Model-based Safety Assessment using OCL and Petri Nets

Ricardo J. Rodríguez, Elena Gómez-Martínez
Babel Group, DLSIIS, Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Madrid, Spain
Email: {rjrodriguez, egomez}@babel.ls.fi.upm.es

***Abstract*—Safety becomes a primordial assessment in safety-related systems where human lives can be somehow put in risk, needing to comply with safety requirements defined by industry standards such as IEC 61508, ISO 26262 or DO-178C. Safety contracts are useful to specify these requirements (as assumptions and guarantees), thus assuring an expected level of confidence. To verify the safety requirements is measured to represent more than a half of the overall system development costs. In this paper, we propose a model-based verification that addresses safety verification from the early beginning of system development, thus saving costs. Namely, we use UML for system design and Object Constraint Language (OCL) for specifying safety contracts, while its verification is carried out using Petri nets. As case study, we assess the safety of an embedded system that models a fire prevention system in a hospital building.**

I. Introduction

Verification of safety properties during system development usually induces overruns in the system development budget. For instance, verification has been quantified as representing more than a half of the overall costs in the avionics software domain [1]. A safety assessment becomes a primordial task when dealing with safety-related systems, such as industrial equipment, road vehicles or avionics. In these domains, safety requirements are usually specified by industrial standards, such as IEC 61508 [2], ISO 26262 [3] or DO-178C [4].

A (software) contract is commonly used to specify the relation between system artifacts (or components), expressing the pre- and post-conditions of a system component [5]. A safety contract is a similar idea but instead of having pre(post)-conditions, it contains assumptions and guarantees assuring a certain level of confidence (integrity) of such a component [6]. Thus, safety contracts can be used to specify safety standard requirements in the design phase, as it is stated in [6]–[8].

Bringing safety contracts specification in design phase enables its early verification, thus detecting potential problems in earlier development phases, and saving overruns normally induced by later verification [1]. Formal models are a good framework to verify and validate safety requirements [9]. However, formal specification is usually ignored by engineers [10] that rely their confidence on graphical descriptions instead of formal specification. In this sense, the Unified Modelling Language (UML) [11], standard de-facto as modelling language, provides an unified understanding and insight in system and software design. Safety specification in UML has been explored in the research community using the Object Constraint Language (OCL) [12] (a standard UML extension to specify constraints in the UML models), such as the works in [5], [7], [8], [13]. Other works [14], [15] use specific UML profiles (a UML extension in terms of *stereotypes* and *tags*), such as SysML and OMEGA, to express safety or correctness contracts in a UML model system. These contracts are usually verified by model-checking techniques, such as ATL [13], [16], or Timed Input/Output Automata [14]. Safety specification and analysis has also been explored using model-based approaches such as AADL [17], [18].

In this paper, we mix both approaches of OCL and UML profiles. Namely, we explore the idea of specifying the safety contracts (assumptions, guarantees) by means of OCL in UML models enriched with the MARTE [19] profile. This profile is used to specify the performance system information. We consider the UML Sequence Diagrams (UML-SD) and UML State-Machine diagrams (UML-SM) to model the dynamic part of the system, while the UML Class Diagram (UML-CD) for the static one. Then, these contracts are analysed using Petri nets [20]. As Lutz established in [21], the integration between informal and formal methods can enhance safety analysis. We assess the safety in a high safety-critical system (namely, a fire prevention system in a hospital building) throughout the paper to explain our approach.

The outline of this paper is as follows. Section II gives some background. Section III is devoted to the specification of safety contracts in OCL and its verification using Petri nets. Finally, Section IV concludes the paper.

II. Background

UML is a semi-formal modelling language commonly used for systems and software specification that can be tailored for specific domains by profiling [22]. A UML profile defines an extension of UML, in terms of *stereotypes* (concepts in the target domain) and *tagged values* (the attributes of the stereotypes). For instance, the UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE [19]) provides an analysis framework called Quantitative Analysis Model (GQAM) that enables performance specification in UML models. In this paper, we use the `gaStep` stereotype (and the `hostDemand` tagged value) of MARTE to indicate the duration of activities in a UML model.

However, a UML design annotated with MARTE is not suitable for a performance evaluation or model-checking. In this sense, some model transformation is needed. In this work, we use the approach in [23] to obtain a formal model (namely, a Generalized Stochastic Petri Net (GSPN) [24]) suitable for quantitative and qualitative evaluation.

