ABSTRACT

The development of cyber-physical systems typically involves the association between multiple coupled models that capture different aspects of the system and the environment where it operates. Due to the dynamic aspect of the environment, unexpected conditions and uncertainty may impact the system. In this work, we tackle this problem and propose a taxonomy for characterizing uncertainty in coupled models. Our taxonomy extends existing proposals to cope with the particularities of coupled models in cyber-physical systems. In addition, our taxonomy discusses the notion of uncertainty propagation to other parts of the system. This allows for studying and (in some cases) quantifying the effects of uncertainty on other models in a system even at design time. We show the applicability of our uncertainty taxonomy in real use cases motivated by our envisioned scenario of automotive development.

1 INTRODUCTION

In recent years, the automotive industry has undergone significant changes in the design and development of vehicles, where software has become a driving factor for innovation [11]. Today, a vehicle is a complex cyber-physical system (CPS) where several software, electric/electronics, and hardware components interact to provide different functionalities. It is also characterized by the need to interact with both humans and changing environments as well as being able to deal with unexpected events and uncertainties that permeate today’s world. Designing and engineering these complex systems call for a cooperation between different disciplines.

The model-based approach to design and develop such systems has proven to be instrumental to guarantee not only their correct behaviour, but also to analyse their non-functional properties, like performance and reliability (cf. examples in [9, 36, 46]). When multiple models devised by engineers from different disciplines are created during the design of a system, a major challenge arises to keep these models consistent. Design models for CPS are usually coupled in the sense that they share common elements, such that a change in one model may need to be reflected in another model as well. In fact, our expert interviews with developers in the automotive industry [43] indicated that systematic consistency preservation is the most desired aspect for any development methodology. Recent research has developed several approaches and consistency management tools (CMTs) to (semi-)automatically preserve consistency in so-called coupled models, i.e., models that together describe the system-under-design, e.g. [30]). However, model-based development and analysis include per-se a certain degree of uncertainty [47, 49], which is further exacerbated in this context where different coupled models coexist at the same time. Recognizing the presence of uncertainties and managing them across coupled models, would minimize their influence and increase the level of trust in the models. The missing corrective potential of managing uncertainties could lead to exaggerate the claims of the models’ validity and to their uninhibited application to problems far beyond their capabilities.

Thus far, few approaches jointly consider coupled models and uncertainty. An exception is the work by Famelis, Chechik et al. [13, 19, 20], who present an approach for specifying so-called partial models to reflect uncertainties about decisions not yet made, which allows for managing models with this type of uncertainty. However, as Chechik et al. state for the context of CPS, an open challenge is to augment existing taxonomies to consider more kinds of uncertainties and managing them across coupled models. In this paper, we fill this gap and present a classification of uncertainties in CPS that can be used as a basis in CMTs to propagate and more generally manage uncertainties in coupled models. In particular, we contributes to:

- understanding which uncertainty types emerge in different models for automotive development, taking the experience in mechanical, electrical, and software engineering into account,
- conceptualise the uncertainty propagation between the different coupled models,
- reify this understanding in a taxonomy that increase the awareness of different types of uncertainty and enable their automated management, and
- illustrate with examples from the automotive domain how the proposed taxonomy can be applied.
The remainder of the paper is organized as follows. Section 2 provides a description of existing uncertainty classifications that form the basis of our proposal. The automotive development context and the adoption of coupled models is illustrated with an example in Section 3. Next, in Section 4 we present our proposal for a classification of the types of uncertainty, a conceptualisation on the uncertainty propagation in coupled models together with application examples in the automotive domain. In Section 5, we summarize related work, focusing on existing work dealing with uncertainty in the areas of software engineering, CPS and in coupled models. Finally, Section 6 concludes with an outlook to future work.

2 FOUNDATIONS

Dealing with uncertainty in coupled models typically involves:

(i) **Awareness:** Recognize the presence of uncertainty in a system.
(ii) **Classification:** Discover the type and source of the recognized uncertainty.
(iii) **Propagation:** Propagate the classified uncertainty through a single model or coupled models.
(iv) **Mitigation:** Apply appropriate mitigation strategies to tame the uncertainty.

For step (i), what often happens in reality is that the existence of uncertainty is not known in advance, and it can be suspected only when strange results and/or behaviours are observed. Some proposals can be found on this theme. For example, [39] proposes a methodology that guides the software engineers in recognizing the existence of uncertainty. A Bayesian approach has been proposed in [8] to evaluate the presence of uncertainty (denoted surprise) using a metric that measures the distance between the prior and the posterior probability distributions. Another method is the adoption of **multiple conceptual models** [42], which proposes the analysis of several models of the same system to realize the existence of uncertainties if their results differ from each other.

Once the presence of uncertainty has been recognized, it is necessary to understand which type of uncertainty we are dealing with and where its source might be in step (ii). Here, classifications that fit the domain of the system under study can be applied [34]. The scope of the classification presented in this paper are CPS.

