Robustness Analysis for Indoor Lighting Systems

An Application of Model Checking in Large-Scale Distributed Control Systems

Richard Doornbos, Jacques Verriet
TNO-ESI
Eindhoven, Netherlands
e-mail:{richard.doornbos, jacques.verriet}@tno.nl

Mark Verberkt
Philips Lighting
Eindhoven, Netherlands
e-mail: mark.verberkt@philips.com

***Abstract*—Modern lighting systems are configurable systems-of-systems that have to operate in an environment that they cannot fully control. These systems have to guarantee the continuation of their functionality regardless of the events in their environment. As testing and simulation are not able to identify all possible interactions of a lighting system and its environment we propose a model checking approach to analyze a lighting system's robustness. To allow easy integration in lighting system development, the approach uses the same configuration options as the lighting systems under study. We apply our approach to an office lighting system and show how model checking can be used to analyze the robustness against network failures and to investigate communication protocols to improve system robustness.**

Keywords-distributed systems; system robustness; model checking; Uppaal; indoor lighting systems.

I. INTRODUCTION

Many believe that current trends like Internet of Things (IoT) will only thrive when a well-established industry will take them up. This industry could well be the lighting industry, which is going through a number of paradigm shifts: from traditional light sources (incandescent, fluorescent lamps) to LED (light emitting diode), from simple light sources to intelligent, networked, multi-sensor luminaires, and from individually controlled light points to large-scale distributed control systems. These major changes fit well to the IoT idea of having many internet-connected sensor/actuator nodes distributed over an area of interest, e.g., an office building.

A lighting system for an office building is a prime example of a complex system: it is a large-scale distributed system, containing thousands of sensor and actuator components, which exhibit event-based behavior. The system can have very many configurations, but is comprised of only a limited number of types of components. Typical for modern systems is the complicating factor that it has to cooperate with other systems (heating/cooling, network, power, security), sometimes leading to conflicting requirements.

A. Related Work

In this paper, we present a model checking approach to analyze the robustness of large-scale indoor lighting systems to (erroneous) events in its environment. Robustness is the ability of a system to continue to operate correctly across a wide range of operational conditions, and fail gracefully outside of that range [7].

To the best of our knowledge, model checking has not been applied to large-scale lighting systems. There are many examples of other (industrial) systems that have been analyzed using model checking. Examples include elevator control systems [9] and railway interlockings [12].

In another example, van den Berg et al. [11] use a domain-specific language for specifying medical imaging systems. This domain-specific language is translated into Uppaal [10] for performance analysis. The output of this analysis in translated in information understandable to system designers: analysis results are transformed into lower and upper bounds of system response times. Hendriks et al. [8] have applied a similar approach to create optimal schedules of a wafer scanner.

Similar results have been achieved using MechatronicUML, an Eclipse-based tool suite for the design of cyber-physical systems [2]. It comprises a modeling language and a development process. To validate software correctness, Gerking [3] has developed transformations from MechatronicUML to Uppaal [10] and vice versa: a MechatronicUML design is transformed into an Uppaal model and counterexamples identified by Uppaal are translated back into the MechatronicUML language. This allows system designers to formally validate their system designs without knowledge of Uppaal's timed automata formalism.

Combemale et al. [1] present a formal approach to tracing back analysis results. Their approach requires an input language with an operational semantics definable as finitely-branching transition systems; it transforms analysis results back to the syntax and operational semantics of a domain-specific input language. Input for their approach is a formal relation between the states of the input language and those of the target language. They illustrate their approach using a timed process modeling language as input language and a timed Petri net language as analysis language.

The systems for which formal analysis and back transformations have been used are quite different from the types of systems that we consider in this paper. The basis of our approach is the configurability of lighting systems: a huge variety of lighting systems can be constructed from a few configurable component types. This also holds for logistic control systems. Verriet et al. [13] have shown how

 ISBN: 978-1-61208-399-5

warehouse controllers can be configured using domain-specific tooling. A controller can be configured by selecting the appropriate planning and scheduling components and instantiating their behaviors from a library of configurable protocols. This tooling allows warehouse designers to specify a warehouse control system without knowledge of implementation details. A back transformation is not required, because the generated controller provides feedback in terms understandable to the warehouse designer.