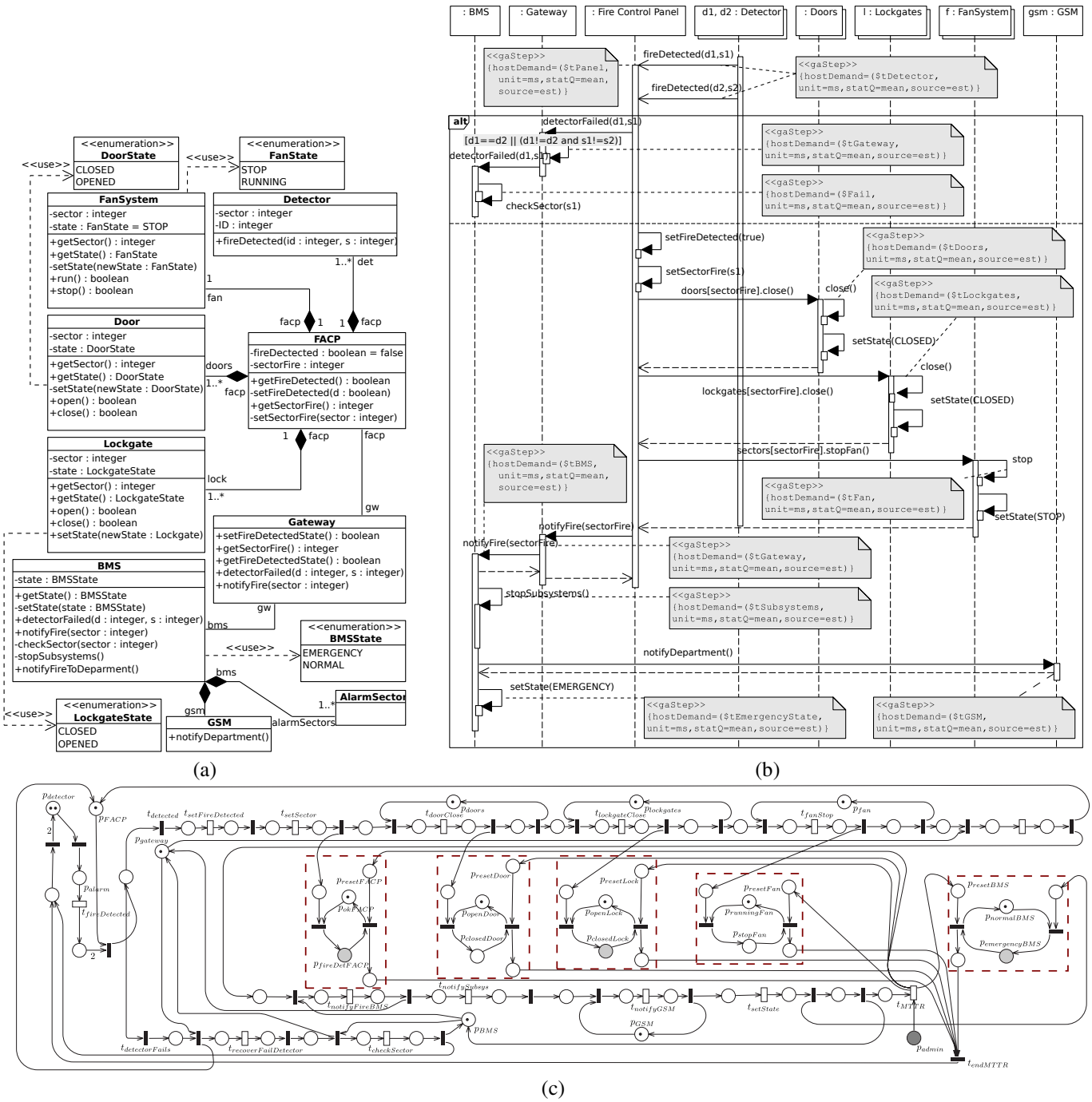

Figure 1. (a) UML Class diagram, (b) UML Sequence Diagram of the example and (c) its GSPN representation.

A GSPN is a graphical and mathematical formalism used for the modelling of concurrent and distributed systems. For the sake of space, we refer the reader to [24] for a gentle introduction to GSPN. Once a GSPN is obtained from an annotated UML model, it can be used for performance analysis and for qualitative analysis. In this paper, we use the GreatSPN [25] tool to compute the probability of (eventually) reaching a safe state represented as a place being (eventually) marked. This model enables also performance evaluation, and we aim at exploring it as future work.

