MoDeS3: Model-based Demonstrator for Smart and Safe Cyber-Physical Systems

András Vörös[1,2], Márton Búr[1,4], István Ráth[2,3], Ákos Horváth[2,3], Zoltán Micskei[2], László Balogh[2], Bálint Hegyi[2], Benedek Horváth[2], Zsolt Mázló[2,3], and Dániel Varró[1,2,4]

[1] MTA-BME Lendület Cyber-Physical Systems Research Group, Budapest, Hungary
[2] Department of Measurement and Information Systems
Budapest University of Technology and Economics, Budapest, Hungary
[3] IncQuery Labs Ltd., Budapest, Hungary
[4] Department of Electrical and Computer Engineering
McGill University, Montreal, Canada
{vori,bur,ahorvath,micskei,varro}@mit.bme.hu, rath@incquerylabs.com

Abstract. We present MoDeS3, a complex research demonstrator illustrating the combined use of model-driven development, formal verification, safety engineering and IoT technologies for smart and safe cyber-physical systems. MoDeS3 represents a smart transportation system-of-systems composed of a model railway and a crane which may automatically load and unload cargo from trains where both subsystems need to fulfill functional and safety requirements. The demonstrator is built by using the model-based software engineering principle, while the system level safety is ensured by the combined use of design-time and runtime verification and validation techniques.

Keywords: smart cyber-physical systems, model-driven engineering, formal methods, education, demonstrator

1 Introduction

Motivation. A smart and safe cyber-physical system (CPS) autonomously perceives its operational context and adapts to changes over an open, heterogeneous and distributed platform with a massive number of nodes, dynamically acquires available resources and aggregates services to make real-time decisions, and resiliently provides critical services in a trustworthy way [9, 12].

These challenges and the multidisciplinary nature of CPS make the engineering of such systems very complex. On the one hand, traditional techniques used for developing safety-critical systems may have limited applicability for CPS [8]. Moreover, both research and education of CPSs necessitate well-documented open-source demonstrator platforms which capture and reflect the essence of problems and challenges, yet it is reasonably complex to highlight the key characteristics of CPSs and present them in the context of modern technologies.

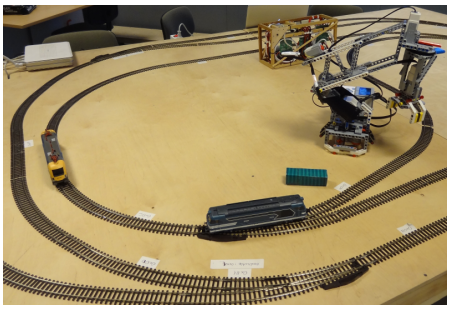

Fig. 1: Physical layout

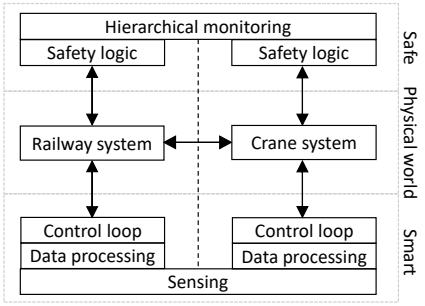

Fig. 2: Architectural overview

Objectives. We introduce *MoDeS3: the Model-based Demonstrator for Smart and Safe Cyber-Physical Systems*[5], which aims to illustrate the combined use of model-driven development, intelligent data processing, safety engineering and IoT technologies in the context of safety-critical system of systems with emerging safety hazards. This open source project simultaneously serves as (1) a *research platform used for experimental evaluation* of CPS-related research, (2) a complex *educational platform* used for graduate and undergraduate teaching, and (3) an *IoT technology demonstrator* used by industrial partners and collaborators.

The MoDeS3 demonstrator as a smart and safe CPS. The physical layout of MoDeS3 is depicted in Figure 1. As its core is a *model railway transportation system*, guarantees for the safe operation of trains, switches, and semaphores are required. Connected to a specific segment of the track, an automated *crane system* loads cargo on and off the trains. As such, it is a critical system in itself since the cargo cannot be dropped by the crane.

Additionally, the MoDeS3 demonstrator represents a system-of-systems, since the railway and the crane system are physically located next to each other. In this case, new kind of hazardous situations may emerge which are not incorporated in any of the constituent systems. For instance, a rotating movement of the crane may physically hit a train passing by along the track.

