UncertainTree: Analyzing Multi-Model Consistency under Uncertainty

Nathan Hagel
nathan.hagel@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Johannes Mäkelburg
johannes.maekelburg@tum.de
Technical University of Munich (TUM)
Munich, Germany

Martin Armbruster
martin.armbruster@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Bowen Jiang
bowen.jiang@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Benedikt Jutz
benedikt.jutz@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Lars König
lars.koenig@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Arne Lange
arne.lange@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Fabian Reinbold
fabian.reinbold@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Robin Maisch
maisch@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Thomas Weber
thomas.weber@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Maribel Acosta
maribel.acosta@tum.de
Technical University of Munich (TUM)
Munich, Germany

Anne Koziolek
koziolek@kit.edu
Karlsruhe Institute of Technology (KIT)
Karlrsuhe, Germany

Abstract

Model-driven development of complex software-intensive or cyber-physical systems involves the creation of heterogeneous artifacts and models. Although most modeling tools seem to convey certainty for each value or model state, the information they represent often relies on assumptions and so, by necessity, include a degree of uncertainty. When specifying and managing such uncertainty within a single model, it is hard to determine its effect on the set of modeling artifacts and as well, on model consistency and correctness. The potential effects of uncertainty can be unclear, especially when models are automatically kept consistent. Effects become even more unclear when consistency and correctness are determined not solely by constraints but for a given model state, also by simulation or other black-box mechanisms. In this paper, we present UncertainTree, an end-to-end process that makes uncertainty a first-class concern of multi-model consistency analysis. The approach's analysis includes interacting uncertainties from various sources and models. Engineers annotate uncertainty in the multi-model environment. The process explores the induced possible models, propagates them, and evaluates their consistency. UncertainTree aggregates the heterogeneous verdicts in a central reporting metamodel and communicates the outcome through stakeholder-specific uncertainty views. To improve explainability, we enrich the views with explanations provided by large language models, accessing the evidence provided by previous analysis steps. To make the analysis scale, we envision interchangeable execution modes, from state-space exploration to symbolic execution of consistency-preservation rules. As evidence of feasibility, we report a pre-study, conducted as a small-scale evaluation with a working prototype on two subject systems, as evidence of feasibility. Further, we lay out a detailed plan to evaluate the end-to-end vision.

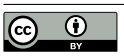
CCS Concepts

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

Keywords

Model-Driven Engineering, Uncertainty, Multi-Model Consistency, Consistency Preservation, View-Based Modeling, Symbolic Execution

ACM Reference Format:
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 *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems*



This work is licensed under a Creative Commons Attribution 4.0 International License.

MODELS 2026, Málaga, Spain

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2809-9/2026/10
https://doi.org/10.1145/3822455.3838782

(MODELS 2026), October 04–09, 2026, Málaga, Spain. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3822455.3838782

1 Introduction

Engineers develop large-scale software-intensive and cyber-physical systems with a large number of interrelated models drawn from many domains [17]. To reflect a change made in one model in the models of other domains, and to maintain the relations between them, automated consistency-preservation mechanisms can be used during development [21, 28] as part of the general task of model management [25]. Engineers who rely on such mechanisms reasonably place trust in their tool chain. As long as the models are reported to be consistent, the current design decisions appear sound. This, however, creates a dangerous illusion of certainty [23], because the values in the models often rest on provisional, guessed, or only partially known information and are hence uncertain [7, 18].

Uncertainty rarely remains local to the model element in which it originates. Through consistency-preserving transformations, an uncertain parameter in one domain propagates into dependent models in other domains [12]. For example, an uncertain diameter in a mechanical specification may change the geometry of the corresponding Computer-Aided Design (CAD) model [15] and thereby decide whether surrounding components still fit or whether downstream analyses remain valid. Uncertainty is, thus, not a property of a single model element, but a composite property of the *state of the whole multi-model environment* [7, 11]. Consequently, a system may appear consistent for one nominal realization yet become inconsistent for other, equally plausible, realizations of the same uncertain parameters.

Reasoning about consistency under uncertainty is hard for three reasons. First, a single uncertain value induces a whole *space of possible concrete models*, and critical states are often revealed only by the interaction of several uncertainties [9, 11]. Second, deciding consistency of a possible model is not a static lookup. The consistency-preservation rules must be *executed* to propagate the uncertain change, and only then can the resulting state be judged. Third, that judgement is made by *heterogeneous evaluators* that go far beyond constraints, ranging from simulations to third-party black-box tools whose internal reasoning is unavailable for the model management consistency preserving tool. Existing consistency-management approaches keep related models synchronized for concrete values [21, 30], but to the best of our knowledge, none natively reason about uncertainty across these three dimensions, nor communicate the result to the diverse stakeholders who must act on it.

