Torsten Bandyszak, University of Duisburg-Essen
Lisa Jöckel, Fraunhofer IESE
Michael Kläs, Fraunhofer IESE
Sebastian Törsleff, Helmut Schmidt University Hamburg
Thorsten Weyer, University of Duisburg-Essen
Boris Wirtz, OFFIS e.V.

7

Handling Uncertainty in Collaborative Embedded Systems Engineering

As collaborative embedded systems operate autonomously in highly dynamic contexts, they must be able to handle uncertainties that can occur during operation. On the one hand, they must be able to handle uncertainties due to the imprecision of sensors and the behavior of data-driven components for perceiving and interpreting the context to enable decisions to be made during operation. On the other hand, uncertainties can emerge from the collaboration in a collaborative group, related to the exchange of information (e.g., context knowledge) between collaborative systems. This chapter presents methods for modeling uncertainty early in development and analyzing uncertainty during both design and operation. These methods allow for the identification of epistemic uncertainties that can occur when various, potentially heterogeneous systems are required to collaborate. The methods also enable graphical and formal modeling of uncertainties and their impact on system behavior (e.g., in the course of dynamic traffic scenarios). Furthermore, this chapter investigates the quality of outputs issued by data-driven models used to equip collaborative embedded systems with uncertainty-resilient machine learning capability.

7.1 Uncertainty in Collaborative Embedded Systems

Collaborative embedded systems (CESs) are often safety-critical, operate in dynamic contexts within a collaborative system group (CSG), and must be capable of reacting to unforeseen situations without human intervention. Uncertainty that can occur during operation should be considered systematically during engineering to enable CESs to cope with uncertainties autonomously. This section first introduces an ontology for understanding uncertainty in CESs. It then continues with an explanation of different kinds of uncertainty using the platooning example.

7.1.1 Conceptual Ontology for Handling Uncertainty

*Core concepts for
describing uncertainty*

In order to lay the foundation for understanding uncertainty that can occur during the operation of CESs, we have developed a lightweight ontology. This ontology defines core concepts for describing the different facets of uncertainty. It thereby provides the basis for seamless consideration of uncertainty in the engineering and operation of CESs. However, the ontological concepts are generic enough to describe different kinds of uncertainties, as we will see in Section 7.1.2. The ontology focuses on uncertainty that occurs during runtime, which differs from design-time uncertainty (e.g., ambiguous requirements) [Ramirez et al. 2012]. Nevertheless, the ontology is used during the design of CESs. [Figure 7-1](#) shows the ontology as a UML class diagram.

*Uncertainty related to
information*

As can be seen in [Figure 7-1](#), *uncertainty* is the central concept of our ontology. However, other concepts are necessary to fully account for uncertainty and its consequences. Uncertainties are always related to certain *information* (e.g., about the operational context in which a CES is operating) that stems from a specific *source* (e.g., a sensor or another CES). With regard to information, uncertainty means that it is not clear whether the information is valid as processed in a specific *scenario* during the operation of CESs. In that scenario, *information* is processed by an *agent* to make a decision or to perform some activity based on the information. Hence, an agent perceives the uncertainty related to specific information. In order to cope with uncertainty, it needs to be quantified to allow for automated uncertainty handling approaches to be applied — for example, using probabilistic or fuzzy approaches.

Fig. 7-1: Core ontology for describing uncertainty

In a *scenario*, *risks* may occur due to uncertainty. The term *risk* is used in a generic way here; it covers safety hazards but also, for example, economic risks. A risk can have a negative *impact*—for example, an accident—as well as a specific *likelihood* of occurrence. Uncertainty can be mitigated in two ways: 1) by mitigating the risk—that is, by reducing the likelihood or severity of an *impact*—or 2) by reducing the uncertainty that originally triggers the *risk*. Hence, the ontology includes the *risk mitigation* and the *uncertainty mitigation* concepts.

An uncertainty also always has at least one *cause* — that is, a reason why the uncertainty occurs. We can use the *type* concept to distinguish between different causes of uncertainty. We decided to relate the type of an uncertainty to its cause and not to the uncertainty itself, since categorizing uncertainties often relates to abstract sources of uncertainty, such as imprecision of sensors (cf., e.g., [Ramirez et al. 2012]), failures of communication networks, or insufficient trust in other agents (cf., e.g., [Yu et al. 2013]). Section 7.1.2 illustrates such types of uncertainty using the example of autonomous vehicles.

Relationship between uncertainty and risk

Cause and types of uncertainty

7.1.2 Different Kinds of Uncertainty

In this section, we consider the use case of “Cooperative Vehicle Automation,” which enables vehicles to form platoons on highways in order to reduce fuel consumption and increase traffic throughput (cf. [Jia et al. 2016]). As explained in Chapter 1, there are specific

Uncertainty in “Cooperative Vehicle Automation”

uncertainty-related challenges that emerge from this use case. In the following, we consider these challenges in more detail in order to illustrate different kinds of uncertainty that can occur during the operation of CESs. Specifically, we place particular emphasis on uncertainties rooted in the operational context of CESs, which is highly dynamic. The operational context comprises other CESs (e.g., other cars equipped with cooperative ACC (CACC) systems, which are engaged in or able to join a platoon) as well as non-collaborative context objects (e.g., pedestrians or vehicles steered manually).