Optionally in step (iii), the uncertainty can be propagated. The propagation can be based on structural information or custom rules defined per uncertainty type. In addition, the transition to coupled models is possible. Although uncertainty can already be mitigated after classification in step (ii), the propagation helps to describe the impact of uncertainty more precisely and comprehensively [24].

Based on the results of step (ii) and step (iii), we are able to manage the uncertainty in the most suitable way in step (iv).

Henceforth, we posit on the awareness of uncertainty and focus on steps (ii) and (iii). We propose a classification that exploits and combines existing proposals to better characterize the automotive domain. We also discuss the propagation of uncertainty in coupled models for CPS systems. As for step (iv), we plan to exploit existing methods able to tame the different types of uncertainties that characterize CPS systems. Hereafter, we summarize the classifications on which we found our work. The first one has been introduced to deal with uncertainty in technical systems [38] (Section 2.1), while the second one focuses on self-adaptive systems [40] (Section 2.2).

For each classification, we shortly report the suggested methods for step (iv) to tame uncertainty. In Section 2.3, we discuss foundations regarding coupled models and view-based development, which are relevant in the domain of CPS.

2.1 Uncertainty in Technical Systems

The main assumption behind the approach by Pelz et al. [38] is that a system can be modeled with mathematical functions. In particular, a system function \( g(x) \) can be represented by a model

\[
    f(u, y, z, m, \ldots) = 0
\]

where \( u \) are inputs, \( y \) the internal variables, \( z \) the output and \( m \) the parameters. The proposed classification moves along two orthogonal aspects (cf. Figure 1).

For step (i), what often happens in reality is that the existence of uncertainty is not known in advance, and it can be suspected only when strange results and/or behaviours are observed. Some proposals can be found on this theme. For example, [39] proposes a methodology that guides the software engineers in recognizing the existence of uncertainty. A Bayesian approach has been proposed in [8] to evaluate the presence of uncertainty (denoted surprise) using a metric that measures the distance between the prior and the posterior probability distributions. Another method is the adoption of multiple conceptual models [42], which proposes the analysis of several models of the same system to realize the existence of uncertainties if their results differ from each other.

Based on the results of step (ii) and step (iii), we are able to manage the uncertainty in the most suitable way in step (iv).

Henceforth, we posit on the awareness of uncertainty and focus on steps (ii) and (iii). We propose a classification that exploits and combines existing proposals to better characterize the automotive domain. We also discuss the propagation of uncertainty in coupled models for CPS systems. As for step (iv), we plan to exploit existing methods able to tame the different types of uncertainties that characterize CPS systems. Hereafter, we summarize the classifications on which we found our work. The first one has been introduced to deal with uncertainty in technical systems [38] (Section 2.1), while the second one focuses on self-adaptive systems [40] (Section 2.2).
with a probabilistic or a non-probabilistic approach distinguishes the different types of uncertainties in Figure 1.

- **Model uncertainty**: exists if the functional relations between the variables in Eq. 1 as well as the scope and complexity of the model are unknown or incomplete. This can be due, for example, to the lack of knowledge about the employed materials or about the effects of the environment or to the decision to neglect some aspects in the model definition.

To capture the existence of uncertainty Eq. 1 becomes:

\[
f(u, y, z, m, \ldots) + \delta_f = 0
\]

where \(\delta_f\) is a discrepancy function that represents the difference between the model and reality, albeit its analytical expression is usually unknown. As for data uncertainty, the ability to represent \(\delta_f\) with a probabilistic or a non-probabilistic (fuzzy, interval, etc.) approach distinguishes the different types of uncertainties in Figure 1. When \(\delta_f\) cannot be quantified, then there is ignorance.

- **Structural Uncertainty**: refers to the fact that not all the possible solutions (e.g., materials) are evaluated with respect to uncertainty. In this sense, the model of the system is incomplete. For example, different materials, with different quality, might exist for the realization of a given system, all satisfying the same specific system function \(g(x)\), but not all of them are evaluated.

**Taming the uncertainty.** Monte Carlo Simulation (MCS) methods [23, 44] have been suggested as a way to deal with probabilistic data uncertainty, as well as sensitivity analysis [3]. When there is uncertainty, the possibilistic approaches that analyse whether an event is possible or impossible [25] can be applied, as well as fuzzy analysis techniques [35] and interval analysis [2]. To handle model uncertainty, examples of adopted techniques are discrepancy functions [28], error estimates, or Bayesian calibration techniques [38]. Discrete mathematical optimisation methods [37] can be useful to handle structural uncertainty.

### 2.2 Uncertainty in Self-Adaptive Systems

The taxonomy defined by Perez-Palacin and Mirandola [40] pertains to self-adaptive software systems, which can be modelled using both formal and semi-formal notation, e.g. stochastic automata or architectural description languages.

The focus of this classification is on the uncertainties that are present in the models of the system and are classified regarding the following dimensions: location, level and nature.