In this paper, we present a vision for an *end-to-end process* that turns uncertainty into a first-class concern of multi-model consistency analysis. Based on uncertainty annotations [14] the process derives the induced possible models, propagates them through the existing consistency-preservation rules, evaluates each resulting state with composable, possibly black-box evaluators, aggregates the heterogeneous verdicts (from booleans over constraint violations to value ranges) in a central reporting metamodel, and communicates the outcome through stakeholder-specific uncertainty views enriched by Large Language Models (LLMs). Crucially, the costly analysis is served by *interchangeable execution modes*. State-space exploration for the general case, and symbolic execution of consistency-preservation rules where their structure permits it, trading enumeration for compact, range-based results.

We ground this vision in a working prototype that already realizes the exploration and evaluation steps, and we report a pre-study, conducted as a small-scale evaluation on two subject systems, as evidence that the core mechanism is feasible while exposing what is still missing, most notably scalable analysis and effective communication of results (section 5). In summary, we contribute (i) a vision of an end-to-end, uncertainty-aware multi-model consistency analysis process (section 4), (ii) the concept of interchangeable execution modes for scalability, including symbolic execution of consistency-preservation rules, (iii) a central reporting metamodel feeding stakeholder-specific, LLM-enriched uncertainty views, and (iv) feasibility evidence from a pre-study together with a plan to evaluate the complete vision (section 6).

The remainder is structured as follows. Section 2 presents background and related work, and section 3 introduces a running example. Section 4 describes our vision, section 5 reports the pre-study, section 6 details the future and evaluation plan, and section 7 concludes.

2 Background

2.1 Consistency Management for Multi-Models

A central concern of model-driven development is maintaining consistency between the heterogeneous artifacts created during development. While for small domains a single monolithic metamodel may be sufficient, most practical cases require a pragmatic setting that combines multiple metamodels [3]. Consistency can then be ensured both by *consistency checking* [31], which evaluates a consistency specification on a model state, or by *consistency preservation*, which keeps every developer-visible state consistent through incremental model transformations. Approaches such as Vitruvius [21] realize consistency preservation across a Virtual Single Underlying Model (VSUM) using Consistency Preservation Rules (CPRs) that propagate changes between models.

CPRs can be specified in dedicated languages such as the Reactions language [22]. These languages can be Turing-complete. Consequently, statically predicting the model states reachable through such CPRs is in general undecidable, so reasoning about the effect of a change may require executing the rules or using external analyses.

2.2 Uncertainty in Modeling

Following Walker et al., we regard uncertainty as *any deviation from the unachievable ideal of completely deterministic knowledge of the relevant system* [33]. Uncertainty in model-driven engineering has been studied across design and run time [7, 27], surveys propose conceptual models to represent it [24, 32], and further work studies how it propagates and interacts in adaptive systems [9] and how transformations over partial models can be made uncertainty-aware [10]. For consistency, prior work targets change propagation, traceability, and artifact-level annotations [20], but typically assumes deterministic states or treats uncertainty locally. What remains open is an *end-to-end* treatment that explores the possible

models induced by interacting uncertainties across interconnected models, judges their consistency with heterogeneous evaluators, and communicates the result to the responsible stakeholders.

3 Running Example

We use the model-driven development of an automotive brake system as a running example, based on a case study from the literature [15]. The system is described by four models that form a Multi-Model Environment (MME) [21]. The *Brake Specification* model (product manager) defines the product and component specifications negotiated with suppliers. The *CAD* model (mechanical engineer) captures the 3D shapes, dimensions, and positions of the assembly. The *Simulation* model (simulation engineer) draws on both to check properties such as thermal limits, and a fourth model captures the *uncertainty* annotated on the others. Changes are propagated across models automatically by Vitruvius [21], and consistency is judged by single- and cross-model constraints and by a black-box simulation that, given a model state, returns whether it complies with regulations. This black box stands in for third-party tooling whose decision process the team cannot inspect.

