Formal methods for residual risk reduction in cyber-physical systems

David Kaufmann[1], Radu Mateescu[2], Lucie Muller[2], Wendelin Serwe[2], Franz Wotawa[3,*]
[1]AVL List GmbH, Graz, Austria
[2] Univ. Grenoble Alpes, INRIA, CNRS Grenoble INP, LIG, Grenoble, France
[3] Institute of Software Engineering and Artificial Intelligence, Graz University of Technology, Graz, Austria
david.kaufmann@avl.com, radu.mateescu@inria.fr, lucie.muller@inria.fr, wendelin.serwe@inria.fr, wotawa@tugraz.at
*corresponding author

Abstract—Assuring quality for cyber-physical systems has been a significant concern, leading to various proposed solutions. Faults in cyber-physical systems lead to security and safety issues in communication and operation, respectively. To prevent harm, verification and validation methodologies are applied during development. However, there might be no guarantee that the final deployed system is fault-free, i.e., a residual risk always remains. This paper focuses on involved risks, identifies their sources, and discusses methods for risk reduction in cyber-physical systems. For this purpose, a holistic approach to risk reduction in cyber-physical systems is utilized. Further, different stages of system development and operation are explained, and methodologies for finding defects and evaluating risks are discussed. Finally, concepts and methods using an industrial battery management system are presented. Specifically, the benefits of using formal methods to reduce risks in the context of autonomous driving and ADAS functionality are illustrated.

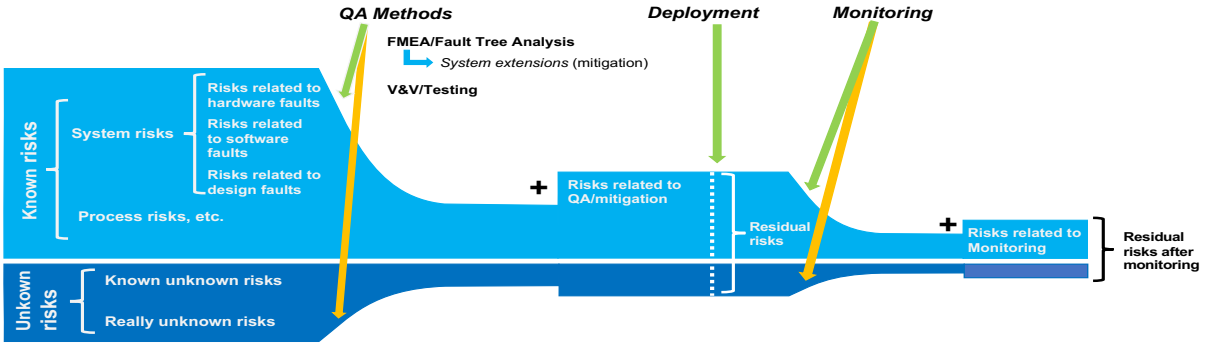
Keywords—management system, system monitoring, cyber-physical systems, verification and validation, formal methods, residual risks

1. INTRODUCTION

Cyber-physical systems (CPSs) are engineered systems whose operations are monitored, coordinated, controlled, and integrated by a computing and communication core (see [1]). Like every engineered system, a CPS interacts with its surrounding environment. This interaction is ideally based on sensor and user input and directs actuators to take particular values that impact the environment. Faults in such systems affect the behavior and might have severe consequences, and measures are required to ensure quality. For example, let us consider an autonomous emergency braking function (AEB) that takes action to prevent crashes with other cars or pedestrians in front. An AEB is a CPS because it monitors the environment and the internal state of the vehicle and computes a decision, i.e., braking or not braking, in case there is a likelihood of a crash with an obstacle ahead. Verifying and validating the correct implementation of an AEB is essential to ensure the proper functionality. Faults in the implementation may cause two issues, namely, the AEB stopping the car in cases where this is not required or not stopping in instances where it is needed. The former may degrade people's comfort inside the vehicle, whereas the latter is more severe, potentially harming people. Hence, there is a need for verification and validation methodologies that identify such critical faults before such systems are deployed (see, e.g., automated testing [2]).

Unfortunately, all verification and validation methodologies have assumptions and limitations, making fault detection during development impossible. Reasons are failing to obtain all requirements, the sheer search space for formal verification and test case generation that can hardly be traversed completely, not considering unexpected input values, or assuming the impact on a particular but incomplete set of hardware faults when testing the reaction of software, among others. In summary, finding only some faults during development is due to computational limitations and incomplete knowledge and assumptions. Hence, to avoid issues still in a CPS after verification and validation, other means for tracking the behavior of a CPS during operation are required. Smart monitoring (see [3], [4], and [5]) to observe the behavior of a CPS and to check its consistency with given regulations, requirements, and other constraints at all system levels might be a solution allowing not only to find issues and failures but also to predict consequences and to provide appropriate mitigation actions. For example, overheating the batteries of an electrified vehicle might be acceptable to some extent, allowing it to complete a mission, i.e., reaching the planned destination. Such an informed decision can only be taken considering all influencing factors, like the remaining distance to drive and conservative predictions of safe battery operation. However, in any case, whether limiting oneself to verification and validation during development or monitoring after deployment, achieving completely risk-free use of engineered systems is hardly possible. For this purpose, standards like IEC 61508-1 [6] define boundaries for the probabilities of dangerous failure per hour, e.g., ≥ 10-9 to < 10-8 for high demand or continuous mode systems and safety integrity level (SIL) of 4. Hence, estimating failure probabilities and corresponding risks during development is required. Handling and mitigating risks during the operation of a CPS is an option that further reduces risks. However, it must be considered that introducing monitoring and diagnosis functionality may also introduce additional risks (see, for example, [7]).

Residual risk also includes unknown unsafe behavior that requires an appropriate detection method. This method is essential for applying general mitigation actions to increase safety and reliability in unforeseen scenarios. To address this, formal methods based on abstracted system models may be implemented for analyzing system behavior and



:

Figure 1. The origins of risks during development and operation.

finding uncertainties in logical correctness but may struggle with unforeseen scenarios not represented in the model. Another method is related to AI-based anomaly detection [8], which may be valuable to detect a deviation from expected normal behavior. However, AI-based approaches heavily rely on time-series data with a high coverage of the operational domain. Although the ability to adapt to uncertainties in the behavior of the system is beneficial, the lack of mathematical verifiability could also introduce new risks, such as false positives.

In this paper, a contribution is made to discuss how to reduce risks in CPSs. Particularly, and in contrast to previous work, the utilization of formal methods [9] for risk reduction is outlined. The case of employing formal methods during development as well as during operation is considered. To demonstrate the impact of formal methods, their use in a power management system for electrified vehicles is explained. It demonstrates how the specifications and requirements can be formalized and used to detect issues during development and operation. Formal methods rely on an abstract model of a system and its requirements to be analyzed. This model can be used for checking formal properties, such as that the system always reacts to input (i.e., liveness) or that a particular behavior is present, e.g., that a light turns on after a switch is closed. The model can also be used to obtain test cases to check the conformance of concrete implementations to the model. Last but not least, the formal model can also be used to monitor the formal requirements and check the behavior during runtime, allowing us to identify failures and detect faults after the deployment of a CPS. It is also worth mentioning that the methodology is general, and can be applied to any CPS: it is only necessary to formally model the behavior of the CPS and write the glue code to connect the model with the physical implementation or a simulator of the CPS.

