Uncertainty-aware Design Decisions through Probabilistic Uncertainty Quantification and Consistent Merging

Nathan Hagel
Karlsruhe Institut of Technology
Karlsruhe, Germany
nathan.hagel@kit.edu

Johannes Mäkelburg
Technical University of Munich
München, Germany
johannes.maekelburg@tum.de

Alireza Maleki
Karlsruhe Institute of Technology
Karlsruhe, Germany
alireza.maleki@kit.edu

Claus Hammann
Karlsruhe Institute of Technology
Karlsruhe, Germany
claus.hammann@kit.edu

Raffaela Mirandola
Karlsruhe Institute of Technology
Karlsruhe, Germany
raffaela.mirandola@kit.edu

Maribel Acosta
Technical University of Munich
München, Germany
maribel.acosta@tum.de

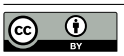
Anne Koziolek
Karlsruhe Institute of Technology
Karlsruhe, Germany
koziolek@kit.edu

Abstract

When developing complex, software-intensive or cyber-physical systems, uncertainty must be managed during all stages. Especially if made explicit, it can significantly affect the design decisions of various stakeholders. A necessary requirement for determining the effect of uncertainty on a system and deciding whether it must be resolved is explicit quantification of uncertainty. However, that alone is not enough. Often, in multi-model development processes, dependent parameters are affected by such quantified uncertainty. To ensure efficient development and uncertainty-aware decision making, we present an approach that allows to quantify uncertainty probabilistically or through a sample set. This quantified uncertainty is then kept consistent across all models. Furthermore, uncertainty is propagated and merged onto downstream model elements that depend on model elements or parameters with quantified uncertainty, and the contribution of each uncertain input to the derived uncertainty is quantified using Sobol sensitivity indices. We evaluate the approach using two case studies, comprising seven scenarios from cyber-physical systems and software engineering, based on industry scenarios and literature. The results show that quantified uncertainty can be expressed and correctly propagated and merged across all scenarios, matching an independent analytical oracle.

CCS Concepts

• **Software and its engineering → Model-driven software engineering**.



MODELS Companion 2026, Málaga, Spain

ACM ISBN 979-8-4007-2903-4/2026/10
https://doi.org/10.1145/3837062.3838899

Keywords

Uncertainty Quantification, Uncertainty Propagation, Model-Driven Engineering (MDE), Consistency Preservation, Cyber-Physical Systems, Sobol Indices

ACM Reference Format:
Nathan Hagel, Johannes Mäkelburg, Alireza Maleki, Claus Hammann, Raffaela Mirandola, Maribel Acosta, and Anne Koziolek. 2026. Uncertainty-aware Design Decisions through Probabilistic Uncertainty Quantification and Consistent Merging. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026), October 04–09, 2026, Málaga, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3837062.3838899

1 Introduction

Uncertainty is an inseparable challenge in the development of complex software-intensive or cyber-physical systems [10]. Decisions rely on assumptions about system properties, external components, requirements and the execution environment. These assumptions are often incomplete, evolving, or only partially communicated across the development artifacts and models created by different roles. Consequently, the specification contained in development artifacts may diverge from the implemented system or even the actual requirements [2, 10]. If left unmanaged, such uncertainty can lead to design decisions that appear sound during development but later cause costly problems in operation.

In modern development processes, these uncertainties are distributed across multiple semantically overlapping artifacts and models, such as requirements specifications, models of the system architecture, simulation models, and implementation artifacts. Consistency preservation mechanisms can help keep such artifacts aligned [18]. However, existing approaches typically treat uncertainty only qualitatively, for example, as an annotation or a concern noted in a single artifact [13, 22]. This is insufficient when downstream analysis and decision making depends on quantified assumptions, such as simulations conducted repeatedly during development to verify design decisions. In such cases, uncertainty must not only be represented explicitly, but also quantified in a way

that can be processed automatically across related models. Existing approaches rely on parameterized uncertainty representations specified by developers, such as fuzzy numbers, to capture uncertainty in decisions [11, 20].

Consistency-preservation frameworks already keep such artifacts aligned and can propagate qualitative uncertainty annotations across them, so that an uncertainty introduced in one artifact becomes visible in all dependent artifacts [13, 18]. However, they treat uncertainty purely qualitatively. They record *that* an element is uncertain and where this uncertainty propagates, but neither represent *how* uncertain an element is in a machine-processable form nor compute the resulting uncertainty of downstream elements that depend on several uncertain inputs. Consequently, the quantified assumptions needed for repeated downstream analysis, such as simulations, still have to be reconstructed and combined manually whenever an uncertain input changes.

This paper presents an approach that closes this gap by adding probabilistic quantification to cross-artifact uncertainty propagation. Quantified uncertainty is attached to the artifact in which it originates, propagated consistently across related artifacts, and automatically merged where downstream properties depend on several uncertain inputs. Concretely, this paper makes the following contributions:

- **Stochastic expression language extension.** We extend an existing stochastic expression language (StoEx [5]) with constructs for probability distributions and sample-based uncertainty, providing a human-readable and machine-processable representation of quantified uncertainty that can be attached to uncertainty annotations.
- **Interpreter for merging and propagation.** We provide an interpreter that evaluates, combines, and propagates these quantified uncertainties, so that the uncertainty of a downstream element depending on several uncertain inputs is computed automatically. It additionally quantifies the contribution of each uncertain input to the output using Sobol sensitivity indices.
- **Realization in a propagation workflow.** We integrate both into a model-based consistency-preservation and propagation workflow, so that quantified uncertainty is propagated and merged automatically as artifacts evolve.

We evaluate the approach using scenarios from model-driven development of a cyber-physical system and software engineering. The results indicate that the approach is expressive enough to represent the required quantified uncertainties and that quantified uncertainties can be propagated and merged consistently into downstream artifacts. Overall, the approach contributes a practical step toward making uncertainty-aware analysis available in consistency-automated development processes.

2 Foundations

Uncertainty characterizes the distributional behavior of variables within a system. These variables may follow predefined distributions, such as Gaussian, Gamma, or Poisson, or their distributions may be approximated through stochastic simulation techniques, most notably Monte Carlo methods, which estimate a target distribution by repeatedly drawing random samples from the underlying process. According to the Law of Large Numbers, the empirical distribution of these samples converges toward the true distribution as the number of samples increases [31]. The resulting distribution can then be represented, for example, using histograms or kernel density estimation.

To propagate uncertainty, the objective is to determine the output distribution induced by the joint distribution of the inputs. For two independent input variables X_1 and X_2, the distribution of the output $Z = f(X_1, X_2)$ may either admit a closed-form expression or require numerical approximation, depending on the properties of the inputs and the function. For example, if $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \forall i \in \{1, 2\}$, then for the sum $Z = X_1 + X_2$, the resulting distribution is also Gaussian, with $Z \sim \mathcal{N}(\mu_z, \sigma_z^2)$, where $\mu_z = \mu_1 + \mu_2$ and $\sigma_z^2 = \sigma_1^2 + \sigma_2^2$.