To make the demonstrator more realistic, we adopted various safety assurance techniques ranging across design-time formal verification and validation (V&V), runtime monitoring or testing on various levels of abstraction (see Section 2). A conceptual overview is provided in Figure 2. Multiple levels of safety are applied: a distributed safety logic is responsible for the accident-free operation of the trains. Hierarchical monitoring is used to ensure the safe cooperation of the subsystems. The details are given in Section 2. A wide range of sensors serves as a rich information source for smart control and data analytics (see details in Section 3). Educational use of MoDeS3 is covered in Section 4. The project timeline and conclusions are drawn in Section 5.

[5] http://modes3.inf.mit.bme.hu/

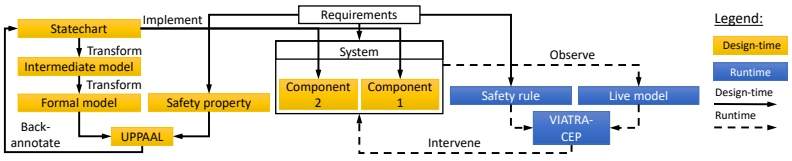

Fig. 3: Overview of design-time and runtime verification in MoDeS3

2 Design- and runtime assurance

The development of safety-critical systems has a long history with well-established methodologies to ensure safe operation. The MoDeS3 demonstrator was built using Model-based Systems Engineering (MBSE) where models are first-class citizens of the engineering process. SysML models are used to define the functional and the platform architecture of the system, while the Gamma Statechart Composition Framework[6] is used for the precise definition of the component level behaviour. Gamma supports the design, verification and code generation for component-based reactive systems.

The MoDeS3 demonstrator incorporates various V&V approaches (such as model checking, structural completeness and consistency analysis) as well as fault-tolerance techniques — all of which are widely used in real systems. However, due to its complex and multidisciplinary nature, design-time assurance cannot guarantee in itself the safe operation of inherently dynamic smart CPSs. Therefore, runtime certification [13] using techniques like runtime monitoring [10] or runtime verification [7] complement design-time assurance. Therefore, MoDeS3 integrates runtime monitoring and verification techniques on both component and system-level to flag violations of safety properties during the operation of the system and trigger appropriate counter-measures such as immediately stopping or slowing down trains. Our emphasis is on the combined use of design-time and runtime V&V techniques when building MoDeS3 to address its safety requirements. A high-level overview of V&V techniques is illustrated in Figure 3.

2.1 Design-time formal V&V of timing properties

As a primary design-time verification task, we carried out a formal analysis of logical and timing properties of the distributed safety logic of the accident prevention subsystem. We used the Gamma Statechart Composition Framework [11] to form the composite behavior of Yakindu statechart models. This composite model serves as the engineering input for the design-time analysis. Gamma introduces an intermediate state machine language with some high-level constructs and precisely defined semantics [14] to serve as a bridge between engineering and formal models. This intermediate language also helps in the back-annotation of analysis results to statechart models. Formal verification is performed using UPPAAL model checker [2], which is widely used for analyzing timing properties.

[6] http://gamma.inf.mit.bme.hu/

The generated formal models address the verification of a single component against local properties as well as their interaction against global properties. However, these models are insufficient to reason about the correctness of the system in themselves. For that purpose, one needs to ensure the interaction between the physical world and the cyber world.

For this purpose, formal models are built to capture the (logical and physical) behavior of trains. Then a combined design-time verification can reveal potentially unsafe situations, e.g. if trains move too fast, some accidents cannot be prevented. Investigating the counterexample retrieved by Gamma highlights that the situation could only happen if the trains are faster than the messages transmitted between the components. Unless there is a denial-of-service attack with flooding of messages, this is hardly the case in practice, but it is still a potential security threat. After extending the statechart models with timing assumptions on communication speed, we can formally prove that the safety logic prevents multiple trains from entering the same section of the track.

2.2 System-level runtime monitoring

As smart and safe CPSs have complex interactions with an evolving environment and the physical world, we complement design-time verification in MoDeS3 with runtime monitoring techniques on both component and system level. For space considerations, here we only provide a summary of the *hierarchical system-level runtime monitoring* technique using graph reasoning with live models and complex event processing techniques (see right part of Figure 3).

As traditional monitoring techniques consume events but do not cover data-dependent behavior or structural properties, runtime knowledge about the operational system is captured by a runtime (live) model [4]. A runtime model captures the current abstract snapshot of the system and its operational context, and changes in the underlying running system are constantly incorporated. Unlike a detailed design model, a runtime model only captures those aspects of the system, which are relevant for runtime monitoring and intervention.

