal and formal artefacts. In our case study, we modelled formally each requirement through verification patterns, which permits the formal verification of its validity through the NuSMV[4] model checker. Model checking offers an automatic and exhaustive proof of the system. The patterns are used for the presentation, codification and reuse of formal properties for AutoFOCUS 3 models. We associate verification patterns from the requirement to the functional architecture with refinement rules, which define a correspondence between the i/o ports of the formalized requirements and the functional architecture.

The verification patterns integrated in the AutoFOCUS 3 model view are of two types: patterns based on the specification patterns presented in [DAC99] and specifications conceived for AutoFOCUS 3 model. Further information about the specification of properties and the model checking integration in AutoFOCUS 3 are in [CHN11]. A verification pattern represents common and recurrent properties, as for instance a state or event of the system that must be reached after a state or event has happened. They are saved with the verification results along with the model itself. Each pattern is structured in a predefined logical part of the property, which determines its semantics and text fields to be filled. Typically, the user writes in these fields expressions that correspond to states, events, or port values and logical operators to combine them. When the pattern is completed an automatic type and name check is made to prove the correctness of the information inserted. Short descriptions help the user to understand the role of each text field.

A verification context has to be specified, in terms of components to be considered, to execute the verification. Then the tool translates the property to a formal logical property and the AutoFOCUS 3 model to an input model for NuSMV and executes it. A counterexample is reported if the result of the verification is negative. A counterexample may represent in the state machine either a path from the initial state to an erroneous state or a loop, where a desired behaviour never happens. Each step of the counterexample sequence is described by the actual value of variables and i/o ports of the verified model. The counterexample can be stepwise simulated in a graphical view of the tool to debug the AutoFOCUS 3 model. In each simulation step the variables and i/o ports values are visualized. The verification of the requirements in the functional architecture and the counterexample simulation are shown in Figure 7.

[4]http://nusmv.fbk.eu



Specification Atom	Verification context	Last check date	Last check result
If 'QON' then 'FB_Closed && !FB_Open' immediately.	Run_BeachWell_i	Fri Jan 09 18:03:01 CET 2015	SUCCESS
'!FB_Closed && FB_Open' Responds to 'QON' Globally.	Run_BeachWell_i	Fri Jan 09 18:03:01 CET 2015	SUCCESS
'FB_Closed && !FB_Open' Responds to 'QOFF' Globally.	Run_BeachWell_i	Fri Jan 09 18:03:01 CET 2015	SUCCESS
Property 'FB_Closed && !FB_Open' is true Before 'QON'.	Run_BeachWell_i	Fri Jan 09 18:03:00 CET 2015	FAIL

Timeout in seconds: 50

Prover to be used: NuSMV

Number of analyzed steps: 10

Counterexample Viewer

```
== Verified System: 'Run BeachWell_i' ==
=== State 1 ===
Run BeachWell_i : Stand By
Run BeachWell_i.FB_Closed: NoVal
Run BeachWell_i.FB_Open: NoVal
Run BeachWell_i.QOFF: NoVal
Run BeachWell_i.QON: NoVal

=== State 2 ===
Run BeachWell_i : Error
Run BeachWell_i.FB_Closed: true
Run BeachWell_i.FB_Open: false
```

Run BeachWell_i I/O

Input P...	Current	Next	Hold	Output Port	Current	Next
FB_Cl...	NoVal	n/a	☐	QOFF	NoVal	n/a
FB_O...	NoVal	n/a	☐	QON	NoVal	n/a

Run BeachWell_i State

Variable	Curr...	Next

Figure 7: Model checking of the requirement BW4 in the functional architecture: representation of the requirement through verification patterns (upper part) and the simulation of the counterexample that is generated because the requirement BW4a was violated (lower part).

4 Discussion

In the presented case study we formally verified requirements over the system behaviour expressed in the functional architecture. We studied a seamless model integration between requirements, formal properties definition, model checking and counterexample simulation. AutoFocus 3 provided a suitable environment to apply these concepts to our automation case study of beach wells. Our verification approach in the early development phases allows to avoid errors and incomplete requirements that can be much more expensive, if detected in successive development phases. In the requirements definition and elicitation phases, we specified for each requirement a typed i/o interface in a functional specification and we formalized behaviours in logical formulas expressed in verification patterns. These specifications make the requirements much more precise and with a defined semantics, in comparison with textual informal or semi-formal requirements descriptions. Furthermore, we verified and executed counterexample analysis without requirements of specific skills in formal verification.

Besides erroneous behaviour and incompleteness, the definition of dependencies between functions helps to solve interaction conflicts. Moreover, there is an improvement in the communication between experts of different disciplines, which are normally involved in the development of automation software. In fact, functional descriptions do not require all implementation details in contrast to the following logical and technical representations of the system. In these representations the functions modelled in the functional architecture are refined. Our verification approach proofs the system in a representation that abstracts from many details. Some of them are specific for a certain discipline. Therefore, we have an executable and verifiable system representation in a sort of discipline neutral language.

5 Conclusion & Outlook

In this paper we investigated the application of the SPES development method for the automation domain. In particular, a case example of a seawater desalination plant was modelled according to that method. Special emphasis was given to the verification of functional requirements at an early stage of the development process. Moreover, AutoFOCUS 3 permits to model the successive logical and technical representations of the system.

Thus, the intention of this paper was to provide an example in order to exhibit the additional value provided by SPES development method in the automation domain. Last but not least, possible future direction will be to explore the scalability of our approach, how to integrate real-time aspects and how deployment of these verified models to the engineering tools and the hardware used in the automation domain is possible. This gives a chance to transfer know-how to other application domains such as the smart grid domain and the execution of custom case studies within these domains.

References

[BDH+12] Manfred Broy, Werner Damm, Stefan Henkler, Klaus Pohl, Andreas Vogelsang, and Thorsten Weyer. Introduction to the SPES Modeling Framework. In Klaus Pohl, Harald Hönninger, Reinhold Achatz, and Manfred Broy, editors, *Model-Based Engineering of Embedded Systems*, pages 31–49. Springer Berlin Heidelberg, 2012.

[BS01] M. Broy and K. Stølen. *Specification and development of interactive systems: focus on streams, interfaces, and refinement*. Springer, 2001.

[CHN11] Alarico Campetelli, Florian Hölzl, and Philipp Neubeck. User-friendly Model Checking Integration in Model-based Development. In *24th International Conference on Computer Applications in Industry and Engineering*, 2011.

[DAC99] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in Property Specifications for Finite-State Verification. In *ICSE*, pages 411–420, 1999.

[HC02] Anthony Hall and Roderick Chapman. Correctness by construction: Developing a commercial secure system. *Software, IEEE*, 19(1):18–25, 2002.

[HF11] Florian Hölzl and Martin Feilkas. AutoFOCUS 3 - A Scientific Tool Prototype for Model-Based Development of Component-Based, Reactive, Distributed Systems. In *Model-Based Engineering of Embedded Real-Time Systems*, volume 6100 of *Lecture Notes in Computer Science*, pages 317–322. Springer Berlin / Heidelberg, 2011.

[LG97] Luqi and Joseph A Goguen. Formal methods: promises and problems. *Software, IEEE*, 14(1):73–85, Jan 1997.

[PHAB12] Klaus Pohl, Harald Hönninger, Reinhold Achatz, and Manfred Broy. *Model-based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, 2012.

[TMR13] Sabine Teufl, Dongyue Mou, and Daniel Ratiu. MIRA: A Tooling-Framework to Experiment with Model-Based Requirements Engineering. In *RE*, 2013.