In particular, as illustrated in Figure 2 and detailed in [40], the location of uncertainty refers to the place where the uncertainty manifests itself within the model. An uncertainty can be located in the context, the structure of the model, or the input parameters as described in the following.

- **Context uncertainty** is an identification of the boundaries of the model; that is uncertainty about the information to be modeled. This uncertainty concerns the completeness of the model with respect to the real world. It refers to the kind of information that should be included in the model and the kind of information that should be abstracted away from it.

**Figure 2: Model uncertainty dimensions (inspired by [40])**

- **Model structural** uncertainty concerns the form of the model itself. This uncertainty refers to how accurately model’s structure represents the subset of the real world that has to be modeled.

- **Input parameters** uncertainty is often identified as parameter uncertainty and it is associated with the actual value of variables given as input to the model and with the methods used to calibrate the model parameters.

The level describes where the uncertainty manifests itself along the spectrum between deterministic knowledge (0th level) and total ignorance (3rd level) passing through awareness of uncertainty (1st level) and unawareness of uncertainty (2nd level). The nature indicates whether the uncertainty is due to the lack of accurate information (Epistemic) or is due to the inherent variability of the phenomena being described (Aleatory).

To manage uncertainties in software systems, researchers have investigated their possible sources [49]. Examples of sources of uncertainty are among others: simplifying assumptions, noise in sensing, future parameter values, humans in the loop. The impact of these uncertainties on the trustworthiness of the information in the models and their relation with the taxonomy dimensions have been analysed in [40, 49]. For example, simplifying assumptions is classified as being epistemic with location that can be both context or model structural.\(^1\)

**Taming the uncertainty.** According to the identified type of uncertainty, several methods can be applied to reduce their impact. For example, to tame uncertainties located in input parameters, reliability bound, confidence intervals, probability distributions [52], fuzzy methods, range of values, mean and variance, and sensitivity analysis [14] have been applied so far. Two powerful techniques that are generally applicable to reduce uncertainty are model averaging [10] and model discrepancy [45]. Some of the above mentioned methods were proposed in computer science, while others are brought from other research areas. However, currently none of them is able to completely eliminate uncertainties.

### 2.3 Consistency among Coupled Models

The need to keep multiple sources of information about the design of a system consistent has been extensively studied in various informatics subdisciplines, such as, databases, software engineering, in general, and model-driven software engineering, in particular. In software engineering, the notion of views and view types has

\(^{1}\)Details about this classification can be found in [40, 49].
been studied [21, 51]. One approach to view-based development is to assume a single underlying model from which views for different developers are projected [4]. Another approach is the synthetic approach [26], in which the overall system description is a composition of views (or models). The term “coupled models” focuses on the synchronization of information between different models of a system. Technically, the automation of consistency preservation in such coupled models involves the specification of consistency-preserving (bidirectional) model transformations [16]. A Consistency Management Tool (CMT) aims to support the management and preservation of consistency in coupled models by supporting the automated and semi-automated propagation of changes from one model to the others. One approach for integrating view-based development and coupled models is the idea to treat the coupled models as a virtual single underlying model (V-SUM) and thus separate the steps of view creation on such a V-SUM (similar to the projective view creation with SUMs) and the consistency preservation among the coupled models. The Vitruvius approach [30] is such a hybrid approach.

In this paper, we build upon the existing idea of consistency preservation in a V-SUM. Our contribution is the discussion of how uncertainty can be integrated in a V-SUM-approach to coupled models and a classification of the different kinds of uncertainty that needs to be handled (differently) in a resulting V-SUM-based CMT. Although we expect that the classification can be transferred to other approaches with views and coupled models, this hypothesis must be studied in more detail in the future.

3 ENVISIONED AUTOMOTIVE DEVELOPMENT WITH COUPLED MODELS

To illustrate the different models involved in model-based automotive development and how we envision automated consistency management to support it, we present an exemplary automotive development scenario. Based on this scenario, we discuss use cases of our proposed uncertainty classification in Section 4.3.

We intentionally consider the development of a self-driving electric car because its design involves many engineers from various disciplines. This shows that the impact of a design decision can affect many views involving different areas of expertise. Moreover, it makes clear that it is neither feasible to involve all disciplines in every design decision nor practical that every discipline decides for themselves. In such a situation, uncertainty needs to be represented and propagated across models, such that all engineers are aware of current uncertainties.

Figure 3 shows a small part of the envisaged V-SUM meta-model focusing on three meta-models ranging from a computer-aided design (CAD) model for the automotive domain (a), a meta-model for the electronic/electric (E/E) architecture (c), through the component meta-model for the Robot Operating System (ROS) components responsible for trajectory planning within the self-driving car (e). Moreover, it comprises consistency specifications (CS) [31] between these meta-models (shown in (b), (d) and (f)).

