David Santiago Velasco Moncada, Fraunhofer IESE
Daniel Schneider, Fraunhofer IESE
Ana Petrovska, Technical University of Munich
Nishanth Laxman, Technical University of Kaiserslautern
Felix Möhrle, Technical University of Kaiserslautern
Stefan Rothbauer, Siemens AG
Marc Zeller, Siemens AG
Chee Hung Koo, Robert Bosch GmbH
Samira Safdari, Expleo Germany

8

Dynamic Safety Certification for Collaborative Embedded Systems at Runtime

Traditionally, integration and quality assurance of embedded systems are done entirely at development time. Moreover, since such systems often perform safety-critical tasks and work in human environments, safety analyses are performed and safety argumentations devised to convince certification authorities of their safety and to certify the systems if necessary. Collaborative embedded systems, however, are designed to integrate and collaborate with other systems dynamically at runtime. A complete prediction and analysis of all relevant properties during the design phase is usually not possible, as many influencing factors are not yet known. This makes the application of traditional safety analysis and certification techniques impractical, as they usually require a complete specification of the system and its context in advance. In the following chapter, we introduce new techniques to meet this challenge and outline a safety certification concept specifically tailored to collaborative embedded systems.

8.1 Introduction and Motivation

Embedded systems are presently evolving from stand-alone closed embedded systems towards open and collaborative embedded systems (CESSs). In CESSs, collaboration can take place on different scales—from small, predefined groups of systems to large, heterogenous collectives of systems at a global level—and there will be an evolution from smaller scales to larger scales of collaboration [Damm et al. 2019]. Collaboration between CESSs means sharing information concerning context perception, reasoning, and actuation. It means acting in unison with each other to reach a suoperordinate goal or to render a higher level service that could not be achieved by single systems alone.

The potential in this is very significant. For many existing applications, it will become possible to improve important performance properties (e.g., speed or efficiency), decrease resource consumption (e.g., fuel consumption), and improve safety (e.g., in a platooning scenario) simultaneously. The implications for society and the economy are correspondingly huge, as the systems envisioned have the potential to completely transform and improve our societies and economies in a groundbreaking way.

There are, however, a number of challenges that must be tackled before this vast potential can be unlocked. Most importantly, established engineering approaches—and safety engineering approaches and standards in particular—focus on closed systems and cannot be applied to future CESSs without further ado.

Traditional safety engineering approaches typically require a complete understanding and specification of the system (and its context) under consideration at design time. CESSs, however, may be integrated with other CESSs at runtime and thus form collaborative system groups (CSGs) dynamically. Some of these systems may not even be on the market during the design phase of others. Therefore, the goal of obtaining a full specification of all possible variations of CSGs is generally not feasible at design time, and new safety certification approaches are required.

CrESt set out to investigate corresponding solution ideas based on a range of complementary industrial use cases. A key premise of our proposed safety assurance approach is that the required information is only partially available at design time and must be completed at runtime. This assumption means that certain parts of the assessment

process that traditionally take place at design time have to be postponed until runtime, when all variables can be resolved. This applies in particular to the final verdict as to whether the integrated CSG is safe or not. Nevertheless, we consider it essential to conduct as much preparatory work as possible during the design phase to ensure that the final assessment at runtime can be performed efficiently and in a largely automated way.

The remainder of this chapter is structured as follows: Section 8.2 gives a brief overview of the proposed safety certification process. Section 8.3 describes the integration of modular safety cases at runtime to obtain a coherent safety case for the systems group. Section 8.4 addresses the inner details of the modular contracts at different levels and how they can be standardized. Finally, Section 8.5 concludes this chapter with a summary.

8.2 Overview of the Proposed Safety Certification Concept

Our safety certification concept stipulates a two-stage process. The first step concerns the preparatory work at design time. Here, each CES is equipped with a modular safety case. In each modular safety case, the respective CES is conceived as a stand-alone system and an interface for the integration with other modular safety cases is defined. The main purpose of these modular safety cases is to provide safety arguments and evidence to enable better decision-making at runtime. To this end, modular safety cases include the working conditions of their respective CESs, such as requirements that are placed on the environment or other CESs. Furthermore, they specify guarantees that may or may not be given by the CES for its own services provided, depending on whether certain conditions are met or not. Hazard and risk analysis make it easier to understand potential hazards that may originate from a system. Context modeling and analysis allow identification of the uncertainties that may arise in a particular context, as well as shared resources that have to be coordinated between systems. Fault analysis investigates possible malfunctions and drives the definition of safety measures. Finally, all these aspects are then combined into a modular safety case for each system.

The second step in our safety certification concept concerns the integration of the modular safety cases at runtime with respect to the planned collaboration. For this purpose, information on how the CESs

are organized among themselves and what types of dependencies emerge as a result is required. This information can, for example, originate from a human integrator who arranges the CESs (adaptable factory) or it can be generated by a machine — for example, from onboard computers (vehicle platooning). The resulting system architecture therefore reflects the interdependencies between the CESs, which are needed to integrate their modular safety cases. Finally, the integrated safety case is evaluated and used to assess the safety of the CSG as a whole.

To perform the second step, both semi-automated approaches with human intervention (cf. Section 8.3) and fully automated approaches (cf. Section 8.4) are feasible, depending on the context. In the case of a semi-automated approach, a human operator can be kept in the loop through manual selection of a suitable configuration, for example. In the case of full automation, the systems can negotiate their relevant properties on a peer-to-peer basis. This can be done on a contract basis by providing and demanding guarantees for services exchanged. At runtime, the safety of the collaboration is continuously monitored in a feedback loop. If the conditions for safe cooperation are no longer met, the systems must react accordingly — for example, by graceful degradation or termination of cooperation.

