ISSN 2186-7437

**NII Shonan Meeting Report**

No. 232

# Uncertainty Interaction in Software-intensive Systems (UNISON)

Javier Cámara
Raffaela Mirandola
Kenji Tei

March 2–5, 2026


National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

# Uncertainty Interaction in Software-intensive Systems (UNISON)

Organizers:
Javier Cámara (University of Málaga)
Raffaela Mirandola (Karlsruhe Institute of Technology)
Kenji Tei (Institute of Science Tokyo)

March 2–5, 2026

## Background and introduction

The world is undergoing a profound transformation in which systems controlled by software are increasingly used to support critical tasks across essential domains (e.g., healthcare, transportation, banking) characterized by high degrees of uncertainty introduced by the complex interactions with their human users, the use of machine learning components, nontrivial interdependencies between their physical elements and software, and rapidly changing environmental conditions. Hence, providing assurance about the safety and performance of such "software-intensive" systems (SiS) under specified levels of uncertainty is crucial to their adoption.

During the last decade, researchers have made an important effort in supporting the analysis and management of software-intensive systems that operate under uncertainty by devising modeling notations, analysis, and assurance mechanisms that have increasingly started to capture and mitigate the effects of different types of uncertainty [13]. However, these solutions tend to tackle different types of uncertainty in isolation; yet, different uncertainty types are rarely independent and often interact, causing emergent effects that impact the achievement of system goals in subtle and often unpredictable ways [2, 3].

Indeed, these interactions can hinder the assurance and adoption of software-intensive systems. Consider, for instance, an autonomous service robot operating in a healthcare facility. When navigating between two hospital locations, this robot may face uncertainty due to: (i) its limited knowledge of the environment (e.g., presence of people in corridors, remaining energy in the battery – which has to be estimated based on measured voltage), and (ii) an overly abstract model of the environment that does not represent the geometry of obstacles in detail and can increase the chance of collision and the need for subsequent recovery routines that increase energy consumption. These uncertainty sources, when considered together, can cause the robot to deplete its battery before completing its task, while individual sources of uncertainty would not have caused the same situation. For instance, if the robot has an abstract model that causes

a collision, an accurate knowledge of the remaining battery and presence of people can allow re-planning that might still allow it to reach its target location. However, the same situation with uncertainty in the remaining battery, or the presence of humans who delay the progress of the robot through a corridor can lead to the generation of a plan based on unrealistic estimates, and therefore prone to make the robot fail its mission (e.g., due to battery depletion).

UNISON has been aimed at furthering the advances made by other relevant Shonan and Dagstuhl seminars that have discussed the engineering of software-intensive systems under uncertainty, but have not explicitly acknowledged and therefore have not explored the pivotal role of the Uncertainty Interaction Problem (UIP) [2, 3] and the need for an explicit management of uncertainty interactions in building safer and more resilient software-intensive systems.

To achieve this goal, the following topics were proposed for discussion:

**UIP concepts and terminology.** Concepts related to uncertainty (e.g., nature, category, sources) have been coined and developed in different fields like statistics, economics, and computer science. Even within computer science, there are multiple taxonomies that employ different concepts, categorisations and terminology [20, 18, 4, 19, 13]. Hence, one of the main topics of the seminar was dedicated to disentangling this mixture of terminology and concepts to reach a clear definition of uncertainty interaction and uncertainty-related concepts.

**State-of-the-art methods for taming uncertainty and their integration.** The need to manage different types and sources for uncertainty in SiS has fostered the development of various ad-hoc methods that address the specific issues induced by these uncertainties, often isolated from interactions with other sources of uncertainty, and for individual applications [1, 5, 15, 7]. These ad-hoc methods include: (i) representation of uncertainty and its propagation, (ii) analysis techniques able to provide guarantees about system behavior under prescribed levels of uncertainty, for instance using quantitative verification techniques such as probabilistic model checking [10], and (iii) mitigation of the effects of uncertainty, e.g., through adaptation techniques that are able to anticipate disruptions and mitigate their effects proactively [15, 7] . The discussion of these methods during the seminar have been key for understanding their merits, limitations, and for making progress towards a common conceptual framework that allows their integration and exploitation.