For discrete random variables, the distribution of Z can be computed exactly rather than approximated. It is obtained by enumerating all outcome pairs and summing the corresponding joint probabilities:

$$P_Z(Z = z) = \sum_{\substack{x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2 \\ f(x_1, x_2) = z}} P(X_1 = x_1, X_2 = x_2), \tag{1}$$

where $\mathcal{X}_i$ denote the supports of X_i, respectively. This generalizes convolution, which is the special case for addition [24]. While this procedure is exact, its computational cost may become high for large supports.

If the output distribution cannot be derived analytically or computed exactly, it can be approximated using Monte Carlo (MC) methods. Specifically, one draws N independent samples $(x_1, x_2)_j$ from the distributions of X_1 and X_2, evaluates $z_j = f(x_1[j], x_2[j])$, and constructs an empirical approximation of the distribution of Z from the resulting samples. The set $\{z_j\}_{j=1}^N$ can then be aggregated into a histogram, providing a practical representation of the output distribution. To quantify the sensitivity of the model output to the uncertainty of each input parameter, we compute the first-order Sobol sensitivity indices [16]. Sobol indices measure the contribution of each input parameter to the variance of the model output, thereby quantifying the influence of input uncertainty on output uncertainty. The first-order Sobol index of an input variable X_i is defined as:

$$S_i = \frac{\mathrm{Var}\left(\mathbb{E}[Z \mid X_i]\right)}{\mathrm{Var}(Z)}. \tag{2}$$

By computing Sobol indices, we identify which input parameters contribute the most to the uncertainty of the output. This information helps identify sources of instability and guides design decisions, such as refining uncertain inputs or modifying the architecture to improve the robustness of the resulting quality attributes.

3 Running Example

We use the cross-domain development of an automotive brake system as a running example, adapted from the industrial cyber-physical system (CPS) case study of Hagel et al. [14]. The example is representative of CPS development, in which several roles work on semantically overlapping models, see Figure 3: the *project manager* maintains a textual *brake specification model* that captures the required brake-system characteristics, the *design engineer* details the geometry in a *3D CAD model*, and the *simulation engineer*

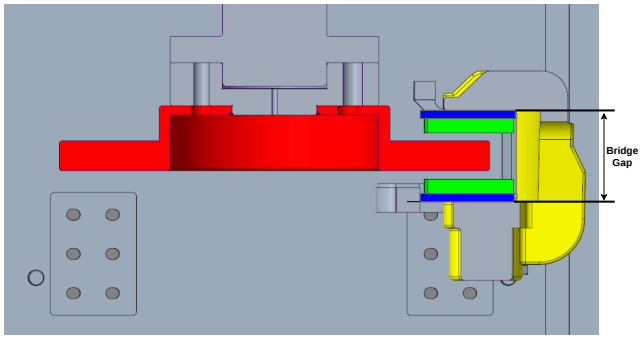


Figure 1: CAD 3D model of the brake system under development [14]. The brake disc (red) is enclosed by the caliper (yellow). The friction pad (green) is mounted on its steel backing plate (blue). The *bridge gap* is the caliper opening that has to seat the disc together with the pad and its backing plate, and is therefore derived from the brake-disc thickness, the pad thickness, and the backing-plate thickness.

maintains *Simulink simulation models* used to validate the design on a test bench. These artifacts are strongly interdependent and are automatically kept consistent using a consistency-preservation framework for multiple models. A characteristic recorded in the specification model constrains geometric parameters in the CAD model, and the same parameters must ultimately be reflected in the simulation models used for validation. In practice, however, the parameter values underlying such a design are often uncertain, distributed across the models of different domains, and only partially communicated between teams.

Concrete scenario. Consider the assembly of the disc-brake caliper. Three geometric parameters drive whether the pads and the disc physically fit into the caliper: the friction-pad thickness `thicknessPadInMM`, the steel backing-plate thickness `thicknessBackingPlateInMM`, and the brake-disc thickness `brakeDiskThicknessInMM`. During consistency preservation, two design parameters are derived from them. First, the total brake-pad width is

$$\texttt{padWidth} = \texttt{thicknessPadInMM} + \texttt{thicknessBackingPlateInMM}$$

and, using it, the required caliper bridge gap, i.e., the clearance the caliper must provide to seat the disc and the pad (Figure 1), is

$$\texttt{bridgeGap} = \texttt{brakeDiskThicknessInMM} + \texttt{padWidthInMM}.$$

With the nominal values from the specification

$$(\texttt{thicknessPadInMM} = 8, \texttt{thicknessBackingPlateInMM} = 2, \texttt{brakeDiskThicknessInMM} = 30)$$

, this yields a pad width of 10 mm and a required bridge gap of 40 mm. The design is only consistent as long as the caliper still provides a positive residual clearance for this required bridge gap. Once the combined thickness of disc and pad consumes the available caliper opening, the assembly no longer fits and the design becomes inconsistent.

Sources of uncertainty. None of the three parameters is known exactly at design time, and each is uncertain for a different, possible reason:

- *Friction-pad thickness* is uncertain because the friction material is pressed and sintered, a process that introduces a manufacturing variance around the target thickness.
- *Backing-plate thickness* is uncertain because the plates are sourced from several suppliers whose stamping tolerances differ.
- *Brake-disc thickness* is uncertain because of quality deviations between the casting batches of the disc.

These sources are exactly the ones a stakeholder would later act upon, e.g., by changing the pad manufacturing method, switching supplier, or increasing the caliper size (cf. Figure 5).

First requirement: explicit, probabilistic quantification. A first insight from this scenario is that uncertainty must be stated not only qualitatively but also quantitatively. Merely annotating that the disc thickness is "uncertain" is insufficient for automated design validation. What is needed is an explicit probabilistic characterization of the uncertain quantities. Depending on the available information, this characterization takes different forms: the project manager can quantify the disc thickness analytically as a normal distribution $\mathcal{N}(30, 1)$ derived from the supplier datasheet, whereas the procurement side provides the pad and backing-plate thickness as sample sets obtained from incoming-goods measurements (cf. the two quantification paths in Figure 4). This turns uncertainty into an analyzable modeling concept: the simulation engineer can then evaluate not just a single nominal caliper geometry, but the range of plausible geometries induced by the same specification.

Second requirement: propagation and automated merging. However, explicit quantification alone is still not enough. The uncertain inputs are not consumed only in the model in which they originate. The pad and backing-plate thickness first have to be combined into the derived pad width, and this pad width then has to be combined with the disc thickness to obtain the bridge gap in the CAD and simulation models. The derived bridge gap is therefore not a copy of a single uncertainty but is jointly determined by three uncertain inputs. If such derived quantified uncertainty is maintained manually, each stakeholder must repeatedly reinterpret and recompute the consequences of upstream uncertainty whenever an input changes. This is error-prone, leads to divergence between the brake specification, CAD, and simulation models, and creates substantial communication overhead. This motivates a second requirement: automated merging of quantified uncertainty during consistency preservation, so that the bridge-gap distribution is derived and kept up to date automatically from the pad, backing-plate, and disc uncertainties instead of being lost or inconsistently interpreted. For the concrete scenario, merging the disc uncertainty with the two sampled pad-related uncertainties yields a bridge-gap distribution with a mean of 40 mm and a standard deviation of $\sqrt{3} \approx 1.73$ mm.