System-level safety monitoring is carried out using graph queries and complex event processing (CEP) [5], which detect runtime violations of safety rules (by the identification of changes in the match sets of graph queries) and trigger appropriate reactions. While graph models and queries are widely used in *design tools* of CPS and CEP is a key technique in stream processing for web applications, their use in the context of smart and safe CPS is an innovative aspect of the MoDeS3 demonstrator.

Graph-based runtime techniques nicely complement traditional, component-level, automaton-based monitors deployed to embedded computers since critical signals raised by low-level monitors can be further propagated to the system-level as a hierarchy of events. As a consequence, we obtain a technique for the runtime monitoring of system-of-systems [15] where emerging and ad hoc hazardous situations can be incorporated and detected automatically also in the presence of complex structural (graph) constraints.

3 Smart IoT technologies

Intelligent services and technologies are integrated into MoDeS3 at various levels. First, distributed autonomous intelligent control is used both for driving the trains and also to load and unload cargo on trains by the robot crane. Moreover, multiple sensors and surveillance cameras are used, and initial processing of the data stream is carried out close to the information source in accordance with fog and edge computing [6,9] principles. Such sensor data can be consumed by multiple data processing services and different subsystems by offering generalized sensing services. This way, reusable smart sensing services may initiate actuation and control according to the collected environmental and operational information. The software stack is based on open-source Eclipse IoT solutions.

System-level runtime verification exploits events obtained from track sensors and general-purpose surveillance cameras. The visual information is processed using state-of-the-art computer vision (OpenCV) and neural network (TensorFlow) technologies. Distributed components are using state-of-the-art IoT communication protocols with open connectivity to share sensor data with different data processing services (and different subsystems). MQTT[7] provides a lightweight protocol for exchanging messages in a publish/subscribe model, which is widely used in communication between embedded devices and sensors.

Open-source microcontrollers (Arduino) and industrial embedded computers (Raspberry Pi, BeagleBone Black) provide the hardware elements of the platform. Cloud computing technologies are used for integrating hardware devices, service APIs and real-time data analytics.

4 MoDeS3 in education

One of the goals of MoDeS3 is to support education with realistic examples and case studies. The demonstrator currently fulfills this purpose at the Budapest University of Technology and Economics at various stages of education.

Undergraduate level. At the first year introductory *System Modeling* course, the demonstrator is used for illustration purposes: students are introduced to modeling by the simplified models of the platform. Third year undergraduate students of the *Systems Engineering* course face the problem of designing the railway system by going through the development process. All phases of the development process result in a model which is then evaluated by the instructors. Undergraduate students choose thesis project after completing the Systems Engineering course which may include developments of the MoDeS3 platform itself.

Graduate level. At the master's level, three courses actively use the demonstrator platform. The course on *Model-Driven Software Development* introduces domain specific languages and development of model transformations for the students. The *Cyber-Physical Systems* course integrates the knowledge from the previous courses and introduces the modeling and controlling of hybrid systems.

[7] http://mqtt.org/

Beside the theoretical foundations, practical skills for integrating IoT technologies and cloud computing is also part of the curriculum. CPS course also covers fault-tolerance and other extra-functional aspects of cloud-based CPS. *Software and Systems Verification* is a course for further enhancing the knowledge of the students on testing with a specific focus on model-based testing or hardware-in-the-loop and model-in-the-loop testing. The course also summarizes runtime verification with a special focus on the hierarchical composition of the verification tasks according to the specification. At this part of the course, the advanced verification approaches are illustrated to the students on the MoDeS3 platform.

5 Project timeline and conclusion

Since its inception in 2014, the project has been proceeding by major milestones which have been organized along public demonstrations and presentations. At each milestone, some new features have been introduced, and critical maintenance tasks have been completed. These milestones are illustrated in Fig. 4 together with the new features.