B. Outline

The paper is organized as follows. Section II explains the robustness challenges in the design of lighting systems. The indoor lighting case is introduced in Section III. Section IV describes the lighting system model, which is used for the robustness analysis described in Section V. Section VI describes the validation of the model by coupling the model to a test setup. In Section VII, we present strategies to improve the robustness of lighting systems against failures in the communication network. Section VIII reflects on the modeling approach and its industrial applicability and describes future work. Section IX summarizes the paper.

II. Problem Statement

The paradigm shifts mentioned in Section I lead to many technical and business challenges. The technical challenges can be summarized in non-functional aspects, such as interoperability (cooperation/collaboration between components from different vendors), availability and robustness (correct lighting behavior at all times), and security (system integrity and user privacy).

A. Goal

For a lighting company, it is crucially important to guarantee correct behavior of a lighting system. In this paper, we use a model-based approach to address this challenge. In particular, we apply model checking to assess the robustness of a lighting system's distributed control system.

Our goal to identify robustness issues in a lighting system has led to a two-step approach. The first step focuses on modeling normal system behavior and identifying the system's reactions to events in its environment. These events include normal events like occupancy detection events, but also failures, such as loss or delay of messages. The second step involves finding improvements to make the system less sensitive to undesirable environmental events.

B. Approach

Understanding and predicting the behavior of lighting systems is hard without formal modeling. This is due to the complex interaction of a vast number of parallel processes and events. Simulation provides little chance of identifying (rare) robustness flaws. Since model checking allows even the rarest events to be found, we apply formal modeling and model checking as the main direction for our research. We propose a model checking approach that fits the configurability of distributed lighting systems. We require therefore that a large variety of system models can be configured from a small set of model elements in the same manner that many lighting systems can be configured using a small set of component types.

Because formal modeling skills are not commonly available in industry, we propose an approach where a system configuration is converted into a formal model that is used to analyze a lighting system. An important aspect of our work is the translation of the model checking results back to the lighting system domain.

We aim to eventually hide the complexity of the formal modeling tools completely by adaptation of tooling so that it can easily be integrated in an industrial way of working.

III. Case Description

Our robustness analysis is instigated by a real-world application: an office lighting system being developed by Philips Lighting. This office lighting system provides a complete lighting solution in office spaces (cell office, open plan) and central spaces, such as corridors, lounges and entrance areas. The system is typically deployed in an office building and cooperates with other systems, such as HVAC (heating, ventilation, and air conditioning), security, network and power systems. The system comprises networked luminaires with LED-based light sources, a set of sensors (occupancy, luminosity, etc.), and a microcontroller. The number of luminaires in a building is in the order of thousands; typically half of them have a set of sensors.

The office worker, i.e., a lighting system user, has control over the lighting behavior of an area via a button panel (and via a mobile app, but this is not considered in this paper). A button panel allows the selection of a number of predefined lighting settings (relax, concentrate, presentation, etc.), called *presets*. Occupancy sensors provide automatic switching on/off behavior of the lights in an area. Another feature is daylight regulation, which uses the amount of available daylight for setting the light intensity to a constant level. Other features like linking rooms to corridors (keeping the corridor lights on as long as neighboring rooms are occupied, etc.) are not considered in this paper.

System control is distributed in the sense that each luminaire has a controller that exchanges control and synchronization messages with other luminaire controllers. This paper focuses on robustness issues to understand the impact of a change in the technology for message transport. In particular, we investigate the robustness issues when changing from an RS485-based system to an IP-based system, either using Ethernet or wireless. These issues are of extreme importance for Philips Lighting as the possible issues are usually immediately visible to the users. For example, a delayed response to a button press (poor responsiveness) and a single 'dark' luminaire in a 'lit' room (inconsistency) are very noticeable.