8.3 Assuring Runtime Safety Based on Modular Safety Cases

As already motivated, CESs must react to ongoing, constant changes in their open, dynamic context. However, for the systems to react at runtime, the CESs must first be aware of their relevant context [Petrovska and Grigoleit 2018] so that they can subsequently monitor and assess the type and impact of the context change. Additionally, in parallel, uncertainties resulting from these changes have to be handled effectively. To allow an efficient certification for the CSG at runtime, a dynamic risk assessment is performed. The systematic documentation of all relevant evidence enabled by modular safety cases helps safety engineers during the certification process.

In the context of an adaptable factory, major trends such as the growing individualization of products and volatility of product mixes lead to a situation where every product is produced differently and is routed according to the current production situation [Koren 1999], [Yilmaz 1987]. In this chapter, we demonstrate our methods using a small case study as a running example. Reconfigurable industrial CESs

(such as a robot arm and a tray as a storage unit) are used to assemble a small roll that consists of a roll body, an axle, and two metal discs as depicted in [Figure 8-1](#).

Fig. 8-1: Case study description for an adaptable factory

8.3.1 Modeling CESs and their Context

In practice, CESs are typically developed either within one original equipment manufacturer or by different suppliers. Moreover, when CESs form collaborative system groups (CSGs), the CSGs can hardly be analyzed a priori as relevant context because they are typically not explicitly defined at design time. On the contrary, they are formed, at least to a certain extent, emergently at runtime, which is actually a key trait and strength of CESs and the open ecosystems they enable. A CSG fulfills a global goal that an individual CES cannot fulfill alone. Of course, as already motivated, the increased complexity of the functionality requires different verification, validation, and certification approaches.

One method for testing a CES for consistency and correctness is the use of executable models, referred to as monitors. A monitor observes the execution of a system and determines whether it meets the given requirements [Goodloe and Pike 2010]. The monitor can then register and log the violations found during the test. In particular, in CSGs, monitors may help in detecting specification violations when the requirements are described as goals. One of the main characteristics of these goals is that they are influenced by the orchestration of the different CESs. However, for the systems to react at runtime, the CESs must first be aware of their relevant context through the constitution of runtime models of the context where the CESs operate, so that they can subsequently monitor and assess the type and impact of context changes on the systems (this is explained further in Section 8.3.3).

Context awareness is generally accomplished through the creation of context models, which depict relevant aspects of the context for the CES. The context models are initially created at design time of the systems, and updated accordingly during runtime. However, in practice, there are differences in the context modeling concepts among different manufacturers and suppliers. This exacerbates the integration effort of different data models to integrate CESs, causing a "semantic heterogeneity" [Jirkovský et al. 2016]. The use of ontologies unfolds the potential to serve as a conceptual as well as a technological representation of such data models to cope with the semantic heterogeneity and enable semantic interoperability [Negri et al. 2016].

We propose an ontology, shown in [Figure 8-2](#), that integrates elements of two types of context: (1) classes and relationships of the interacting CESs, known as the operational context, and (2) sources of information with respect to the CESs, seen as the context of knowledge.

The operational context models the interaction between a system under analysis and other systems in the environment, whereas the context of knowledge focuses on relevant knowledge sources that possess information about the system under analysis [Daun et al. 2016]. The aim of integrating these two types of context together is to

gather relevant classes for constructing a context model that includes relevant information that can subsequently be used to check specification violations during runtime monitoring.

Our proposed ontology allows a distinction between the system under analysis and the parts of the context that may influence the system but that cannot be changed. The system under consideration/analysis is called the “context subject.” The context, composed of context objects, is that part of the environment that is relevant to the context subject. To allow the distinction between parts of the context that are collaborative and context objects that do not collaborate, we name a context object “collaborative context object.” This distinction permits the identification of the entities that interact with the context subject, their dependencies with the context subject, and dependencies among context objects.

From a functional perspective, collaborative context objects provide services or functions that are accessible to the context subject. In our ontology, context object function entities are used to document the dependencies and the exchange of data between the context subject and these context functions. From a behavioral perspective, to enrich the documentation of a context function, we use context object state and context state variable entities. These entities provide information about the different states, and their related variables, that define the behavior of a context function. Furthermore, these context states define the context object behavior of collaborative objects in the context.

In the context of knowledge, the ontology integrates entities that provide and/or constrain the collaborative objects in the context. In particular, we are interested in safety guarantees and hazards that provide information and constrain context objects and the context subject based on standard rules.

Modeling Context in the Adaptable Factory

The creation of a context model is a process that is executed at both design time and runtime. At design time, the functional, structural, and behavioral aspects of the operational context of the CESs are modeled into a generic context ontology. This generic ontology can then be refined to create a domain-specific ontology that captures all relevant information of the domain — in our case, the adaptable factory use case. Both ontologies are represented as OWL files. The resulting ontology enables CESs to store context-related information and draw conclusions from this information. The domain-specific ontology of the adaptable factory serves two purposes: the ontology

(1) defines the input data, defining in particular where the data is located and what the relationships between different data are. For a mechatronic object, for instance, the ontology may specify where this CES is located (i.e., its position in the machine cell). This results in purpose (2), where the ontology can be used to find constraints in the data. A mechatronic object may specify, for instance, the maximum speed at which the CES moves.

At runtime, the CESs are identified and a CSG configuration is selected. This information is then replicated into the adaptable factory context ontology: for each CES identified, a new individual (i.e., the instance of an entity in the ontology) is created and all the relevant information is stored as data properties of the new individual. Finally, the information stored in the context ontology is queried to build a runtime context model. For implementation and evaluation purposes, the runtime context model is stored in an XML file.