**Uncertainty interaction classification and patterns.** To develop a common conceptual framework for managing uncertainty interactions, there is a need to identify the common categories of uncertainty interaction that affect the quality (e.g., safety, security) of SiS across strategic domains. Hence, this proposed line of discussion included: (i) identification of common types of uncertainty interaction across different domains and classes of system (e.g., ML-enabled systems, CPS), (ii) how to devise appropriate notations and patterns to represent such types of uncertainty interactions, as well as mitigation actions and strategies for their impact on system properties.

## Overview of the meeting

The meeting started with an introduction from the organizers, describing the topic of the meeting, the overall plan, the expected outcomes of the meeting, and a round of lightning introductions.

After a session with introductory talks that set the context for the seminar, there was a round of debate to identify discussion topics. Based on the discussion, as well as on the topics of interests shared by participants prior to the seminar, three main topics were identified:

- **Understanding**: Characterization, representation and categorization of uncertainty interaction.
- **Analysis**: Quantification and propagation of uncertainty, combination, compositional analysis.
- **Mitigation**: Methods and patterns for mitigation, explainability.

After that, participants split for discussion into three focus groups that corresponded to each of the identified topics. Focus groups kept on working independently and reporting back to the whole group in plenary sessions during three iterations in the remaining days of the seminar.

During the second day, there was a plenary session devoted to discussing case studies, during which Alessandro Papadopoulos prensented an extensive case study in the area of robotics. There was also additional discussion regarding the usefulness of putting together a catalog of case studies based on the compiled set of 17 examples provided by seminar participants prior to the meeting. An additional focus group who volunteered to curate the set of case studies and package it as an artifact after the seminar was identified.

In the following, we first report the meeting schedule and summary of the context-setting talks. Then, we include three sections that provide a summary of the discussion, findings, and challenges identified during the seminar by each of the focus groups (understanding, analysis, and mitigation).

# Overview of Talks

## Uncertainty Interaction in Robotic Application

Shaukat Ali, Simula Research Laboratory

This talk presents an LLM-based framework for systematic uncertainty identification and quantification in robotics. Within the EU RoboSAPIENS project, multiple robotic use cases are considered, including an autonomous vessel, a warehouse robotic swarm, human–robot interaction, and an industrial laptop-disassembly robot. Practitioners interact with LLMs via role-based prompts, guided questionnaires, and an uncertainty taxonomy to elicit, refine, and rank potential uncertainties along dimensions such as nature, temporal characteristics, scope, propagation, and resolution. Results show that domain experts agreed with roughly 63–88% of LLM-generated responses across different models (e.g., GPT-4o, Claude 3.5 Sonnet, LLaMA 3.3, Mistral, Perplexity Sonar), indicating that expert-guided prompting can effectively support uncertainty analysis. The work delivers two key artifacts: an uncertainty taxonomy for self-adaptive robotics and RoboULM, a tooling framework that operationalizes this taxonomy for real-world robotic systems.

## Proactive Adaptation and Uncertainty Reduction

Gabriel Moreno, SEI/CMU

This talk presents a decision-making framework for proactive, latency-aware self-adaptation under uncertainty. Instead of reacting after problems appear, the system anticipates future conditions using a stochastic environment model based on DTMCs and Markov decision processes. Adaptation tactics (e.g., VM provisioning, firmware updates, human actions) have non-negligible latency, so deciding early is crucial to keep response times and other quality attributes within acceptable bounds.

The talk then introduces uncertainty reduction as a first-class tactic: the system may spend resources (and risk user annoyance) to better characterize its current or future state, which can improve adaptation decisions. Using probabilistic model checking, it derives optimal policies that balance the cost of actions, including uncertainty reduction, against expected benefits. Case examples such as intrusion detection show that reducing uncertainty is not always optimal; the framework decides at run time when and how to measure, adapt, or do nothing.

## Addressing the Uncertainty Interaction Problem in Software-Intensive Systems

Javier Cámara, Universidad de Málaga

This presentation addresses the Uncertainty Interaction Problem (UIP) in software-intensive systems, where multiple sources of uncertainty—ranging from environment behavior to sensing errors, model abstractions, ML components, and third-party services—interact in unexpected ways and jeopardize system

goals. Through examples such as mobile robotics and the adaptive news service Znn.com, the authors illustrate how individually manageable uncertainties can combine to cause failures, as when discretization and sensing errors jointly prevent necessary adaptations. The talk argues that uncertainty interactions must be treated as a first-class concern in system development and that modeling is central to detecting, representing, and managing these interactions. It promotes reusable modeling patterns, runtime models, and cross-paradigm interoperability as foundations for more robust and efficient software-intensive systems.