The technology transition leads to a more cost-effective solution as the building's existing network can be used. A second important improvement is the use of Power over Ethernet (PoE) technology that not only leads to discarding the power lines for each luminaire, but it also allows installation by less expensive personnel. Using the existing network, however, entails that the network has become part of the lighting system's environment. There are risks

 ISBN: 978-1-61208-399-5

involved in migrating from an internal to an external network. In particular, there is no control of the environment: a high network load may cause loss or delay of messages, and the start-up behavior of routers and switches may influence the lighting system behavior.

IV. LIGHTING SYSTEM MODEL

In this section, we present the models that we have used to analyze the robustness of indoor lighting systems. These models are based on the observation that there are a few basic events in a lighting system:

- An *occupancy event* is triggered by the detection of occupancy by a sensor in an area and causes a change of the area's preset.
- A *vacancy event* is triggered by the absence of area occupancy for a certain time period, called the *hold time*, and causes a change of the area's preset.
- A *button event* is triggered by the press of a button in an area and causes a change of the area's preset.

Using these events, one can configure a controller for an area. This configuration involves selecting the appropriate parameters for the events: the presets, the *dwell times* (a period without preset change), and the hold times.

This will be illustrated using a simple example, which includes all three events. Consider a workspace within an open office. When someone enters the workspace area, the lights switch on to a low light level (of 50%). If the workspace is occupied for more than a dwell time of 15 seconds, the light switches to a medium light level (of 75%). A workspace occupant can manually toggle the light level between medium (75%) and high (100%) by pressing a button on a panel. The light switches off (i.e., 0%) automatically if no occupancy has been detected for a hold time of 10 minutes.

The workspace example involves four presets with seven transitions between them. The presets and the transitions are visualized in Figure 1.

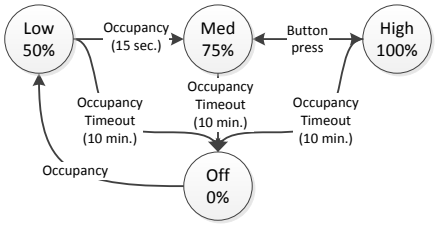

Figure 1. Workspace preset transitions.

Based on the parameterized events, we have created an Uppaal model that can be configured in the same manner as the events. The model consists of two main elements, a controller model and an environment model. These are described in the following subsections.

A. Controller Model

The controller model contains a parameterized timed automaton for each of the events described earlier. The values of the automata's parameters for the workspace example in Figure 1 are given between brackets.

- The *occupancy automaton* has four parameters: a controller id, an active preset (*Off*/*Low*), a dwell time (0/15 seconds), and a new preset (*Low*/*Med*).
- The *vacancy automaton* has four parameters: a controller id, an active preset (*Low*/*Med*/*High*), a hold time (10 minutes), and a new preset (*Off*).
- The *button automaton* has three parameters: a controller id, an active preset (*Med*/*High*), and a new preset (*High*/*Med*).

The occupancy automaton is shown in Figure 2. As explained earlier, the automaton has four parameters: a controller id (*lumId*), an active preset (*p1*), a dwell time (*tDwell*), and a new preset (*p2*). The Uppaal automaton consists of three *locations* (denoted with a circle symbol) connected by five *edges* (denoted with an arrow symbol). Edges are annotated with *selections* (e.g., *p: preset_t*), *guards* (e.g., *sensor[lumId]*), *synchronizations* (e.g., *PB[lumId][p]*), and *updates* (e.g., *t=0*) [10].

The automaton's initial location (indicated with a circle in the location symbol) is the *Off* state. If the corresponding luminaire has a sensor and its power supply gets enabled (channel *powerOn*), it changes to the *On* state. It then resets its internal clock *t*, which determines the time since the last preset change. If occupancy is detected (channel *occupancy*) after the dwell time *tDwell* has elapsed, it communicates a preset change via channel *PLocal* and resets clock *t*. Clock *t* is also reset if a preset change is reported via channel *PB*. If the luminaire loses power (channel *powerOff*), it changes back to the *Off* state.