8.3.2 Runtime Uncertainty Handling

The term “uncertainty” is used in different ways in different research fields. Uncertainty and its impact are being extensively explored by research communities in areas such as economics, software and systems engineering, robotics, artificial intelligence, etc. In the field of cyber-physical systems, there are multiple definitions for uncertainty, and the one provided by [Ramirez et al. 2012], serves as a rationale for our collaborative systems: “Uncertainty is a system state of incomplete or inconsistent knowledge such that it is not possible for the system to know which of two or more alternative states/configurations/values is true.” As explained, CESs interact and integrate at runtime; the uncertainties that occur during runtime, more specifically the ones that might create safety-critical scenarios for CSGs, are of prime importance here.

Two types of uncertainties can be distinguished: epistemic uncertainties and aleatory uncertainties [Perez-Palacin and Mirandola 2014]. The epistemic type refers to the uncertainties that arise due to incomplete knowledge or data, and the aleatory type of uncertainty is the result of the randomness of certain events. Epistemic uncertainties can be handled effectively by collecting additional information, meaning that the uncertainty then ceases to exist. In contrast, aleatory uncertainties are relatively complicated because of their inherent randomness. The concept presented will help to address most of the epistemic types of uncertainties and few

of the aleatory type. The main viewpoint of handling uncertainties explained in this section would be from that of safety assurance.

Concept Overview

The outline of the concept is to provide a quantified, well-reasoned, and well-defined mapping of the uncertainties identified to their corresponding mitigation steps. The CSG is constantly monitored at runtime for occurrences of uncertainty and, based on the definitions and parameters of these occurrences, runtime adaptations of configurations for CEs or any further specific measures defined in the mapping are undertaken to ensure safety.

Development of a U-Map for the Adaptable Factory

The solution approach is centered around the development of an uncertainty map (U-Map) artifact during design time. This artifact is used as the knowledge base at runtime for monitoring and executing mapped mitigation measures for uncertainty occurrences. The first step in the development of a U-Map is identifying the relevant uncertainty and its classification. This step is the most vital and also the most time-consuming. Here, all possible uncertainties are listed based on various classifications from research, the most recent and extensive being the one from [Cámara et al. 2015]. To aid the process of identifying uncertainties with respect to the information exchange between CEs from an ontological perspective, the classification provided by [Hildebrandt et al. 2019] is used. Both of these classifications are used as a checklist to identify possible uncertainties at runtime, specific to the use case. Once identified, concrete instances of uncertainty must be defined. In due process, uncertainties that can be resolved during the design of the CSG but have not been considered in general system development have to be updated. These instances have to be further iterated and quantified as monitor specifications so that they can be detected at runtime. Examples include ambiguity in sensor measurements, inconsistency in service descriptions, incompleteness in self-descriptions of CEs, or incompleteness in information exchange. The next step involves identifying all possible failures that might occur from these uncertainties that might put the system into a hazard state and might subsequently lead to an accident or harm. To aid this, standardized hazards and failures from [ISO 2010] are considered for the adaptable factory and from [ISO 2018] for vehicle platooning. Bayesian networks [Halpern 2017] and the Dempster-Shafer theory [Shafer 1976] based on probability theory are found to be effective for mapping the uncertainties identified to

possible failures and hazards. As an outcome, we notice that each uncertainty can lead to multiple hazards and every hazard can be a result of one or more uncertainty occurrences. The next step involves mapping these hazards to their corresponding mitigation measures. For the use case of the adaptable factory, an intermittent step of rectification acts as an additional layer of safety assurance, which is feasible because of the semi-automated approach employed. The uncertainties that can be eliminated by rectification measures occur predominantly in information exchange between individual CESs. In certain cases, the system may still be in a hazardous state even after the uncertainty has been eliminated through rectification. To maintain safety, these hazards must be further mapped to appropriate mitigation measures. The mitigation measures can be either based on present industrial standards or they can be reconfiguration identified as degradation modes. In certain scenarios, these degradation modes alone are not sufficient and additional protective measures have to be taken.

Fig. 8-3: Visualization of a U-Map

In the end, an extensive set of identified uncertainties is mapped to an even bigger set of possible hazards, which in turn is mapped to a rather small set of degradation modes and protective measures. This U-Map makes implementation simple and does not have an exploded range of mitigation measures that have to be undertaken specifically to handle every uncertainty. However, creating such a map and ensuring its completeness to handle all possible uncertainties at runtime can be a complex task, which presently relies greatly upon the research communities' identified sources of uncertainty, which in themselves might not be complete. Furthermore, we consider subjective probability for uncertainty occurrence [Shafer 1976],

which in itself might be imprecise. A U-Map can be visualized as shown in [Figure 8-3](#).

At runtime, with the help of this U-Map, the necessary rectification measures are taken by the safety engineers, thereby eliminating relevant uncertainties before safety approval. The degradation modes and additional protective measures serve as an input for further explanation of dynamic safety certification, in that they enable the appropriate configuration and safety measures to be chosen.

8.3.3 Runtime Monitoring of CESs and their Context

The generated runtime context model from Section 8.3.1 can be used to deliver relevant information that enables runtime analysis/monitoring.

With the information available from the context model explained in Section 8.3.1 and the specifications for uncertainty detection from the DTU map as explained in Section 8.3.2, monitors can be created to monitor the properties of interest in a given CES. The monitors and the specifications are created at design time; however, the monitors are executed during runtime. For example, it may be desirable to monitor the speed of a mechatronic object to determine whether the said speed obeys safety requirements. A common way to create a runtime monitor is to translate assertions about the state of a context element into rigorous specification formalisms [Bartocci et al. 2018], such as LTL formulas, to subsequently create instrumentation files with the monitor specifications. In our example, a domain expert can provide the assertion “It is always the case that CES1 moves at a speed of 2 mm/s” and this can be translated into the LTL formula $G(ces1.speed \leq 2)$; this formula can be used to create the monitor specification [Bartocci et al. 2018] as instrumentation files¹ that have to be integrated into the CES. The runtime monitor specification must be created during design time and the instrumentation files generated should be integrated during development. At runtime, these monitor specifications, including the specifications from the DTU map, will be represented in the form of modular safety cases. In the context of an adaptable factory, a centralized software that is responsible for task orchestration and system assessment can identify and compile the monitoring requirements dynamically to allow for the final approval by safety engineers in a semi-automated certification process.