## List of Participants

- Prof. Shaukat Ali, Simula Research Laboratory
- Dr. Nelly Bencomo, Durham University
- Dr. Matteo Camilli, Politecnico di Milano
- Prof. Javier Cámara, Universidad de Málaga
- Dr. Nicolas Cardozo, University of Los Andes
- Dr. Ivana Dusparic, Trinity College Dublin
- Dr. Fuyuki Ishikawa, National Institute of Informatics
- Prof. Anne Koziolek, Karlsruhe Institute of Technology
- Dr. Livia Lestingi, Politecnico di Milano
- Dr. Jialong Li, Waseda University
- Prof. Marin Litoiu, York University
- Prof. Raffaela Mirandola, Karlsruhe Institute of Technology
- Dr. Gabriel Moreno, SEI/CMU
- Prof. Alessandro Papadopoulos, Mälardalen University
- Prof. Danilo Pianini, University of Bologna
- Dr. Diego Pérez, Linnaeus University
- Prof. Ralf Reussner, Karlsruhe Institute of Technology / FZI Forschungszentrum Informatik
- Dr. Genaína Rodrigues, University of Brasilia
- Dr. Patrizia Scandurra, University of Bergamo
- Dr. Vincenzo Scotti, Karlsruhe Institute of Technology
- Prof. Gabriel Tamura, ICESI University
- Dr. Kenji Tei, Institute of Science Tokyo
- Dr. Christos Tsigkanos, University of Athens
- Dr. Karthik Vaidhyanathan, IIIT Hyderabad
- Dr. Gricel Vazquez, University of York
- Dr. Norha Villegas, Universidad ICESI / University of Victoria

## Meeting Schedule

**Check-in Day: March 1 (Sun)**

- Welcome Banquet

**Day 1: March 2 (Mon)**

- Opening, Meeting aims and planned outcomes
- Round of short introductions
- Short context-setting talks
  - Shaukat Ali, Simula Research Laboratory
  - Gabriel Moreno, SEI/CMU
  - Javier Camara, Universidad de Málaga
- Round 1 break-out groups:
  - Topic definition and grouping
  - Alignment of vocabulary, concepts and challenges across disciplines
  - Specific technical aspects of UiP

**Day 2: March 3 (Tue)**

- Round 1 break-out groups: Work continues and Plenary reports
- Talks about use cases: Christos Tsigkanos and Alessandro Papadopoulos
- Round 2 break-out groups: Topic definition, use case definition, and uncertainty patterns

**Day 3: March 4 (Wed)**

- Round 2 break-out groups: Work continues and Plenary reports
- Group Photo Shooting (during break)
- Excursion: Kamakura tour (Jomyoji and Hokokuji temples)
- Main Banquet: Kaiseki-style Dinner in Kamakura

**Day 4: March 5 (Thu)**

- Planning of next steps: UIP research agenda, seminar report, and summary article
- Planning of joint publications: Target venues and contributor groups
- Wrap up and closing

# Summary of discussions

## Understanding

The Understanding Uncertainty Interactions group focused on establishing terminology, developing conceptual frameworks, and bridging perspectives from software engineering and control theory.

### Terminology Clarification

Much of our discussion centered on clarifying the distinction between related concepts: "sources of uncertainty," "uncertainties," and "uncertainty interaction." We established that *sources of uncertainty* refer to the origins or locations where uncertainty arises (e.g., environment, sensors, models, goals), while *uncertainties* refer to the epistemic or aleatory limitations in knowledge or predictability at those locations.

We developed a layered conceptual framework distinguishing propagation, confluence, interaction, and relevance:

**Propagation and Translation** The flow of uncertainty from one location to another. We distinguished two sub-cases: *within-formalism propagation*, the classical case where uncertainty flows from inputs to outputs within a single modeling formalism (e.g., Monte Carlo simulation through a differential equation model), and *cross-formalism translation*, where uncertainty must be mapped across different model representations, for instance, when coupled models maintain consistency through transformation rules. Cross-formalism translation is non-trivial because different formalisms may use fundamentally different uncertainty representations (probability distributions, intervals, fuzzy sets), and there is no general-purpose mechanism for converting between them. Despite this added difficulty, both sub-cases concern the *flow* of uncertainty rather than its *combination*; they are preconditions for interaction but not interaction itself.