III. Safety Contracts Specification and Verification

A. Safety Contracts Specification

Recall that safety contracts define safety constraints (or assumptions) and guarantees in the context of a system artifact, guaranteeing its functional properties and a certain level of integrity [6]. In this work, we explore the idea of specifying safety constraints and guarantees (forming a safety contract fragment, SCF [6]) of artifact components using OCL invariants within UML models. In this paper, we restrict the logic of SCF assumptions to AND and OR.

Let $\mathcal{S} = \langle \mathcal{C}, \mathcal{G} \rangle$ be a SCF [6], where $\mathcal{C} = \mathcal{C}^+ \bigcup \mathcal{C}^*$ is the superset of disjoint sets $\mathcal{C}^+, \mathcal{C}^*$ of OR and AND safety constraints, respectively, and $\mathcal{G}$ is a guarantee. Recall that OCL is an extension of UML to express constraints into UML designs. Therefore, the question we raise here is: *is there any easy way to translate a SCF, usually expressed in text form, to an OCL constraint in the context of a UML model?*. In this paper, we propose to express a SCF of a system with information extracted from the UML Class Diagram (UML-CD) of the system, thus making easier the construction of OCLs.

Running Example. Hospital buildings require some of the most demanding facilities with a very strict building regulation. A *Building Management System* (BMS) controls most of these facilities, such as air conditioner, lights and elevators, among others. Figure 1(a) depicts a BMS of a hospital supervising and coordinating the interaction between several subsystems. In addition, a BMS coordinates the interaction between these subsystems and checks their functionality. The most safety-critical subsystem is the fire alarm system, since human lives depend on it. The fire alarm system is located at the *Fire Alarm Control Panel* (FACP), which communicates with BMS via a Gateway. A FACP is in charge of fire detection of a building area that are divided in sectors. Each sector is composed of a set of environmental detectors, fire doors, lockgates and ventilation system fans.

Consider now the following safety contracts in the context of the running example: 1.- *When a fire is positively detected in some sector, eventually the system reaches emergency state*; and 2.- *When a fire is detected in a sector s, the lock gates of s are eventually closed*.

Attending to the UML-CD in Figure 1(a), these safety contracts can be respectively defined as:

$$\mathcal{S}_1 = \langle \{FACP.getFireDetected()\}, \{BMS.getState() = EMERGENCY\} \rangle$$
$$\mathcal{S}_2 = \langle \{FACP.getFireDetected() \wedge Lockgate.getSector() = Lockgate.sector.fcp.getSectorFire()\}, \{Lockgate.getState() = CLOSED\} \rangle$$

Let us describe now how these safety contracts can be expressed in terms of OCLs. Note that a SCF $\mathcal{S} = \langle \mathcal{C}, \mathcal{G} \rangle$ will be defined in system level (i.e., it interrelates several classes), but OCL is defined in a concrete class. Thus, we need to find the more suitable context for $\mathcal{S}$ when expressed in OCL. Let us assume that $\mathcal{G}$ always contains an equation relating some class attribute that is private, only accessible through setters and getters class methods, being compared to some value. Under these assumptions, the setters class methods become the more suitable context. Thus, a SCF $\mathcal{S} = \langle \mathcal{C}, \mathcal{G} \rangle$ will be transformed to an OCL invariant in such a context. Following the running example, $\mathcal{S}_1, \mathcal{S}_2$ can be therefore mapped to the OCL rules listed in Code 1. As it can be seen, the transformation rule from a SCF to a OCL is intuitive.

OCL invariants are useful to state conditions that must always be met by all instances of a context type, as the SCF does. This transformation step can be easily automated by navigating through the UML-CD. However, OCL does not take into account dynamic or timing behaviour of UML models. This drawback has been overcome in the literature with an extension of OCL [26]. Here, we propose to use

Code 1. OCL of safety contract $\mathcal{S}_1$ (above) and $\mathcal{S}_2$ (below).

```
context BMS::setState(state: BMSState): void
   inv: gw.facp.getFireDetected() implies state = EMERGENCY

context Lockgate::setState(newState: LockgateState): void
   inv: gw.facp.getFireDetected() and
        self.getSector() = self.sect.fcp.getSectorFire()
        implies newState = CLOSED
```

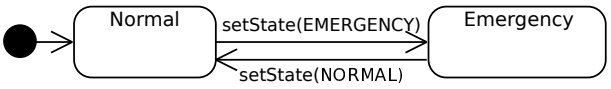

Figure 2. UML State-Machine of Building Management System (BMS).