Let us consider a typical development situation, in which one engineer makes a change to one model which needs to be propagated to other models and requires subsequent decisions. In the following, we describe our vision of how this development scenario will be supported by automated consistency propagation and management. Suppose that the team leader decides to incorporate braking by recuperation (i.e., slowing down the car by recovering energy via the electric engine) and approaches the motor electronics engineer to modify the engine’s control software. In contrast to the negotiations between multiple disciplines required today, we envision that each engineer can make individual design decisions while the CMT preserves the overall consistency of the car under design.

First, the motor electronics engineer opens the engine’s control software (not shown in Figure 3) and implements the additional functionality, performs the corresponding tests, and once all pass, commits the control software and test results to the CMT. This change triggers the CMT’s consistency preservation mechanism, which computes the impact of this change by tracing the consistency relations. This change primarily affects the brake coefficient when combining the recuperating and mechanical brake. Thus, the CMT computes the new brake coefficient and, afterwards, generates and tests the ROS components responsible for trajectory planning within the self-driving car. This change can be performed automatically without user involvement assuming that there is a consistency relation between the brake coefficient and the corresponding parameter within the ROS components.

Additionally, the CMT then notifies the brake system engineer that the brake coefficient of the mechanical brake can be reduced, as a portion is contributed by the recuperating brake, to save weight and production costs. The engineer reduces the brake disc’s size in the CAD model, which prompts the CMT to rerun the tribological simulations of the brake disc’s heat distribution as well as the collision detection algorithm for updated CAD parts. If any violations of requirements are detected, the engineer is informed.

The change also affects the power electronics, because the engine now recuperates energy while braking which must be transmitted to the battery. Here, the CMT can automatically run tests and determine that, in our example, the energy capacity of the power electronics is exceeded due to the electrical recuperation. Assuming that such a violation of requirements cannot be automatically repaired in our example and involves manual changes, the power electronics engineer is notified by the CMT to increase the capacity of the power electronics in the E/E architecture model.

Later, during the development of the autonomous car, the team leader can task the CMT to check whether all safety and security requirements have been fulfilled by running X-in-the-Loop (XiL) tests, i.e., where models, software, and hardware are assembled in a physical test infrastructure. This is done, in our example, after both the newly manufactured brake disc and the modified wiring have been assembled in the XiL test infrastructure.

Although the presented scenario is simplified and discusses only a low number of models, it already highlights the benefits of systematic, (semi-)automated consistency checks. Note that in such an approach, only known consistency relations between two models can be handled, i.e. there needs to be a consistency specification between the respective metamodels (cf. Figure 3). Ideally, such consistency specifications can be defined for two metamodels independently of a concrete project, so that the consistency specifications can be reused across projects and even across organizations. In cases where an automated consistency preservation for a given consistency specification is not possible but human input is needed, a
Figure 3: Example of a coupled models in the automotive domain. A (partial) V-SUM meta-model for the automotive domain containing a computer-aided design (CAD), an E/E architecture, and a ROS component meta-model with viewtypes for 3D models and tribological simulations, electrical topology with hardware components (PREEvision), software components and component diagrams, as well as a combination of view types for a simplified vehicular model used by the trajectory planning component and for testing the drive-train and brake system of a car in a XiL test infrastructure (© IPEK).

CMT can at least notify engineers where in the models information may have become outdated.

In conclusion, by using a central CMT, the various engineers involved in the design of a (self-driving) car can individually design and evaluate which of their design decisions offer the best trade-off between production cost, reliability, and future extensibility and save the effort for manual consistency preservation across development teams.

After introducing this envisioned automotive development scenario with coupled models and automated consistency preservation, we next discuss our approach to uncertainty in coupled models of cyber-physical systems (CPS). While our example here assumed a CMT based on a V-SUM, we expect that the approach described in the next section might also be applicable to other approaches for coupled models. In Section 4.3, we revisit the envisioned scenario and discuss possible use cases for uncertainty.

4 PROPOSED APPROACH
This section presents our proposal for uncertainty classification in the automotive domain (Section 4.1) and a first discussion of how to quantify the uncertainty propagation among the different views of the system model (Section 4.2).

4.1 Uncertainty Classification
We propose a classification that exploits and combines the existing results tailoring them to the automotive domain. Table 1 summarizes our effort by highlighting the comparison with the taxonomies described in Section 2.

In regard to the considered types of models, the automotive domain requires to be inclusive and cover hardware, electric/electronic, and software components. To this end, as system models, we consider mathematical models like differential equations as well as formal and semi-formal models like Markovian models and UML-like component and behavioural diagrams.

Regarding the effect of uncertainty, we mainly adopt Pelz’s classification considering the known effect that is the only one analysed. Besides, we add one additional level: the level of tagging uncertainty. The rationale behind this choice is that while developing a system, it may be helpful to tag a model element as uncertain even if one is not able to further characterize the effect of the uncertainty. Such an annotation can most likely not be processed further, as more information has to be collected (either to characterize the uncertainty further or to even resolve it). These tags might be used to trace the propagation of uncertainty described in step (iii) in Section 2 by tainting all the elements in the coupled models that are related to the annotated model element as somewhat uncertain.
Regarding **handling uncertainty**, we can devise two main classes of approaches: the ones that can be applied at design time, like a Monte Carlo simulation, and the ones that can be applied at runtime, like self-adaptation.