Third requirement: contribution analysis for informed decisions. Even a correctly propagated bridge-gap distribution only tells the stakeholders *how* uncertain the assembly is, not *where* that uncertainty comes from. Yet the design decisions differ fundamentally in cost and effect: changing the pad manufacturing method, qualifying a different backing-plate supplier, or enlarging the caliper each address a different source. To decide rationally, stakeholders need to know how much each uncertain input contributes to the

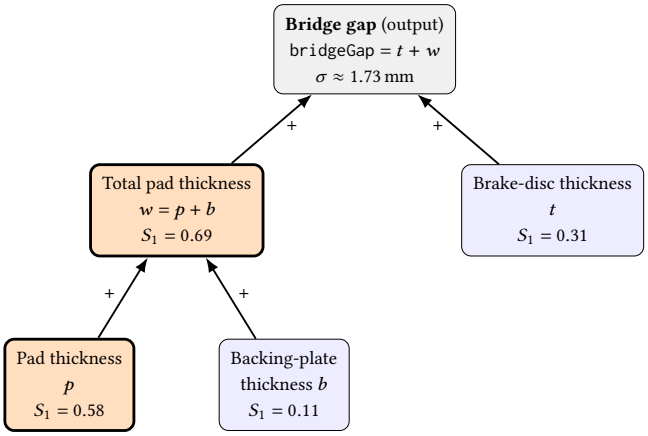


Figure 2: Derivation tree of the caliper bridge gap for the running example. The uncertain input parameters (leaves) are merged by addition into the derived total pad thickness and, together with the disc thickness, into the bridge gap (root). The first-order Sobol index S_1 annotated at each node quantifies its contribution to the variance of the bridge gap. The index of the aggregated total pad thickness equals the sum of its children (in this example). The highlighted path shows that the friction-pad thickness dominates, indicating where refining an assumption improves robustness the most.

variance of the bridge gap. This is precisely what first-order Sobol sensitivity indices provide, which our approach attaches to each node of the bridge-gap derivation tree (Figure 2). In the scenario, the friction-pad thickness is the single largest contributor to the bridge-gap variance ($S_1 = 0.58$), followed by the disc thickness ($S_1 = 0.31$) and the backing-plate thickness ($S_1 = 0.11$). Aggregated, the two pad-related inputs account for roughly two thirds of the variance while the disc accounts for the remaining third. Consequently, refining the pad manufacturing process reduces the risk of an infeasible assembly far more effectively than qualifying a different backing-plate supplier, re-specifying the disc, or oversizing the caliper. Without such a contribution analysis, this ranking of otherwise equally plausible design decisions would remain hidden.

The scenario thus highlights three capabilities required for an uncertainty-aware CPS development process. First, uncertainty must be captured explicitly and probabilistically so that it can inform analysis rather than remain a qualitative annotation. Second, once quantified, it must be propagated and automatically merged across related models so that all stakeholders work with a consistent and up-to-date view of both direct and derived uncertainty. Third, the contribution of each uncertain input to the derived uncertainty must be quantified so that stakeholders can prioritize the design decisions that improve robustness the most. Otherwise, the most important consequences of uncertainty, and the levers to control them, remain hidden precisely in those downstream models.

4 Approach

We present an approach to explicitly quantify probabilistic uncertainty in development artifacts and consistently propagate and merge it throughout a multi-model-based development process. The approach builds on automated consistency preservation across models and incorporates probabilistic expressions, using a stochastic expression Domain Specific Language (DSL) to define the metamodel. In particular, the approach contributes (1) an extension of StoEx to represent probability distributions and sample-based uncertainty, (2) an interpreter for combining quantified uncertainties, and (3) the integration of both into a consistency-preservation, uncertainty propagation, and uncertainty contribution analysis workflow. The approach comprises three steps:

(1) Specify quantified uncertainty referencing an artifact.
(2) Propagate and merge quantified uncertainty across related artifacts.
(3) Use the propagated quantified uncertainty in downstream design and analysis and quantify the contribution of uncertain inputs to the output uncertainty using Sobol indices.

As in many CPS development processes, different roles work on different models that partially describe the same system entities. In the running example, the project manager maintains the brake specification model, the design engineer maintains the CAD model, and the simulation engineer maintains the simulation models. These models are semantically overlapping because they refer to the same components, geometric parameters, and design characteristics.

Our approach assumes that such semantic overlap is made explicit through correspondence relationships or trace links. A correspondence indicates that two model elements describe the same underlying concept or information, even if they use different notations or appear in different models or artifacts. For example, the brake-disc thickness recorded in the brake specification model may correspond to a dimension of the disc in the CAD model and to a parameter of the corresponding simulation model.

To establish and maintain these links, we rely on automated consistency specifications as provided by view-based consistency-preservation frameworks such as Vitruvius [18]. These correspondences provide the basis for propagating uncertainty information across artifacts. As a result, the same uncertainty can be referenced consistently from the ticket level down to artifacts at the architecture, analysis, and implementation levels. Second, uncertainty can also be automatically propagated across such consistency-preservation rules. At the same time, it is refined by automatically merging probabilistic quantification.

In the brake-system example, this means that the pad, backing-plate, and disc thicknesses in the brake specification model, the corresponding dimensions in the CAD model, and the derived caliper bridge gap are linked through correspondence relationships. Their overlap, meaning which information is kept consistent automatically is defined using consistency preservation rules, which we leverage to propagate uncertainty even further by merging it to downstream artifacts. The overall approach overview is given by Figure 4.

4.1 Specify quantified uncertainty referencing an artifact

The first step is to annotate artifact elements with quantified uncertainty. For this purpose, we extend an existing uncertainty model [13, 22] with a formal probabilistic expression featuring the defined