MARTE annotations for the timing behaviour of UML models. An annotated UML model can be then transformed to Petri nets to verify and validate the OCL constraints defined in the dynamic UML models. The automation of the whole model transformation process and the model-checking of OCL invariants, as it is described in the sequel, are currently our ongoing work.

B. Safety Contracts Verification

We use a subset of the UML behavioural models, namely the UML Sequence Diagram (UML-SD) and the UML State-Machine diagram (UML-SM), to express the dynamics of the system. The UML-SD of the running example is depicted in Figure 1(b). The FACP needs to receive events from two different fire detectors distributed in the same sector to raise a fire alarm, changing its status to *FireDetected* and keeping note of the suspected sector. Otherwise, the FACP notifies the failure of the detector(s) to the BMS. Then, the FACP closes firewall doors, air lockgates and halts the air fan system, changing their corresponding states. Finally, the FACP notifies to the BMS that a fire has been detected through the gateway. Then, the BMS stops all the associated subsystems and sends a GSM alarm to the Fire Department. Lastly, the BMS updates its status to *Emergency* state until an administrator resets the entire system. The transition between *Emergency* and *Normal* state is depicted by the UML-SM in Figure 2. The rest of UML-SM of state changes (firewall doors, air lockgates and air fan system) are similar to this one. Recall that the OCL of the safety contract $\mathcal{S}_1$ (in Code 1) is associated to `BMS::setState`.

The UML-SD and the UML-SM (red-dash boxes) of Figures 1(b) and 2 can be translated to a combined GSPN, depicted in Figure 1(c), as proposed in [23]. We have added a place, p_{admin} (dark grey), which models the deactivation of emergency state, usually performed by some building administrator. We can now use the GreatSPN tool [25] to validate the model. Recall that we are interested in model-checking the OCL constraints defined in Code 1. In terms of Petri nets, they are equivalent to compute the probability of the following conditions in the GSPN: ($\mathcal{S}_1$) the places $p_{fireDetFACP}$ and $p_{emergencyBMS}$ are marked; and ($\mathcal{S}_2$) the places $p_{fireDetFACP}$ and $p_{closedLock}$ are marked (we assume that the second safety constraint of the $\mathcal{S}_2$ is always fulfilled). These places have been depicted in light grey in Figure 1(c). The computation of these conditions with GreatSPN shows that they are fulfilled.

IV. Conclusions

An early verification of safety during a safety-critical system design helps to detect potential problems that contradict the safety requirements. A safety requirement can be specified with a safety contract, which defines the assumptions and guarantees of an artifact, assuring its level of confidence. Safety contracts are usually expressed in informal ways, such as descriptive text. In this paper, we propose to express them using a predefined syntax to make a transformation to Object Constraint Language (OCL) rules easier. These rules can be added to the UML system diagrams, thus embedding all safety-related information in a single picture. Moreover, these UML diagrams can also incorporate profile annotations (e.g. MARTE profile) to enrich their expressiveness. For instance, to indicate the duration of activities, probabilities of execution paths or number of resources in the model.

In this paper, we propose to transform a UML model, with MARTE annotations and OCL-based safety contracts, to a formal model, namely Generalised Stochastic Petri Nets (GSPN), thus enabling a qualitative and quantitative analysis during design phase. This early detection can help the engineers during system design to check the fulfilment of safety requirements, thus assuring unsafe states are not reached in the system. Let us remark that final effort must be focused on assuring that system implementation corresponds with the system model, because otherwise implementation may lead the system to an unsafe state.

We believe the integration of informal and different formal models can help in system safety assessment during design phase, and it deserves further research as it was already stated in [21]. As future work, we aim at formalising the model transformation of our approach, to automatise the model transformation steps and to perform an extensive evaluation of our approach. We also plan to explore the transformation of OCL-based safety contracts to Othello [27]. Othello enables a Linear-time Temporal Logic checking, then providing a different system analysis.

Acknowledgements

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n^o 295373 (project nSafeCer) and from National funding. Part of this work was done while Dr. Rodríguez was a visiting researcher at Mälardalen University, Västerås (Sweden).

References

[1] F. Randimbivololona, "Orientations in Verification Engineering of Avionics Software," in *Informatics*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2001, vol. 2000, pp. 131–137.

[2] *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems*, International Electrotechnical Comission Std., 2010.

[3] *ISO 26262: Road vehicles – Functional safety*, International Organization for Standardization Std., 2011.