Regarding the **locus of uncertainty**, we combine the classifications of Pelz et al. and Perez et al. and introduce the notion of **locus**. This differs from the term location in Perez et al. as it specifies which model elements are affected (locus) and the "location in the real world", which we introduce as a new dimension below. Specifically, we distinguish the following categories:

- **Parameters**: refers to the uncertainty of all the types of parameters that characterize a model and covers the input parameters in [40] and one of the aspects of **data uncertainty** in [38].
- **Model**: this type refers to the uncertainty related to the model definition, formalism selection, boundaries, and structure, and incorporates the model structure in [40] and one of the aspects of **data uncertainty** in [38].
- **Analysis**: this category refers to the uncertainty due to the evaluation of (design) decisions using model-based evaluation methods. Models always abstract from lower-level details, which gives rise to uncertainty on the predicted quality of the system. Besides, the analysis algorithms themselves could add uncertainty, when, for example, approximation techniques for mathematical models are used. In [38] this is partially covered by the **model** category, while [40] does not discuss this aspect.
- **Decision making**: this type of uncertainty is related to the decision making process when different options are available, concerning, for example, possible design decisions, or when the decision is based on incomplete information and only estimates of the values of interest are available.

Regarding the **source of uncertainty** (**in the real world**), we introduce this element to make explicit the difference between the uncertainty that is related to the system itself (what we design), and uncertainty that instead depends on the environment (everything outside the system boundary). This characterization is particularly helpful in CPS, as it can drive taming or mitigating uncertainty in a more informed way.

Finally, our classification explicitly excluded the aleatory and/or epistemic nature of uncertainty, because the distinction between these two natures often depends on the point of view of the observer and could become a philosophical debate more than a useful distinction helping the classification and management of uncertainty [17, 29].

### 4.2 Propagation of Uncertainty in Coupled Models

Uncertainty propagation allows for further understanding the impact of uncertainty in other parts of the system. In this section, we discuss several operations on how uncertainties in coupled models could be propagated by CMTs along the known consistency specifications. We characterize propagation mechanisms according to the locus of uncertainty based on our taxonomy.

**Propagation of parameter uncertainty**: depending on the parameter, the effect of uncertainty and the relationship of parameters to other elements in the models. Uncertainty propagation can be defined employing the following approaches:

- **Identity function**: indicates a direct effect of the parameter annotated with uncertainty over another part of the model.
- **Functions for non-probabilistic parameters**: for example, using interval analysis [27] or T-norms to handle fuzzy values (e.g., [1] for data uncertainty). In coupled models, the specific T-norms or T-conorms (depending on the relationship of parameters and other parts of the model annotated with uncertainty) used to
compute the propagated uncertainty need to be specified manually by the author of the consistency specification or the model transformation. Additionally, the relation can also state that the uncertainty is not propagated, e.g., with the bottom operator ⊥.

- **Approaches for probabilistic parameters:** sampling-based approaches, simulations, and other solutions for propagating the uncertainty that occurs in probabilistic parameters.
- **Domain-specific functions:** in some cases, the propagation of uncertainty needs to be defined in the consistency preservation rule based on domain knowledge.

**Propagation of model uncertainty:** The propagation of model uncertainty depends on the locus of uncertainty within the model. For instance, when uncertainty occurs in parameters, then the above described approaches can be used to capture the overall uncertainty of the model and propagate it to other parts of the system. However, structural uncertainty in the model is not easy to quantify. In this case, we hypothesise that there is no generic solution for uncertainty propagation and specific domain knowledge is needed to define the propagation.

**Propagation of analysis uncertainty:** In our experience, the occurrence of uncertainty in analyses is reflected on the output elements of the analyses. Therefore, the propagation of analysis uncertainty can be treated as a special case of parameter uncertainty, where uncertainty is represented (and “quantified”) in the output parameter of the analysis. In this case, we need to map the analysis uncertainty to parameter uncertainty, e.g., by applying a series of transformations/combinations from the elements that introduce uncertainty in the analyses to its output. Our hypothesis is that this mapping again needs to be defined in the consistency preservation rule based on domain knowledge.

In general, uncertainty located in parameters or models can be propagated to analysis. This can be handled in two ways:

- **Uncertainty-aware analysis:** Some analyses already support uncertainty in their input parameter or models. In this case, the uncertainty in coupled models needs to be mapped to the model elements expressing uncertainty in the input model of the analysis. This, we expect, can only be done by humans specifying the consistency preservation rules.
- **Analyses that ignore (some kind of) uncertainty:** For analyses that are not aware of a given kind of uncertainty, techniques such as Monte-Carlo simulation can be used to repeatedly execute the analysis and thus characterize the uncertainty of the analysis results for the given uncertainty.