The paper is organized as follows: First, the discussion begins with an exploration of risks occurring in CPSs, focusing on residual risks, i.e., risks that remain even when countermeasures are introduced. Afterward, an overview of the foundations behind formal methods is briefly presented, including referencing essential articles. Furthermore, the application of formal methods is demonstrated using the battery management system use case. Finally, the obtained findings and future research challenges are summarized.

2. Risk Reduction for Cyber-Physical Systems

Many standards use risks to deal with the degree of safety-criticality of systems, like IEC 61508-1 [6], ISO 26262 [10], or ISO/PAS 21448 [11] for automotive systems. From a high-level point of view, risk is always defined as a function or product of the probability of a harmful event and its corresponding costs. In ISO 26262, the degree of controllability after the event happens is considered, and the risks are computed utilizing a qualitative degree of the event probability, the corresponding costs, and controllability leading to an assignment of automotive safety integrity levels (ASILs) to given systems and their subsystems. Regarding the ISO 26262 standard, ASIL A characterizes the least safety-critical system and ASIL D the most. It is also worth noting that ASIL can be mapped to other similar safety integrity levels like SIL in the case of the IEC 61508-1 standard.

Regardless of the concrete definition of risk, the risks of a system represent the degree of harm that occurs as a consequence of a fault during operation, considering its likelihood. In this paper, risk is defined as the product of the probability of the fault (or, in general, an event) and the costs of its corresponding effects. For a system, there are usually several risks that sum up to the total risk that needs to be coped with. This section discusses the origins of risks during development and operation under certain assumptions.

Figure 1 summarizes the origins of risks, not necessarily claiming completeness. For all systems, there are known and unknown risks. The former are typically considered during development and operation, where activities like testing, mitigation, or monitoring are introduced to decrease the likelihood of severe issues. The latter are risks that might be completely unknown or (partially) known, like shortcomings of processes dealing with risks. An example of such a shortcoming is a limitation of the failure mode and effect analysis (FMEA) method (see, e.g., [12]), where only single failures are typically considered. Another example is the limitation of testing where the focus is on reaching Modified-Condition/Decision Coverage (MC/DC), but only considering some potential test cases. Completely unknown risks (or unknown unknown risks) are all remaining risks that cannot be addressed due to limited knowledge.

In any case, known and unknown risks, if not handled appropriately, may lead to an event that causes harm or other unwanted effects. Hence, quality assurance (QA) methodologies like testing or FMEA must be introduced to reduce risks. It is worth noting that such methods themselves may cause additional risks. For example, a mitigation action like triggering a warning for a driver during the operation of a car might cause the driver to temporarily divert their attention from the road to deal with the warning, which may finally lead to an accident that would not occur without raising an alert. As another example, there might be a mitigation action that requires introducing new hardware, which may fail, too. There might be consequences like stopping the car immediately, which may lead to a disaster. It is worth noting that QA also affects unknown risks. All activities for risk reduction that occur during development lead to a final residual risk of a system.

Other techniques may be introduced to reduce the residual risks further. During operation, monitoring of system behavior and state may reduce risks. However, again, this activity presents a higher risk than expected. Hence, awareness of ensuring risk reduction is necessary when using monitoring.

After discussing the origins of risks during development and operation, focus is now directed towards methods and techniques to keep the risks within predefined boundaries, which is especially important for safety-critical systems like cars.

1. Standards like ISO 26262 already introduced measures for reducing risks during development. These measures include static testing, e.g., code analysis, and dynamic testing, i.e., executing the system and evaluating its behavior according to the given requirements. Ammann and Offutt [13] provide an introduction to testing, and the work is used as a reference for further explanation. Code coverage metrics, like statement coverage or MC/DC, are suggested to evaluate the testing quality. All these measures relate to a system's expected ASIL or parts. For example, in the case of ASIL D, MC/DC has to be used.

   To apply testing and, in particular, the most appropriate testing methodology that balances the testing effort and its effectiveness for detecting faults to reduce the risks, it is necessary to characterize the safety integrity level of the CPS. For this, methods like FMEA are used in practice. There, the underlying idea is to assume the components' failure modes and estimate the impact of the corresponding fault on the CPS. This includes assessing the probability of the fault and the related costs in case the fault manifests. After FMEA, countermeasures or failure mitigation are introduced to avoid potential effects. This includes adding hardware and software to detect the fault and take mitigating action, like stopping the car or raising a warning or error message. When testing the CPS, it is required to also ensure that the implemented mitigation is tested. This requires considering tests where faults are injected, and the system’s behavior is observed.

   Despite applying FMEA, implementing mitigation actions, and conducting testing, there are still residual risks that remain. First, the FMEA might not be complete, not considering all combinations of failure modes, which would be intractable. Although such cases might be very unlikely, there might be adverse effects on safety. Second, testing can hardly be complete, too. Even when MC/DC coverage is used to determine required test cases, faults might be undetected. Finally, the countermeasures themselves might cause trouble when they fail during operations because, for instance, the countermeasures reach their end of life.
2. To further reduce risks, the problem of the incompleteness of testing must be tackled. A well-known method is formal verification, where the behavior of (an abstraction of) the system under tests is formally proven correct. Formal methods provide a basis for this verification. What is required is a model of the system, which usually is an abstraction of the real system and formalized requirements, i.e., a formal specification. Methods like the one outlined in this paper can be used to demonstrate the fulfillment of requirements and other properties such as liveness.

   Although the use of formal methods reduces risks further, new risks may be added. First, the system model used for verification may not capture the system's behavior with sufficient accuracy. This risk can be minimized by testing the model itself. Secondly, only some requirements may be considered. And finally, the underlying formal method may itself be faulty, delivering only sometimes the correct results. However, obviously, after applying formal methods, the overall remaining risks are lower because the mentioned residual risks are less likely to occur.
3. The final measure of risk reduction is to observe the system's behavior during operation. Although many issues can be detected during development, there is no guarantee that all faults are detected. In addition, there are other effects, like the aging of components, that finally lead to a system breakdown. Hence, it is necessary to monitor lifetime constraints and also compare the current behavior with the expected one. Formal methods provide a perfect basis for the latter. Further, requirements and properties can be used to check the observed behavior. In addition, the abstract model can be utilized to predict and evaluate the current behavior. In case of deviations, countermeasures need to be implemented, like a minimum risk maneuver in the case of autonomous driving.

   Again, there are remaining risks to consider. The monitoring itself is part of the system and may fail unexpectedly. Hence, monitoring also needs to be tested and, ideally, formally verified. Challenges may also arise when facing the incompleteness of the abstract model and requirements so that faults cannot be detected. That could cause a wrongly evaluated behavior, leading to unnecessary mitigation actions, which potentially lead to harm in the particular situation. For a more detailed analysis of the effects when

introducing monitoring and diagnosis, interested readers are referred to [7].

This proposed three-step process for risk reduction, based on formal methods, can be applied to all CPSs. Under reasonable assumptions, i.e., that the probability of failure of a method used for risk reduction like monitoring is lower than the probability of failure of the ordinary system, the discussed approach should reduce risks sufficiently. However, as discussed, there is always a residual risk that cannot be avoided. It is also worth noting that formal methods can also be used for risk estimation and verification, e.g., see [14].