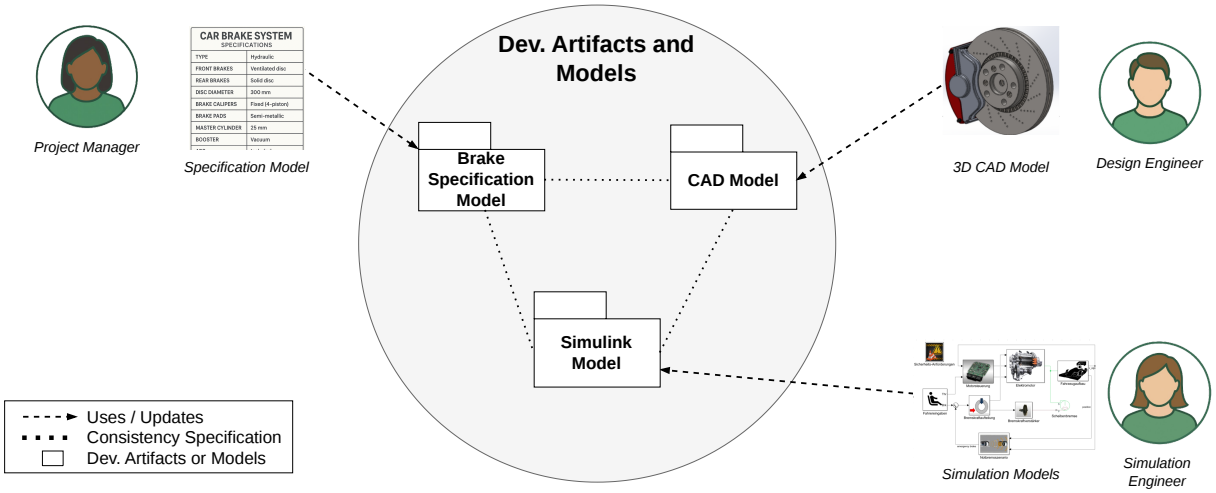


Figure 3: Running example: the cross-domain development process of an automotive brake system [14]. Different roles work on the brake specification model, the CAD model, and the simulation model, which are kept consistent by consistency-preservation specifications that also propagate the quantified uncertainty.

Stochastic Expressions (StoEx) DSL[1]. Concretely, the *Effect* of an uncertainty is extended by an *expression* attribute of type *Expression*, which references the top-level concept of the StoEx metamodel. This allows uncertainty to be represented not only qualitatively, but also in a form that can be processed automatically during propagation and analysis.

To support this, StoEx is reused and extended as a domain-specific language for stochastic expressions. The original Palladio StoEx [5] already provided an expression language for stochastic terms, but it did not directly support explicit probability distributions in the required form. Therefore, the language is extended with dedicated constructs for continuous and discrete probability functions.

This representation supports both parameterized [17] and sample-based quantification. Parameterized distributions are useful when a stakeholder can characterize uncertainty analytically, for example, by specifying a normal distribution for a request rate or an exponential distribution for an inter-arrival process. Sample-based specifications are useful when uncertainty is derived from measurements, experiments, or monitoring data. The *SampledDistribution* construct allows such observed values to be represented directly in StoEx. According to the grammar, a sample-based uncertainty can be written explicitly as a list of values, for example using the `Sampled[...]` notation.

In the brake-system running example, this step allows the stakeholders to annotate uncertain inputs such as the friction-pad thickness, the backing-plate thickness, or the brake-disc thickness with explicit probabilistic expressions, for example a normal distribution $\mathcal{N}(30, 1)$ for the disc thickness or a sample set for the measured pad thickness. These annotations are attached to the model elements where the uncertainty is introduced, for example a parameter of the brake specification model or of the CAD model, while remaining part of the shared uncertainty model.

[1]Inspect the respective metamodel in the replication package [15]

4.2 Propagate and merge quantified uncertainty across related artifacts

Once quantified uncertainty has been attached to an artifact element, it must be propagated to all related elements connected through correspondence relationships. For quantified uncertainties, however, propagation is not always a direct copy. In software development, downstream model elements often depend on several uncertain inputs at once. In such cases, the quantified uncertainty of a target element must be derived from multiple upstream uncertainties.

To enable this, the approach introduces a StoEx interpreter that evaluates and combines uncertainty expressions. The interpreter accepts either a StoEx DSL String or an instantiated *Expression* model element and returns the resulting expression after evaluation. It supports the arithmetic combination of probability distributions and scalar values. The implemented operators include addition, subtraction, multiplication, and division for probability distributions, and scalar operations on literals. The evaluation result is, again, returned as a StoEx expression, which can then be attached to a propagated uncertainty annotation.

The combination strategy depends on the type of quantified uncertainty. For discrete probability mass functions, the result can be computed exactly by enumerating value pairs and aggregating their probabilities. For continuous distributions, the interpreter uses closed-form solutions where available and otherwise falls back to Monte Carlo simulation.

This is the key extension needed for CPS scenarios in which downstream quantities are derived and plain propagation along trace links reaches its limits. In the brake-system example, the uncertainty of the caliper bridge gap does not originate from one single parameter. It is derived from the pad thickness and the backing-plate thickness, which are first merged into the pad width, and from the disc thickness. The interpreter makes it possible to express such dependencies as StoEx expressions (here `p + b` for the pad width

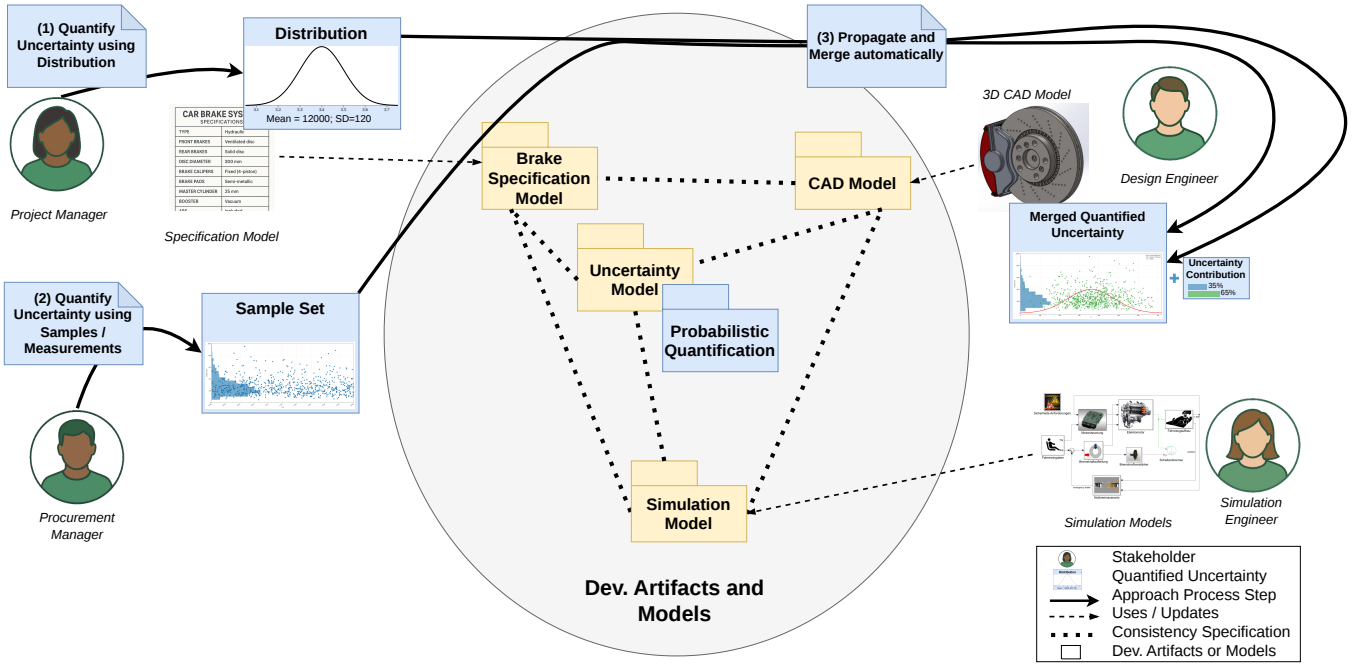


Figure 4: Overview of the approach. Stakeholders quantify uncertainty annotated to model elements or parameters. If these affect dependent parameters as defined in consistency preservation rules, the uncertainty quantification of the dependent element is calculated by merging the quantified uncertainties accordingly. Sobol indices are calculated during that process.

and `t + w` for the bridge gap) and to compute the resulting quantified uncertainty automatically when uncertainties are propagated through the Virtual Single Underlying Model (VSUM). The updated UnCertaGator metamodel therefore, serves as the integration point between the uncertainty model and the StoEx-based quantification.