### 4.3 Example Use Cases

In Table 2, we list a number of use cases in which uncertainty related to our future automotive development scenario may arise and classify these uncertainties employing our classification. These use cases are formulated based on discussions with project partners from electrical and mechanical engineering about coupled models in the automotive domain.

In connection to the motivating scenario presented in Section 3, the use cases (UC) discussed in this section are related to some of the models depicted in Figure 3. In particular, UC 1 is related to

the E/E architecture, UC 2, 3a, and 3b are related to the automotive CAD model, and UC 4 is linked to the ROS model. Lastly, UC 5 is presented as uncertainty in some material (for example occurring in the CAD model or another part of the system not depicted in Figure 3), and UC 6 describes uncertainty in the environment (i.e., outside of the system).

### 5 RELATED WORK

We group related work into three sections. First, we give an overview over the state of the art in uncertainty classification and analysis in software engineering. Afterwards, we discuss uncertainties in CPS and coupled models in more detail as these are closer to our work.

#### 5.1 Uncertainty in Software Engineering

The topic of uncertainty has been studied in software engineering, especially in the community of self-adaptive systems. We group related work into three categories: general discussion and studies about uncertainty research, available classifications and taxonomies, as well as approaches to handle uncertainty.

A decade ago, Garlan [22] already proposed to include uncertainty as first-class concern into the software design process to create more resilient systems. Troya et al. [47] gave an overview over the current state of the art. They conducted a systematic literature review with 123 primary studies and gathered uncertainty types, approaches, and application domains. Mahdavi-Hezavehi et al. [34] conducted a survey in the community of self-adaptive systems to better understand which sources, methods, and quality requirements are considered. They also discuss the lack of systematic approaches to understand and deal with uncertainty.

To better understand uncertainty, several taxonomies and classifications have been proposed in the field of software engineering. Perez-Palacin and Mirandola [40] define a three dimensional classification of uncertainty based on its location, level, and nature. Ramirez et al. [41] present a template for describing sources of uncertainty and present a taxonomy of uncertainties at the level of requirements, design time, and run time. Besides them, Esfahani and Malek [17] characterize uncertainty in self-adaptive systems. They discuss different sources and their reducibility.

Mahdavi-Hezavehi et al. [33] conducted a systematic literature review on architecture-based approaches to handle uncertainty. We highlight several of such approaches. Esfahani et al. [18] present GuideArch, an approach for design space exploration under uncertainty based on fuzzy values. Similarly, Lytra and Zdun [32] propose to use fuzzy values to define reusable and uncertainty-aware design decisions. Famelis and Chechik [19] apply partial models to handle design time uncertainty and related design decisions. Whittle et al. [50] propose to already consider uncertainty in requirements by defining points of flexibility.

The major difference to our approach is that our classification is build for propagation in coupled models, such as CPS. Bringing together software engineering approaches with model-driven engineering and uncertainty in mechanical engineering [38] requires a new view. Thus, we present a more detailed discussion of CPS and coupled models in the following.
When designers evaluate whether to introduce a new sensor for autonomous
analysis, the temperature of the environment is uncertain, but can be characterized with
a system. Assume a function that estimates the stopping time of a vehicle based on
sensor measurements. Consequently, they will initially only estimate the timing
behaviour to evaluate whether the design alternative is promising.

5.2 Uncertainty in CPS

More recent approaches have focused on representing uncertainty
in CPS [6, 12, 15, 53], which extend the uncertainty models developed
for software engineering (cf. Section 5.1).

Zhang et al. [53] characterise uncertainty in CPS in three levels:
application related to data and events, infrastructure, as well as
integration, which relates uncertainty within or between the application
and the infrastructure levels. Based on this distinction, Zhang
et al. propose the U-Model conceptual model to represent uncertainty
in three sub-models: the Belief Model, Uncertainty Model, and
Measure Model. The Belief Model captures the agent that perceives
the uncertainty, the source and the evidence of the uncertainty.
The Uncertainty Model represents self-associations of uncertainty
among the different levels specialized as content, environment, geographi-
cal location, occurrence, and lifetime. The Measure Model
aims at describing uncertainty either qualitatively (e.g., ambiguity)
or quantitatively (e.g., with fuzzy logic). The work by Chatterjee
and Reza [12] extends the U-Model to express new patterns of
spatio-temporal uncertainty that combine geographical location and
occurrence. Chipman et al. [15] classifies uncertainty in CPS in
three dimensions: location as Walker [48], level which corresponds
to the nature dimension by Perez-Palacin and Mirandola [40], and
distinguishes between static and dynamic uncertainty. The latter
captures whether an unknown variable is constant (e.g., process
variations) or can change over time (e.g., noise). Chipman et al.
propose a model of uncertainty which is in line with the taxonomy
by Pelz [38] and discuss different approaches for performing verifica-
tion of CPS with uncertainty. Bandyszak et al. [6] study the forms
of uncertainty that can occur in real-time embedded and reactive
systems, e.g., CPS. This work introduces the ECDC reference model
for categorising techniques that handle uncertainty at run time. The
ECDC model comprises four core concepts, i.e., Execution platform,
Communication infrastructure, Data processing, and Coordination.
Each of these four concepts can manifest different forms of run time
uncertainty that can be mitigated by strengthening the physical
infrastructure by design or with software-based solutions. Our pro-
solved solution advances the state of the art in the following ways.