System under consideration and context objects

An exemplary situation is depicted in Figure 7-2. It shows a platoon consisting of a certain number of vehicles (three of which are shown), as well as another non-collaborative vehicle driving ahead of the platoon leader (labeled “LV” in Figure 7-2). Furthermore, the car labeled “JV” attempts to join the platoon, which requires message exchange and coordination with the platoon leader. In the subsequent sections, we make use of this example to illustrate various concepts. Where applicable, we distinguish between the system under consideration (SUC) and its context objects (COs). For instance, in Figure 7-2, the platoon leader could be the SUC, whereas the messages it receives from other cars, as well as the other cars themselves and surrounding objects (e.g., road signs), are considered COs.

Fig. 7-2: Uncertainty in the open and dynamic context of CESs

Uncertainty relates to perception, data quality, and communication

There are three different sources of uncertainty under consideration:

1. *Perception-related uncertainty:* Due to the inherent imprecision of sensor devices, information representing the current context situation of a system may be invalid. CESs also share context information in order to jointly mitigate uncertainties. Each vehicle is equipped with sensory devices — such as a radar sensor or cameras for perceiving obstacles, other traffic participants, and

traffic signs. For example, an onboard camera sensor may be affected by dirt on the camera lens, resulting in uncertainty regarding the context information. Furthermore, the timing of events related to the context perception may cause uncertainties (e.g., caused by time delays in the transmission of sensor signals, see Section 7.2.2).

2. *Data-related uncertainty*: This kind of uncertainty is caused by limitations in the results of data-driven components (DDCs). These DDCs extract context knowledge from information sources (e.g., sensors). This context knowledge can then be processed further by some application logic (in this case, a CACC system). For instance, traffic signs detected by onboard cameras are recognized by embedded DDCs that contain artificial intelligence (AI) models that process camera images (see Section 7.3.2).
3. *Epistemic runtime uncertainty*: CESs exchange context information in order to reason about the current context situations they face. Uncertainty can be caused by epistemic concerns pertaining to the representation of the information that is exchanged and processed by CESs (see Section 7.3.1). For instance, context information about the goals of platoon members may be subject to ambiguity or incompleteness.

Since these uncertainties have different causes (cf. Section 7.1.1), uncertainty is a multi-faceted and cross-cutting concern that requires a multitude of methods and techniques to enable CESs to handle such uncertainty autonomously during operation. The following sections present different methods for modeling and analyzing uncertainty that are complementary in our comprehensive uncertainty handling framework. Each method aims at handling a specific kind of uncertainty listed above. Furthermore, the different methods target both the development time (e.g., modeling methods) as well as runtime methods (e.g., wrapper components enabling the quantification of a DDC's uncertainty).

*Complementary
methods for handling
uncertainty*

7.2 Modeling Uncertainty

In the following, we present approaches for modeling uncertainty, including a language for modeling early uncertainty information and relating it to other artifacts and a domain-specific method for capturing behavioral uncertainty in traffic scenarios.

7.2.1 Orthogonal Uncertainty Modeling

In the engineering of CESs, uncertainties that may occur during operation must be considered systematically. In the early stages of requirements engineering in particular, potential runtime uncertainties must be identified and documented so that stakeholders can analyze and consider them appropriately. The ultimate goal of considering such uncertainty during engineering is to design CESs that are able to safely handle uncertainties during operation.

*Orthogonality of
uncertainty*

Depending on project-specific demands, various different artifact types are used in a complementary manner to model a CES under development and other relevant information, such as the operational context. Typical modeling perspectives used in requirements engineering distinguish structural, functional, and behavioral aspects of the CES and its context. Uncertainties are not limited to a single artifact but rather affect different aspects modeled in several artifacts. For example, perception-related uncertainty due to the failure of a sensor (see Section 7.1.2) is reflected in all three modeling perspectives: the interface between the CES and the sensor device—where the uncertainty appears in the form of missing or corrupted data—is captured in structural models; sensor data is processed by CES functions modeled in the functional perspective; and finally, sensor failure may result in certain (mis-)behavior, where uncertainty can be detected, and uncertainty may affect the CES behavior.

*Difficulties in capturing
uncertainty information*

Hence, uncertainty can be considered a cross-cutting concern, similar to variability (cf., e.g., [Bachmann et al. 2003], Chapter 18). Describing uncertainty by attaching uncertainty information to model elements across different artifacts (e.g., denoting sensor data as the information subject to uncertainty) has two essential drawbacks:

1. The information necessary to specify uncertainty is documented redundantly at multiple locations attached to model elements. For instance, sensor data may be modeled in structural models and as input for functions in functional models.
2. Uncertainty information is spread across several models. This makes it difficult to structure uncertainty information and trace essential parts of an uncertainty description (e.g., the cause of uncertainty) throughout the various artifacts. Furthermore, it is difficult to capture relationships between different uncertainties.

*Orthogonal uncertainty
models*

In the following, we present a modeling technique [Bandyszak et al. 2018b], [Bandyszak et al. 2020] to support the systematic identification and documentation of runtime uncertainties during the

entire engineering process. The core idea is to capture information that describes uncertainty (see Section 7.1.1) in a separate artifact and connect this information to other engineering artifacts (called “base artifacts”) through trace links. These first-class uncertainty artifacts created using a dedicated graphical modeling language are called “*orthogonal uncertainty models*” (OUMs). OUMs provide a central artifact for analyzing uncertainties and how they affect and must be accounted for during engineering.