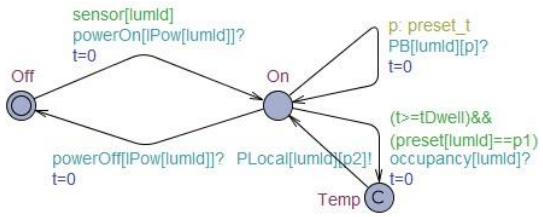

Figure 2. Occupancy automaton.

The event automata receive triggers from the system's environment (see Section IV-B) and update the internal states of the corresponding controllers accordingly. In our model, a controller's internal state includes the active preset and a number of clock values. The clock values are updated by the event automata; the model includes a separate automaton to update a controller's active preset.

We model a lighting system that is controlled in a distributed manner: one area may have several controllers. To avoid undesired behavior, e.g., a controller recalling preset *Off* while another controller has recently detected occupancy, the internal states of the controllers need to be synchronized. This is done by a synchronization event, for which we have created a fourth parameterized automaton. This synchronization automaton informs the other controllers in the same area of its internal state.

 ISBN: 978-1-61208-399-5

B. Environment Model

As our goal is to analyze lighting systems' robustness against events in their environment, we have created automata that model a lighting system's environment. In particular, we have created timed automata for occupancy generation, power supplies, and network communication.

We will focus on the network automata. These automata model loss and delay of messages in a generic manner, i.e., they are independent of the underlying network technology and protocols. The network automata decouple transmission and reception of messages. A network automaton for communicating presets is shown in Figure 3. The automaton uses two channels: an incoming channel *PA* and an outgoing channel *PB*. If the network automaton receives a message via channel *PA*, then it either forwards it via channel *PB* or discards it. The latter models message loss. Message delays have been modeled in a similar manner.

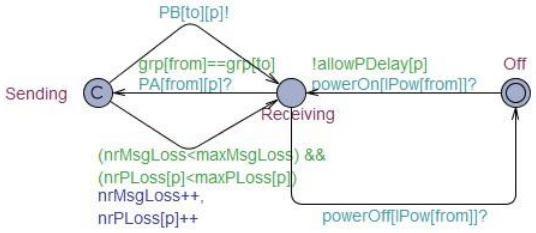

Figure 3. Network automaton.

To fully control loss and delay of messages, the model includes a network automaton for each combination of a source controller, a destination controller, and a preset. This allows a message to be received by any subset of the intended recipients, enabling the analysis of all possible scenarios involving message loss and message delay.

V. Robustness Analysis

As explained in Section III, Philips Lighting is considering porting a distributed control concept from an RS485 network that is part of the lighting system, to an IP network that is part of the lighting system's environment. Philips wants to know the influence of an external network on the behavior of their lighting systems.

The CAP theorem shows that a distributed system cannot simultaneously be Consistent, Available, and Partition tolerant [6]. Since high network loads may cause message loss (a form of partitioning), this practically means that inconsistencies between luminaires in one area cannot be avoided or that (part) of the system becomes unavailable, i.e., unresponsive.

This can easily be exemplified; consider the preset transitions of Figure 1 for an area with two luminaires, one of which has an occupancy sensor. Figure 4 shows a Gantt chart for a scenario in which this area ends up in an inconsistent state. The area starts in preset *Off* (black). The occupancy sensor of luminaire 1 detects occupancy (blue) and its controller recalls preset *Low* (cyan event). Fifteen seconds later, this sensor still detects occupancy and preset *Med* is recalled (magenta event). The corresponding message from luminaire 1 to luminaire 2 gets lost. This leads to an inconsistent situation: luminaire 1 is in preset *Med* (light gray) and luminaire 2 in preset *Low* (dark gray). This inconsistency ends when an occupant presses a button and preset *High* (white) is recalled (yellow event).