¹ <http://fsl.cs.illinois.edu/index.php/MOP>

8.3.4 Integrated Model-Based Risk Assessment

Due to frequent changes in the products being manufactured, adjusting a factory quickly is a major challenge. This raises concerns with regard to dependability due to unknown configurations at runtime. Thus, apart from functional aspects (i.e., the check of whether a factory is able to manufacture a specific product), safety aspects as well as product quality assurance aspects must be addressed. In flexible production scenarios, a risk assessment must be conducted after each reconfiguration of the production system. Since this is a prerequisite for operating the factory in the new configuration, a manual approach can no longer effectively fulfil the objectives for assuring safety in highly flexible manufacturing scenarios. During production, every process step has the potential to influence the quality of the product in an undesirable way, for example depending on the precision of the equipment used, or random failures while executing the process step. This is captured in a Process Failure Mode and Effects Analysis (process FMEA) with the concept of failure modes of a process step as well as the respective severity. The process FMEA also defines measures for detecting and dealing with unwanted effects on product quality. Since both the factory's configuration and its products change constantly in adaptable factory scenarios, a process FMEA must be performed dynamically during operation.

In the context of industrial production systems, the safety standards ISO 13849 [ISO 2006] or IEC 62061 [IEC 2005] provide guidelines for keeping the residual risks in machine operation within tolerable limits. For every production system, a comprehensive risk assessment is required, which includes risk reduction measures if necessary (e.g., by introducing specific risk protective measures such as fences). The resulting safety documentation describes the assessment principles and the resulting measures that are implemented to minimize hazards. This documentation lays the foundation for the safe operation of a machine and it proves compliance with the Machinery Directive 2006/42/EC of the European Commission [European 2006].

In this section, we present an approach for the model-based assessment of flexible and reconfigurable manufacturing systems based on a meta-model. This integrated approach captures all information needed to conduct both risk assessment and process FMEA dynamically during the runtime of the manufacturing system in an automated way. The approach thus enables flexible manufacturing

scenarios with frequent changes in the production system up to a lot size of one.

Meta-model SQUADfps

To address the aforementioned problem statement for a dynamic assessment at runtime, a meta-model called SQUADfps (machine Safety and product QUALity assessment for a flexible proDuction system) is presented [Koo, Rothbauer et al. 2019]. This metamodel considers hazards and failure modes due to both safety and quality issues. Four categories are introduced within the SQUADfps metamodel: *process definition*, *abstract services*, *production equipment*, and *process implementation*. This depicts the modularity within an adaptable factory scenario. This integrated model-based approach allows information not only from each item of modular production equipment (i.e., CESs within CrESt) to be considered during the assessment, but also from the production context.

With the focus on quality assurance, an integrated CES that provides services for production (EquipmentService) steps brings along information about its possible failure modes (EquipmentFailureMode) at runtime. Equipment that provides quality measures (CoveredFailureMode) brings along the information about the effectiveness of the measures (e.g., detection) regarding specific failure modes (EquipmentFailureMode). The suitability of the planned production schedule—that is, the equipment’s suitability to provide the required services—can be analyzed by conducting a model-based quality assessment process FMEA, taking the production recipe and the services required into account, as shown in [Figure 8-4](#). For the risk assessment, possible hazards introduced into the overall production system during process implementation can be captured and checked against the available SafetyFunction to determine whether safety requirements are fulfilled.

The benefits of applying SQUADfps for the dynamic certification of CSGs in an adaptable factory are twofold: firstly, this metamodel allows risk-related information to be captured dynamically at runtime. Secondly, the risk information—be it hazards or failure modes along with the analysis of this information—provides input for the modular safety cases systematically. The process of conducting a dynamic safety certification is discussed in subsequent paragraphs.

Fig. 8-4: Meta-model SQUADfps for a dynamic machine safety and product quality assessment at runtime [Koo, Rothbauer et al. 2019]

Based on the case study described, we now present the results generated using SQUADfps to aid understanding.

Case Study Example

Table 8-5 shows the product recipe $R = r_1, \dots, r_6$ for producing a pulley wheel, specifying the required process steps. For each recipe step required, the relevant failure modes are listed and a measure of their severity (Sev) is given as they would impact the final product. This information can be added by the design team of the product, as they know exactly how each failure mode will impact the final product. For each failure mode in the product recipe, a measure of the detectability (Det) by scheduled quality measures is also given. For example, the failure mode Misplacement of the service Pick & place can be detected visually with high certainty (detection value is one) but the failure mode Crimping will likely go unnoticed as these can only be detected by stress tests that are not considered in this process instantiation.

Consider the first process instantiation P in Table 8-5, consisting of process steps $p_1, \dots, p_{6a}$. Process P is capable of producing the pulley wheel, as it provides all the services required and in the correct order. For each deployment of a required recipe step to a process step on a concrete item of equipment, the occurrence—the information regarding failure mode frequency (Occ)—can be added to the model. This information is provided by the vendor of the production equipment that provides production services and the operator will

possibly also update these values based on local production experience (e.g., environment conditions).

