Uncertainty in Coupled Models of Cyber-Physical Systems

Maribel Acosta
Ruhr University Bochum
Bochum, Germany
maribel.acosta@rub.de

Sebastian Hahner
Karlsruhe Institute of Technology
Karlsruhe, Germany
sebastian.hahner@kit.edu

Anne Koziolek
Karlsruhe Institute of Technology
Karlsruhe, Germany
koziolek@kit.edu

Thomas Kühn
Karlsruhe Institute of Technology
Karlsruhe, Germany
thomas.kuehn@kit.edu

Raffaela Mirandola
Politecnico di Milano
Milano, Italy
raffaela.mirandola@polimi.it

Ralf Reussner
Karlsruhe Institute of Technology
Karlsruhe, Germany
reussner@kit.edu

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.

ACM Reference Format:
Maribel Acosta, Sebastian Hahner, Anne Koziolek, Thomas Kühn, Raffaela Mirandola, and Ralf Reussner. 2022. Uncertainty in Coupled Models of Cyber-Physical Systems. In *ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion), October 23–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3550356.3561539

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 relevant for system assurance [13].

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.

MODELS '22 Companion, October 23–28, 2022, Montreal, QC, Canada
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9467-3/22/10.
https://doi.org/10.1145/3550356.3561539

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).

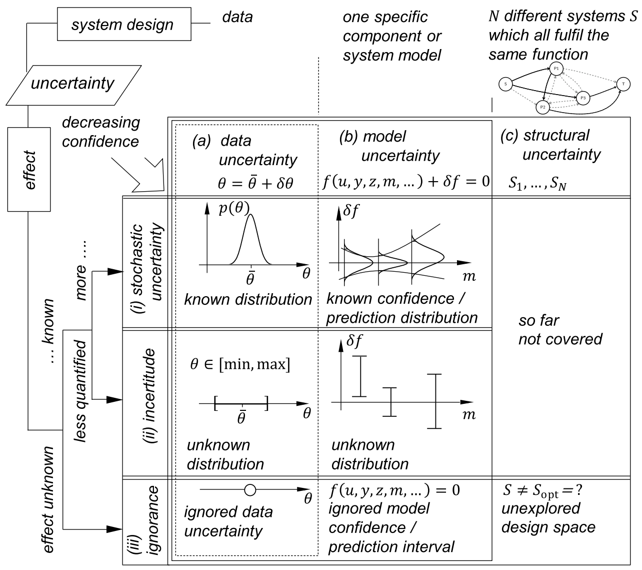
system design
data
one specific component or system model
N different systems S which all fulfil the same function
uncertainty
decreasing confidence
effect
(a) data uncertainty $\theta = \bar{\theta} + \delta\theta$
(b) model uncertainty $f(u,y,z,m,\ldots) + \delta f = 0$
(c) structural uncertainty $S_1,\ldots,S_N$
... known
more ...
less quantified
effect unknown
(i) stochastic uncertainty
$p(\theta)$
$\bar{\theta}$
θ
known distribution
δf
m
known confidence / prediction distribution
so far not covered
(ii) incertitude
$\theta \in [\min, \max]$
unknown distribution
unknown distribution
(iii) ignorance
ignored data uncertainty
$f(u,y,z,m,\ldots) = 0$ ignored model confidence / prediction interval
$S \neq S_{\text{opt}} = ?$ unexplored design space

Figure 1: Classification of Uncertainty by Pelz et al. [38] (CC BY 4.0)

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

$$f(u, y, z, m, \ldots) = 0 \tag{1}$$

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).

The first one takes into account whether *the effect* of an uncertain property on the system model is *known* or *unknown*. If the effect is known, and it is expressed in terms of probability, then it is classified as *stochastic uncertainty*. If the effect is only partially quantifiable (e.g., with interval or fuzzy metrics), it is classified as *incertitude*. When the effect is unknown, it is classified as *ignorance*. The first two cases lead to a non-deterministic system design, while the *ignorance* implies a deterministic system design. This means that the unknown effect is not modelled at all, it is disregarded.