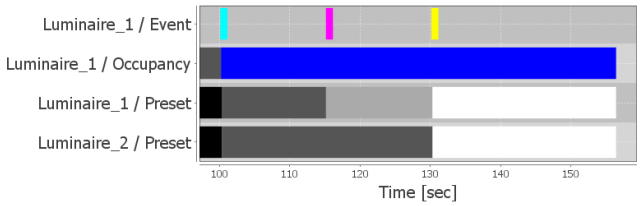

Figure 4. Inconsistency due to message loss.

Section IV has described the automata from which lighting system models can be constructed. The example shown in Figure 1 involves seven event automata per controller. Moreover, all luminaires have an automaton that maintains the active preset. On top of that, there are network automata for each combination of a source luminaire, a destination luminaire, and a preset. For the example area, this involves sixteen network automata.

Using the model elements described in Section IV, we have configured and analyzed a number of lighting systems including a cell office configuration that Philips has used in customer projects. Details regarding these configurations are omitted because of confidentiality.

Uppaal has a requirements specification language in which one can formulate system properties [10]. We have used this language to formulate desired lighting system properties. Combined with a consistency monitoring automaton, we have formulated properties regarding the occurrence and maximum duration of area inconsistencies. With these requirements, we have identified several scenarios in which loss or delay of messages causes an area to end up in an inconsistent situation. The identified message loss scenarios are similar to the one in Figure 4; the inconsistencies caused by message delays are mainly due to out-of-order message reception.

VI. Model Validation

As explained in Section V, we have used Uppaal models to identify robustness issues, e.g., scenarios in which an area becomes inconsistent. These formal models have been created from a variety of informal documents and observations of actual systems. To gain confidence in the correctness of the outcome of the robustness analysis, we have coupled our model to a test setup (see Figure 5). This test setup is a realistic, small-scale lighting system composed of commercial products: luminaire controllers, (simulated) sensors, and PoE switches. The simulated sensors allow injecting occupancy events into the system when needed. Message loss and message delays are controlled by an IP bridge. This bridge is realized by two network cards in the PC connecting two separate networks.

 ISBN: 978-1-61208-399-5

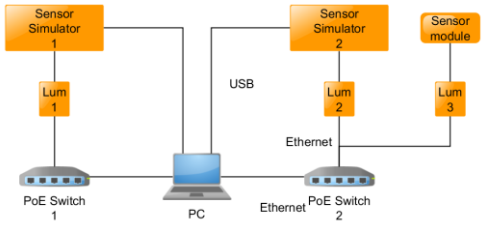

Figure 5. Lighting system test setup.

The Uppaal model and the test setup can both be configured with occupancy, vacancy, button, and synchronization events.

To validate our models, we have created a two-way transformation between the model and the test setup. This transformation is based on a domain-specific language (DSL) that we have created for lighting system scenarios. This DSL has been developed using Xtext technology [4] and the transformations using Xtend [3] and Java.

The test setup allows observing the luminaire controllers and the communication network over time and creates a record expressed in the DSL. A recorded scenario includes occupancy detections, active presets, and output levels. This scenario can be translated into an Uppaal automaton that replays the occupancy detections and checks whether the response of the model corresponds to the response observed in the test setup.

Reversely, the inconsistency scenarios identified using the Uppaal models need to be checked on the test setup. We have developed a transformation that transforms an inconsistency scenario into a script that can be replayed on the test setup to check whether the inconsistency actually occurs. In the same manner, we can check good-weather scenarios.

Using the described two-way transformation, we have successfully validated the Uppaal model. On the one hand, the responses of the model correspond to responses observed on the test setup. On the other hand, all of the inconsistency scenarios identified using our model have been successfully reproduced by filtering or delaying the appropriate network messages.

This gives confidence in the correctness of the model and therefore in a lighting system's behavior in case of loss or delay of messages. The actual probability of message loss and message delay is subject for further study, and involves measurements and estimation of loss rates, and other implementation-specific details.