Recipe R	Failure Mode	Sev	Service	Occ	Process P	Det	RPN	Occ'	Process P'	Det'	RPN'
r ₁ : Deliver tray	Misplacement	4	Convey	2	p ₁ : Belt conveyer	1	8	2	p' ₁ : Belt conveyer	1	8
	Shock	5		1		1	5	1		1	5
r ₂ : Mount axle	Misplacement	4	Pick & place	2	p ₂ : Robot arm	1	8	2	p' ₂ : Robot arm 2	1	8
	Crimping	2		4		5	40	2		2	8
r ₃ : Circular grease roll	Too little	5	Apply liquid	4	p ₃ : Robot arm	1	20	4	p' ₃ : Robot arm 2	1	20
	Too much	2		4		1	8	4		1	8
r ₄ : Mount 1 st disc	Misplacement	4	Pick & place	2	p ₄ : Robot arm	1	8	2	p' ₄ : Robot arm 2	1	8
	Crimping	3		4		5	60	2		2	12
r ₅ : Circular grease 2 nd disc	Too little	5	Apply liquid	4	p ₅ : Robot arm	1	20	4	p' ₅ : Robot arm 2	1	20
	Too much	2		4		1	8	4		1	8
r ₆ : Mount roll on 2 nd disc	Misplacement	4	Pick & place	2	p ₆ : Robot arm	1	8	2	p' ₆ : Robot arm 2	1	8
	Crimping	5		4		5	100	2		2	20
			Visual inspection		p _{6a} : Worker				p' _{6a} : Laser scanner		

Table 8-5: Case study process definition and two possible deployments of the Process P and P' (production schedules)

Looking at the risk priority numbers (RPN), the chosen process deployment P for producing the product seems to come at a high risk of not reaching the required quality goals, which is indicated by the high value of the RPN. An alternate process instantiation using more reliable equipment and higher precision quality measures can be seen in Process P' in Table 8-1. The equipment Robot arm 2 has a lower probability of introducing the critical crimping failure mode (Occ value 2 for Robot arm 2 vs. Occ value 4 for Robot arm 1) and a high-precision laser scanner is used as a quality measure. As we can see, the concrete instantiation of the process on actual equipment influences the occurrence values for each failure mode of a production step as well as detection values. As a consequence, the risk measured—for example, using the product of occurrence and severity (RPN)—will differ and the highest values of RPN are lowered from 100 to 20.

Considering machine safety, the results generated using model-based risk assessment for the various integrated CESs can be seen in Table 8-6 (for the production schedule P'). In this table, the combination of the risk parameters F (Frequency), S (Severity) and P (Possibility for avoidance) will determine the risk level (which is represented as the Performance Level PL used for safety analysis).

Process P'	Interaction Task T	F	Hazard H	S	P	PL'	Safety Function	PL
Robot arm 2	Loading of roll body with axle	F2 (high)	h1: Shearing due to robot movement	S2 (high)	P2 (high)	PL e	Safety-sensitive cover	PL e
Belt conveyor	Loading of roll body with axle	F2 (high)	h2: Squeezing due to belt getting caught	S1 (low)	P1 (low)	PL b	Light curtain	PL d
Robot arm 2	Maintenance of robot's handling tool	F1 (low)	h3: Bruising due to robot movement	S1 (low)	P1 (low)	PL a	Safety-sensitive cover	PL e

Table 8-6: Exemplary results for the model-based risk assessment

In the exemplary safety risk assessment shown, we can see that the integrated robot (CES) might cause a hazard h_1 shearing when the operator loads the material into the assembly cell. The runtime assessment system evaluates this risk as PL e (very high risk according to ISO 13849-1) based on different data from the context and allocates a possible existing safety function to h_1 . As the integrated safety-sensitive cover for the robot has a very high reliability (also PL e), it provides proof that the risk of h_1 can be mitigated during the interaction task. A similar analysis procedure is also performed for all relevant hazards to generate the foundation for the safety risk assessment.

This approach is of a qualitative nature, which in practice is very effective for prioritizing measures for the main problems. It can be extended to deliver quantitative measures of production risk. The approach aims to assist humans in finding an optimal solution for producing a product while considering both machine safety and product quality aspects.

8.3.5 Dynamic Safety Certification

The goal of a dynamic and runtime safety certification in the context of an adaptable factory allows an accelerated operational safety approval (i.e., certification) after system modifications are performed. With the dynamic safety certification method presented, automated capture and analysis of runtime data can be performed more efficiently. In the production domain, the human’s role as the person responsible remains significant to guarantee system safety in accordance with the European Machinery Directive [European 2006]. Therefore, a human-in-the-loop assurance based on the concept of modular safety cases [Kelly 2007] is proposed for the adaptable factory use case.

The concept of using a modular safety case allows relevant requirements (i.e., safety goals) and analysis results (i.e., argument and evidence) to be documented in a systematic way for the required certification process. The initiation of these modular safety cases highlights the context-relevant requirements that must be fulfilled by the specific runtime system configuration—as already mentioned earlier—to deal with monitoring, uncertainty, and risk requirements. Dealing with all these requirements successfully and the completion of modular safety cases at runtime will contribute to the overall certification of the adapted CSG.

Fig. 8-7: *Dynamic safety certification for adaptable factory*

An interactive tool called AutoSafety [Koo, Vorderer et al. 2019] has been developed to help operators and safety engineers to assess and approve the adaptable assembly demonstrator at runtime (the dynamic safety certification process is shown in [Figure 8-7](#)). This semi-automated certification approach builds up the safety case of the CSG by integrating modular safety cases of the integrated modular systems while considering relevant runtime safety aspects (e.g., runtime measures) identified during reconfiguration. Moreover, AutoSafety will be able to highlight the status of each modular safety case individually with regard to whether they are successfully fulfilled based on runtime data. When automated analyses of certain runtime variables are conducted, the respective modular safety cases can be updated automatically. Humans can also perform updates to ensure the correctness, accuracy, and completeness of the results. For highly adaptable factory scenarios in the future, this dynamic runtime

certification approach will be able to accelerate the safety approval procedure and minimize manual engineering efforts required for assessment and documentation.