The second aspect is related to system design and includes:

- *Data uncertainty*: is present, if the amount, type, and distribution of required data are incomplete, unknown, or insufficient. With respect to the model in Eq. 1, state variables u and z describe the input and output conditions. Data uncertainty is captured in parameters y and m, i.e., y describes the internal variables, and m describes the characteristics of the technical system. In regard to Figure 1, data uncertainty is represented as $\theta = \bar{\theta} + \delta_\theta$, where the value θ is expressed as the expected value $\bar{\theta}$ of a given distribution plus its discrepancy δ_θ. The ability to represent δ_θ

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:

$$f(u, y, z, m, \ldots) + \delta_f = 0 \tag{2}$$

  where δ_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 δ_f with a probabilistic or a non-probabilistic (fuzzy, interval, etc.) approach distinguishes the different types of uncertainties in Figure 1. When δ_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 incertitude, 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.

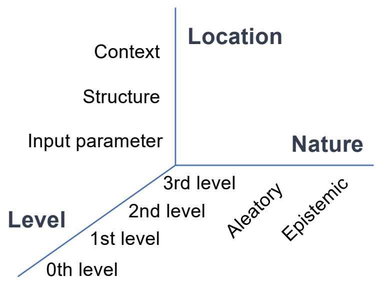

Figure 2: Model uncertainty dimensions (inspired by [40])

- *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 electrical 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

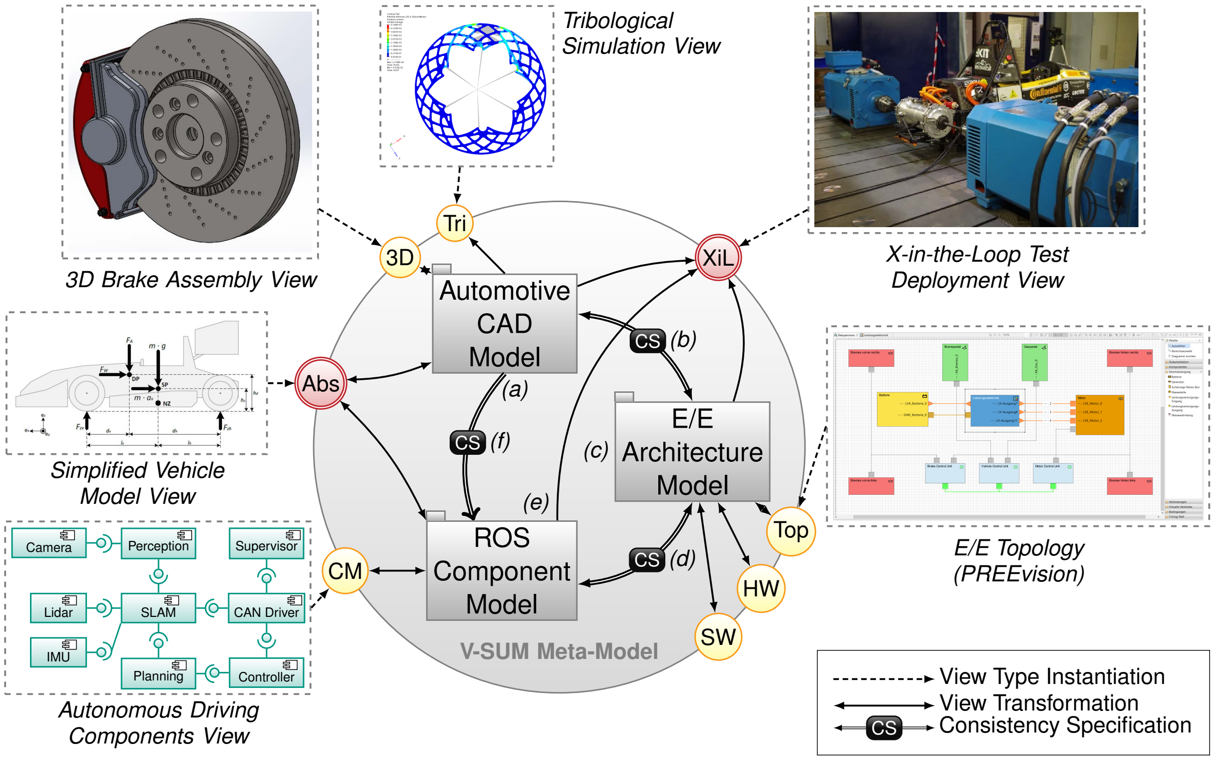

Figure 3: Example of a coupled models in the automotive domain. A (partial) V-SUM meta-model for the automotive domain containing a computer-aided design (CAD), an E/E architecture, and a ROS component meta-model with viewtypes for 3D models and tribological simulations, electrical topology with hardware 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 fir 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*.