Modeling Concepts and Notation

OUMs employ a specific perspective on the operation of an SUC. The modeling concepts and the corresponding graphical notation are introduced in Table 7-3. As we can see, the core concepts for describing uncertainty are captured using UML-like stereotype notation and specific iconic representations to foster understanding of the uncertainty information modeled. The visual notation comprises node types and connector elements for modeling uncertainty relationships within the OUM, as well as establishing relationships to other artifacts.

Graphical modeling language

In order to model uncertainty from the dedicated perspective of a CES during operation (i.e., the “agent” under consideration), we define some specialized extensions to the ontology defined in Section 7.1.1. While the actual information subject to uncertainty is captured in the base artifacts, the OUM uses the *observation point* concept to specify where uncertainty becomes visible. Furthermore, the cause of an uncertainty is subdivided into its *rationale* (i.e., the reason why uncertainty occurs) and *activation condition* elements. Rationales and activation conditions can be combined in a tree-like structure using conjunction/disjunction nodes to describe the cause of uncertainty.

Modeling uncertainty observation and cause

The possible effects of an uncertainty (i.e., risks and impacts of uncertainty, see Section 7.1.1) are captured using *effect* links that point to base artifact elements — for example, system functions affected by uncertainty. In contrast, *uncertainty mitigations* are modeled in the base artifacts (e.g., specific functions that cope with uncertainty) and related to the uncertainty modeled in OUMs. The OUM also allows relationships between different uncertainties to be modeled, including causal and amplification relationships (see Table 7-3).

Modeling uncertainty relationships

Modeling element	Explanation	Visual notation
Uncertainty	This element is used to capture the uncertainty itself and identify it, which allows related uncertainty information (further specified by the other model elements) to be grouped and also allows distinction between different uncertainties.
Uncertainty relation	This connection node visualizes trace relationships of the concepts outlined above by means of an n-ary association.
Uncertainty element connector	This edge type establishes relationships between <i>uncertainty</i> and <i>observation point</i> elements as well as uncertainty causes (causal combinations of <i>rationale</i> and <i>activation condition</i> elements).
Rationale	This element describes the root cause of the uncertainty — i.e., the source that originally introduces uncertainty.
Activation condition	This element describes the condition under which an uncertainty can become active during operation.
Uncertainty cause conjunction/disjunction	The cause of an uncertainty can be considered a causal (AND/OR) combination of a set of <i>uncertainty rationales</i> and a set of <i>activation conditions</i> . These manifest in a binary tree with two kinds of control nodes.
Observation point	This element documents the artifacts in which an uncertainty is present and where the system can detect the said uncertainty.
Uncertainty relations	These relationships document situations where one uncertainty can cause or amplify another uncertainty (i.e., when active, it may sustain effects and thereby complicate the mitigation of the amplified uncertainty).
Trace relation	This relationship is used to trace uncertainty information to base artifact elements.
Effect relation	This relationship refers to the effect an uncertainty has on either the SUC or its context.
Mitigation relation	This relationship, pointing from a base artifact element to a corresponding OUM uncertainty element, captures possible mitigations that either counteract uncertainty once it has occurred (i.e., to reduce possible negative effects), or prevent uncertainty from becoming active.

Tab. 7-3: OUM concepts and notation [Bandyszak et al. 2020]

Example

Figure 7–3 shows an example OUM (middle part) as well as some exemplary base artifacts (upper and lower part). The latter include a structural context model, a functional model, and a behavioral model of the exemplary CACC system. To reduce complexity, only excerpts related to the joining maneuver are depicted. Moreover, a sequence

diagram shows interactions between two vehicles—each equipped with a CACC—in the course of the joining maneuver.

Fig. 7–4: Orthogonal uncertainty model example

The exemplary OUM specifies two uncertainties. First, there can be uncertainty related to inconsistent context information. In this example, the context is perceived in the form of camera data processed by a DDC (cf. Section 7.3.2), as well as by context information communicated by other vehicles, because in the joining maneuver, the leader vehicle informs the joining vehicle about the gap to be taken. The uncertainty of inconsistent information occurs when the gap information obtained by the DDC (e.g., indicating an obstacle

Inconsistent information from different sources

in the gap) differs from the information communicated by the platoon leader (e.g., gap is free). The rationale of this uncertainty could be a failure of the camera sensor or misbehavior of the DDC that interprets the camera data. A potential effect may be the inability to safely complete the joining maneuver. This uncertainty is described further in [Bandyszak et al. 2018a].

Uncertainty in inter-vehicle communication

Another uncertainty modeled in this exemplary OUM relates to uninterpretable messages from other vehicles. In [Figure 7-4](#), this message exchange is illustrated in a sequence diagram. This uncertainty may be caused by technical communication failures of the V2V network, or inconsistent ontologies underlying the representation of the data that is exchanged (cf. Section 7.3.1). For example, during the join maneuver, goals (e.g., target destination and driving preferences) must be exchanged between the vehicle that requested to join a platoon and the platoon leader. However, when inconsistent ontologies are used to specify goals (e.g., considering different concepts for electric vehicles), the goals of the joining vehicle cannot be interpreted, which in turn affects the corresponding CACC function.

7.2.2 Modeling Uncertainty in Traffic Scenarios

When designing a CES, it is important to estimate the impact of uncertainties (as early as possible) in order to be able to make qualified design decisions. Therefore, semi-formal uncertainty modeling techniques (such as the OUM presented in Section 7.2.1), which are especially helpful for understanding uncertainty in early design phases, must be complemented with formal modeling approaches, which allow quantification of the possible influence on CSG behavior throughout the design process.