8.4 Design and Runtime Contracts

This part of the chapter explains the use of contracts within the specification, development, and standardization of safety-critical collaborative systems. The concepts are illustrated in connection with the use case "Vehicle Platooning"

One of the biggest challenges with collaborative systems is to ensure that the systems behave safely — not only as individuals but also as an integrated system. At the same time, a collaborative system can only be successfully introduced into the market if its safety can be assured — for example, based on an adequate certification process. This could be achieved as part of the design time engineering or based on a combination of design time engineering and runtime assurance/certification measures. In the first case (i.e., safety assurance achieved during design time engineering), a traditional method of system development would be pursued, with the difference being that this would be done for the integrated system, which is an abstract construct. However, this requires that the CESs and CSGs are known to a sufficient extent — for example, by means of comprehensive standardization of system and service characteristics for a domain. The second case (i.e., safety assurance based on a combination of design time engineering and runtime assurance/certification measures) is ideal, given the natural dynamics of collaborative systems. To achieve this, we require a fully integrated and comprehensive solution (e.g., runtime certification, an infrastructure for communication, certain standardization — for example, with regard to interoperability, tracking, evaluation, enforcement, etc.). However, it is impossible to decide what this solution should look like for future systems, although we can make pragmatic assumptions as far as we require these aspects for our work. In this project, we have focused in particular on closing the gap between traditional design time certification and runtime certification. We have done this by introducing an approach for collaborative systems specification that relies heavily on contract-based design and engineers the exchange of guarantees and demands/assumptions during runtime. During the design phase, contracts allow the distribution of responsibilities of the participants

to be defined, and during runtime, they allow safe behavior to be enforced.

8.4.1 Design-Time Approach for Collaborative Systems

One of the main drivers in the definition of our approach is the lack of understanding of how to establish a safe collaboration. Therefore, it is not our aim to find the best solution for a particular aspect. Instead, we are aiming for a more comprehensive solution that could help us to better understand the problem and thus distinguish and highlight the more important aspects when considering certification for safe behavior. For this reason, the approach defines the need to specify and certify the CSG itself. The goal is to make the CSG specification the standard that defines the minimum requirements for collaboration in a specific scenario (in our case, a vehicle platoon). System developers who want to participate in such collaboration must then comply with the specification and the associated domain regulations.

Creating the CSG Specification

To build a CSG specification systematically, we consider the following refinement steps.

At the business/domain level, the CSG designer must initially define the aim or subject matter of the collaboration. We believe that, given the nature of collaborative systems, service-oriented architectures (SOA) [Bell 2008] offer useful concepts for specifying this aspect. These include the specification of the functions and objectives of collaboration, roles, allowable system compositions, structural configurations, environmental constraints, and the definition of service contracts.

At the functional level, and by following a traditional top-down approach, the reference architecture that defines how functionality and responsibilities are distributed among roles is built. This includes defining the minimum requirements, the behavior, and functions of the roles and their dependencies, and setting the flexibility points. As mentioned above, runtime contracts could be used to enable such flexibility points. We consider ConSerts [Schneider 2013] to be a useful technique for realizing this concept since they are contracts that are specifically designed to be exchanged during runtime. ConSerts include concepts for defining the quality of the data to be exchanged, and they can be used to define the reactions to contract violations and discrepancies that will guide the change of behavior in the system.

At the contract level, the design decisions have been refined enough so that the CSG designer can define the final list of requirements in the form of verifiable contracts. This should be done in a more formal way to avoid misinterpretations by the CES developers.

Safety-Relevant Activities

In parallel to the design, the safety activities are performed:

At the business/domain level, the safety engineer should be able to perform the hazard and risk analysis for the CSG thanks to an initial list of system functions. This takes place at two levels:

At the CSG level, the consequences of the fail behavior of functions at CSG level must be investigated — for example, platoon deceleration. A lack in deceleration of a platoon can lead to a mass collision. This clearly has a higher severity than a single vehicle collision.

At the role level, the fail behavior of the subsystems in the collaboration, according to the initial function distribution among the roles, must be investigated with regard to its effect on the integrated system.

At the functional level, and given the specification of the safety goals and a first draft of the functional architecture, fault analysis can be performed in the form of a Component Fault Tree (CFT) [Domis and Trapp 2009]. This allows safety measures to be identified and the current design to be adapted to avoid or mitigate these failures. Safety measures are represented by safety requirements, which can be mapped directly to the collaboration roles. With a safety strategy in mind and the design that reflects it, it is then possible to create the functional safety concept for the CSG level.

At the contract level, and similar to the procedure for the architecture design, the safety requirements mentioned must be defined in terms of verifiable safety contracts.

8.4.2 Contracts Concept

As mentioned above, the CSG specification defines the functionality and behavior that the roles will take on in a collaboration. This is partly defined by functional and safety contracts. These contracts are considered as pure design-time contracts since they are exchanged and consumed only during the CSG-CES development time. On the other hand, ConSerts should be exchanged during runtime. In this approach, this means that ConSerts must also be developed and be

standardized as part of the CSG specification so that they can also be successfully exchanged and consumed at runtime.

In the context of the vehicle platoon use case:

- Functional contracts were primarily defined based on the state machine of each role. They define the behavior relevant for collaboration in a particular state.
- Safety contracts define the reaction to failure situations. Therefore, they mainly refer to the transitions in the state machine that connect normal states and failed operational states (including degraded states).
- ConSerts were engineered as an additional function of the system in close relation to the service contracts defined in the context of the service-oriented architecture.
- Service contracts define the specific messages exchanged between leader and follower. Therefore, ConSerts were defined in the form of guarantees of the safety-relevant data being exchanged.
- ConSerts are consumed according to the reference architecture for three purposes: to support flexibility, to allow valid CSG compositions, and to drive change of states.