Now consider uncertainty. The product manager specifies a brake disc diameter of 330 mm, but the manufactured discs vary, so the admissible diameter is the range $[327, 332]$ (Figure 2, Figure 1). Because the incremental consistency-preservation mechanism only reacts to concrete changes, this uncertainty is not propagated, so the nominal model state conveys certainty while engineers base decisions on a state that does not represent reality. Some admissible diameters let the disc intersect the caliper, rendering the assembly inoperable. Others push heat absorption past the thermal limit, risking brake failure. Crucially, the local model state where the uncertainty is introduced is insufficient to decide consistency. The uncertain change must first be propagated through the CPRs into the CAD and simulation models, and only the resulting multi-model state can be judged, by constraints and by the black-box simulation. This is often infeasible to do manually even for a single uncertainty.

Moreover, uncertainties interact. The one-dimensional position of the caliper is itself uncertain but bounded in the CAD model. A single uncertainty may be harmless, yet the combination of the disc diameter and the caliper position may produce an intersection in some realizations.

Not all realizations are equally likely, though. The supplier can characterize the diameter by a probability distribution estimated from measurements, so values near the interval bounds are rare. An inconsistency arising only from a rare combination, such as the largest disc with the caliper at its extreme position, may then have negligible probability, and engineers may reasonably decide not to address it. Making such trade-offs requires knowing not only *whether* an inconsistency is possible, but also *how likely* and severe it is, so that limited effort is prioritized toward the most probable and critical cases. The open questions are therefore which uncertainties actually threaten consistency, how likely and severe the resulting inconsistencies are, which evaluator they violate, and how this insight reaches the right stakeholder.

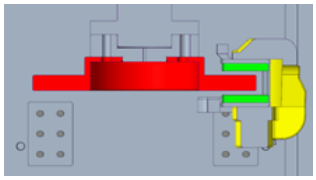


Figure 1: CAD top view as specified. [15]

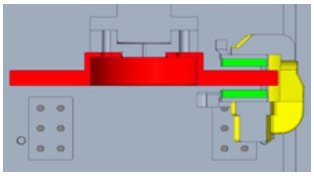


Figure 2: CAD top view of a possible state under diameter uncertainty. [15]

4 Vision: End-to-End Uncertainty-Aware Consistency Analysis

Our vision is an end-to-end process that makes uncertainty a first-class concern throughout multi-model consistency analysis, *from the point* where an engineer expresses what is uncertain and *to the point* where each affected role understands what this means for their work. Figure 3 and Figure 4 give an overview. Our envisioned process has eight conceptual stages. (1) Engineers *annotate* uncertainty on model elements of the multi-model environment. (2) The annotations induce a space of *possible concrete models*, which an interchangeable *execution mode* navigates. (3) For each possible model, the existing CPRs are executed to *propagate* the uncertain change, so that a fully materialized multi-model state results. (4) The consistency of that state is judged by heterogeneous *evaluators*. (5) Findings about inconsistencies are generated, and (6) the results are aggregated in a *central reporting metamodel* from which (7) *stakeholder-specific uncertainty views* are derived. Finally, (8) stakeholders may use the LLM interface to receive additional explanations of the impact of uncertainty in the multi-model environment. Our holistic vision is to combine all eight stages into one complete, scalable, and stakeholder-oriented process, so that both explainability and decision support are improved in cross-domain, multi-model development.

4.1 From Annotated Uncertainty to Possible Models

Uncertainty that goes uncharacterized also cannot be analyzed. Therefore, we build on an uncertainty metamodel from the literature [14, 24] and let engineers attach *uncertainty annotations* to model elements which they are responsible for. Each annotation describes the possible values or states of one source of uncertainty. An annotation may be a value range (fixed or relative to nominal value, discrete or continuous), a choice among a finite set of alternatives, existence or absence of an element, or alternative containment structures. Additionally, where the evidence is available, a realization may carry a probability, as well. Conceptually, each annotation defines one independently realizable source of uncertainty that may jointly affect several model elements. We call this the *uncertainty dimension*, and it is through combining all uncertainty dimensions that the space of all possible concrete models is spanned. Crucially, this space is not a property of any one single models alone but instead, it is a property of the whole multi-model environment. The reasons are that (a) after propagation, an uncertain value in one model induces changes in dependent models, and also that (b) in