Formal modeling of uncertainties

For this aim, uncertainties must be modeled mathematically—for example, as random variable or probabilistic distribution—and the effects of uncertainties on behavior must be identified and modeled, resulting in probabilistic CES/CSG behavior models. Probabilistic behavior models can be used as a base for mathematical analysis—for example, by probabilistic model checking or stochastic analysis (e.g., by using repeated simulation).

Assessing uncertainty impact according to defined criteria

Within the mathematical analysis, the probabilistic behavior model is assessed with regard to an evaluation criterion within the context of a scenario. Typical evaluation criteria are risk or quality of service. Mathematical assessment of the impact of uncertainties can help to guide the design process in many areas, including:

1. Estimating the kind of mitigation required for uncertainties (or the degree of weakening the influence of uncertainties, if they cannot be completely mitigated)
2. Developing and choosing more uncertainty resilient implementations and strategies
3. Assessing the fulfilment of safety requirements (especially if homologation is required)

Summarizing the above, for behavioral modeling of uncertainties, a behavior model is extended to a probabilistic or stochastic behavior model. For assessment, a scenario model and an evaluation criterion are also required. These model components are described in more detail in the following. We start with scenario modeling, which is necessary to describe the situation for which the assessment is performed.

*Scenario-based
behavioral modeling*

Modeling Traffic Scenarios for CSGs

Modeling a traffic scenario with the participation of CSGs requires a specification language with the ability to express dynamically changing relationships (including spatial relationships) between a variable number of CESs involved in a CSG.

Traffic sequence charts (TSCs) [Damm et al. 2017], [Damm et al. 2018] are one such formalism that allow intuitive and concise graphical specification of dynamic constraints based on formal semantics. TSCs (which—with a different focus—are also described in Chapter 6) are based on acyclic graphs of chart nodes. Chart nodes capture constraints over a time interval and can be combined into a chart using sequence, choice, or parallel composition. In a TSC, the constrained time intervals are seamless — that is, there is no time gap within a sequence of two nodes.

*Traffic sequence charts
(TSCs)*

Chart nodes include simple invariants (constraints that hold throughout the complete constrained interval) and conditions (constraints holding at least once), as well as complex nodes that specify communication/event patterns or contain complete charts. Spatial views describe spatial constraints between objects, represented by symbols in a topological view. Spatial views can be used as base constraints for invariant and condition nodes.

For example, the TSC excerpt in [Figure 7-5](#) describes a scenario in which the car with index $n+1$ joins a platoon consisting of the vehicles with index 1 (platoon head) to n (platoon tail) driving at speed v . The TSC consists of four chart nodes: the first chart node is an invariant

TSC example

that contains a spatial view and describes the phase where the approaching car is still far away from the platoon (which is represented by an abstract symbol, see Chapter 6 for more detail). Phase 2 is described by two nodes holding in parallel: the spatial invariant expresses that the approaching car is closer to the platoon but has not yet reached the distance for initiating a join maneuver. During phase 2, the approaching car sends a join request to the last car of the platoon, which eventually answers granting the right to join. In phase 3, the approaching car actually joins the platoon, and then phase 4 is reached, in which the car is part of the platoon.

Fig. 7–5: TSC excerpt describing a car joining a platoon

TSCs are interpreted with respect to a modular world model, which defines object classes, interfaces, and behaviors. World model modules are exchangeable, as long as suitable interfaces are provided. Consequently, behavior modules can be specified using different specification languages (with suitable interfaces) — for example, hybrid automata or stochastic/probabilistic hybrid automata. This flexibility can be used for combining the same scenario specification with different CES implementation variants or CESs whose granularity has been refined during the development process.

World models

Behavioral Uncertainty Modeling

As a first step for behavioral uncertainty description, the occurrences of uncertainties must be identified and modeled mathematically. For behavioral modeling, it is mostly runtime uncertainties—which influence the behavior of the model—that are relevant. These uncertainties can be separated into two classes, depending on the timing of interactions. *Discrete time uncertainties* deal with discrete events, such as sending and receiving messages, or sensors resulting in discrete signals (see Section 7.1.2). *Continuous time uncertainties* deal with continuous interactions, such as following a trajectory or sensors continuously reporting the location of objects (at least the start and end of object detection are discrete events). Both classes include uncertainties due to time delays and data distortion, but only discrete time events can be completely missed.

Considering uncertainty related to timing

Discrete time event misses can be modeled mathematically by a random variable connected with a probability. Data distortions occur with a certain probability distribution. A discrete time delay can be modeled by a random variable with a probability distribution. Time delays for continuous uncertainties may involve time compaction or dilation and are a little more complicated, but can still be covered by a (more complex) probability distribution.

Discrete time and continuous time uncertainties

These mathematical uncertainty models can be embedded in hybrid automata behavior models, resulting in probabilistic hybrid automata incorporating uncertainty behavior.

Risk Assessment

The last missing building block describes the use of the probabilistic behavioral CES/CSG model involving uncertainty. There are two main analysis methods: *probabilistic model checking* [Katoen 2016] can compute the probability with which an evaluation criterion (such as a criticality or a quality of service measure) is reached. Probabilistic

Probabilistic and stochastic model checking

model checking is mathematically more stable but is more applicable for small systems due to its complexity. *Stochastic model checking* [Kwiatkowska et al. 2007] (which is done by performing many simulation runs) is not as exact, but the performance is less challenging. During simulation, the probabilistic elements of the probabilistic behavior model must be instantiated according to the associated probability distribution. The resulting simulation runs are then assessed with respect to the evaluation criterion. Similar to probabilistic model checking, the results can be used as a quality measure for the behavior modeled.