In the following sections, the use of formal methods for risk reduction is illustrated using a high-voltage battery system as a use case. In particular, FMEA is used to show how risks can be obtained during the development process. Furthermore, the application of formal methods is used to show that a system fulfills its requirements, e.g., handles critical faults identified using the FMEA in a good way. Afterward, a discussion follows on how the same formal methods can identify problems even during operation and further reduce the remaining risks.

3. Foundations of Formal Methods

Complex and critical systems, such as the communicating Electronic Control Units (ECUs) embedded in automated vehicles, are subject to strong safety requirements. Malfunctioning these systems may have severe consequences in terms of human lives and economic damage. Therefore, such systems must be designed and built according to rigorous methodologies that involve, as much as possible, formal methods.

The general term formal methods [9] encompasses the languages (with mathematically defined semantics) used to model the behavior and the expected properties of systems, as well as the analysis tools (simulation, verification, testing, etc.) supporting these languages. Formal methods can be inserted into a design process in order to bring additional guarantees about the quality of (a critical part of) the system under design. The fact that formal languages have precise, unambiguous semantics guarantees accurate answers about the validity of desired properties. Moreover, formal verification techniques, such as model checking [15], provide precise diagnostics (counterexamples and witnesses) explaining the validity of properties on a model.

A convenient and general way of representing the behavior of a complex system is provided by the achievements of concurrency theory [16]. The behavior of a sequential component interacting with its environment is modeled as an automaton, or Labeled Transition System (LTS), representing all reachable states of the system. State changes are represented by transitions labeled with events (or actions) that may carry data values. The model of a system consisting of several components that execute concurrently and communicate is obtained by the parallel composition of their individual LTSs synchronized appropriately. In concurrency theory, LTSs and their composition are described as terms in process calculi, such as CCS [17], CSP [18], ACP [19], and their variants.

In this work, the LNT language [20] to model the behavior of concurrent systems is used. LNT is a high-level modeling language with a syntax close to mainstream programming languages and mathematical semantics stemming from process calculi. In a nutshell, LNT provides constructed types and functions (built from usual instructions and pattern-matching) to operate on data, parallel processes that communicate by multiway rendezvous, and modules that enable the management of large system models. LNT is the main modeling language of the CADP toolbox [21], which provides a rich set of functionalities assisting the design process: compilation and rapid prototyping, interactive and guided simulation, model checking and equivalence checking, conformance test case generation, and performance evaluation. CADP is based on explicit-state verification, which consists of enumerating the states and transitions of the LTS of a concurrent system modeled in LNT. LTSs can be represented either explicitly (in a binary, compressed file format equipped with read and write operations) or implicitly (as C programs implementing an API for exploring the states and transitions). To deal with large models, CADP provides several techniques that can be combined together: on-the-fly verification (on-demand exploration of an implicit LTS starting from its initial state), compositional verification (incremental construction of the LTS of a concurrent system by hierarchical composition and minimization of the LTS of its components), and distributed verification (construction of a distributed LTS using several machines connected by a network).

The properties (or requirements) describing the desired good functioning of a system are specified formally as temporal logic properties in MCL (Model Checking Language) (see [22] and [23]) of CADP. MCL is an extension of the alternation-free mu-calculus with data values, programming language constructs, regular expressions on transition sequences, and fairness operators similar to generalized Büchi automata [24]. MCL is supported in CADP by the EVALUATOR model checker, which evaluates temporal logic formulas on LTSs on the fly and generates diagnostics (fragments of the LTS) explaining the verification results.

The formal methods methodology supported by CADP consists of several steps. Starting from a (possibly informal) system specification, an LNT model of the behavior is built, and the requirements are expressed as MCL formulas. Next, the LNT model is analyzed by step-by-step interactive simulation with backtracking, then by LTS generation (direct or compositional) and visual inspection whenever possible. This is useful, especially in the early model building, to gain confidence in the proper behavior of the LTS. Then, the MCL formulas expressing the requirements are checked on the LTS; when all requirements are satisfied by the LTS, the LNT model can be considered as a golden reference for further analyzing a physical implementation of the system, for instance by co-simulation [25] or conformance test case generation ([26]; [27]).

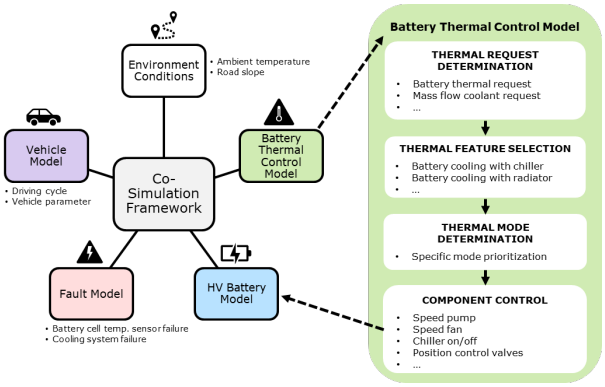


Figure 2. The co-simulation framework enables the simulation of complex systems comprised of subsystems and models. In the case of a BMS, an environment model, a vehicle model, an HV battery model, and a thermal control unit are joined. In addition, a fault model is integrated, enabling the injection of faults during runtime.

When applying formal methods to increase the rigor of a design process, it is necessary to abstract away from irrelevant details and model only the system's essential features, concentrating on the critical parts in which errors are more likely to occur (typically, complex interactions between concurrent components). In practice, obtaining a good abstraction level for a formal model requires a good knowledge of both the system and the capabilities of verification tools, such as capturing the essential aspects and keeping the model size tractable.

4. The Battery Management System Use Case

This section discusses the provided use case, the high-voltage (HV) battery management system (BMS). The discussion starts with the system, its requirements, and its failure analysis. Afterward, the application of formal models to find and detect faults during development is demonstrated. Finally, the use of formal models during operation and the relationship to risk reduction, as introduced in Section 2, is discussed in all parts.

4.1. The Battery Management Systems

For the time being, the state-of-the-art energy storage for modern electrified vehicles is dominated by the lithium battery system, ensuring an efficient and reliable operation (see [28]). The reasons for this are, of course, the high energy density, fast charging capabilities, and long cycle life. However, what must not be neglected are drawbacks like potential safety critical issues, e.g., thermal runaway, overheating, or total discharge, to mention only some in the context of battery damage. The appearance of such failures not only limits the vehicle's operation but could also cause harm to the overall system environment or even endanger lives. To minimize the risk of such cases, a BMS is applied to monitor and control the health status of the battery by tracking internal resistance, capacity degradation, cell temperatures, and voltage measurements, among others. Further, a thermal management, or more specifically, a thermal control model, is part of the BMS to monitor and manage the batteries' temperature to ensure safe operation by initiating mitigation actions to prevent overheating or even thermal runaway.

The BMS is a truly CPS comprising sensors for obtaining measures for cell temperature, voltages, and other physical quantities, actuators like heating and cooling, and a control component that manages to keep the status of the batteries within a pre-defined range, among other tasks. In the following, a BMS use case is described, focusing on the thermal control unit of a battery system, which will be used to highlight possible risks and, in this context, develop risk reduction methods. For validation and verification of the introduced risk reduction methods, AVL's Model.Connect™ [29] software is utilized to create a co-simulation framework (see Figure 2), which comprises a vehicle model, an HV battery model, a separate thermal control unit, and an environment model in order to execute various realistic driving scenarios and simulate different behaviors of BMS-related components under predefined conditions. Further, the framework enables the connection of additional models as a monitoring device, a formal model to compute a system diagnosis, or even a fault model, which enables fault injection during runtime into the powertrain elements such as the BMS's subsystems as the battery pack or coolant system. The simulation of the BMS provides the capability to initiate various state requests. These requests include battery cell temperatures, specification of coolant inlet temperatures and flow rate, and the execution of direct commands such as cooling, heating, or battery system homogenization. The other simulated powertrain elements are the sensors, which deliver continuous information about the battery cell temperature, coolant temperature, pressure, or flow rate. The developed fault model makes use of the provided information and enables the modification of received commands or observations, which then may lead to unexpected or faulty behavior of the BMS.