The proposed approach requires an annotation of existing consistency preservation rules to define whether uncertainty should be merged and propagated or not.

4.3 Using the propagated quantified uncertainty in downstream analysis

The final step is to use the propagated and, where necessary, merged quantified uncertainty in downstream analyses that support design decisions. Since uncertainty and its contribution to the propagated uncertainty remain attached to the relevant artifact elements across the linked views, all stakeholders can work with the same up-to-date quantified assumptions.

For the brake-system example, this means that once the stakeholders introduce quantified uncertainty on the pad, backing-plate, and disc thicknesses, the simulation engineer can evaluate not only the distribution of the plausible caliper bridge gap but also the contribution of each uncertain input to the resulting output uncertainty using Sobol indices. Consequently, downstream analyses can operate on the current uncertainty-aware design state, enabling stakeholders to identify the parameters that have the greatest impact on the bridge-gap variability and to prioritize design refinements, such as changing the pad manufacturing method or supplier, accordingly.

This step is not a separate uncertainty formalism but the intended use of the previous three steps. StoEx provides the analyzable representation, the interpreter provides the combination mechanism, and the integration into consistency frameworks [18] provides the propagation infrastructure. Together, they allow uncertain design assumptions to remain available across models and to inform analysis consistently throughout the development process. The overview figure Figure 4 of the approach shows this interaction.

4.4 Implementation

The approach is implemented by extending the UnCertaGator [13] Eclipse Modeling Framework (EMF) based metamodel with StoEx. The StoEx DSL is implemented as a standalone Xtext-based project whose grammar serves as the single source of truth for generating the corresponding Ecore metamodel. This makes the language reusable in EMF-based environments and allows the existing uncertainty metamodel to reference StoEx expressions directly.

The interpreter is implemented as a separate evaluation component. It can be invoked from consistency-preservation rules to evaluate propagated expressions.

Overall, the implementation provides the technical basis for quantified uncertainty management in development processes: explicit modeling via stochastic expressions, automated evaluation via the interpreter, and cross-artifact propagation via consistency-preservation rules. The implementation is provided in the replication package [15].

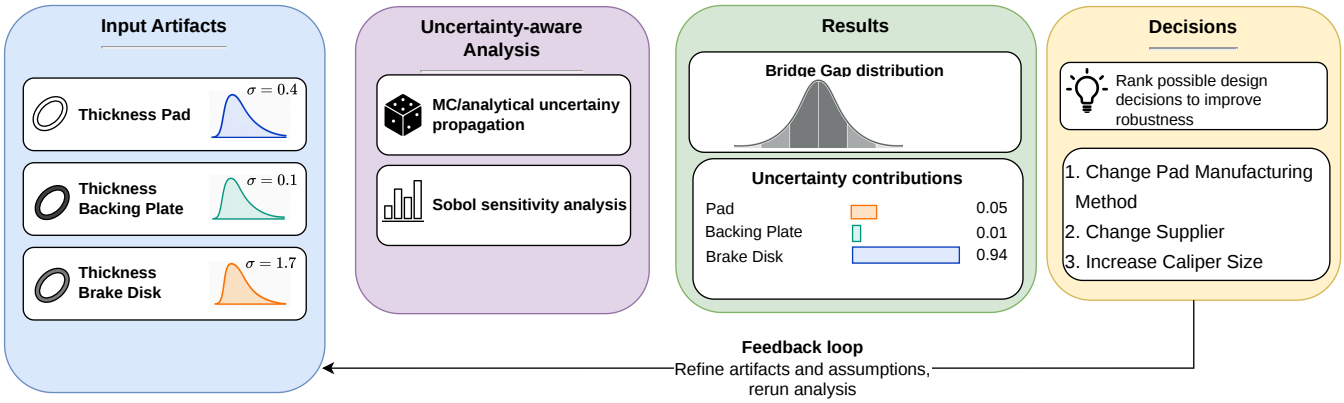


Figure 5: Overview of the presented approach featuring the sensitivity analysis

5 Evaluation

Methodology. We evaluate the approach along two lines. First, we follow the Goal-Question-Metric (GQM) method [4] [3] to assess whether quantified uncertainty can be expressed and correctly propagated and merged across models. We use two case studies. The main case study is the automotive brake system from the CPS domain [1] [14], which is also our running example (section 3). It provides six scenarios, CPS-1 to CPS-6, that vary the representation and the variance of the uncertain inputs. As an additional software-architecture case study, we reuse the TeaStore scenario from [13], denoted SA, which is adapted from an industrial scenario and transferred to the TeaStore [28] for confidentiality reasons. It complements the additive brake scenarios with a multiplicative and division-based combination. This yields seven evaluation scenarios in total, summarized in Table 1. Second, we report the feasibility of the approach in terms of the runtime overhead it induces, and we make a coherent argument for its practical benefit based on the running brake-system scenario.

Goal: Evaluate the proposed approach with respect to its ability to correctly express, propagate, and merge probabilistically quantified uncertainty across models, from the perspective of the involved engineers, in the context of multi-model CPS and software development.

Q1 (Expressibility): *Can the probabilistically quantified uncertainties from the case-study scenarios be specified using the proposed StoEx DSL?*

Metric: *Distribution coverage*, i.e., the fraction of the quantified uncertainties required by the scenarios that can be expressed.

Q2 (Propagation correctness): *Can such specified uncertainty be propagated and merged onto downstream elements so that the derived quantified uncertainty is correct?*

Metric: *Propagation output agreement*, i.e., whether the propagated and merged downstream uncertainty matches an independent correctness oracle.

Expressibility (Q1). *Q1* refers to whether the required quantified uncertainties can be expressed using the presented approach. The scenarios, see Table 1, require 20 quantified uncertainties in total, ranging from closed-form normal distributions to sample-based quantifications and, in the SA scenario, a log-normal distribution combined by multiplication and division. Using the presented metamodel, all of them were expressed, so the *distribution coverage* is 100%. The metamodel additionally covers more distribution types than are exercised by the scenarios, so its expressibility exceeds what was required. *Q1* can be answered with *yes*.