7.3 Analyzing Uncertainty

In addition to modeling uncertainty semi-formally and formally, more specific analysis methods are necessary to fully account for the various kinds of uncertainty (see Section 7.1.2). Uncertainty analysis methods cover, among other things, guidance for identifying potential uncertainty sources as well as eliciting more specific information for (automatically) estimating uncertainty. In the following, we present a classification scheme that helps identify epistemic uncertainties, as well as an approach to provide situation-aware estimates of the uncertainty in an outcome of a DDC during operation of CESs.

7.3.1 Identifying Epistemic Uncertainties

This section presents a classification scheme that facilitates the identification and classification of epistemic uncertainties resulting from information exchange in CSGs (see Section 7.1.2). For this purpose, we first describe the different types of epistemic uncertainties that may occur in this context. We then present the epistemic uncertainty classification scheme for runtime information exchange (EURECA) and its application during requirements engineering. For an in-depth presentation of the approach, including formal modeling of knowledge and epistemic uncertainties (as well mitigation strategies, etc.), please refer to [Hildebrandt et al. 2019].

*Epistemic uncertainties
in platooning*

In this section, we use an example in which the SUC is a platoon leading vehicle that runs on a combustion engine and the CO is an electric vehicle that wants to join the platoon (see Section 7.1.2). As epistemic uncertainties result only from a lack of understanding regarding a message the SUC *receives*, all examples concern messages sent from the CO to the SUC. As we show in the following, epistemic

uncertainties can appear at the type level and at the instance level. Uncertainty sources at the type level are rooted in the terminological knowledge utilized by CEs to specify messages. In simple terms, terminological knowledge is comparable to a vocabulary that includes the relationships between vocabulary items and is from here on referred to as the TBox [Krotzsch et al. 2014]. At each level—that is, type and instance level—four different classes of uncertainties can be distinguished. Figure 7–6 illustrates a subset of these classes (T1, T3, I1, and I2). As we can see, the set difference of the SUC TBox (upper ellipse at the top of Figure 7–6) and the CO TBox (lower ellipse) is not empty, resulting in various uncertainties. The uncertainty classes illustrated and additional classes are detailed in the following.

Uncertainty Sources at the Type Level

There are four classes of uncertainties that may occur at the type level. All of these uncertainties result from a mismatch between the TBox of the SUC and the CO used for specifying and interpreting the messages to be exchanged at runtime.

Fig. 7–6: Epistemic uncertainties at type and instance level

The first class of type-level uncertainties results from a *known difference in scope* (T1) between the TBox of the SUC and the CO. This uncertainty occurs whenever the CO sends the SUC a message that includes a TBox element that is not included in the SUC’s TBox. The difference in scope is *known* insofar as the *unknown* TBox element has a *known* relationship to a *known* TBox element, which may allow for

Known difference in scope

some kind of mitigation at runtime. For instance, if the message contains an element “MinimizeEnergyConsumption” with the relationship “hasSupportingGoal” connecting it to the element “IncreaseDownhillRecuperation,” the SUC may understand the first two elements but not the last one since it is not an electric vehicle.

*Unknown difference in
scope*

The second class occurs whenever there is an *unknown difference in scope* (T2). In this case, the message contains an *unknown* TBox element that has an *unknown* relationship to a *known* TBox element. This makes mitigation at runtime more challenging. For instance, if the message from the CO contains an element “DesiredStateAtDestination” with the relationship “hasStateOfCharge” to the element “StateOfCharge,” the SUC may understand the first element but not the second and third as these are concepts specific to an electric vehicle.

*Distinct ontologies
scope*

The third class of type-level uncertainties results from a *distinct scope* (T3). In this case, the *unknown* TBox element has no attachment to the SUC’s TBox whatsoever and mitigation is barely possible. For instance, a message from the CO containing the goal “ReachNextChargingStation” might not be understood by the SUC at all because it has no relationship to any known concept.

*Inconsistent ontological
commitment*

Uncertainties of the fourth class occur whenever a message indicates an *inconsistent ontological commitment* (T4). For instance, if the CO informs the SUC about its consumption in the measurement unit kWh, whereas the SUC measures its consumption in liters, this may lead to uncertainty even if the SUC’s TBox contains the elements kWh and consumption.

Uncertainty Sources at the Instance Level

While the previously described type-level uncertainties result from terminological differences, the following uncertainties pertain to the actual information contained in a message. The first class, *semantically inconsistent information* (I1), occurs when a message contains a value that violates the semantic definition of a relationship that it refers to. For example, the SUC is informed by the CO about its “MaximumPowerConsumption” with regard to the goal “MinimizeEnergyConsumption.” Even if the SUC understands the former concept in a different context (such as its power outlet), this message will result in uncertainty as the SUC running a combustion engine would consider only “MaximumFuelConsumption” a valid concept in this context.

*Situationally incomplete
information*

Situationally incomplete information (I2) as a source of uncertainty occurs whenever a message contains a set of statements that does not satisfy the requirements of the situation at hand. For instance,

consider a CO that wants to join a platoon and the SUC as the platoon leader requires the information “MaximumFuelConsumption.” However, if the CO provides only the “Destination” in its message, the SUC will perceive the message as incomplete.