Table 1: Comparison of existing [38, 40] and own uncertainty classification.

Taxonomy element	Pelz et al. [38]	Pérez-Palacin and Mirandola [40]	Our Solution
Type of models	Mathematical models expressed as a function of model parameters, internal variables, input and output $f(u, y, z, m, ...)$	Software architecture models like component diagrams, Queuing networks, Petri nets, Markovian models and mathematical models	Both: A broad range of models, from discrete structural like UML diagrams to mathematical models such as differential equations
Modelling the effect of uncertainty	• probabilistic approaches • incertitude-based approaches (i.e. possibilistic, fuzzy, or intervals)	Aspect related to the model formalism adopted and mentioned in the different methods to manage uncertainty	• probabilistic approaches • incertitude-based approaches (i.e. possibilistic, fuzzy, or intervals) • just representing uncertainty without characterizing it further
Handling uncertainty	at design time only, cf. Sect. 2.1	• at design time • at runtime, cf. Sect. 2.2	• at design time, cf. Sect. 2.1 • at runtime, cf. Sect. 2.2
Locus of uncertainty (in the model)	• Data • Model • Structure	Partially captured in *location:* • Input parameters • Model structure • Context	• Parameters • Model • Analysis • Decision making
Source of uncertainty (in the real world)	• not considered	• included in location	• System • Environment

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 run time, 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.

Table 2: Uncertainty classification in envisioned CPS use cases.

ID	Use Case (UC)	Effect	Handling	Locus	Source
1	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.	representing uncertainty without characterizing it further	design time	decision making	system
2	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.	probabilistic or incertitude	design time	parameters	system
3a	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.	mix of probabilistic and possibilistic	design time	model	system
3b	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.	representing uncertainty without characterizing it further	design time	model	system
4	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.	any, depending on how well the introduced uncertainty can be characterized	run time	analysis	system
5	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.	any, depending on the available knowledge	design time	analysis	both
6	The temperature of environment is uncertain, but can be characterised with an interval (-30 degree to 60 degree).	incertitude (interval)	design time	parameter	environment

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, geographical 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 verification 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 proposed 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 or from the environment. Finally, our work differs from the state of the art, as it studies the propagation of uncertainty in CPS according 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 focuses on meta-models. The *Orthogonal Uncertainty Model* (OUM) [5] is a meta-modelfor representing uncertainties and their connections to other engineering artefacts (e.g., requirements or behavioural specifications) in CPS development. OUM is defined as an ontological model that captures the core concepts to describe uncertainty, i.e., uncertainty instance, observation point, activation condition, rationale, mitigation, and effect. The OUM language proposes an extension of UML diagrams to visually represent the previous concepts. OUM is complementary to our work as our proposed taxonomy allows for modelling 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.

Famelis 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 considers 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 classification to propagation, and mitigation. We then proposed a novel classification of uncertainty that describes the locus, source, modeling, 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 uncertainty 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 Security Research Labs (46.23.03).

REFERENCES

[1] Maribel Acosta, Elena Simperl, Fabian Flöck, and Maria-Esther Vidal. 2017. Enhancing answer completeness of SPARQL queries via crowdsourcing. *J. Web Semant.* 45 (2017), 41–62. https://doi.org/10.1016/j.websem.2017.07.001

[2] Götz Alefeld and Günter Mayer. 2000. Interval analysis: theory and applications. *J. Comput. Appl. Math.* 121, 1 (2000), 421–464. https://doi.org/10.1016/S0377-0427(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. 2007. *Sensitivity Analysis: From Theory to Practice.* John Wiley & Sons, Ltd, Chapter 6, 237–275. https://doi.org/10.1002/9780470725184.ch6

[4] Colin Atkinson, Dietmar Stoll, and Philipp Bostan. 2009. Orthographic software modeling: a practical approach to view-based development. In *Evaluation of Novel Approaches to Software Engineering*. Springer, 206–219.