Propagation correctness (Q2). *Q2* refers to the propagation and merging of uncertainty within and across models along consistency-preservation rules. For each scenario, the consistency-preservation rules were annotated to make use of the presented approach, and each scenario was executed by annotating quantified uncertainty on the input model elements. A scenario is successful if (1) a new downstream uncertainty is created at the dependent element and (2) the merged quantified uncertainty matches an independent oracle. As an oracle we use manually derived closed-form solutions. For the brake scenarios the bridge gap follows a normal sum, so the expected mean is $\mu = 30 + 8 + 2 = 40$ mm and the expected variance is the sum of the input variances. For the SA scenario the closed-form log-normal result is used. For the closed-form scenarios (CPS-1, CPS-4, SA) the propagated distribution must equal the analytical solution exactly. For the Monte Carlo and mixed scenarios (CPS-2, CPS-3, CPS-5, CPS-6) the sampled result must match the analytical mean and standard deviation within a tolerance (±2.0 mm for the mean and ±0.1 mm for the standard deviation, using 10,000 samples and a fixed seed). In addition, the first-order Sobol indices are estimated by the Saltelli–Jansen estimator [16], and then attached to the derived bridge gap were checked against the analytically expected variance split. Uncertainty estimation and sensitivity analysis can be further improved by increasing the number of samples. All scenarios satisfied these criteria, resulting in a *propagation output agreement* of 100%.

Feasibility: overhead. Beyond expressibility and correctness, we assess the feasibility of the approach by analyzing the runtime overhead it induces. Applying the approach adds overhead on three dimensions. (1) specifying the quantified uncertainty on model elements, (2) the automated consistency preservation that now merges uncertainty, and (3) a one-time annotation of consistency-preservation rules to mark which dependencies should be uncertainty-propagation aware. The specification effort (1) is limited to a single

Table 1: Overview of the evaluation scenarios and their characterization. All brake scenarios (CPS-1 to CPS-6) merge the three input uncertainties into the derived bridge gap via addition. CPS-3 is the running example.

Scenario	# Quant. Unc.	σ [A,B,D]	Representation	Propagated output	Oracle match	Sobol ind. [A,B,D]
SA (TeaStore)	1	[1.1, None, None]	Closed form (log-normal)	LogNormal$(-0.13, 1.1)$	exact	[1, None, None]
CPS-1	3	[0.1, 0.1, 0.1]	Closed form (Normal)	$\mathcal{N}(40, 0.17)$	exact	[0.33, 0.33, 0.33]
CPS-2	3	[0.1, 0.1, 0.1]	Monte Carlo (sampled)	Sampled $(40, 1.73)$	within tol.	[0.32, 0.33, 0.34]
CPS-3 (Running ex.)	3	[0.1, 0.1, 0.1]	Mixed (Normal + sampled)	Sampled $(40, 1.73)$	within tol.	[0.34, 0.34, 0.34]
CPS-4	3	[0.4, 0.1, 1.7]	Closed form (Normal)	$\mathcal{N}(40, 1.75)$	exact	[0.05, 0.01, 0.94]
CPS-5	3	[0.4, 0.1, 1.7]	Monte Carlo (sampled)	Sampled $(40, 1.75)$	within tol.	[0.07, 0.01, 0.94]
CPS-6	3	[0.4, 0.1, 1.7]	Mixed (Normal + sampled)	Sampled $(40, 1.75)$	within tol.	[0.05, 0.01, 0.95]

model element or a short string following the DSL, or a list of values for sample-based quantification. The rule annotation (3) is a single-time modification performed once when the approach is integrated into a development process. The dominant runtime cost is dimension (2), since derived distributions must be sampled whenever the interpreter falls back to Monte Carlo. We therefore measured the most expensive task, merging uncertainty, over 100 repeated propagation operations per scenario on an AMD Ryzen 5 3600 XT CPU with 32 GB of memory, using an approach [13] with uncertainty information propagation, which has no mechanism to propagate quantified uncertainty, as the baseline. For single propagation operations, the relative overhead ranged from about 3× (closed form, one uncertain input) to about 7× (Monte Carlo, two uncertain inputs). For merge operations that combine several uncertain inputs into a derived downstream element, the relative overhead reached up to two orders of magnitude (roughly 130×–190×), with the higher factors occurring for the Monte Carlo and mixed scenarios that must sample the derived distribution. In absolute terms, however, the means stayed below $\approx$ 2.3 s for the most expensive scenario and below $\approx$ 666 ms for the others, which is acceptable for non-real-time design-time analysis.

Qualitative evaluation using the running scenario. Beyond the quantitative results, the approach provides a practical benefit for uncertainty-aware CPS development. Previous work has already shown that making uncertainty explicit and consistently propagating it across semantically overlapping models improves traceability, reduces communication overhead, and supports earlier detection of design problems that would otherwise remain implicit in a single stakeholder's artifact [13]. Our extension strengthens this benefit by turning such uncertainty into an analyzable quantity rather than a purely qualitative annotation.

In the running brake-system scenario, the relevant assumptions are not limited to the existence of uncertainty. They concern geometric parameters such as pad thickness, backing-plate thickness, and disc thickness, whose combined effect only becomes visible in the derived caliper bridge gap used in downstream CAD and simulation models. If these assumptions are maintained informally, the simulation engineer must repeatedly reinterpret the specification and manually recompute the derived bridge-gap input for the analysis. This is error-prone and risks divergence between the brake specification, CAD, and simulation models. By contrast, our approach keeps quantified uncertainty attached to the model element from which it originates and automatically propagates it to dependent elements. In addition, it merges several quantified uncertainties into derived downstream properties, such as the pad width and the caliper bridge gap, and attaches the Sobol contribution of each uncertain input. Thus, downstream analysis operates on the current design assumptions rather than on manually reconstructed point estimates, and stakeholders can rank design decisions, e.g., changing the pad manufacturing method, qualifying a different supplier, or increasing the caliper size, by their effect on robustness (cf. Figure 5).

This yields three concrete benefits. First, it preserves information that would otherwise be lost when uncertainty is propagated only qualitatively. Second, it improves consistency between the specification, uncertainty model, CAD, and simulation models, because dependent parameters are updated automatically through formal consistency-preservation rules. Third, it enables uncertainty-aware downstream analysis. Stakeholders can evaluate not only a single nominal design but also the range of plausible outcomes induced by quantified uncertainty, and, through the Sobol indices, identify the inputs that drive it. We argue that this additional modeling effort is justified because the integration is a one-time effort and the specification of an uncertainty requires only a single model element or a short DSL string. For CPS development, this means that uncertainty-aware simulation and analysis can reveal fragile design decisions before manufacturing or physical validation, when adaptations are still comparatively cheap.

Overall, the benefit of the approach is not merely that uncertainty is represented more precisely. Its main value is that quantified uncertainty remains consistent across models and becomes directly usable in downstream analysis, bridging the gap between design assumptions and their evaluation, and enabling more informed, uncertainty-aware analysis of design decisions in multi-model development processes.

Threats to Validity. We follow Wohlin et al. [30] and Runeson and Höst [25] to describe the threats to validity.