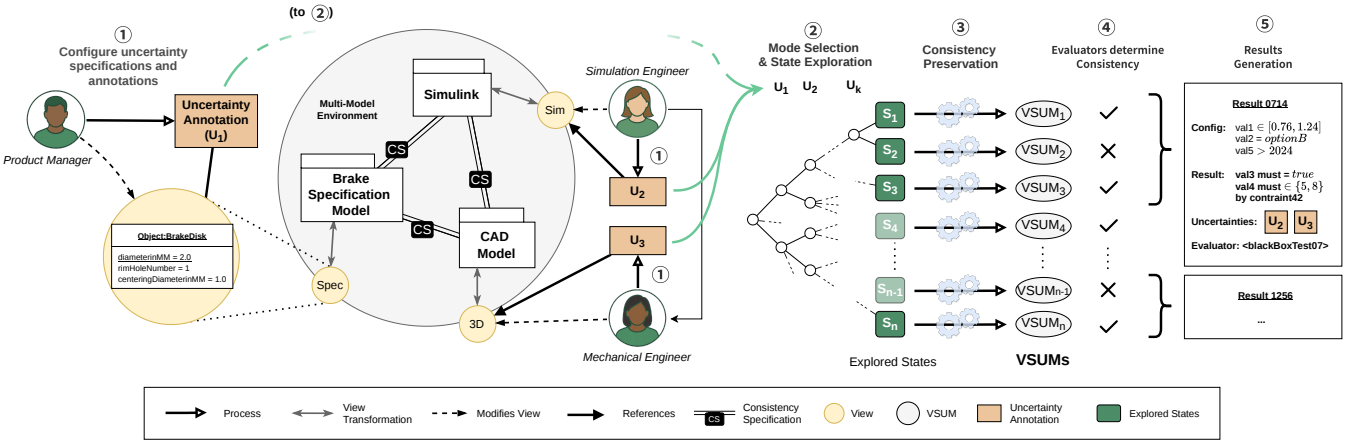


Figure 3: Overview of stages 1-5: Stakeholders annotate model elements with uncertainties (1). The analysis navigates the induced possible models (2). Consistency preservation propagates each realization (3) and evaluators judge the consistency of every resulting state (4). The aggregated results feed reporting and stakeholder-specific views (5).

the case where a realization carries a probability, the probability of every possible model necessarily follows from the realization of that same model. In this way, our envisioned analysis yields not only which states are consistent but also the *probability mass* of the consistent and inconsistent outcomes. Essentially, we reformulate the question "Is an inconsistency possible?" instead as the question "How likely is an inconsistency?" Likewise, we gain a basis for the risk-based prioritization to be exploited in the Reporting Model (Stage 6) and the stakeholder views (Stage 7).

The scope of the uncertainty analysis is also important. Any uncertainty outside of scope must keep its nominal value, though the uncertainty still may change it indirectly through propagation. The reason is that in a realistic environment, not every uncertainty should be analyzed at once. Engineers should be able to select a subset of dimensions to focus the analysis so as to avoid unnecessary effort.

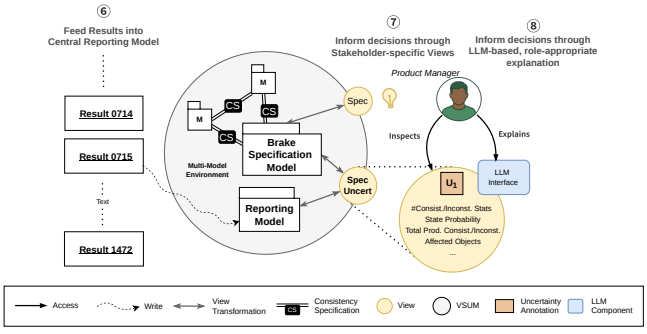


Figure 4: Overview of stages 6-8: (6) Results are fed into the Reporting Model. (7) Stakeholders consider uncertainty specific views and (8) extend their understanding of the impact of uncertainty on cross-model consistency using LLM explanations.

4.2 Deciding Consistency under Uncertainty

The annotated model state, on its own, is insufficient to know whether the system is consistent in a possible model. Only after Stages 3 and 4 does such knowledge become decidable. Stage 3 yields one fully materialized, consistency-preserved multi-model state. Specifically, the uncertainty realizations are *applied and propagated*, the realized change is committed and so triggers the existing CPRs, and these, in turn, update the dependent models according to the defined consistency relations. Stage 4 has an evaluator receiving a multi-model state which they, then, return *judgement* on. Importantly, our notion of *evaluator* is deliberately generic, including, for example, intra- and cross-model constraint checkers (e.g., OCL [34] or VitruviusOCL [26]), simulations, model checkers, geometric analyses, and *third-party black-box tools* (e.g., the regulatory simulation in our running example). Moreover, evaluators are composable, meaning that several analyses can be combined for one run. Evaluator verdicts are the product of both the multi-model state and its realization. Hence, traceability is preserved from an observed inconsistency back to the cause uncertain assumptions.