[5] Torsten Bandyszak, Marian Daun, Bastian Tenbergen, Patrick Kuhs, Stefanie Wolf, and Thorsten Weyer. 2020. Orthogonal Uncertainty Modeling in the Engineering of Cyber-Physical Systems. *IEEE Transactions on Automation Science and Engineering* 17, 3 (July 2020), 1250–1265. https://doi.org/10.1109/TASE.2020.2980726

[6] Torsten Bandyszak, Thorsten Weyer, and Marian Daun. 2020. Uncertainty 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 Bastin, Dan Cornford, Richard Jones, Gerard B.M. Heuvelink, Edzer Pebesma, Christoph Stasch, Stefano Nativi, Paolo Mazzetti, and Matthew Williams. 2013. Managing uncertainty in integrated environmental modelling: The UncertWeb framework. *Environmental Modelling & Software* 39 (2013), 116–134. https://doi.org/10.1016/j.envsoft.2012.02.008

[8] Nelly Bencomo and Amel Belaggoun. 2014. A world full of surprises: bayesian theory of surprise to quantify degrees of uncertainty. In *36th International Conference on Software Engineering, ICSE '14, Companion Proceedings*, Pankaj Jalote, Lionel C. Briand, and André van der Hoek (Eds.). ACM, 460–463.

[9] Matthias Bernaerts, Bentley Oakes, Ken Vanherpen, Bjorn Aelvoet, Hans Vangheluwe, and Joachim Denil. 2019. Validating Industrial Requirements with a Contract-Based Approach. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 18–27. https://doi.org/10.1109/MODELS-C.2019.00010

[10] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2022. Taming Model Uncertainty in Self-adaptive Systems Using Bayesian Model Averaging. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2022, Pittsburgh, PA, USA, May 22-24, 2022*. IEEE, 25–35. https://ieeexplore.ieee.org/document/9800082

[11] Robert N. Charette. 2021. How Software Is Eating the Car. *IEEE Sprectrum for the Technology Insider* (June 2021). https://spectrum.ieee.org/software-eating-car

[12] Amrita Chatterjee and Hassan Reza. 2020. Toward Modeling and Verification of Uncertainty in Cyber-Physical Systems. In *IEEE Intl. Conf. on Electro Information Technology (EIT)*. 568–576. https://doi.org/10.1109/EIT48999.2020.9208273

[13] Marsha Chechik, Sahar Kokaly, Mona Rahimi, Rick Salay, and Torin Viger. 2019. Uncertainty, modeling and safety assurance: towards a unified framework. In *Working Conf. on Verified Softw.: Theories, Tools, and Experiments*. Springer, 19–29.

[14] L. Cheung, L. Golubchik, N. Medvidovic, and G. Sukhatme. 2007. Identifying and Addressing Uncertainty in Architecture-Level Software Reliability Modeling. In *Int. Parallel and Distributed Processing Symposium. IPDPS 2007*. 1–6. https://doi.org/10.1109/IPDPS.2007.370524

[15] William Chipman, Christoph Grimm, and Carna Radojicic. 2015. Coverage of Uncertainties in Cyber-Physical Systems. In *ZuE 2015; 8. GMM/ITG/GI-Symposium Reliability by Design*. 1–8.

[16] Anthony Cleve, Ekkart Kindler, Perdita Stevens, and Vadim Zaytsev. 2019. Multidirectional transformations and synchronisations (Dagstuhl seminar 18491). In *Dagstuhl Reports*, Vol. 8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[17] Naeem Esfahani and Sam Malek. 2013. *Uncertainty in Self-Adaptive Software Systems*. Springer, Berlin, Heidelberg, 214–238. https://doi.org/10.1007/978-3-642-35813-5_9

[18] N. Esfahani, S. Malek, and K. Razavi. 2013. GuideArch: Guiding the exploration of architectural solution space under uncertainty. In *2013 35th Intl. Conf. on Softw. Engineering (ICSE)*. 43–52. https://doi.org/10.1109/ICSE.2013.6606550

[19] Michalis Famelis and Marsha Chechik. 2019. Managing design-time uncertainty. *Software & Systems Modeling* 18, 2 (Apr 2019), 1249–1284. https://doi.org/10.1007/s10270-017-0594-9

[20] Michalis Famelis, Rick Salay, Alessio Di Sandro, and Marsha Chechik. 2013. Transformation of models containing uncertainty. In *International conference on model driven engineering languages and systems*. Springer, 673–689.