**Confluence** The point where multiple propagation paths converge at a shared variable. This is where interaction becomes structurally possible.

**Interaction** The combined effect of multiple uncertainties at a confluence point. Even additive composition counts as interaction, because the combined effect can differ qualitatively from what any individual uncertainty produces alone.

**Relevant interaction** An interaction where the combined effect causes the system to cross a decision boundary that would not be crossed by any proper subset of those uncertainties or where their combination makes crossing much more likely, and where crossing this boundary affects the system's ability to satisfy its requirements.

This framing clarifies that propagation (even across formalisms) is not yet interaction; interaction requires multiple uncertainties to meet at a shared variable where their combined effect influences decisions.

### Bridging Control Theory and Software Engineering Perspectives

Our group included control theory experts alongside software engineering researchers. The control theory perspective introduced concepts such as parametric uncertainty (unknown but fixed parameters), stochastic uncertainty (random disturbances), and adversarial uncertainty (worst-case assumptions). We developed a diagram incorporating the plant, controller, disturbances, sensor noise, and learning components, which helped establish common ground between the communities.

From control theory, we discussed the role of feedback loops in managing uncertainty: continuous monitoring and averaging over time windows reduce the impact of measurement noise and prevent uncertainty from accumulating unboundedly. This contrasts with situations where decisions are made on single measurements, where sensing uncertainty has more impact.

A key insight from this cross-disciplinary exchange was that both communities deal with fundamentally similar challenges, reasoning about system behavior under incomplete knowledge, but frame them differently. Control theorists tend to work with well-defined mathematical models of the plant and characterize uncertainty parametrically, allowing for formal robustness guarantees. Software engineers, by contrast, more frequently face structural uncertainty (is the model itself adequate?) and must contend with systems whose architecture and environment evolve at runtime. The separation principle from control theory, which states that state estimation and control can be designed independently under certain conditions, resonated with software engineering participants as a potentially useful decomposition strategy, though its applicability to self-adaptive systems with runtime model changes remains an open question.

### The Decision Threshold Problem

Our discussions returned multiple times to decision discontinuities at thresholds. We explored how discretization combined with sensing uncertainty can lead to situations where the probability of making a wrong decision increases. Using examples from server response time adaptation to robot navigation, we analyzed how uncertainties interact at decision points.

### Normative versus Quantitative Requirements

A distinction emerged regarding the treatment of normative requirements (privacy, safety, legal compliance) versus quantifiable requirements (response time, throughput). The group recognized that normative requirements often cannot be monitored continuously in the same manner as quantitative requirements. They may require different monitoring loops (slower, event-triggered) and different mitigation strategies. Edge cases and emergency situations require different handling than normal operation, and these should be treated separately in system design.

## Analysis

Previous work had identified challenges in the area of analysis of uncertainty interaction in self-adaptive systems, namely *identifying uncertainty interactions*,

*quantifying the impact of uncertainty interactions*, and *determining the uncertainty interactions that require mitigation* [2].

The discussion in the analysis group discovered that these challenges are not only interrelated but also constitute key steps in the analysis of uncertainty interaction. The discussions then focused on two topics: (i) a workflow with steps for the analysis of uncertainty interaction, and (ii) an approach to formally carry out some of these steps.

Overall, the discussions helped clarify the main steps involved in analyzing uncertainty interactions and highlighted how rigorous modeling can support this analysis. The proposed workflow provides a structured process for identifying, filtering, and quantifying uncertainty interactions, while the formalization offers a systematic way to reason about how uncertainties affect requirement satisfaction. At the same time, the challenges identified above indicate that further work is needed to refine the workflow, develop scalable analysis techniques, and better support the identification and interpretation of uncertainty interactions in complex software systems.