Our taxonomy, when compared to the work by Zhang et al. [53],
captures different models for representing and studying the effect
of uncertainty, while distinguishing between design time and run
time uncertainty. In contrast to the taxonomies in [6, 12, 15], our
solution distinguishes between the locus, i.e., where uncertainty
manifests, and the source of uncertainty, i.e., whether uncertainty
steams from imprecision in the system that should be represented

Table 2: Uncertainty classification in envisioned CPS use cases.

<table>
<thead>
<tr>
<th>ID</th>
<th>Use Case (UC)</th>
<th>Effect</th>
<th>Handling</th>
<th>Locus</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>When designers evaluate whether to introduce a new sensor for autonomous driving, they may not yet have made subsequent decisions about the detailed E/E design. Thus, they cannot predict the detailed timing behaviour yet based on E/E models. Consequently, they will initially only estimate the timing behaviour to evaluate whether the design alternative is promising.</td>
<td>representing uncertainty without characterizing it further</td>
<td>design time</td>
<td>decision making</td>
<td>system</td>
</tr>
<tr>
<td>2</td>
<td>If the weight of the brake disc is uncertain, the uncertainty assessment may determine that there are no consequences in a mechanical view, because the axle mounting is designed to be strong enough.</td>
<td>probabilistic or incertitude</td>
<td>design time</td>
<td>parameters</td>
<td>system</td>
</tr>
<tr>
<td>3a</td>
<td>If it is yet uncertain which of the available materials out of a set of possible materials with each well-defined characteristics with stochastic uncertainty each shall be used for the brake disc, this uncertainty will be propagated to the thermal stress model. Thermal designers may then notice that not all options ensure that the brake disc is sufficiently cooled by the expected air flow. In this UC, we assume that not just a single parameter value is used to model the material, but multiple model elements.</td>
<td>mix of probabilistic and possibilistic</td>
<td>design time</td>
<td>model</td>
<td>system</td>
</tr>
<tr>
<td>3b</td>
<td>If it is yet uncertain which of the available materials with unknown characteristics shall be used for the brake disc, this uncertainty will be propagated to the thermal stress model. This model then cannot be solved and no results are produced (i.e. the results are unknown as well). In this UC, we assume that not just a single parameter value is used to model the material, but multiple model elements.</td>
<td>representing uncertainty without characterizing it further</td>
<td>design time</td>
<td>model</td>
<td>system</td>
</tr>
<tr>
<td>4</td>
<td>Assume a function that estimates the stopping time of a vehicle based on input parameters like speed of the vehicle, temperature of the breaks, friction of the grounds. This function is a simplification of the real world so that it can fulfill real-time requirements.</td>
<td>any, depending on how well the introduced uncertainty can be characterized</td>
<td>run time</td>
<td>analysis</td>
<td>system</td>
</tr>
<tr>
<td>5</td>
<td>The characteristic curve of the thermal properties of a material have been measured in a certain context, and it is uncertain whether the material will behave in the same way in the context of the system.</td>
<td>any, depending on the available knowledge</td>
<td>design time</td>
<td>analysis</td>
<td>both</td>
</tr>
<tr>
<td>6</td>
<td>The temperature of environment is uncertain, but can be characterised with an interval (-30 degree to 60 degree).</td>
<td>incertitude (interval)</td>
<td>design time</td>
<td>parameter</td>
<td>environment</td>
</tr>
</tbody>
</table>
or from the environment. Finally, our work differs from the state
of the art, as it studies the propagation of uncertainty in CPS ac-
cording to the locus where the uncertainty occurs. This allows for
understanding the effects of uncertainty even at design time.

Another line of research for representing uncertainty in CPS fo-
cuses on meta-models. The Orthogonal Uncertainty Model (OUM) [5]
is a meta-model for representing uncertainties and their connections
to other engineering artefacts (e.g., requirements or behavioural
specifications) in CPS development. OUM is defined as an ontologi-
cal model that captures the core concepts to describe uncertainty,
.i.e., uncertainty instance, observation point, activation condition,
rational, mitigation, and effect. The OUM language proposes an
extension of UML diagrams to visually represent the previous con-
cepts. OUM is complementary to our work as our proposed taxon-
omy allows for modeling specific aspects of uncertainty at another
level of abstraction. For instance, our taxonomy can be used to
express whether the locus of uncertainty is in the input data or in
the model using the input data. In OUM, for instance, the source
of uncertainty can only be traced to the observation point in the
system (e.g., laser sensor) where uncertainty is present, yet, the
nature of uncertainty is not represented. The previous examples
aim to illustrate the differences between our taxonomy and OUM;
this distinction in the levels of abstraction is also true in the other
elements of our taxonomy.