[21] Anthony Finkelstein, Jeff Kramer, Bashar Nuseibeh, Ludwik Finkelstein, and Michael Goedicke. 1992. Viewpoints: A framework for integrating multiple perspectives in system development. *Intl. J. of Softw. Eng. and Knowledge Eng.* 2, 01 (1992), 31–57. https://doi.org/10.1142/S0218194092000038

[22] David Garlan. 2010. Software engineering in an uncertain world. In *Proc. of the FSE/SDP workshop on Future of softw. eng. research - FoSER '10*. ACM Press, 125. https://doi.org/10.1145/1882362.1882389

[23] Carl Graham and Denis Talay. 2013. *Stochastic simulation and Monte Carlo methods: mathematical foundations of stochastic simulation*. Vol. 68. Springer Science & Business Media.

[24] Sebastian Hahner. 2021. Architectural Access Control Policy Refinement and Verification under Uncertainty. In *Companion Proceedings of the 15th European Conference on Software Architecture (ECSA-C)*.

[25] Jon C. Helton, Jay D. Johnson, William Oberkampf, and Cédric J. Sallaberry. 2010. Representation of analysis results involving aleatory and epistemic uncertainty. *Int. J. General Systems* 6 (2010), 605–646. http://dblp.uni-trier.de/db/journals/ijgs/ijgs39.html#HeltonJOS10

[26] ISO/IEC/IEEE 42010:2011(E). 2011. *Systems and software engineering – Architecture description*. Intl. Organization for Standardization, Geneva, CH. 1–46 pages. https://doi.org/10.1109/IEEESTD.2011.6129467

[27] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. 2001. Interval analysis. In *Applied interval analysis*. Springer, 11–43.

[28] Marc Kennedy and Anthony O'Hagan. 2001. Bayesian Calibration of Computer Models. *Journal of the Royal Statistical Society Series B* 63 (02 2001), 425–464. https://doi.org/10.1111/1467-9868.00294

[29] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural Safety* 31, 2 (Mar 2009), 105–112. https://doi.org/10.1016/j.strusafe.2008.06.020

[30] Heiko Klare, Max E. Kramer, Michael Langhammer, Dominik Werle, Erik Burger, and Ralf Reussner. 2021. Enabling consistency in view-based system development—The Vitruvius approach. *Journal of Systems and Software* 171 (2021), 110815. https://doi.org/10.1016/j.jss.2020.110815

[31] Max Emanuel Kramer. 2017. *Specification Languages for Preserving Consistency between Models of Different Languages*. Ph. D. Dissertation. Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. https://doi.org/10.5445/IR/1000069284

[32] Ioanna Lytra and Uwe Zdun. 2013. Supporting architectural decision making for systems-of-systems design under uncertainty. In *Proc. of the First Intl. W. on Softw. Eng. for Systems-of-Systems (SESoS '13)*. ACM, New York, NY, USA, 43–46. https://doi.org/10.1145/2489850.2489859

[33] Sara Mahdavi-Hezavehi, Paris Avgeriou, and Danny Weyns. 2017. A Classification Framework of Uncertainty in Architecture-Based Self- Adaptive Systems with Multiple Quality Requirements. (2017), 33.

[34] Sara Mahdavi-Hezavehi, Danny Weyns, Paris Avgeriou, Radu Calinescu, Raffaela Mirandola, and Diego Perez-Palacin. 2021. Uncertainty in Self-Adaptive Systems: A Research Community Perspective. *ACM Transactions on Adaptive and Autonomous Systems* (2021).

[35] Bernd Möller and Uwe Reuter. 2007. *Uncertainty Forecasting in Engineering*. https://doi.org/10.1007/978-3-540-37176-2

[36] Lorenzo Pagliari, Raffaela Mirandola, and Catia Trubiani. 2020. Engineering cyber-physical systems through performance-based modelling and analysis: A case study experience report. *J. Softw. Evol. Process.* 32, 1 (2020). https://doi.org/10.1002/smr.2179

[37] R Gary Parker and Ronald L Rardin. 2014. *Discrete optimization*. Elsevier.