Internal Validity: Regarding the performance overhead induced by propagating the uncertainties, we measured under identical conditions to reduce the risk of confounding factors. As the approach uses tools like Vitruvius and its implementation, the results are affected by both and by the consistency-preservation rules used. The brake scenarios were designed within this work, which may have resulted in unconscious design choices favoring the presented approach. We mitigate the risk of a circular "does what it was built to do" evaluation by comparing every propagated result against an

independent analytical oracle rather than against the tool's own output.

External Validity: The evaluation draws on seven scenarios from the CPS and software-architecture domains using a variety of uncertainty quantifications. However, they cannot cover the full variability and scale found in real development processes. The brake scenarios use additive derivations only, and the SA scenario is a scenario translated into the TeaStore case study because original artifacts cannot be used for publication and was therefore abstracted from the actual structure and scale. This reduces the realism of the evaluation setting, so generalizability is limited and further empirical validation is required.

Construct Validity: The evaluation uses distribution coverage, propagation correctness against an oracle, and runtime overhead as operational measures for expressibility, correctness, and feasibility. While suitable at the feasibility stage, they only partially capture the approach's practical usefulness in real development processes. In particular, representing a quantified uncertainty in the DSL does not by itself show that stakeholders would model it consistently or correctly in practice, and successful propagation demonstrates technical correctness of the implemented rules rather than the meaningfulness of the resulting quantified uncertainties for downstream decision-making. The Monte Carlo results are approximations. We bound the resulting error through a fixed sample count, a fixed seed, and reported tolerances, but a residual approximation error remains. Finally, the current scenarios use only additive and multiplicative derivations without cyclic dependencies between propagated uncertainties, so the behavior on larger or cyclic propagation graphs is not evaluated.

6 Related Work

Classifying and Representing Uncertainty. Uncertainty representation has been addressed across different domains, including mechanical engineering [1], cyber-physical systems [2], and software engineering [27]. Mahdavi-Hezavehi et al. [21] and Acosta et al. [2] categorize uncertainty across development phases and architectural concerns. Mäkelburg et al. [22] survey existing representations and highlight that current approaches remain fragmented and lack a unified modeling perspective. Weyns et al. [29] outline a research agenda for uncertainty in self-adaptive systems, emphasizing challenges such as uncertainty propagation, interaction, and runtime management. These works provide a structured understanding of uncertainty but treat it primarily as descriptive information, not directly integrated into architectural artifacts or analysis models. In contrast, our approach enables explicit probabilistic quantification using stochastic expressions, allowing uncertainty to be embedded into artifacts and processed automatically during analysis. Esfahani et al. [11] represent uncertainty using fuzzy numbers through optimistic, expected, and pessimistic parameter values. These fuzzy values are then used to evaluate and select the most suitable architecture. However, these approaches assume a predefined design space, where uncertainty is primarily used to rank decision candidates. Consequently, they provide limited insight into how individual sources of uncertainty propagate through the model, and they offer limited support for incrementally refining alternatives during the design process.

Quantifying and Analyzing Uncertainty. A variety of techniques exist for quantifying and analyzing uncertainty, including sampling-based methods such as Monte Carlo simulation [31], analytical approaches such as Markov models and moment-based techniques, and model-based strategies such as model averaging [23]. These methods support reasoning about system behavior under uncertainty by deriving output distributions from uncertain inputs. In addition, uncertainty can be analytically characterized in closed-form when the input distributions and their combinations permit it [17].

However, such approaches operate within individual models and do not address how uncertainty is propagated and combined across multiple interrelated artifacts. Our approach extends probabilistic analysis beyond single models by enabling the systematic combination of multiple uncertainty sources across architectural artifacts using a dedicated expression language and interpreter.

Consistency Preservation Across Artifacts. Software architecture is described through multiple artifacts, such as architectural decision records, component models, and performance models, which must remain consistent with one another. Frameworks such as Vitruvius [18] establish correspondences between artifacts and propagate changes through consistency preservation rules. Complementary work addresses cross-domain consistency checking [26] and change propagation across views [19], focusing on maintaining structural and semantic alignment between models.

These approaches ensure consistency of deterministic information but do not account for quantified uncertainty or its interaction across artifacts. Our approach extends consistency preservation by treating uncertainty as first-class information that is propagated and merged along these relations.

Uncertainty in Transformations and System Evolution. Prior work studies uncertainty in different contexts, including model transformations [9], runtime adaptation in self-adaptive systems [6, 7], system evolution and digital twin scenarios [8], or exploring how uncertainty effects multi-model consistency through state space exploration [12]. These approaches analyze how uncertainty affects specific transformations or decision points, often focusing on localized interactions or runtime behavior.

While they acknowledge uncertainty in model interactions, they do not provide a mechanism for systematically propagating and combining quantified uncertainty across multiple artifacts. Our approach addresses this limitation by enabling uncertainty-aware propagation and merging across interrelated artifacts in a view-based development process.

7 Conclusion

We presented an approach to explicitly specify and automatically propagate quantified probabilistic uncertainty in multi-artifact development processes using consistency preservation. For the specification, a DSL and metamodel were defined. An interpreter was implemented to automatically merge such quantified uncertainties. The approach is evaluated using scenarios from industry and literature. Our results show its capability to express and automatically propagate uncertainties in multi-artifact multi-team development.

For future work, we plan to extend the approach by providing further automated analysis to provide more feedback and intuition on the automatically propagated uncertainties.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1608 – 501798263 and by the Topic Engineering Secure Systems of the Helmholtz Association (HGF) and supported by KASTEL Security Research Labs, Karlsruhe.

References

[1] Peter F. Pelz, Peter Groche, Marc E. Pfetsch, and Maximilian Schaeffner (Eds.). 2021. *Mastering Uncertainty in Mechanical Engineering* (Cham, 2021). Springer International Publishing. doi:10.1007/978-3-030-78354-9

[2] Maribel Acosta, Sebastian Hahner, Anne Koziolek, Thomas Kühn, Raffaela Mirandola, and Ralf Reussner. 2022. Uncertainty in coupled models of cyber-physical systems. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings* (Montreal, Quebec, Canada) *(MODELS '22)*. Association for Computing Machinery, New York, NY, USA, 569–578. doi:10.1145/3550356.3561539

[3] Victor R Basili. 1992. *Software modeling and measurement: the Goal/Question/Metric paradigm.* University of Maryland at College Park.

[4] Victor R Basili and David M Weiss. 1984. A methodology for collecting valid software engineering data. *IEEE Transactions on software engineering* 6 (1984), 728–738.

[5] Steffen Becker, Heiko Koziolek, and Ralf Reussner. 2009. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software* 82, 1 (2009), 3–22.