Before complementing the co-simulation framework with a formal model for diagnosis of the HV battery thermal control model health state, a prerequisite step is the detailed investigation of the thermal controller's specifications. Therefore, a specification document on the HV battery thermal control model is used, providing statements, indicators, thresholds, signal relations, state transitions, limitations, and boundary parameters. Figure 3 shows an

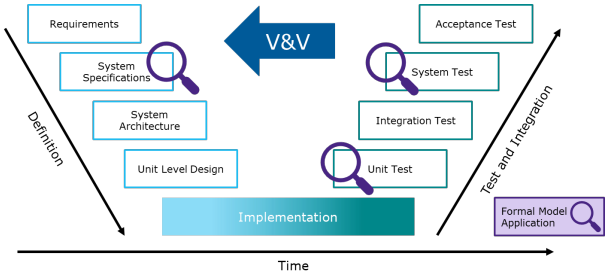


Figure 4. V-model of system/component design and implementation showing the corresponding phase application of the formal model to detect and identify failures due to specification violations.

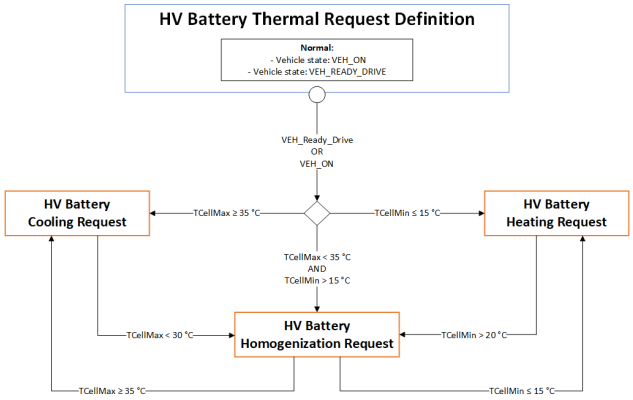


Figure 3. HV battery thermal request definition required for control actuators in the thermal system. The function specifies the battery needs (coolant mass flow, coolant inlet temperature, heating/cooling/homogenization) based on the actual cell temperatures.

extraction of the document for the thermal request definition sub-module, the decision process, and the related battery cell temperature boundaries for the corresponding thermal request.

Considering the complete specification document for the thermal control unit, a formal model can be built to be applied to different component life phases for the identification of failures and condition violations. The corresponding phases are shown in Figure 4, highlighted in the V-model, and detailed as follows:

- **System specification phase**
  - The use of formal modeling reduces risks because of a more complete capture and discussion of requirements, models, and their concrete interpretation.
  - For instance, during the formal modeling of the thermal controller, it became clear that the original specification mixed temperature values in Kelvin and Celsius and did not specify what happens if both conditions "TCellMax ≥ 35°C" and "TCellMin ≤ 15°C" hold (this latter situation was modeled as a non-deterministic choice between a Cooling and a Heating request). In addition, formal modeling also enables the provision of models for monitoring devices, allowing them to test and at least partially formally verify their intended functionality.
- **Unit test phase**
  - Specification monitoring during implementation to ensure a correct representation of the definitions.
- **System test phase**
  - Property monitoring during runtime of the observed signals and states and specific relations to ensure a safe and reliable operation.
  - For instance, the BMS thermal control unit uses an active (controlled coolant system, e.g., chiller/radiator) and passive (ambient temperature and airflow interaction with the vehicle frame) coolant system. In a scenario where the ambient temperature sensor fails, and the vehicle is operated in a very demanding driving scenario within a high ambient temperature beyond the HV battery system's design domain, the coolant system may fail to adequately lower the HV battery temperature as expected. A property check could be performed on the temporal relation between the applied coolant system performance and the battery cell temperature. In case the battery cell temperature is not decreasing or can be held at a certain level, although cooling is activated, a property violation will be reported, and correlated mitigation actions will be initiated.

Table 1. FMEA for HV battery-related thermal condition faults.

Id	Component	Function	Fault	Consequences	Severity Level	Explanation
1	Battery	Thermal conditioning	Battery too warm	No electric driving	8	Coolant system performance too low Cell overtemperature
				Vehicle immobility	8	
				Release of toxic gases	10	
				Thermal event	10	
2	Battery	Thermal conditioning	Inhomogeneous temperature spread inside the battery (hot spots)	Vehicle immobility	8	Cell temperature spread is too high Misplacement of temperature sensor
				No electric driving	8	
				Reliability not achieved	7	
				Durability / lifetime target not achievable	8	
3	Battery	Thermal conditioning	Battery too cold during driving	Vehicle range not achieved	8	Coolant system performance too low Coolant system pressure drop too high
				Driving discomfort	7	
5	Battery	Thermal conditioning	Component overheating	Vehicle immobility	8	Improper filling Coolant system pressure drop too high
				Durability / lifetime target not achievable	8	

In the following section, the approach towards the system test phase is presented, and a use case based on the correlated FMEA table for the BMS is described. This use case provides deeper insights into the associated risks when a certain failure is active during operation. To identify the related risks for specific fault modes in the HV battery system, the corresponding levels in Table 1 are used. These levels are established on AVL's expert knowledge. The table shows a selection of the battery's thermal conditioning functional faults, which are relevant to the specific HV battery example.

Each fault, e.g., battery too warm, is further divided into possible consequences, where each has its own rating in terms of severity level.

These corresponding severity levels are linked to Table 3, where the severity category and level are explained with the effect on the product as well as on the user experience. The severity levels are scaled from 1, no effect, to 10, indicating a severe hazard with safety and regulatory compliance violations. In general, high risks starting from level 9 require immediate stopping of the operation due to a hazardous scenario caused by severe component faults. The last column in the FMEA table gives an explanation

Table 2. HV battery detailed analysis for an overheating scenario and potential impacts, causes, and consequences.

ID 1 Fault: Battery too warm	
Direct component impact	Too high discharging power requested from the battery at a resulting cell temperature.
Component and system interference	No heat transfer between high voltage battery (HVB) cell and cooling circuit leading to a high cell temperature. Mechanical failures of the cooling package in combination of increased cooling power request, due to possible low vehicle speed and consequently low air mass flow rate used for passive cooling. HVB cooling system design issues, resulting in low coolant flow in certain channel spots.
Consequences to the system and components	Derated electrical power output of the HVB to keep the cell temperature on a certain specified level. Immediate vehicle stop and power shut down to avoid reaching the HVB cell limits. Thermal runaway of the HVB.
Thermal control unit	The thermal control unit raises flags in case of a fault to indicate a detection in the components. Error handling and mitigation is solved at system vehicle level.

referencing a potential cause of the failure. For the BMS use case simulations, the focus is on the "battery too warm" failure and actively injected faults during simulation runtime in order to observe a BMS component behavior, which leads to unexpected results or even to observations operating outside the component's specification. Thus, Table 2 is introduced, which provides a detailed analysis of the "battery too warm" faults by considering the direct impact on the component, the components and system interface, the consequences to the component or system, and the mitigation actions triggered by the thermal control unit in case the failure is detected as expected.