The third class of instance-level uncertainties results from *situationally inconsistent information* (I3), which occurs whenever the content of a message is inconsistent with regard to the information expected by the SUC for the situation at hand. Consider a platoon where the vehicles regularly broadcast their range in kilometers. If the platoon now descends a steep road, the virtually constant braking might actually increase the reported range of the CO (the electric vehicle). The SUC, however, might expect a range decrease over time and thus considers the reported range to be *situationally inconsistent*.

*Situationally
inconsistent information*

The fourth class of instance-level uncertainties results from *missing type membership* (I4). This class of uncertainty occurs when a message contains an information item that lacks a type membership. For instance, the SUC might receive geographical coordinates from a CO but no information on what these coordinates refer to.

*Missing type
membership*

EURECA

In order to systematically analyze and capture the previously described uncertainties at the instance and type level, we developed an epistemic uncertainty classification scheme for runtime information exchange (EURECA). The two-dimensional scheme is depicted in Table 7-7.

SUC ontology	Ontology element	Type-level uncertainty				Instance-level uncertainty			
		T1	T2	T2	T4	I1	I2	I3	I4
Goal ontology	hasSupportingGoal	x							
	DesiredStateAtDestination		x						
	ReachNextChargingStation			x					
	hasConsumption				x				
	MaximumPowerConsumption					x			
	MaximumFuelConsumption						x		
	Range							x	
	GeographicalCoordinates								x

Tab. 7-7: EURECA applied to the platooning example

The first column is populated with the SUC ontology that the uncertainties captured relate to. In our examples, we considered only the SUC’s goal ontology, but other ontologies obviously might be subject to epistemic uncertainties as well. The second column lists the concrete ontology elements that are subject to epistemic uncertainty according to the analysis performed. A checkmark indicates which

*Two-dimensional
classification scheme*

specific epistemic uncertainty has been identified for an ontology element. Note that in our example, each ontology element is subject to exactly one type of epistemic uncertainty; however, the elements could also be subject to multiple types. Instead of checkmarks, we could also indicate the specified communication scenario in which an element is subject to uncertainty (cf. [Hildebrandt et al. 2019]). This approach is generally recommended due to improved traceability when new scenarios are added or scenarios are changed during subsequent requirements engineering iterations — for example, when an agile development approach is applied.

The epistemic uncertainty classes identified are rooted in an analysis of the underlying ontological foundations of knowledge exchange and are not the result of domain-specific considerations. Hence, while evaluation has been performed only for the automotive domain, we are confident that EURECA can also be applied to other domains — for example, distributed energy production and adaptive factories.

7.3.2 Assessing Data-Driven Uncertainties

Components with data-driven models [Solomatine and Ostfeld 2008], such as those obtained when applying AI and machine learning methods, are becoming increasingly important for complex software-intensive systems. In particular, CESs intended to collaborate in an open context have to process various kinds of sensor inputs to recognize and interpret their situation in order to handle changes in their environment and interact with previously unknown agents. Unlike traditionally engineered components, which are developed by software engineers who define their functional behavior using code or models, the functional behavior of a DDC is learned automatically from gathered data through an algorithm.

*Assuring correct
behavior of a DDC*

As a consequence, the functional behavior expected from a DDC can be specified only in part on its intended application domain, usually based on a number of example cases for which data was previously collected and augmented with additional information. Consequently, we cannot provide assurance that such a component will behave as intended in all cases [Klås 2018], resulting in data-related uncertainty (see Section 7.1.2).

Example situation

In our example, the current speed limits have to be considered when a vehicle wants to join an existing platoon (cf. [Figure 7–2](#)). One *information source* for current speed limits can be a DDC that provides traffic sign recognition (TSR) based on data of the camera sensor in

the CESs of interest (as shown in [Figure 7-3](#)). At the latest since 2012, deep convolutional neural networks have proven their superior performance for this kind of task and can be considered as a state-of-the-art approach for TSR [Krizhevsky et al. 2017]. Nevertheless, uncertainty remains inherent in the outcome of our TSR component since we cannot specify for all possible combinations of pixel values within an image what kind of traffic sign should be reported.

Because the use of DDCs is an important source of uncertainty in CESs, the uncertainty they introduce must be appropriately understood and managed not only during *design time* but also during *operation*. In the following, we first present a classification for the different sources of uncertainty relevant when applying a DDC, and then introduce "*uncertainty wrappers*" as a means of quantifying and analyzing the level of uncertainty for any specific situation at runtime.

Considering both design and runtime of DDCs

Three Types of Uncertainty Sources

The sources of uncertainty in DDCs can be separated into three major *types*: uncertainties caused by limitations in terms of model fit, data quality, and scope compliance [Klås and Vollmer 2018]. Whereas *model fit* focuses on the inherent uncertainty in data-driven models, *data quality* covers the additional uncertainty caused by their application to input data obtained in suboptimal conditions, and *scope compliance* covers situations where the model is probably applied outside the scope for which the model was trained and validated.

In our example, limitations in model fit may be caused by restrictions in the number of model parameters, input variables considered, and data points available to train the model. Limitations in data quality may be caused by weather conditions, such as rain, natural and artificial backlight, a dirty camera lens, and other factors that make it more difficult for the TSR component to reliably recognize the correct traffic sign on the given image. Finally, limitations in scope compliance occur when a model is used outside its target application scope (TAS). For example, if our TSR component was intended to be used for passenger cars in Germany, its application in a different country will make its outcomes questionable because the component was most likely built and validated only for German traffic signs.