4.3 Interchangeable Execution Modes for Scalability

The model space grows with the Cartesian product of the uncertainty dimensions, and so the efficient navigation of this space poses the central scalability challenge. Here, we argue that no single analysis technique fits all CPRs and evaluators. Hence, our vision calls for a common interface but behind that, a multiplicity of *interchangeable execution modes*, each offering another tradeoff between generality, guarantees, and cost.

Mode 1: State-space exploration. The most general mode enumerates the possible models, materializes each through the CPRs, and runs the evaluators on the result. This is the mode realized in our pre-study. Its strength is that it is *universally applicable*. It works for Turing-complete CPRs, for rules that call external tools during propagation, and for black-box evaluators (since they only ever

observe concrete, fully materialized states). However, the weakness of this mode is combinatorial cost, which can be mitigated but not removed. For example, concrete states are independent and thus trivially parallelizable, continuous dimensions can be discretized at varying granularity, and the scope can be narrowed or widened as needed.

Mode 2: Symbolic execution of consistency preservation. Unlike Mode 1, the *symbolic execution* of consistency preservation is not universally applicable. CPR and evaluator must satisfy analyzability requirements: Transformation logic must be amenable to symbolic execution (i.e., with branching conditions and updates which are expressible as solver-tractable constraints over the symbolic dimensions). Calls to external tools during propagation must themselves be symbolically modeled. Evaluators must expose their verdict as a constraint over the configuration space (instead of as an opaque decision). Our aim is to replace enumeration by an uncertainty dimension as a symbolic variable. We envision executing the transformation symbolically, collecting path constraints at branching points, and using SMT solving to characterize whole *ranges* of realizations which converge on the same consistency verdict. Toward that end, we build on concolic exploration of consistency-preserving paths [19], and so obtain results like "the design stays consistent for all diameters ≤ 331" rather than separate verdicts per discretized point. It will be a key part of our research agenda to determine for realistic rule sets (1) which fragments meet the necessary analyzability requirements and (2) for fragments which do not, how to revert to Mode 1.

Mode 3: Hybrid modes. Modes 1 and 2 are the endpoints of a spectrum, along which we can imagine many additional *hybrid modes*. Nonetheless, the unifying idea remains populating one and the same reporting metamodel by all modes, so that an engineer need not change how results are reported or consumed in order to switch the analysis engine. One candidate mode might symbolically analyze the analyzable rule fragments and evaluators but enumerate only the opaque dimensions. Here, the advantage is that the cost of exhaustive exploration is paid only where unavoidable. Another candidate mode might explore sampling or boundaries for early feedback and abstraction-based modes that group equivalent realizations.

4.4 A Central Reporting Metamodel

The execution modes produce fundamentally heterogeneous results. To make these comparable and aggregable, our envisioned *central reporting metamodel* has all evaluators writing into it through adapters, thereby decoupling *what* a finding means from *how* it was obtained. The core concept is the portion of the configuration space to which a finding applies. This *result extent* uniformly captures a single realization, a bounded interval, or an unbounded predicate. But if the extent cannot be normalized without loss, then the evidence native to the evaluator is preserved. A result extent is linked by its result to the involved uncertainties, to the affected model elements, to the evaluator that produced it, and where available, also to the probability mass of the extent. This common representation lets an engineer ask, across heterogeneous analyses, which uncertainties or model elements are most often implicated and which configurations remain safe.

Table 1: Small-scale evaluation over 11 scenarios from a Cyber-Physical System (CPS) brake system [15] and the Corona-Warn-App [1, 16]. Size: model elements; St.: states; Inc.: inconsistent; Int.: revealed uncertainty interaction.

ID	Size	Unc.	St.	Inc.	Int.
CPS-1	155	1	13	8	-
CPS-2	163	2	52	32	-
CPS-3	40	3	1008	552	✓
CPS-4	163	2	273	163	✓
CPS-5	163	2	26	16	-
CPS-6	163	2	260	160	-
CPS-7	163	2	16	6	-
CPS-8	198	7	1280	1088	✓
CWA-1	1599	1	2	1	-
CWA-2	1610	1	2	0	-
CWA-3	1622	2	4	1	✓