With the aim of testing the introduced formal method, the customizable co-simulation framework is used to apply configurations, which potentially lead to a demanding operation of the BMS. Thus, a WLTC (Worldwide Harmonized Light Vehicles Test Cycle) driving cycle is selected, which operates the vehicle within a speed range from 0 km/h up to 130 km/h. In addition, a very high ambient temperature of +40 °C and a continuous road inclination of +5% are configured to simulate a stressful driving scenario for the propulsion system. Furthermore, the fault model is activated for a targeted intervention in the system to trigger unexpected behavior in the HV battery system and the correlated thermal control unit. For the use case, the focus is set on four scenarios: the first serves as a healthy system reference, and three cases with fault injection during runtime. Figure 5 illustrates the simulated behavior of all four cases, which are discussed in the following in detail.

- The *Reference* case outlines the correct behavior of the BMS under a given specific operation. In this case, the BMS tries to keep the temperature of the batteries within a pre-defined range.

Table 3. Description of the severity category and effects on the product and user experience.

Severity Category	**Effect on Product**	**Severity Category**	**Effect for User**	**Severity Level**
Safety and/or regulatory compliance	Potential failure mode affects safety vehicle operation and/or involves noncompliance with regulations/standards	Severe hazardous	Safety risk, failure to comply with legal requirements, breakdown. May endanger user without warning.	10
Safety and/or regulatory compliance	Potential failure mode affects safe vehicle operation and/or involves noncompliance with regulations/standards	Hazardous	Safety risk, failure to comply with legal requirements, breakdown. May endanger user with warning.	9
Loss of primary function	Loss of primary function	High	Functionality strongly restricted, immediate maintenance at the workshop mandatory, function restriction of important subsystems. High degree of customer dissatisfaction due to the nature of the failure, such as inoperable item or system.	8
Reduction of primary function	Degradation of primary function	High	Functionality strongly restricted, immediate maintenance at the workshop mandatory, function restriction of important subsystems. High degree of customer dissatisfaction due to the nature of the failure, such as inoperable item or system.	7
Loss of secondary function	Loss of secondary function	Moderate	Product showed damages which requires actions. Functionality of vehicle restricted, immediate maintenance at the workshop is not absolutely necessary. Function restriction of important comfort systems.	6
Reduction of secondary function	Degradation of secondary function	Moderate	Product show abnormalities, actions recommended. Functionality of vehicle restricted, immediate maintenance at the workshop is not absolutely necessary. Function restriction of important comfort systems.	5
Annoyance	Appearance or audible noise, vehicle operable, item does not conform. Defect noticed by > 75%	Moderate	Product show single weak points. Functionality of vehicle restricted, immediate maintenance at the workshop is not absolutely necessary. Function restriction of important comfort systems.	4
Annoyance	Appearance or audible noise, vehicle operable, item does not conform. Defect noticed by 50%	low	Low functional impairment of the vehicle. Limitation of operation and comfort systems.	3
Annoyance	Appearance or audible noise, vehicle operable, item does not conform. Defect noticed by 50%	low	Low functional impairment of the vehicle. Limitation of operation and comfort systems.	2
No effect	No discernible effect	No effect	Customer will probably not notice the failure.	1

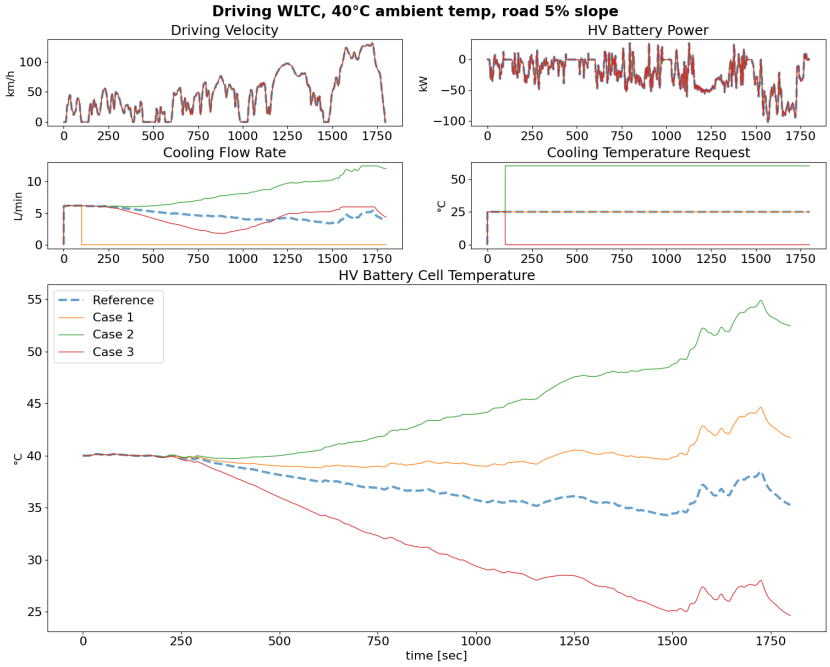


Figure 5. HV battery simulation test for a WLTC driving cycle and fault injection during runtime.

- The first fault case, *Case 1,* represents a thermal control system failure, where the coolant flow rate drops to null due to a mechanical or electrical fault of the active coolant system component (e.g., fan). This leads to a slowly increasing HV battery cell temperature over time.
- The second case, *Case 2,* simulates a scenario where the thermal control unit requests a too-high coolant temperature of 60 °C due to a sensor failure (e.g., sensor misplacement) in the battery cell temperature measurement. The battery heats up very fast, leading to a stressful and demanding situation for the HV battery system.
- The last fault case, *Case 3,* shows a scenario where the coolant temperature is set to 0 °C due to a temperature sensor fault, e.g., a signal interference in the battery management system, which leads to an undesired fast cooling of the batteries.

Finally, a link between the individual test cases and the associated FMEA failure is made, as shown in Table 4. The assigned severity level is used to further evaluate the potential risk reduction.
In this subsection, the use of FMEAs is illustrated for identifying problematic cases and simulation utilizing specific test cases to show the effect of faults identified in the FMEAs. To reduce risks, countermeasures have to be introduced. According to the FMEA, using a temperature sensor having a lower fault probability can be chosen to reduce risks. Alternatively, considering additional sensors for the BMS to identify critical cases and to raise warnings or initiate preventive actions would reduce the risk. For

Table 4. Test cases linked to the FMEA related scenarios and potential failure description.

Test Cases	FMEA Fault	FMEA Scenario Description	FMEA Failures
1	Battery too warm during driving.	No heat transfer between HV battery cell and cooling circuit leading to a high cell temperature. Extreme warm environmental conditions in combination with mechanical or electrical failures of the cooling package fan and an increased fan power request.	product design failures, electrical interferences, mechanical vibrations, material degradation, improper filling or low vehicle speed and consequently low air mass flow rate.
2, 3	Battery too warm during driving.	Issue in the cooling circuit temperature controller or battery cell sensors leading to wrong temperature requests by the thermal control unit of the HV battery. Battery cell temperature sensor error indicating wrong values causing misleading interpretation of the control unit. This leads to wrong cooling system temperature request of the thermal control unit. A too high cooling system temperature request leads to a heat up of the battery cells by the cooling system.	product design & sensor placement failures, electrical interferences on the temperature sensors, sensor material degradation, sensor communication error.