Examples for uncertainty sources

Managing Uncertainty during Operation

The *uncertainty wrapper* approach was developed to better deal with uncertainty inherent in the outcomes of data-driven models during

Characteristics of
uncertainty wrappers

operation. It supports this purpose by encapsulating the data-driven model to enrich the model outcomes with dependable and situation-aware estimates of uncertainty. This allows the CES that processes these outcomes to make more informed and dependable decisions.

The approach is *holistic* in the sense that it addresses uncertainty caused by limitations in model fit, data quality, and scope compliance. Moreover, it is *generic* in the sense that it is model agnostic — that is, it states no requirements on the data-driven model it encapsulates. Specifically, it does not require that the existing model be adapted nor that the model provides specific kinds of outcomes.

The uncertainty wrapper approach consists of a selection of concepts that extend and refine the uncertainty ontology introduced in Section 8.1, equations that allow quantification and aggregation of uncertainties from different types of sources [Klås and Sembach 2019], an architectural design proposal, and a tooling framework for building and validating uncertainty wrappers [Klås and Jöckel 2020].

Uncertainty Wrapper – Architecture and Application

In the following, we introduce the most prominent elements of the uncertainty wrapper architecture and relate them. Moreover, we illustrate them on a simplified application example in the context of the previously introduced example case.

Extension of DDC
outcomes

The wrapper encapsulates the existing data-driven model and extends its *outcome* with *dependable uncertainty estimates* (Figure 7–8).

Fig. 7-8: Uncertainty wrapper architecture and example applications

Example for
application

In our example, the outcome of the model could be the information about whether the data-driven model has recognized a “speed limit

50” sign or not (cf. Figure 7–2). The uncertainty is then expressed by the likelihood that the outcome provided is not correct.

Besides the data processed by the data-driven model, the *data-driven model input* may contain further data that is used by the wrapper to assess the degree of uncertainty in the model outcome. For example, the GPS signal may be used to determine whether the vehicle is still in Germany, a task which is conducted by the *scope model*. The result is then provided as a scope factor evaluation result to the *scope compliance model*, which calculates the likelihood of scope incompliance considering the results for all scope factors.

Further data to assess scope compliance

Moreover, the rain sensor signal may be also used as an input. Based on this signal, the *quality model* determines the level of precipitation, which is anticipated to have an influence on the recognition performance of the data-driven model. Together with the results of other quality factors, the *quality impact model* then determines a situation-aware uncertainty estimate using a decision tree model as a white box approach.

Quality and quality impact model

Finally, the uncertainty information provided by the scope compliance and quality impact model are combined to give a dependable uncertainty estimate that considers the requested level of *confidence*.

In Figure 7–8, we illustrate this for three cases. In Case A, we get an extremely high uncertainty because the DDC is used in New York, which is outside the TAS. In Case B, we obtain low uncertainty since the component is used in its TAS under good quality conditions. In Case C, we would end up with a moderate uncertainty since the rain makes the recognition task more difficult.

Examples for application results

Uncertainty Wrappers – Limitations and Advantages

In order to build an uncertainty wrapper, in addition to the existing data-driven model, we need data for a representative set of cases from the TAS, most preferably augmented with labels that indicate quality deficits. Alternatively, quality deficit augmentation frameworks such as the one presented in [Jöckel and Kläs 2019] may be applied.

In comparison to a more traditional approach, where uncertainty is estimated by the data-driven model itself, the use of uncertainty wrappers provides advantages in practice. Its uncertainty estimates are unbiased since it does not rely on data used during the training of the data-driven model. Moreover, the uncertainty assessment is broader because it considers more types of uncertainty sources, including the commonly neglected scope compliance, and also information that is relevant for uncertainty but not for the primary

Unbiased and broader uncertainty assessment

objective of the data-driven model. Finally, uncertainty wrappers can simplify safety assessments of data-driven models by not only providing statistically safeguarded uncertainty estimates at a requested confidence level, but also promising more comprehensible evaluations by domain and safety experts since they decouple the uncertainty analysis from the “black box” as which most data-driven models are still considered.

7.4 Conclusion

CESs operate in highly dynamic contexts and thus have to cope with uncertainties during operation. This uncertainty cannot be fully anticipated during design, since CESs are increasingly autonomous and adaptive, and CSGs may take various forms. Nevertheless, it is crucial to systematically consider potential runtime uncertainties during the engineering of CESs. This requires methods for identifying, modeling, and analyzing uncertainty to develop CESs capable of coping with uncertainty during operation autonomously.

Uncertainty ontology

This chapter approached the complex task of uncertainty handling during CES engineering by first conceptualizing uncertainty. To this end, we presented an uncertainty ontology that defines core concepts to describe uncertainty. Among other aspects, this ontology provides a means of describing and understanding causes of uncertainty and relating uncertainty to information gathered through certain sources (e.g., sensors) and processed by CESs.

*Versatile modeling and
analysis methods*

Based on the ontology and a characterization of different kinds of uncertainty relevant for CESs and CSGs, we presented methods for modeling uncertainty, and for analyzing uncertainty and its effects during both design and operation. To model uncertainty graphically, especially during early phases, we presented a modeling language for specifying uncertainty in dedicated artifacts — that is, orthogonal uncertainty models. As a more formal approach, we presented a behavioral modeling approach based on traffic sequence charts for analyzing risks in dynamic traffic scenarios. As part of the analysis methods, we presented a classification scheme for identifying epistemic uncertainties in the information exchange among CESs. Furthermore, we presented an analysis method to support the safe operation of DDCs by equipping them with a wrapper that provides reliable information on situation-specific uncertainty.