[6] Simona Bernardi, Michalis Famelis, Jean-Marc Jézéquel, Raffaela Mirandola, Diego Perez Palacin, Fiona A. C. Polack, and Catia Trubiani. 2021. *Living with Uncertainty in Model-Based Development.* Springer International Publishing, Cham, 159–185. doi:10.1007/978-3-030-81915-6_8

[7] Javier Camara, Sebastian Hahner, Diego Perez-Palacin, Antonio Vallecillo, Maribel Acosta, Nelly Bencomo, Radu Calinescu, and Simos Gerasimou. 2024. Uncertainty Flow Diagrams: Towards a Systematic Representation of Uncertainty Propagation and Interaction in Adaptive Systems. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (Lisbon, AA, Portugal) *(SEAMS '24)*. Association for Computing Machinery, New York, NY, USA, 37–43. doi:10.1145/3643915.3644084

[8] Julien Deantoni, Paula Muñoz, Cláudio Gomes, Clark Verbrugge, Rakshit Mittal, Robert Heinrich, Stijn Bellis, and Antonio Vallecillo. 2025. Quantifying and combining uncertainty for improving the behavior of Digital Twin Systems. *at - Automatisierungstechnik* 73, 2 (2025), 81–99. doi:doi:10.1515/auto-2024-0036

[9] Romina Eramo, Alfonso Pierantonio, and Gianni Rosa. 2015. Managing uncertainty in bidirectional model transformations. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering* (Pittsburgh, PA, USA) *(SLE 2015)*. Association for Computing Machinery, New York, NY, USA, 49–58. doi:10.1145/2814251.2814259

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

[11] Naeem Esfahani, Sam Malek, and Kaveh Razavi. 2013. GuideArch: Guiding the exploration of architectural solution space under uncertainty. In *2013 35th International Conference on Software Engineering (ICSE)*. 43–52. doi:10.1109/ICSE.2013.6606550

[12] Nathan Hagel, Johannes Mäkelburg, Martin Armbruster, Bowen Jiang, Benedikt Jutz, Lars König, Arne Lange, Fabian Reinbold, Robin Maisch, Thomas Weber, Maribel Acosta, and Anne Koziolek. 2026. UncertainTree: Analyzing Multi-Model Consistency under Uncertainty. In *2026 ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS)*.

[13] Nathan Hagel, Johannes Mäkelburg, Claus Hammann, Maximilian Hummel, Ralf Reussner, and Maribel Acosta. 2026. UnCertaGator: Propagation of Uncertainty Information in Development Processes Using Consistency Automation. In *2026 IEEE 23rd International Conference on Software Architecture (ICSA)*. 315–322. doi:10.1109/ICSA66085.2026.00038

[14] Nathan Hagel, Johannes Mäkelburg, Claus Hammann, Thomas Weber, Thomas Alexander Völk, Francesco P. Urbano, Patrick Grycz, Katharina Bause, Minakshi Kaushik, Vincenzo Scotti, Akhila Bairy, Maike Schwammberger, Maribel Acosta, Albert Albers, Anne Koziolek, and Tobias Düser. 2025. Explainability in Automated Cross-Domain Model-Driven Brake System Development. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*. 204–211. doi:10.1109/ASEW67777.2025.00047

[15] Nathan Hagel, Johannes Mäkelburg, Alireza Maleki, Claus Sebastian Hammann, Raffaela Mirandola, Maribel Acosta, and Anne Koziolek. 2026. *Replication Package: Uncertainty-aware Design Decisions through Probabilistic Uncertainty Quantification and Consistent Merging.* doi:10.5281/zenodo.21943278

[16] Michiel J.W. Jansen. 1999. Analysis of variance designs for model output. *Computer Physics Communications* 117, 1 (1999), 35–43. doi:10.1016/S0010-4655(98)00154-4

[17] Working Group 1 Joint Committee for Guides in Metrology. 2008. Evaluation of measurement data—Guide to the expression of uncertainty in measurement. *JCGM* 100, 2008 (2008), 1–116.

[18] 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. 171 (2021). doi:10.1016/j.jss.2020.110815

[19] Roland Kretschmer, Djamel Eddine Khelladi, Roberto Erick Lopez-Herrejon, and Alexander Egyed. 2021. Consistent Change Propagation within Models. *Software and Systems Modeling* 20, 2 (April 2021), 539–555. doi:10.1007/s10270-020-00823-4

[20] Emmanuel Letier, David Stefan, and Earl T. Barr. 2014. Uncertainty, risk, and information value in software requirements and architecture. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 883–894. doi:10.1145/2568225.2568239

[21] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns. 2017. A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements. In *Managing Trade-Offs in Adaptable Software Architectures*. Elsevier, 45–77. doi:10.1016/B978-0-12-802855-1.00003-4

[22] Johannes Mäkelburg, Diego Perez-Palacin, Raffaela Mirandola, and Maribel Acosta. 2026. Surveying Uncertainty Representation: A Unified Model for Cyber-Physical Systems. *Computing* 108, 5 (April 2026), 68. doi:10.1007/s00607-026-01657-6

[23] 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)*. Association for Computing Machinery, New York, NY, USA, 3–14. doi:10.1145/2568088.2568095

[24] Sheldon M Ross. 2014. *A first course in probability*. Vol. 8. Pearson London.

[25] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. 14, 2 (2009), 131–164. doi:10.1007/s10664-008-9102-8

[26] Weslley Torres, Mark G. J. van den Brand, and Alexander Serebrenik. 2021. A Systematic Literature Review of Cross-Domain Model Consistency Checking by Model Management Tools. *Software and Systems Modeling* 20, 3 (2021), 897–916. doi:10.1007/s10270-020-00834-1

[27] Javier Troya, Nathalie Moreno, Manuel F. Bertoa, and Antonio Vallecillo. 2021. Uncertainty representation in software models: a survey. 20, 4 (2021), 1183–1213. doi:10.1007/s10270-020-00842-1

[28] Jóakim von Kistowski, Simon Eismann, Norbert Schmitt, André Bauer, Johannes Grohmann, and Samuel Kounev. 2018. TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 223–236. doi:10.1109/MASCOTS.2018.00030

[29] Danny Weyns, Radu Calinescu, Raffaela Mirandola, Kenji Tei, Maribel Acosta, Nelly Bencomo, Amel Bennaceur, Nicolas Boltz, Tomas Bures, Javier Camara, Ada Diaconescu, Gregor Engels, Simos Gerasimou, Ilias Gerostathopoulos, Sinem Getir Yaman, Vincenzo Grassi, Sebastian Hahner, Paola Inverardi, Dimitri Van Landuyt, Rogerio de Lemos, Emmanuel Letier, Marin Litoiu, Lina Marsso, Angelika Musil, Juergen Musil, Genaina Nunes Rodrigues, Diego Perez-Palacin, Federico Quin, Patrizia Scandurra, Antonio Vallecillo, and Andrea Zisman. 2023. Towards a Research Agenda for Understanding and Managing Uncertainty in Self-Adaptive Systems. 48, 4 (2023), 20–36. doi:10.1145/3617946.3617951

[30] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer.

[31] Jiaxin Zhang. 2021. Modern Monte Carlo methods for efficient uncertainty quantification and propagation: A survey. 13, 5 (2021). doi:10.1002/wics.1539