Flexibility aspect: If demands and guarantees are met, the collaboration is allowed. Flexibility is supported because demands define a range in which guarantees can satisfy them. If the guarantees remain within this acceptable range, collaboration is allowed.

Valid compositions: A valid composition means that a demand is satisfied by a specific guarantee. If this is not the case, the collaboration should theoretically be terminated. We consider the validation of demands vs guarantees in two ways:

Contract violation: A violation is deemed to have occurred when the vehicle with demands can prove on its own that the service provider is not acting in accordance with its guarantees.

Contract discrepancy: A discrepancy arises when a demand cannot be satisfied by any guarantee.

Change of states: A contract violation is engineered in the platoon scenario such that when it occurs, the vehicle that detects the violation will preventively transition into a degraded mode for a certain time and notify the system causing the problem. In the event of a contract discrepancy, the collaboration with the provider is terminated. This will finally lead to the division of the platoon into sub-platoons.

8.4.3 Runtime Evaluation of Safety Contracts

A full detailed runtime analysis and safety assurance of all collaboration scenarios, including all environmental conditions, is not possible for real systems. Functional and safety contracts provide the means to operate on an adequate abstraction that has been prepared by diligent development time engineering. The use of safety contracts of different CESs requires the development of an environment capable of composing and evaluating these contracts at runtime. In the vehicle platoon use case investigated, safety contracts are used to define the reaction to failure situations, and safety guarantees are expressed as a means for tolerating deviation from a nominal behavior.

Simulative Approach for Validation of Safety Contracts

In order to validate the safety contracts designed and evaluate the behavior of the overall system when failures occur, a simulative approach can be used. Simulations and model-based evaluation of safety contracts are used during the development phase to observe the system behavior and validate the expectation of the safety engineer at design time. In the simulation, various manipulations, such as data corruption, invalid data due to a hardware defect, and other possible failures can be injected into the system [Isermann 2017]. An executable model of the collaborative embedded system should be created first as a means of validating the required safety functionalities.

Safety contracts separate requirements into assumptions and guarantees, which help to decrease the complexity of verifying the implementation against its specifications. Using a formal approach such as failure detection and isolation (failure handling) to do this allows the process of contract evaluation to be automated.

Case Study: Vehicle Platoon Example

The aim of the vehicle platoon use case is due to maintaining a short inter-vehicle distance. This would be achieved by exploiting real-time knowledge of the driving behavior of each vehicle in the platoon through onboard sensors and wireless communication among platoon members. If a sensor or communication failure occurs, or the respective safety guarantees become worse due to context changes, then the real-time knowledge would not be reliable, which puts the platoon in an unsafe mode. Therefore, both failures and changes in safety guarantees must be detected and compensated to keep the system working under any circumstance. Using a graceful degradation concept would help the system to remain operational (with a

degraded performance) in at least some such conditions. Note that the simulative approach used in the CrEST project is not executed in a fully realistic scenario due to effort limitations; instead, a highly simplified scenario has been used. The simulation model focuses on a platoon system that is already running, consisting of three vehicles running on one straight highway without any tunnels, curves, or inclines. In the simulation runs, one predefined safety contract is evaluated as an example. The results of the simulation are presented in [Figure 8-8](#) and [Figure 8-9](#). These figures show the system behavior in the event of a distance measurement sensor failure.

Fig. 8-8: Example: Generation and handling of errors with Simulink

Fig. 8-9: Example of error injection simulation results in a longitudinal platooning model using Simulink

As shown above, the failure injection block (in the left side) in [Figure 8-8](#) is implemented as a MATLAB function in Simulink and is located before the sensor inputs into the controller block. It can generate invalid sensor values at a specified time with the desired repetition rate of the error. Moreover, the failure detection and degradation

function validates the incoming data before it is passed on to the controller. Figure 8-9 shows the course of the platooning without applying error detection and degradation. Here, it becomes obvious that a sensor defect causes a deviation in the platooning distances because of the impacted controller performance. The third vehicle from Figure 8-8 is still following the previous vehicle because it is receiving correct speed data.

8.5 Conclusion

In this chapter, we presented a concept for safety certification of collaborative embedded systems. We highlighted the most distinct characteristics that distinguish them from classical systems. It is mainly their dynamicity that makes predicting their behavior difficult and therefore renders traditional safety certification techniques impractical. Based on these considerations, we presented new techniques and adaptations of existing techniques to enable a safety certification process that is specifically tailored to collaborative embedded systems.

We have outlined a two-step process. On the one hand, this process comprises the preliminary work during the design phase. All CESs are equipped with modular cases that contain an interface for integration with other safety cases. Since there are still many unknowns during the design phase, the second part of the safety certification process is performed at runtime, when all variables can be resolved. At runtime, the modular safety cases are integrated and evaluated according to the planned collaboration. Our concept comprises the monitoring of context changes at runtime and facilitates the handling of uncertainties. This enables a largely automated process that can be repeated efficiently during dynamic reconfigurations at runtime.