4.5 Stakeholder-Specific Uncertainty Views

The usefulness of uncertainty-aware analysis turns upon how well it informs stakeholders employing the multi-model environment for decision-making [2]. Therefore, in the spirit of view-based development [2, 21], we will use our central reporting model to derive *stakeholder-specific uncertainty views*. Each role recognizes which models and uncertainties are under its control and which span to other domains because the role sees all results, result extents, and affected elements expressed in the terms of its own viewpoint. In our running example, the product manager sees which specification ranges remain admissible, the mechanical engineer sees the geometric realizations that intersect, the simulation engineer sees the thermal margins under uncertainty, and all these are linked back to the same findings. Furthermore, stakeholder-specific uncertainty views rank results by likelihood and severity, because every result carries its probability mass. Now stakeholders no longer need to treat every possible inconsistency as equally urgent but instead may focus on probable, high-impact inconsistencies while, perhaps, consciously accepting a rare, low-probability inconsistency.

Finally, we envision extending the stakeholder-specific views by a semantic layer of LLM-based explanation. The LLM lowers the effort of interpreting results across domain boundaries, while the central reporting metamodel keeps LLM-based explanations traceable to their evidence. Our LLM integration would generate role-appropriate explanations of why a combination of decisions is problematic, would summarize large result extents, would suggest which uncertainty to resolve first, would translate findings between stakeholder vocabularies [8], and would even query additional data from the multi-model environment to provide further explanation.

5 Pre-Study: Feasibility Evidence

To show that our vision can be realized, we have built a working prototype that implements stages 1-5 (cf. Figure 3) in the state-space-exploration mode (Mode 1), and we exercised it on two subject systems from different domains. Methodologically, we frame this pre-study as a *small-scale evaluation* [35, 36] rather than a case study. Its purpose is to demonstrate feasibility and prove the concept using

modeling artifacts outside their natural field context and conducted by a small number of researchers, not to claim the real-life context, multiple sources of evidence, and practitioner involvement that a case study requires [29]. Establishing the latter is the goal of the studies planned in section 6. The prototype is built on the Eclipse Modeling Framework and Vitruvius [21]. We extend an existing uncertainty metamodel [14, 24], which is realized in Ecore. Exploration resolves the annotations into concrete realizations and commits them so that the existing Vitruvius CPRs propagate the change into fully materialized multi-model states, each judged by a composable evaluator (currently a multi-model OCL evaluator, with black-box checks composable behind the same interface), from which a report is created (step 5).

The first subject system is the brake-system CPS scenario of the running example [15]. The second is the real-world, open-source *Corona-Warn-App* (CWA), with uncertainty scenarios from the literature [1, 16], where confidentiality is checked on a propagated Palladio-based architecture [6]. Table 1 summarizes the results across 11 scenarios. We provide a replication package with the code and scenarios [13].

The pre-study provides three pieces of evidence. First, the process is *applicable across domains*. It ran on a CPS and a software-engineering environment with model sizes from 40 to over 1600 elements and with 1 to 7 simultaneous uncertainties. Second, it *decides consistency under uncertainty*. Every scenario was explored into a finite number of concrete states and classified into consistent and inconsistent ones, only after propagation through the CPRs and evaluation, exactly the two steps of subsection 4.2. Third, it *reveals interaction effects*. In four scenarios, an inconsistency arose only from the combination of otherwise locally plausible decisions. In CWA-3 an inconsistency only arose after two architectural decisions are propagated into the architecture model.

The prototype also already supports *probability-annotated* uncertainties. Under the current equal-distribution assumption, it computes a probability for each explored state and aggregates the probability mass of the consistent and inconsistent outcomes, providing a first basis for likelihood-aware reasoning.

At the same time, the pre-study exposes precisely the gaps that motivate the rest of the vision. The number of explored states grows combinatorially, up to 1280 states already in a small scenario. Therefore, the single enumerating mode does not scale, and the interchangeable modes of subsection 4.3 are needed. Furthermore, results are emitted as generic per-state files, leaving each stakeholder to extract the relevant part, and the computed probabilities are not yet turned into prioritization support. This is the gap the central reporting metamodel and stakeholder-specific views of subsection 4.4 and subsection 4.5 are designed to close, and richer, empirically grounded distributions remain future work.