[4] *RTCA DO-178C. Software Considerations in Airborne Systems and Equipment Certification*, Radio Technical Commission for Aeronautics Std., 2012.

[5] T. Gezgin, R. Weber, and M. Girod, "A Refinement Checking Technique for Contract–Based Architecture Designs," in *Proocedings of the MoDELS 2011 Workshop – Model Based Architecting and Construction of Embedded Systems (ACES-MB)*, 10 2011.

[6] A. Söderberg and R. Johansson, "Safety Contract Based Design of Software Components," in *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2013, pp. 365–370.

[7] I. Bate, R. Hawkins, and J. McDermid, "A Contract-based Approach to Designing Safe Systems," in *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software - Volume 33*, ser. SCS '03. Australian Computer Society, Inc., 2003, pp. 25–36.

[8] R. Hawkins, I. Toyn, and I. Bate, "An Approach to Designing Safety Critical Systems using the Unified Modelling Language," in *Critical Systems Development with UML - Proceedings of the UML'03 workshop*, 2003, pp. 3–17.

[9] J. Rushby, "Formal Methods and their Role in the Certification of Critical Systems," in *Safety and Reliability of Software Based Systems*, R. Shaw, Ed. Springer London, 1997, pp. 1–42.

[10] W.-T. Tsai, R. Mojdehbakhsh, and S. Rayadurgam, "Experience in Capturing Requirements for Safety-Critical Medical Devices in an Industrial Environment," in *Proceedings of the 2nd High-Assurance Systems Engineering Workshop (HASE)*. IEEE Computer Society, 1997, pp. 32–36.

[11] OMG, *Unified Modelling Language: Superstructure*, Object Management Group, August 2011, version 2.4, formal/11-08-05.

[12] ——, *Object Constraint Language (OCL)*, Object Management Group, February 2010, v2.2, formal/2010-02-01.

[13] E. Cariou, C. Ballagny, A. Feugas, and F. Barbier, "Contracts for Model Execution Verification," in *Modelling Foundations and Applications*, ser. LNCS. Springer, 2011, vol. 6698, pp. 3–18.

[14] I. Dragomir, I. Ober, and C. Percebois, "Integrating verifiable Assume/Guarantee contracts in UML/SysML," in *Proceedings of the 6th International Workshop on Model Based Architecting and Construction of Embedded Systems (ACESMB)*, 2013.

[15] ——, "Safety Contracts for Timed Reactive Components in SysML," in *SOFSEM 2014: Theory and Practice of Computer Science*, ser. Lecture Notes in Computer Science. Springer, 2014, vol. 8327, pp. 211–222.

[16] J. Bézivin and F. Jouault, "Using ATL for Checking Models," *Electronic Notes in Theoretical Computer Science*, vol. 152, no. 0, pp. 69–81, 2006.

[17] P. Feiler, "Model-Based Validation of Safety-Critical Embedded Systems," in *IEEE Aerospace Conference (AERO)*, March 2010, pp. 1–10.

[18] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, and M. Roveri, "Safety, Dependability and Performance Analysis of Extended AADL Models," *The Computer Journal*, vol. 54, no. 5, pp. 754–775, 2011.

[19] OMG, *A UML profile for Modeling and Analysis of Real Time Embedded Systems (MARTE)*, Object Management Group, 2011, document formal/11-06-02.

[20] T. Murata, "Petri Nets: Properties, Analysis and Applications," in *Proceedings of the IEEE*, vol. 77, no. 4, April 1989, pp. 541–580.

[21] R. R. Lutz, "Software Engineering for Safety: A Roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00. New York, NY, USA: ACM, 2000, pp. 213–226.

[22] B. Selic, "A Systematic Approach to Domain-Specific Language Design Using UML," in *10th IEEE Int. Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. Santorini Island, Greece: IEEE Computer Society, May 2007, pp. 2–9.

[23] S. Bernardi and J. Merseguer, "Performance evaluation of UML design with Stochastic Well-formed Nets," *Journal of Systems and Software*, vol. 80, no. 11, pp. 1843–1865, November 2007.

[24] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, ser. Wiley Series in Parallel Computing. John Wiley and Sons, 1995.

[25] S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis, "The GreatSPN tool: recent enhancements," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 4–9, 2009.

[26] M. V. Cengarle and A. Knapp, "Towards OCL/RT," in *FME 2002: Formal Methods – Getting IT Right*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2391, pp. 390–409.

[27] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "Validation of Requirements for Hybrid Systems: A Formal Approach," *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 4, pp. 22:1–22:34, February 2013.