7.5 Literature

- [Bachmann et al. 2003] F. Bachmann, M. Goedicke, J. Leite, R. Nord, K. Pohl, B. Ramesh, A. Vilbig: A Meta-model for Representing Variability in Product Family Development. In: *Software Product-Family Engineering*, LNCS Vol. 3014, Springer, 2003, pp. 66–80.
- [Bandyszak et al. 2018a] T. Bandyszak, P. Kuhs, J. Kleinblotekamp, M. Daun: On the Use of Orthogonal Context Uncertainty Models in the Engineering of Collaborative Embedded Systems. In: *Joint Proceedings of the Workshops at Modellierung 2018*. CEUR Vol. 2060, 2018. pp. 121–130.
- [Bandyszak et al. 2018b] T. Bandyszak, M. Daun, B. Tenbergen, T. Weyer: Model-Based Documentation of Context Uncertainty for Cyber-Physical Systems. In: *Proc. IEEE 14th Int. Conference on Automation Science and Engineering (CASE)*. IEEE, 2018, pp. 1087–1092.
- [Bandyszak et al. 2020] T. Bandyszak, M. Daun, B. Tenbergen, P. Kuhs, S. Wolf, T. Weyer: Orthogonal Uncertainty Modeling in the Engineering of Cyber-Physical Systems. In: *IEEE Transactions on Automation Science and Engineering*, Vol. 17, No. 3, 2020, pp. 1250–1265.
- [Damm et al. 2017] W. Damm, S. Kemper, E. Möhlmann, T. Peikenkamp, A. Rakow: Traffic Sequence Charts – From Visualization to Semantics. Technical Report 14 AVACS, No. 117, 2017.
- [Damm et al. 2018] W. Damm, E. Möhlmann, T. Peikenkamp, A. Rakow: A Formal Semantics for Traffic Sequence Charts. In: *Principles of Modeling – Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday*, LNCS, Vol. 10760, 2018, pp. 182–205.
- [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*, Vol. 34, No. 4, 2019, pp. 177–190.
- [Jia et al. 2016] D. Jia, K. Lu, J. Wang, X. Zhang, X. Shen: A Survey on Platoon-Based Vehicular Cyber-Physical Systems. *IEEE Communications Surveys Tutorials*, Vol. 18, No. 1, 2016, pp. 263–284.
- [Jöckel and Kläs 2019] L. Jöckel, M. Kläs: Increasing Trust in Data-Driven Model Validation – A Framework for Probabilistic Augmentation of Images and Meta-Data Generation Using Application Scope Characteristics. In: *38th Int. Conf. Computer Safety, Reliability and Security (SafeComp)*. Springer, 2019, pp. 155–164.
- [Katoen 2016] J.-P. Katoen: The Probabilistic Model Checking Landscape. In: *Proc. 31st Ann. ACM/IEEE Symp. Logic in Computer Science (LICS)*. ACM, 2016, pp. 31–45.
- [Kläs 2018] M. Kläs: Towards Identifying and Managing Sources of Uncertainty in AI and Machine Learning Models - An Overview. *arXiv preprint, arXiv:1811.11669*.
- [Kläs and Jöckel 2020] M. Kläs, L. Jöckel: A Framework for Building Uncertainty Wrappers for AI/ML-Based Data-Driven Components. In: *3rd Int. Workshop on Artificial Intelligence Safety Engineering (WAISE)* - submitted, 2020.
- [Kläs and Sembach 2019] M. Kläs, L. Sembach: Uncertainty Wrappers for Data-Driven Models – Increase the Transparency of AI/ML-Based Models through Enrichment with Dependable Situation-Aware Uncertainty Estimates. In: *2nd Int. Workshop Artificial Intelligence Safety Engineering (WAISE)*. Springer, 2019, pp. 358–364.

- [Kläs and Vollmer 2018] M. Kläs, A. M. Vollmer: Uncertainty in Machine Learning Applications – A Practice-Driven Classification of Uncertainty. In: 1st Int. Workshop Artificial Intelligence Safety Engineering (WAISE). Springer, 2018, pp. 431–438.
- [Krizhevsky et al. 2017] A. Krizhevsky, I. Sutskever, G. E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM, Vol. 60 No. 6, 2017, pp. 84–90.
- [Krotzsch et al. 2014] M. Krotzsch, F. Simancik, I. Horrocks: Description logics. IEEE Intelligent Systems, Vol. 29, No. 1, 2014, pp. 12–19.
- [Kwiatkowska et al. 2007] M. Kwiatkowska, G. Norman, D. Parker: Stochastic Model Checking. In: Formal Methods for Performance Evaluation. Springer, 2007, pp. 220–270.
- [Ramirez et al. 2012] A. J. Ramirez, A. C. Jensen, B. H. C. Cheng: A Taxonomy of Uncertainty for Dynamically Adaptive Systems. In: 7th Int. Symp. Software Engineering for Adaptive and Self-Managing Systems (SEAMS). IEEE, 2012, pp. 99–108.
- [Solomatine and Ostfeld 2008] D. Solomatine, A. Ostfeld: Data-Driven Modelling: Some Past Experiences and New Approaches. Journal of Hydroinformatics, Vol. 10, No. 2, 2008, pp. 3–22.
- [Yu et al. 2013] H. Yu, Z. Shen, C. Leung, C. Miao, V. R. Lesser: A Survey of Multi-Agent Trust Management Systems. IEEE Access, Vol. 1, 2013, pp. 35–50.

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.