6 Future Plans

We plan to develop the envisioned process into a usable approach along two lines: *Realizing* the missing parts of the process and *evaluating* the end-to-end result. First, we will (a) design the central reporting metamodel (subsection 4.4), and we will (b) derive from it both the stakeholder-specific views and the LLM-based explanation layer, and lastly, (c) we will implement the additional execution modes, starting from symbolic execution of CPRs via concolic exploration [19], to characterize the rule and evaluator fragments to which it applies. Second, we follow the Goal-Question-Metric (GQM) method [4, 5], with a *pre-study* that establishes the requirements for stakeholder-specific views (Goal G1) and a *final study* that assesses the end-to-end workflow (Goal G2) and also assesses the scalability gained from interchangeable execution modes (Goal G3). For both pre-study and final study, we will involve experts in different roles from the CPS and the software-intensive systems domains, drawing on project partners across several projects, with a target of 20 participants per domain.

6.1 Pre-Study: Requirements for Stakeholder Views

Stakeholder views vary by domain as well as by the modeling language or tool used. Hence, we cannot assume the views can be defined generally. To derive reusable view templates, first we must learn what each role needs to see, which aspects are shared, and which are domain-specific. We will conduct a requirements pre-study (Goal G1). We will use walkthrough-based, semi-structured interviews, and Likert-scale surveys to elicit, per role, which results, result extents, and links are relevant, at which level of abstraction, and which questions a view must answer (e.g., admissible ranges, driving uncertainties, supporting evidence). The outcome will be a validated collection of information needs and view mock-ups. These will be our basis for designing the reporting metamodel, stakeholder-specific views, and the LLM-based explanation layer.

6.2 Final Study: End-to-End Workflow and Scalability

First Part: End-to-end usefulness (Goal G2). We will evaluate whether the stakeholder-specific, LLM-enriched views which have been built from the reporting metamodel, let engineers interpret uncertainty-aware consistency results and prioritize decisions in a realistic, end-to-end workflow. We will run task-based sessions on the subject systems and measure perceived usefulness, clarity, and actionability, complemented by coded qualitative feedback and a check of LLM-explanation faithfulness against the traceable evidence in the reporting model.

Second Part: Performance and scalability through execution modes (Goal G3). Here we squarely target the scalability question left open by the pre-study. We will quantify the scalability benefit of the execution modes on the scenarios of both subject systems and on synthetically scaled ones, using state-space exploration (Mode 1) as *baseline* to compare against symbolic execution (Mode 2) and the hybrid mode (Mode 3), wherever rules and evaluators qualify. Metrics will include runtime and memory, concrete states avoided, the "compression" from representing realizations as ranges instead of as points, and verification of agreement among modes on shared verdicts. A key qualitative result will characterize *which* rules and evaluators permit symbolic execution. Those whose branching, updates, and verdicts are expressible as solver-tractable constraints qualify for Mode 2. Turing-complete rules calling external tools and black-box evaluators fall back to Mode 1. Mixtures are handled by the hybrid mode.

7 Conclusion

We presented a vision for treating uncertainty as a first-class concern of end-to-end multi-model consistency analysis. UncertainTree derives the possible models induced by annotated uncertainty, propagates them through the existing consistency-preservation rules, judges each state with heterogeneous (possibly black-box) evaluators, and communicates the aggregated probability-aware verdicts through stakeholder-specific, LLM-enriched views. Scalability of UncertainTree is provided through interchangeable execution modes. Feasibility is initially demonstrated through a pre-study, conducted as a small-scale evaluation on two subject systems. We detail a two-part plan for further development: Realization of as yet missing process parts and evaluation of the end-to-end UncertainTree.

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. This research was edited by our textician, Daniel Shea.

References

[1] 2023. Corona-Warn-App. https://github.com/corona-warn-app Accessed: 2026-03-23..

[2] Colin Atkinson, Dietmar Stoll, and Philipp Bostan. 2008. Orthographic Software Modeling: A Practical Approach to View-Based Development. In *International Conference on Evaluation of Novel Approaches to Software Engineering*. Springer, 206–219.

[3] Colin Atkinson, Christian Tunjic, and Torben Moller. 2015. Fundamental Realization Strategies for Multi-view Specification Environments. In *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*. IEEE, Adelaide, Australia, 40–49. doi:10.1109/EDOC.2015.17

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

[5] Victor R. Basili and David M. Weiss. 1984. A Methodology for Collecting Valid Software Engineering Data. SE-10, 6 (1984), 728–738. doi:10.1109/TSE.1984.5010301

[6] 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.

[7] 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

[8] Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo. 2023. On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Software and Systems Modeling* 22, 3 (2023), 781–793.

[9] Javier Cámara, Javier Troya, Antonio Vallecillo, Nelly Bencomo, Radu Calinescu, Betty HC Cheng, David Garlan, and Bradley Schmerl. 2022. The uncertainty interaction problem in self-adaptive systems. *Software and systems modeling* 21, 4 (2022), 1277–1294.