[38] Peter F. Pelz, Marc E. Pfetsch, Sebastian Kersting, Michael Kohler, Alexander Matei, Tobias Melz, Roland Platz, Maximilian Schaeffner, and Stefan Ulbrich. 2021. *Types of Uncertainty*. Springer International Publishing, Cham, 25–42. https://doi.org/10.1007/978-3-030-78354-9_2

[39] Diego Perez-Palacin and Raffaela Mirandola. 2014. Dealing with uncertainties in the performance modelling of software systems. In *Proc. of the 10th intl. ACM Sigsoft conf. on Quality of Software Architectures (QoSA '14)*. ACM, New York, NY, USA, 33–42. https://doi.org/10.1145/2602576.2602582

[40] Diego Perez-Palacin and Raffaela Mirandola. 2014. Uncertainties in the Modeling of Self-adaptive Systems: A Taxonomy and an Example of Availability Evaluation. In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering* (Dublin, Ireland) *(ICPE '14)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/2568088.2568095

[41] Andres J. Ramirez, Adam C. Jensen, and Betty H. C. Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *2012 7th Intl. Symposium on Softw. Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 99–108. https://doi.org/10.1109/SEAMS.2012.6224396

[42] Jens Christian Refsgaard, Jeroen P. van der Sluijs, James Brown, and Peter van der Keur. 2006. A framework for dealing with uncertainty due to model structure error. *Advances in Water Resources* 29, 11 (2006), 1586 – 1597. https://doi.org/10.1016/j.advwatres.2005.11.013

[43] E. Sax, R. Reussner, H. Guissouma, and H. Klare. 2017. *A Survey on the State and Future of Automotive Software Release and Configuration Management*. Technical Report. KIT, Karlsruhe. https://doi.org/10.5445/IR/100007567

[44] G. Schuëller and Helmut Pradlwarter. 2009. Uncertainty analysis of complex structural systems. *Internat. J. Numer. Methods Engrg.* 80 (11 2009), 881 – 913. https://doi.org/10.1002/nme.2549

[45] Mark Strong, Jeremy E. Oakley, and Jim Chilcott. 2012. Managing structural uncertainty in health economic decision models: a discrepancy approach. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 61, 1 (2012), 25–45. https://doi.org/10.1111/j.1467-9876.2011.01014.x

[46] Casper Thule, Kenneth Lausdahl, Cláudio Gomes, Gerd Meisl, and Peter Gorm Larsen. 2019. Maestro: The INTO-CPS co-simulation framework. *Simulation Modelling Practice and Theory* 92 (2019), 45–61. https://doi.org/10.1016/j.simpat.2018.12.005

[47] Javier Troya, Nathalie Moreno, Manuel F. Bertoa, and Antonio Vallecillo. 2021. Uncertainty representation in software models: a survey. *Software and Systems Modeling* (Jan 2021). https://doi.org/10.1007/s10270-020-00842-1

[48] W.E. Walker, P. Harremoës, J. Rotmans, J.P. van der Sluijs, M.B.A. van Asselt, P. Janssen, and M.P. Krayer von Krauss. 2003. Defining Uncertainty: A Conceptual Basis for Uncertainty Management in Model-Based Decision Support. *Integrated Assessment* 4, 1 (March 2003), 5–17. https://doi.org/10.1076/iaij.4.1.5.16466

[49] Danny Weyns, Nelly Bencomo, Radu Calinescu, Javier Cámara, Carlo Ghezzi, Vincenzo Grassi, Lars Grunske, Paola Inverardi, Jean-Marc Jézéquel, Sam Malek, Raffaela Mirandola, Marco Mori, and Giordano Tamburrelli. 2013. Perpetual Assurances for Self-Adaptive Systems. In *Softw. Eng. for Self-Adaptive Systems III. Assurances (LNCS, Vol. 9640)*. Springer, 31–63.

[50] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J. Bruel. 2009. RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In *2009 17th IEEE International Requirements Engineering Conference*. 79–88. https://doi.org/10.1109/RE.2009.36

[51] Albert Trevor Wood-Harper, Lyn Antill, and David E Avison. 1985. *Information systems definition: The multiview approach*. Blackwell Scientific Publications, Ltd.

[52] Liang Yin, M.A.J. Smith, and K.S. Trivedi. 2001. Uncertainty analysis in reliability modeling. In *Proc. of Reliability and Maintainability Symposium, 2001*. 229–234. https://doi.org/10.1109/RAMS.2001.902472

[53] Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. 2016. Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model. In *Modelling Foundations and Applications (LNCS)*. Springer, 247–264. https://doi.org/10.1007/978-3-319-42061-5_16