8.6 Literature

- [Bartocci et al. 2018] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, S. Sankaranarayanan: Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In: *Lecture Notes in Computer Science*, Springer, Cham. 2018, pp. 135-175.
- [Bell 2008] M. Bell: Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley Publishing, 2008.
- [Cámara et al. 2015] J. Cámara, G. Moreno, D. Garlan: Reasoning about Human Participation in Self-Adaptive Systems. In: 2015 IEEE/ACM 10th International

- Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE, 2015, pp. 146-156.
- [Damm et al. 2019] W. Damm, P. Heidl: Position paper on Safety, Security, and Certifiability of Future Man-Machine Systems, Results of the SafeTRANS Working Group “Resilient, Learning, and Evolutionary Systems”, <https://www.safetrans-de.org/en/activities/Roadmapping.php>, 2019.
- [Daun et al. 2016] M. Daun, J. Brings, T. Weyer, B. Tenbergen: Fostering Concurrent Engineering of Cyber-Physical Systems: A Proposal for an Ontological Context Framework. In: 2016 3rd International Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC), IEEE, 2016, pp. 5-10.
- [Domis and Trapp 2009] D. Domis, M. Trapp: Component-Based Abstraction in Fault Tree Analysis. In: International Conference on Computer Safety, Reliability and Security, Springer-Verlag, 2009, pp. 297-310.
- [European 2006] European Commission: Machinery Directive 2006/42/EC 2006. 2006.
- [Goodloe and Pike 2010] A. Goodloe, L. Pike: Monitoring Distributed Real-Time Systems: A Survey and Future Directions. 2010.
- [Halpern 2017] J. Y. Halpern: Reasoning about Uncertainty. MIT press, 2017.
- [Hildebrandt et al. 2019] C. Hildebrandt, T. Bandyszak, A. Petrovska, N. Laxman, E. Cioroai, S. Törsleff: EURECA: Epistemic Uncertainty Classification Scheme for Runtime Information Exchange in Collaborative System Groups. SICS Software-Intensive Cyber-Physical Systems, 34(4), (2019), pp. 177-190.
- [IEC 2005] International Electrotechnical Commission (IEC): IEC 62061: Safety of machinery — Functional Safety of Safety-Related Electrical, Electronic and Programmable Electronic Control Systems. 2005.
- [Isermann 2017] R. Isermann: Fahrerassistenzsysteme 2017: Von der Assistenz zum automatisierten Fahren — 3. Internationale ATZ-Fachtagung Automatisiertes Fahren. Springer-Verlag (available in German only).
- [ISO 2006] International Organization for Standardization (ISO): ISO 13849-1: Safety of Machinery — Safety-Related Parts of Control Systems – Part 1: General Principles for Design. 2006.
- [ISO 2010] International Organization for Standardization (ISO): ISO 12100: Safety of Machinery - General Principles for Design — Risk Assessment and Risk Reduction. 2010.
- [ISO 2018] International Organization for Standardization (ISO): ISO 26262-3: Road Vehicles — Functional Safety — Part 3: Concept Phase. 2018.
- [Jirkovský et al. 2016] V. Jirkovský, M. Obitko, V. Mařík: Understanding Data Heterogeneity in the Context of Cyber-Physical Systems Integration. In: IEEE Transactions on Industrial Informatics 13, no. 2, 2016, pp. 660-667.
- [Kelly 2007] T. Kelly: Using Software Architecture Techniques to Support the Modular Certification of Safety-Critical Systems. In: ACM International Conference Proceeding Series, Vol. 248, 2007, pp. 53-65.
- [Kephart and Chess 2003] J.O. Kephart, D.M. Chess: The Vision of Autonomic Computing. In: Computer 36, no. 1, 2003, pp. 41-50.
- [Koo, Rothbauer et al. 2019] C. H. Koo, S. Rothbauer, M. Vorderer, K. Höfig, M. Zeller: SQUADfps: Integrated Model-Based Machine Safety and Product Quality for

- Flexible Production Systems. In: International Symposium on Model-Based Safety and Assessment, Springer, 2019, pp. 222–236.
- [Koo, Vorderer et al. 2019] C. H. Koo, M. Vorderer, S. Schröck, J. Richter, A. Verl: Assistierte Risikobeurteilung für wandlungsfähige Plug-and-Produce Montagesysteme. In: VDI-Kongress Automation, 2019, pp. 41–54 (available in German only).
- [Koren et al. 1999] Y. Koren, U. Heisel, F. Jovane, T. Moriawaki, G. Pritschow, A. Galip Ulsoy, H. van. Brussel: Reconfigurable Manufacturing Systems. CIRP Annals, Vol. 48, 1999, pp. 527–540.
- [Negri et al. 2016] E. Negri, L. Fumagalli, M. Garetti, L. Tanca: Requirements and Languages for the Semantic Representation of Manufacturing Systems. In: Computers in Industry 81, 2018, pp. 55–66.
- [Perez-Palacin and Mirandola 2014] D. Perez-Palacin, R. Mirandola: Uncertainties in the Modeling of Self-Adaptive Systems: A Taxonomy and an Example of Availability Evaluation. In: Proceedings of the 5th ACM/SPEC international conference on Performance engineering, 2014, pp. 3–14.
- [Petrovska and Grigoleit 2018] A. Petrovska, F. Grigoleit: Towards Context Modeling for Dynamic Collaborative Embedded Systems in Open Context. In: MRC@ IJCAI, 2018, pp. 41–45.
- [Ramirez et al. 2012] A. J. Ramirez, A. C. Jensen, B. H. Cheng: A Taxonomy of Uncertainty for Dynamically Adaptive Systems. In: 2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), IEEE. 2012, pp. 99–108.
- [Schneider and Trapp 2013] D. Schneider, M. Trapp: Conditional Safety Certification of Open Adaptive Systems. In: ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2013.
- [Shafer 1976] G. Shafer: A mathematical theory of evidence (Vol. 42). Princeton university press, 1976.
- [Yilmaz and Davis 1987] O. S. Yilmaz, R. P. Davis: Flexible Manufacturing Systems: Characteristics and Assessment. In: Engineering Management International, Vol. 4, 1987, pp. 209–212.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.