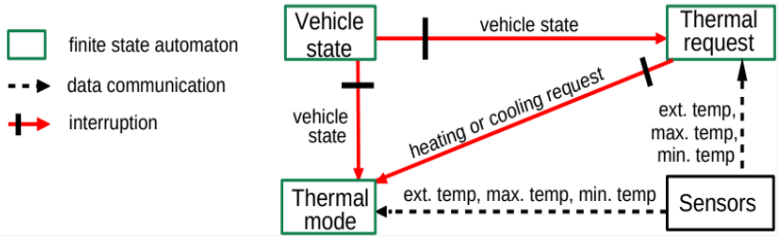


Figure 6. The structure of the formal model of the high-voltage battery management system.

example, when adding an additional temperature sensor, a deviation in cooling demands could be detected, which can be handled by the BMS to prevent not cooling the batteries well enough. Such a case can be tested again using the available tests. However, manually generated tests might not be sufficient to cover all potentially critical scenarios. Hence, an automated approach is required that is accompanied by certain guarantees, like covering all scenarios specified using an underlying model. This second step of risk reduction using formal methods is discussed in the next subsection.

4.2. Applying Formal Methods During Development

Based on the specification of the thermal controller, a formal model is developed. Following the structure of the specification, the model is a parallel composition of four automata, using LNT processes to encode states and LNT transitions to represent the transmission of values and messages between the automata. The scheme depicted in Figure 6 gives an overview of the model structure.

The "Vehicle state" automaton represents the ten states of the vehicle, e.g., OFF (parked and stopped vehicle), POWER_ON (powered up, unmoving vehicle), DRIVING (moving vehicle), and CHARGING (vehicle plugged in a charging station). The remaining states correspond to intermediate states, e.g., the state charging is completed, but the vehicle is not yet powered down, or the STANDBY state. State changes are represented by transitions, on which the "Thermal request" and "Thermal mode" automata can synchronize to monitor the vehicle state, in particular to disrupt their execution when the vehicle is powered down.

The "Thermal request" automaton (see also Figure 6) chooses which action to trigger heating, cooling, or homogenizing, depending on the temperature of the battery components and the vehicle's state. The "Thermal request" automaton has four principal states: OFF (battery turned off), COOLING (cool the battery), HEATING (heat the battery), and HOMOGEN (maintain a homogeneous temperature with the minimal and maximal temperature inside the optimal temperature range). Switching between states is triggered by changes in the minimal or maximal temperature of the battery components (e.g., changing to COOLING if the maximum temperature is above a certain threshold). When the battery is turned on, the automaton returns to the last state before the battery is powered down (initially, the battery is in the HOMOGEN state).

The LNT model for the "Thermal request" is organized as one LNT process per state, representing each state change by a call to the corresponding process. An additional process is required to keep track of the last request (initially homogenization), so the thermal controller can return to this request when reactivated, e.g., when the vehicle is powered on again.

The "Thermal mode" automaton selects among the various heating or cooling modes. It is composed of eleven states, divided into three parts corresponding to the three requests COOLING, HEATING, and HOMOGENization. There are three possible cooling modes (CHILLER, RADIATOR, and COMBINED) and two heating modes (CABIN PLATE and SELF HEATING). The conditions for changing between modes differ according to the vehicle state.

The "Sensors" automaton constrains the possible sensor values to ensure that they obey a realistic evolution (without too large jumps) and keep the size of the state space (of the LTS) small but still relevant. Concretely, temperatures are limited to the range between -10 and 40 degrees Celsius, using only multiples of five, and considering that if a sensor has value V, the next sensed value must be one of the three values V-5, V, and V+5. Using these constraints, the generated LTS has 15 million states, 1.3 billion transitions, and 73,239 labels (corresponding to the state changes and every combination of sensor values conform to the constraints); when minimized for strong bisimulation, the LTS still has six million states and 475 million transitions. Adding further constraints on the sensor values (e.g., considering only those triggering state changes) yields an abstracter and smaller but still relevant model. The corresponding minimized LTS has 120 states, 64,937 transitions, and 1,376 labels, and was produced and minimized in about 40 minutes on a laptop computer. This reduction of the LTS size by just adding appropriate constraints, makes us confident that the approach scales to larger systems, when considering abstractions, constraints and using established techniques to cope with large LTSs, such as compositional and distributed verification.

The 26 functional requirements of the BMS are formalized as 52 temporal logic formulae in MCL: for each requirement, a safety formula (forbidding the presence of undesirable executions) and a liveness formula (requiring the presence of desirable executions) are specified. For instance, the requirement "HV battery thermal state request shall be set to cooling if maximum cell temperature exceeds a calibratable threshold (≥ 35°C)" is translated into the following safety formula

```
[ true * .
  ( "POWER_UP" | "CHARGING" | "DRIVING" ) .
  ( not ( { POWER_DOWN ?any } or "STOP_CHARGING" or
          "CHARGING" or "HV_COOL_REQ !TRUE" or
```

```
            "HV_HEAT_REQ!TRUE" ))*.
  { SENSE ?x:int ... where x >= 35 } .
  ( not ( { POWER_DOWN ?any } or " STOP_CHARGING" or
            "ENABLE_CHARGING" or "READY_RELEASE" ) and
            not ("HV_COOL_REQ !TRUE" ))
] false
```

and the following liveness formula

```
[ true * .
( "POWER_UP" | "CHARGING" | "DRIVING" ) .
( not ( { POWER_DOWN ?any } or "STOP_CHARGING" or
            "ENABLE_CHARGING" or "HV_COOL_REQ !TRUE" or
            "HV_HEAT_REQ !TRUE" ))* .
  { SENSE ?x:int ... where x >= 35 }
] < "HV_COOL_REQ !TRUE" > true
```

In both formulas, the action sequence characterizes that whenever (first action subsequence "true*") the vehicle is either powered on, charging or driving (second action), the vehicle has not left one of these states nor emitted a cooling or heating request (third action subsequence), and a maximal temperature of more than 35 degrees Celsius is measured (fourth action). The safety formula forbids the next action to anything except a cooling request or a change of the vehicle state (these excluded actions describe legal executions of the system without the emission of a cooling request), and the liveness formula requests the existence of at least one execution, where the next action is a cooling request. The other requirements are expressed by formulas of the same structure, varying in the characterization of the actions leading to the state mentioned in the requirement, the expected action of the thermal controller, and the set of actions excluded at the end of the safety formula.

Using the EVALUATOR model checker of CADP, it is checked that the LTS has no deadlocks and satisfies all 52 formulae, giving us confidence that the model faithfully represents the specification and accurately describes (an abstraction of) the behavior of the thermal controller.

While developing the formal model, reading the specification carefully led to the detection of several slight inconsistencies in the naming of variables and signals, in particular, between the natural language description, the state machines, and the requirements. For example, as mentioned before, an inconsistent use of Celsius and Kelvin degrees is detected for the temperature constants as well as an unspecified choice between cooling and heating requests.