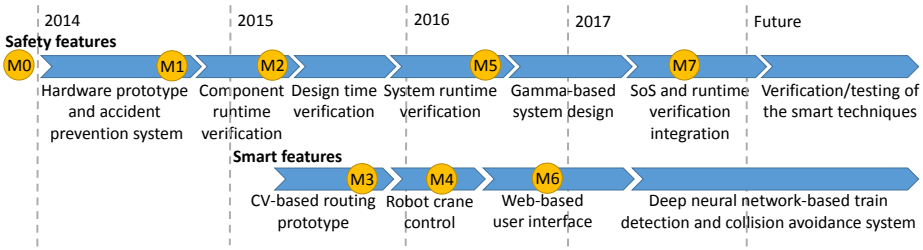

Fig. 4: Project timeline and milestones. **M0**: Project kickoff, **M1**: Researchers' Night 2014, **M2**: Ericsson University Day 2015, **M3**: Researchers' Night 2015, **M4**: 2016 Eclipse IoT Challenge and Ericsson University Day 2016, **M5**: Researchers' Night 2016 and EclipseCon France 2016, **M6**: EclipseCon Europe 2016 Demo, **M7**: EclipseCon Europe 2017 Demo

MoDeS3 demonstrates the innovative use of model-driven engineering approaches, formal methods and intelligent technologies for smart CPS. MoDeS3 proved its innovation at many industrial events: the team won a third prize at the Eclipse Open IoT Challenge 2.0 and MoDeS3 was exhibited twice at the industrial EclipseCon Europe conference and another workshop [1].

As a future work, we plan to further extend the demonstrator with smart technologies, such as a neural network based collision avoidance system and intelligent data analysis. Smart techniques used in for accident prevention have to be extensively tested/verified, where we will exploit the recent advances of the field. In addition, a novel distributed graph-based monitoring approach [3] will be integrated to provide an additional level of safety.

Acknowledgment

MoDeS3 is a joint effort of many participants. It was partially supported by MTA-BME Lendület Research Group on Cyber-Physical Systems the ARTEMIS JU R5-COP project and the NSERC RGPIN-04573-16 project. MoDeS3 also received financial and technical support from our industrial partners: IncQuery Labs Ltd., Quanopt Ltd., Ericsson Hungary and Miniversum. The TITAN Xp used for this research was donated by the NVIDIA Corporation. Colleagues at Dept. of Measurement and Information Systems (BME) worked on the project beside the authors: István Majzik, Gábor Szárnyas, and Oszkár Semeráth. We also thank the hard work of our students: Flórán Deé, Márton Elekes, Anna Gujgiczer, Bence Graics, Raimund Konnerth, Gergő Somos, and Sámuel Várallyay.

References

1. Balogh, L., et al.: Distributed and Heterogeneous Event-based Monitoring in Smart Cyber-Physical Systems. MT CPS workshop (CPS Week 2016)
2. Behrmann, G., et al.: UPPAAL 4.0. In: Third International Conference on the Quantitative Evaluation of Systems. pp. 125–126. IEEE (2006)
3. Búr, M., et al.: Distributed graph queries for runtime monitoring of cyber-physical systems. In: International Conference on Fundamental Approaches to Software Engineering (2018), Accepted
4. Cheng, B.H., et al.: Using models at runtime to address assurance for self-adaptive systems. In: Models@run.time: Foundations, Applications, and Roadmaps (2014)
5. Dávid, I., Ráth, I., Varró, D.: Foundations for streaming model transformations by complex event processing. Software & Systems Modeling pp. 1–28 (2016)
6. Dubey, A., et al.: Resilience at the edge in cyber-physical systems. In: FMEC. pp. 139–146 (May 2017)
7. Havelund, K.: Rule-based runtime verification revisited. STTT 17(2) (2015)
8. Lee, E.A.: Cyber physical systems: Design challenges. 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing pp. 363–369
9. Lee, E.A., et al.: The Swarm at the Edge of the Cloud. IEEE Design & Test 31(3)
10. Medhat, R., et al.: Runtime monitoring of cyber-physical systems under timing and memory constraints. ACM T. Embed. Comput. S. 14(4), 1–29 (2015)
11. Molnár, V., et al.: The Gamma Statechart Composition Framework. ICSE 2018: Demonstrations (2018), Accepted
12. Nielsen, C.B., et al.: Systems of systems engineering: Basic concepts, model-based techniques, and research directions. ACM Comput. Surv. 48(2), 18 (2015)
13. Rushby, J.: Runtime certification. In: RV. pp. 21–35. Springer (2008)
14. Tóth, T., et al.: Verification of a Real-Time Safety-Critical Protocol Using a Modelling Language with Formal Data and Behaviour Semantics, pp. 207–218 (2014)
15. Vierhauser, M., et al.: Reminds: A flexible runtime monitoring framework for systems of systems. Journal of Systems and Software 112, 123–136 (2016)