5.3 Uncertainty in Coupled Models

While uncertainty in software engineering and in CPS in general
has been studied in multiple works (see above), uncertainty in
coupled models (as introduced in Section 2.3) has been considered
only by some approaches.

Fameli et al. [19] present a generic approach to adapt existing
model transformations to so called “May models”, i.e. a formalism
to express uncertainty about decisions not yet taken in software
models [20] (cf. the entry “decision making” in our classification,
described in Section 4.1). However, their approach does not cover
other types of uncertainty, such as probabilistic or possibilistic
uncertainty, often encountered in CPS models. In line with that,
Chechik et al. mention “model management for uncertainty and
assurance” as one open challenge [13].

In the context of environmental sciences, the UncertWeb [7]
approach considers modelling uncertainty with UncertML, which
consider only statistical uncertainty in coupled environmental
models. The approach also discusses Monte Carlo simulations as a
generic way to propagate uncertainty into models and analyses that
assume static values. We transfer this idea to CPS-related models
and uncertainties.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented a classification of uncertainty in CPS.
Here, we focused on coupled models that are common in the domain
of design and development of vehicles. To this end, we discussed
the four steps of handling uncertainty, from awareness, and classi-
fication to propagation, and mitigation. We then proposed a novel
classification of uncertainty that describes the locus, source, model-
ing, and handling of uncertain influences.

This classification enables the propagation of uncertainty to
identify affected parts of coupled models. This benefits engineers
in inspecting and representing uncertainty both in design time and run time. Here, our proposed approach enables them to make
both more precise and more comprehensive statements about the
type and impact of uncertainty. To show this, we have presented a
preliminary analysis of real use cases where uncertainty occurs in
coupled models in CPS development. This analysis exhibits how the
different aspects of our proposed taxonomy allow for representing
or characterizing uncertainty in our envisioned scenario.

In our future work, we want to consolidate the taxonomy to
facilitate an efficient uncertainty management. We then plan to
validate the classification presented in this work by (1) interviewing
experts from the automotive domain, and (2) creating V-SUMs for
the automotive example sketched above, both in a collaboration
with mechanical and electrical engineers at KIT. Furthermore, we
plan to extend the Vitruvius approach to be able to express uncer-
tainty propagation for different kinds of uncertainties in consistency
preservation rules as described in Section 4.2. Based on that, we
will validate whether the uncertainty in realistic dynamic scenarios
can be appropriately propagated and managed. Finally, we plan to
exploit the taxonomy to define blueprints for the application of
suitable methods to tame the different types of uncertainties.

ACKNOWLEDGMENTS

Funded by the Baden-Württemberg Ministry of Science, Research
and the Arts within the “InnovationCampus Future Mobility” and
by the German Federal Ministry for Economic Affairs and Climate
Action within the research project “SofDCar” (19S21002). This work
was also supported by funding from the topic Engineering Secure
Systems of the Helmholtz Association (HGF) and by KASTEL Se-
curity Research Labs (46.23.03).

REFERENCES

[1] Maribel Acosta, Elena Simperl, Fabian Flick, and Maria-Esther Vidal. 2017. En-
hancing answer completeness of SPARQL queries via crowdsourcing. J. Web
0427(00)00342-3
[3] Andrea Saltelli and Marco Ratto and Terry Andres and Francesca Campolongo and
Jessica Cariboni and Debora Gatelli and Michaela Saisana and Stefano Tarantola.
modeling: a practical approach to view-based development. In Evaluation of
[5] Torsten Bandyszak, Marian Daun, Bastian Tenbergen, Patrice Kuh, Stefanie Wolf,
of Cyber-Physical Systems. IEEE Transactions on Automation Science and Engi-
neering 17, 3 (July 2020), 1250–1265. https://doi.org/10.1109/TASE.2020.2980726
Theories for Real-Time Systems. In Handbook of Real-Time Computing. Yu-
Chu Tian and David Charles Levy (Eds.). Springer, Singapore, 1–34. https:
//doi.org/10.1007/978-981-4585-87-3_64-1
[7] Lucy Baston, Dan Corford, Richard Jones, Gerard B.M. Heuvelink, Edeer Pebo-
soma, Christoph Stusch, Stefano Nativi, Paolo Mazzeotti, and Matthew Williams.
2013. Managing uncertainty in integrated environmental modelling: The UncertWeb
//doi.org/10.1016/j.envsoft.2012.02.008
theory of surprise to quantify degrees of uncertainty. In 36th International Con-
ference on Software Engineering, ICSE ’14, Companion Proceedings, Pankaj Jalote,
Lionel C. Briand, and André van der Hoek (Eds.). ACM, 460–463.