Beyond the functional requirements that were part of the initial specification, it is possible to study properties describing the expected behavior of the thermal controller when used within its acceptable operation conditions. For instance, when the thermal controller issues a cooling request, one expects the maximal temperature of the battery to decrease. The following MCL formula can express this:

```
<  true* .
   { HV_COOL_REQ !true } .
   (not ({ SENSE ... } or { HV_HEAT_REQ !true }))* .
   { SENSE ?tmax1:int ?tmin1:int … } .
   (not ({ SENSE ... } or { HV_HEAT_REQ !true }))* .
   { SENSE ?tmax2:int ?tmin2:int … where (tmax2 > tmax1) and
(tmin2 > tmin1) } .
   (not ({ SENSE ... } or { HV_HEAT_REQ !true }))* .
   { SENSE ?tmax3:int ?tmin3:int …
   where (tmax3 > tmax2) and (tmin3 > tmin2) and (tmax3 < 99)
and (tmin3 < 99) }
> true
```

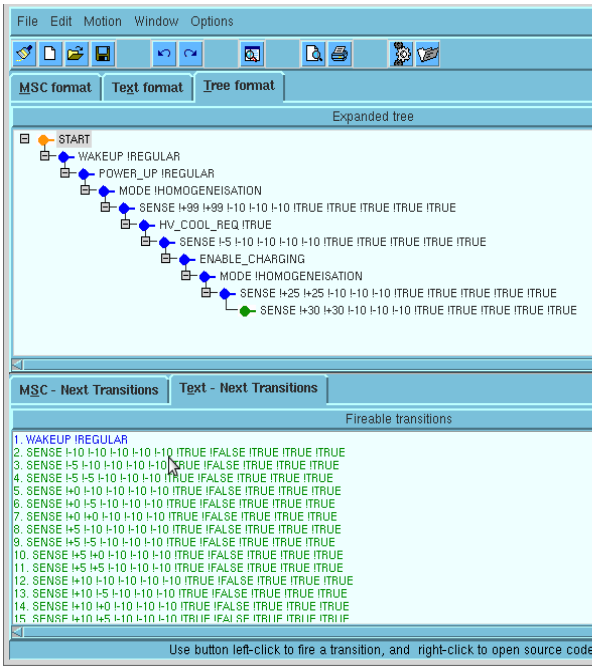


Figure 7. A witness sequence presented in the GUI of the CADP simulator.

As expected, the model checker finds a (shortest path) witness sequence of this undesirable behavior in the formal model, shown in Figure 7.

Indeed, the model does not consider the operation conditions and accurately describes the behavior of the thermal controller, including what happens when the controller is used outside its acceptable operating conditions.

In this subsection, the application of formal methods for testing the BMS is discussed. In addition to particularities, also several cases are shown where the formal methods approach further reduced risks when developing the BMS. First, inconsistencies in the specification are identified, which often lead to faults in the implementation. Second, it is demonstrated that the formal method approach can also be used to identify issues related to assumptions about operating conditions, which might lead to trouble when operating a system. Because of considering all potential behaviors, the formal method approach guarantees that the BMS is working appropriately considering the correctness of the model. Hence, further tests are required, independent of tests generated using formal methods, to avoid faults in the implementation that are not detected due to inaccurate models. Combined with manual testing and tests generated from FMEA cases, a total risk reduction can be compared to system development without the mentioned risk reduction steps. I.e., formal methods help identify additional problems like inconsistent specifications or faulty assumptions about operating conditions and further allow the generation of many test cases from the models.

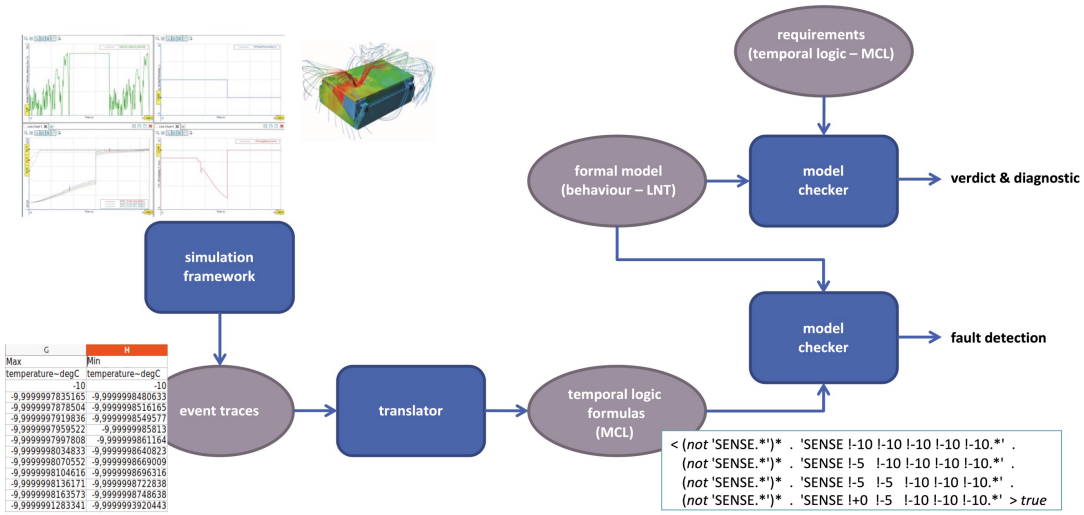


Figure 8. Verification and Co-Simulation based on Formal Models

4.3. Applying Formal Methods During Operation

A formal model can also be used to monitor during operation whether the system behaves correctly and in conformance to the specification, as formalized by the model. Given that this approach is founded on the well-established theory of conformance testing [12], it remains the engineering task of connecting the system under test (a simulator or the real system) to the formal analysis tools and model. In the most straightforward setting, concrete execution traces are to be provided to on-the-fly tools to check the inclusion of the execution trace in the model or verify temporal logic properties. A more involved setting is executing a test case, which interacts with the system under test, observing the outputs and providing appropriate inputs. In any case, such a connection requires some "glue code" to transform the concrete input/output of the system under test to the abstract transitions of the model (and vice-versa).

For the example of the thermal controller, the floating point values are mapped, representing the temperatures measured by the various sensors into the discrete set of values supported by the formal model (integer values corresponding to multiples of five). To improve the performance of the formal analysis tools, experiments are conducted with an optimized abstraction, which compacts a sequence of consecutive identical actions (obtained by our relatively coarse abstraction) into a single action. Then, each step of the execution trace needs to be syntactically expressed as an action of the model. To check the inclusion of the (abstracted) trace in the model, a transformation of the trace into a liveness MCL formula is performed. The utilization of the EVALUATOR model checker enables the verification of the existence of a corresponding sequence in the model. Using this approach, which is depicted in Figure 8, enables the detection of erroneous executions.

Note that monitoring such properties also enables the early detection of faults before they might lead to a critical hazard. To stay in the example of using the battery at a temperature above its acceptable operating domain, detecting that the cooling request is insufficient to reduce the battery's temperature is possible before it reaches a critical level. Hence, using additional properties, the formal model can also reduce residual risks during operation. Therefore, the proposed formal method approach can further reduce operational risks, providing formalized properties to be checked on the fly.

5. Conclusions

In this paper, a comprehensive discussion about risks occurring during the development and operation of CPSs is given. The verification of CPSs is demonstrated, considering safety integrity levels and available standards for safety-critical systems. Methods are discussed as introduced in standards, like the ISO 26262 requiring testing. Furthermore, the remaining residual risks when applying the proposed methods and suggested extending the overall development process are outlined. Based on this discussion, it is suggested to use formal methods during development and monitoring to identify and cope with critical situations.