VII. AUTOMATIC RECOVERY

In Section VI, we have validated that message loss and message delay may lead to periods of inconsistency in a lighting system. Message loss and message delay are rare in practice, but they cannot be avoided by the lighting system, because the communication network is not part of the lighting system. To be robust against message loss, a lighting system needs to cope with these inconsistencies by automatically recovering. This requires the introduction of a recovery strategy. In this section, we analyze such strategies.

We will focus our analysis on lighting systems' recovery capability. In particular, we determine the maximum duration of area inconsistency if a finite number of messages are lost in the communication network. In order to compute this maximum duration, we have extended the model with an area monitoring automaton that continuously checks for inconsistency and starts a clock when it detects area inconsistency.

We have compared three strategies for their recovery capability. All strategies involve repeated synchronization of a controller's internal state. For confidentiality reasons, we omit the details regarding the strategies.

- *Strategy* 1: No additional state synchronization.
- *Strategy* 2: The usage of additional occupancy and vacancy events for state synchronization.
- *Strategy* 3: The usage of new synchronization events for state synchronization.

Table I shows the maximum inconsistency duration for the different strategies, derived using simple argumentation. In this table, N denotes the number of lost messages, T_H equals the maximum hold time of the vacancy events, T_D is the maximum dwell time of the occupancy events, and T_R is the maximum state synchronization time.

TABLE I. MAXIMUM INCONSISTENCY PERIOD DURATION

Strategy	Maximum inconsistency
Strategy 1	$N \cdot \infty$
Strategy 2	$N \cdot (T_H + \max\{T_R, T_D\})$
Strategy 3	$N \cdot T_R$

The results in Table I apply to all system configurations that we have analyzed. These include the example in Figure 1 and configurations used by Philips Lighting. The results in Table I show that if no messages are lost, then the lighting systems behave consistently. It also shows that without additional synchronization, message loss can (in theory) lead to infinite periods of inconsistency. Finally, it shows that repeatedly synchronizing the internal state provides a successful manner to recover from area inconsistency; the speed of recovery depends on the strategy.

VIII. DISCUSSION AND FUTURE WORK

In the previous sections, we have presented formal models to analyze the robustness of distributed lighting systems. This section reflects on the modeling approach and its industrial applicability and describes future work.

A. Industrial Fit

Application of formal modeling and model checking techniques in industrial practice has always been a challenge, mainly due to the distance between the abstract level of reasoning for formal modeling and the concrete level of reasoning required in product development and realization.

In our work at Philips Lighting, we reduce this distance by matching our models to the actual design and implementation concepts and terminology used by the system architects and engineers. Furthermore, we hide the complexity of the model checking tooling by creating a

 ISBN: 978-1-61208-399-5

transformation of configuration information to Uppaal models. This entails the adaptation of our models to the high configurability of the actual system.

The basis of our approach is the configurability of lighting systems: these systems are constructed by selecting and configuring a few component types. In earlier work, we have identified that logistic systems, e.g., warehouses and baggage handling systems, share these characteristics [13]. Our approach can therefore easily be adapted to validate logistic system controllers.

Note that the approach will also have great value in the IoT domain, as there is strong dependency on the good- and bad-weather behavior of the (network) environment.

B. Model and Modeling Experience

The process of creating a formal model from design knowledge and concrete implementations is a challenging one: initially the knowledge has to be acquired from experts or design documentation (which may be hard to find); next the models are created iteratively when the knowledge and insight grows. The large number of features and configurations that are captured in the model quickly leads an unmanageable monolithic model. Therefore, we were forced to decompose the monolithic model into logical parts exhibiting the same behavior. We have created a modular model comprised of parameterized event automata, which fits the configuration tooling of Philips Lighting. This, however, has consequences for scalability. It is a challenge to keep the state space small when having many automata. This scalability problem in a realistic industrial context is a topic for further research.

C. Model Extensions

The current model of the Philips Lighting case captures the control behavior of single-area configurations. The model can be extended with hierarchy of areas (linking the behavior of a corridor to that of connected offices, etc.). We foresee additional scalability challenges in such cases.