[10] Michalis Famelis, Rick Salay, Alessio Di Sandro, and Marsha Chechik. 2013. Transformation of models containing uncertainty. In *Intl. conf. on Model-Driven Engineering Languages and Systems*. Springer, 673–689.

[11] Bijan Fazal, Joanna Schmidt, Ben D Phillips, Irian Ordaz, and Kenneth T Moore. 2025. Integration of uncertainty quantification in a model-based systems analysis and engineering framework. *Journal of Aircraft* 62, 6 (2025), 1680–1686.

[12] 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 *Proceedings of the ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '26)*.

[13] 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. *Replication Package: UncertainTree: Analyzing Multi-Model Consistency under Uncertainty*. doi:10.5281/zenodo.21922839

[14] 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)*.

[15] 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

[16] Sebastian Hahner, Robert Heinrich, and Ralf Reussner. 2023. Architecture-Based Uncertainty Impact Analysis to Ensure Confidentiality. In *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 126–132. doi:10.1109/SEAMS59076.2023.00026

[17] INCOSE. 2023. *INCOSE systems engineering handbook*. John Wiley & Sons.

[18] Jean-Marc Jézéquel. 2025. Taming uncertainty with MDE: an historical perspective: J.-M. Jézéquel. *Software and Systems Modeling* 24, 3 (2025), 615–628.

[19] Bowen Jiang, Miriam Boss, Thomas Weber, Weixing Zhang, Mattias Ulbrich, and Anne Koziolek. 2026. CoCoPath: Concolic Exploration of Consistency-Preserving Paths. *Journal of Object Technology* (2026). doi:10.5381/jot.2026.25.3.a9

[20] Robbert Jongeling and Antonio Vallecillo. 2023. Uncertainty-aware consistency checking in industrial settings. In *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 73–83.

[21] 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.

[22] Max Emanuel Kramer. 2017. *Specification Languages for Preserving Consistency between Models of Different Languages*. PhD Thesis. Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. doi:10.5445/IR/1000069284

[23] Azad M Madni, Norman Augustine, and Michael Sievers. 2023. *Handbook of model-based systems engineering*. Springer Nature.

[24] Johannes Mäkelburg, Diego Perez-Palacin, Raffaela Mirandola, and Maribel Acosta. 2026. Surveying uncertainty representation: a unified model for cyber-physical systems. *Computing* 108, 5 (2026), 68.

[25] Johannes Meier and Andreas Winter. 2018. Model Consistency ensured by Metamodel Integration.. In *MoDELS (Workshops)*. 408–415.

[26] Max Oesterle, Arne Lange, Nathan Hagel, Terru Stübinger, Ralf Reussner, and Anne Koziolek. 2026. VitruviusOCL: A Declarative Language for Multi-Model Consistency Specification. In *Proceedings of the ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '26)*.

[27] Andres J Ramirez, Adam C Jensen, and Betty HC Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 99–108.

[28] Ralf Reussner, Ina Schaefer, Bernhard Beckert, Anne Koziolek, and Erik Burger. 2023. Consistency in the view-based development of cyber-physical systems (convide). In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 83–84.

[29] 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

[30] Patrick Stünkel, Harald König, Yngve Lamo, and Adrian Rutle. 2021. Comprehensive systems: a formal foundation for multi-model consistency management. *Formal Aspects of Computing* 33, 6 (2021), 1067–1114.

[31] 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 (June 2021), 897–916. doi:10.1007/s10270-020-00834-1

[32] Javier Troya, Nathalie Moreno, Manuel F Bertoa, and Antonio Vallecillo. 2021. Uncertainty representation in software models: a survey. *Software and Systems Modeling* 20, 4 (2021), 1183–1213.

[33] 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. doi:10.1076/iaij.4.1.5.16466 _eprint: https://doi.org/10.1076/iaij.4.1.5.16466.

[34] Jos Warmer and Anneke Kleppe. 1998. *The object constraint language: precise modeling with UML*. Addison-Wesley Longman Publishing Co., Inc., USA.

[35] Claes Wohlin. 2021. Case Study Research in Software Engineering—It is a Case, and it is a Study, but is it a Case Study? *Information and Software Technology* 133 (2021), 106514.

[36] Claes Wohlin and Austen Rainer. 2022. Is it a case study?—A critical analysis and guidance. *Journal of Systems and Software* 192 (2022), 111395.