The risk-reduction method is illustrated using an automotive use case, i.e., a battery management system for high-voltage batteries. The introduction of the FMEA analysis, along with test cases related to the analysis demonstrating incorrect behavior in case of faults, is presented. Additionally, the utilization of formal methods to identify inconsistencies in specifications and assumptions about operational conditions is highlighted. Further, the potential of utilizing the underlying formal methods during operation to detect failures is discussed.

Acknowledgments

ArchitectECA2030 received funding within the Electronic Components and Systems For European Leadership Joint Undertaking (ECSEL JU) in collaboration with the European Union's Horizon2020 Framework Programme and National Authorities under grant agreement number 877539. All ArchitectECA2030 related communication reflects only the author's view, and the Agency and the Commission are not responsible for any use that may be made of the information it contains. The work was partially

funded by the Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) under the program “ICT of the Future” project 877587.

REFERENCES

[1] Rajkumar, R.R., Lee, I., Sha, L., and Stankovic, J., 2010. Cyber-physical systems: The next computing revolution, in Proceedings of the 47th Design Automation Conference, DAC ’10, (New York, NY, USA), pp. 731–736, ACM.

[2] Klück, F., Li, Y., Tao, J., and Wotawa, F., 2023. An empirical comparison of combinatorial testing and search-based testing in the context of automated and autonomous driving systems. Inf. Softw. Technol. 160.

[3] Wotawa, F. and Lewitschnig, H., Sept. 16-18, 2021. Monitoring hierarchical systems for safety assurance, in Proceedings of the 14th International Symposium on Intelligent Distributed Computing, Online Conference, Italy.

[4] Wotawa, F., 2023. Which Components to Blame? Integrating Diagnosis into Monitoring of Technical Systems, in Proceedings of the 36th Intl. Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE), Shanghai, China.

[5] Stettinger, G., and Wotawa, F., 2023. Smart Monitoring for Safety-Assurance in Autonomous Driving, in Proceedings of the 35th International Conference on Software Engineering and Knowledge Engineering (SEKE), San Francisco Bay, USA.

[6] IEC 61508-1, 2010. Functional safety of electrical/electronic/programmable electronic safety-related systems.

[7] Hirsch, T., and Wotawa, F., August 10-11, 2023. Analyzing Residual Risks when Introducing Monitoring and Diagnosis into Systems. In Proceedings of the 10th International Conference on Dependable Systems and Their Applications (DSA), Tokyo, Japan.

[8] Muehlburger, H., and Wotawa, F., 2024. Making Systems Fail-Aware: A Semi-Supervised Machine Learning Approach for Identifying Failures by Learning the Correct Behavior of a System, Manuscript submitted for publication to the 12th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes 2024.

[9] Garavel, H. (editor), and Graf, S., December 2013. Formal Methods for Safe and Secure Computer Systems. Bundesamt für Sicherheit in der Informationstechnik, BSI study 875, Bonn, Germany. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/formal_methods_study_875/formal_methods_study_875.html.

[10] ISO26262, 2018. Road Vehicles – Functional Safety. International Organization for Standardization (ISO).

[11] ISO/PAS 21448, 2019. Road Vehicles – Safety of the intended functionality. International Organization for Standardization (ISO).

[12] McDermott, R.E., Mikulak, R.J., and Beauregard, M.R., 2008. The Basics of FMEA (2nd ed.), CRC Press.

[13] Ammann, P., and Offutt, J., 2017. Introduction to Software Testing (2nd ed.). Cambridge University Press.

[14] Horel, J.B., Ledent, P., Marsso, L., Muller, L., Laugier, C., Mateescu, R., Paigwar, A., Renzaglia, A., and Serwe, W., 2023. Verifying Collision Risk Estimation using Autonomous Driving Scenarios Derived from a Formal Model, J. Intell. Robotic Syst., 107(4), p. 59.

[15] Baier, C. and Joost-Pieter Katoen, J.P., 2008. Principles of model checking. MIT Press, ISBN 978-0-262-02649-9.

[16] Bergstra, J.A, Ponse, A., and Smolka, S.A., 2001. Handbook of Process Algebra. North-Holland Elsevier. ISBN 978-0-444-82830-9.

[17] Milner, R., 1989. Communication and concurrency. PHI Series in computer science, Prentice Hall. ISBN 978-0-13-115007-2.

[18] Hoare, C. A. R., 1985. Communicating Sequential Processes. Prentice-Hall, ISBN 0-13-153271-5.

[19] Bergstra, J.A., and Klop, J.W., 1985. Algebra of Communicating Processes with Abstraction. Theoretical Computer Science, 37, pp. 77-121. https://doi.org/10.1016/0304-3975(85)90088-X.

[20] Garavel, H., Lang, F., and Serwe, W., 2017. From LOTOS to LNT. In ModelEd, TestEd, TrustEd - Essays Dedicated to Ed Brinksma on the Occasion of His 60th Birthday. LNCS vol. 10500. Springer Verlag, https://doi.org/10.1007/978-3-319-68270-9_1.

[21] Garavel, H., Lang, F., Mateescu, R., and Serwe, W., April 2013. CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes, International Journal on Software Tools for Technology Transfer (STTT), 15(2), pp. 89-107.

[22] Mateescu, R., and Thivolle, D., 2008. A Model Checking Language for Concurrent Value-Passing Systems. Proceedings of the 15th International Symposium on Formal Methods (FM'2008), LNCS vol. 5014, pp. 148-164. Springer Verlag. https://doi.org/10.1007/978-3-540-68237-0_12.

[23] Mateescu, R. and Requeno, J.I., 2018. On-the-Fly Model Checking for Extended Action-Based Probabilistic Operators. Springer International Journal on Software Tools for Technology Transfer (STTT), 20(5), pp. 563-587. https://dx.doi.org/10.1007/s10009-018-0499-0

[24] Büchi, J.B., 1966. Symposium on Decision Problems: On a Decision Method in Restricted Second Order Arithmetic, Editor(s): Ernest Nagel, Patrick Suppes, Alfred Tarski, Studies in Logic and the Foundations of Mathematics, Elsevier, 44, pp. 1-11, ISSN 0049-237X, ISBN 9780804700962, https://doi.org/10.1016/S0049-237X(09)70564-6.

[25] Lantreibecq, E., and Serwe, W., 2014. Formal analysis of a hardware dynamic task dispatcher with CADP. Sci. Comput. Program., 80, pp. 130-149. https://doi.org/10.1016/j.scico.2013.01.003.

[26] Marsso, L., Mateescu, R., and Serwe, W., April 14-20, 2018. TESTOR: A Modular Tool for On-the-Fly Conformance Test Case Generation, 24th Intl. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2018), Thessaloniki, Greece.

[27] Marsso, L., Mateescu, R., and Serwe, W., December 9-11, 2020. Automated Transition Coverage in Behavioural Conformance Testing, 32nd IFIP International Conference on Testing Software and Systems (ICTSS'2020), Napoli, Italy.

[28] Van Mierlo, J., Berecibar, M., El Baghdadi, M., De Cauwer, C., Messagie, M., Coosemans, T., Jacobs, V.A., Hegazy, O., 2021. Beyond the State of the Art of Electric Vehicles: A Fact-Based Paper of the Current and Prospective Electric Vehicle Technologies. World Electr. Veh. J., 12, 20. https://doi.org/10.3390/wevj12010020.

[29] AVL, Model.Connect: AVL’s open model integration and co-simulation platform, last accessed December 23, 2023. https://www.avl.com/en-at/simulation-solutions/software-offering/simulation-tools-z/modelconnect