Another modeling challenge involves the start-up behavior of system components (including switches, routers, etc.) and the corresponding lighting behavior. This is important in cases of (partial) power failures, or software update scenarios. The challenge lies in the acquisition of detailed knowledge about the component behavior in non-specified situations.

IX. Conclusion

In this paper, we have presented a model checking approach to analyze the robustness of distributed lighting systems. There is a huge variety of lighting systems that can be built from a small set of component types. Our modular and parameterized modeling approach addresses this variety, because it allows the same configuration options as lighting systems. We have shown that our approach is very helpful in identifying robustness issues and in analyzing robustness-improving communication protocols. In particular, we have shown that lighting systems can recover from inconsistent behavior by having their luminaires communicate their internal state periodically. These concrete results are highly appreciated by the development organization of Philips Lighting.

A second benefit of our modular and parameterized approach is its integration in an industrial way of working. Because the models have the same configuration options as the distributed lighting systems, formal models can be generated automatically from the lighting systems' configuration information. We have shown that it is possible to close the loop from formal model to real implementation by having a transformation that translates formal analysis results back into the lighting domain.

References

[1] B. Combemale, L. Gonnord, and V. Rusu: "A generic tool for tracing executions back to a DSML's operational semantics," in Modelling Foundations and Applications, R.B. France, J.M. Kuester, B. Bordbar and R.F. Paige, Eds. Berlin: Springer, pp. 35-51, 2011.

[2] S. Dziwok et al.: "A tool suite for the model-driven software engineering of cyber-physical systems," 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), ACM, November 2014, pp. 715-718, doi:10.1145/2635868.2661665.

[3] Eclipse Foundation: Xtend. [Online]. Available from: http://www.eclipse.org/xtend/ [retrieved: 2015.03.02].

[4] Eclipse Foundation: Xtext. [Online]. Available from: http://www.eclipse.org/Xtext/ [retrieved: 2015.03.02].

[5] C. Gerking: "Transparent Uppaal-based verification of MechatronicUML models," Master's thesis, University of Paderborn, May 2013.

[6] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," ACM SIGACT News, vol. 33, pp. 51-59, June 2002, doi:10.1145/564585.564601.

[7] S.D. Gribble: "Robustness in complex systems," Eighth Workshop on Hot Topics in Operating Systems, IEEE, May 2001, pp. 21-26, doi:10.1109/HOTOS.2001.990056.

[8] M. Hendriks, B. van den Nieuwelaar, and F. Vaandrager: "Model checker aided design of a controller for a wafer scanner," International Journal on Software Tools for Technology Transfer, vol. 8, June 2006, pp. 633-647, doi:10.1007/s10009-006-0025-7.

[9] F. Kammüller and S. Preibusch: "An industrial application of symbolic model checking - The TWIN elevator case study," Informatik - Forschung und Entwicklung, vol. 22, February 2008, pp. 95-108, doi: 10.1007/s00450-007-0032-2.

[10] Uppsala Universitet and Aalborg University: UPPAAL. [Online]. Available from: http://www.uppaal.org/ [retrieved: 2015.03.02].

[11] F. van den Berg, A. Remke, and B.R. Haverkort: "A DSL for performance evaluation of medical imaging systems," Medical Cyber Physical Systems Workshop 2014, April 2014, pp. 80-93, doi: 10.4230/OASIcs.MCPS.2014.80.

[12] L. van den Berg, P. Strooper, and K. Winter: "Introducing time in an industrial application of model-checking," in Formal Methods for Industrial Critical Systems, S. Leue and P. Merino, Eds. Berlin: Springer pp. 56-67, 2008.

[13] J. Verriet, H.L. Liang, R. Hamberg, and B. van Wijngaarden: "Model-driven development of logistic systems using domain-specific tooling," in Complex Systems Design & Management, M. Aiguier, Y. Caseau, D. Krob, and A. Rauzy, Eds. Berlin: Springer, pp. 165-176, 2013.