## Mitigation

The Mitigation Group focused the discussions on how uncertainties interact in software-intensive systems in order to conceptualize how to mitigate the adverse effects of these interactions. Building on prior work on the Uncertainty Interaction Problem (UIP) [3], the group adopted the view that uncertainties rarely occur in isolation but instead interact in ways that can significantly affect system behavior and the satisfaction of system goals. Previous research has identified interaction categories such as dominance, augmentation, and conflict, which characterize how different uncertainties may influence each other and compound their effects on system outcomes [2]. These interaction types provided an important conceptual foundation for the group's discussions on uncertainty mitigation. Building on this foundation, the group explored how reasoning about these interaction categories could inform mitigation decisions through the notion of a mitigation function. The mitigation function is intended as a conceptual mechanism that relates the characteristics of interacting uncertainties (e.g., interaction category, uncertainty types, system design characteristics, and goal criticality) to potential mitigation strategies and their expected impact on system quality attributes. In this way, the mitigation function provides a structured mechanism for analyzing mitigation trade-offs and guides the identification of mitigation tactics appropriate for specific uncertainty interaction scenarios in software-intensive systems, an effort that is currently ongoing within the group.

Having the mitigation function conceptualized at a very high level of abstraction, the group identified the necessity of providing guidelines for gradually achieving a more concrete definition for it. However, given that achieving this goal for software-intensive systems in general was a monumental task, the group agreed on focusing only on a category of software systems as a first step. Thus, based on Gen-AI systems taken from the uncertainty examples documented by the seminar participants, and considering them as relevant and of-current-interest representatives of complex systems, the group chose them as the category to find and explore the key elements the mitigation function should have to consider.

From this agreement, the discussions centered on how uncertainty manifests in Gen-AI based systems. Compared to traditional software-intensive systems, Gen-AI based architectures introduce additional sources of uncertainty, including data-source variabilities, stochastic model outputs, hallucinations, incomplete reasoning chains, context window limitations, and memory staleness, among others [12, 16]. These uncertainties often occur simultaneously and may interact across components and layers of systems that incorporate large language models (LLMs), agentic architectures, and adaptive control mechanisms as part of the system architecture. Understanding these interactions is essential because their combined effect may significantly impact system properties such as reliability, performance, and functional correctness, and in effect, they are useful to finding what is required to mitigate them and to guide mitigation decisions.

Another central topic in the discussions was that mitigation cannot be understood solely as a run-time activity. Instead, it must be addressed throughout the software lifecycle, including requirements specification, architectural design, and runtime adaptation. Runtime mitigation depends on design-time decisions, such as defining monitoring capabilities, adaptation tactics, and controllable parameters that the system can adjust during operation. This observation aligns with previous research on self-adaptive systems [11, 6], which emphasizes the importance of design-time preparation for run-time adaptation.

In this context, the group also examined the role of adaptive feedback loops as mechanisms for uncertainty mitigation. In classical self-adaptive systems, monitoring, analysis, planning, and execution form a continuous loop that detects uncertainties and triggers mitigation actions [11, 22]. Similarly, in Gen-AI based systems, uncertainties may arise not only in the managed system but also in the managing system itself, for example, when LLM-based planning components generate unreliable reasoning. This observation motivated discussions about mitigation across different levels of system dynamics, building on the DYNAMICO reference model [22], which distinguishes among control objectives, adaptation mechanisms, and monitoring infrastructures.

With all of the aforementioned elements, the discussion advanced towards what could be the specification of the mitigation function, as the mechanism intended to provide a structured way of reasoning about how interacting uncertainties influence system qualities and how mitigation strategies might address them. Conceptually, this mitigation function takes as input parameters characteristics such as the interaction category between uncertainties, the types and sources of uncertainty, the architectural design/model of the system, and the criticality of system goals to fulfill. As output, it should return the mitigation tactics to apply in the system, along with the expected impacts on relevant quality attributes (work still in progress). This formulation would provide a conceptual mechanism for reasoning and estimating trade-offs when selecting mitigation strategies for uncertainty interactions in complex Gen-AI based systems.

# Summary of new findings

## Understanding

### The Uncertainty Envelope Concept

The group developed and refined the *uncertainty envelope* concept, drawing an analogy to acceptability envelopes used in control and systems engineering. We defined an uncertainty envelope as the admissible bounds of uncertainty within which system guarantees (safety, performance, trust) remain valid. As a working proposal, the group identified several dimensions that characterize such an envelope:

- **Belief confidence**: The system's confidence in its current state estimation
- **Model error bounds**: Acceptable deviation between model predictions and reality
- **Prediction error tolerance**: Acceptable inaccuracy in forward projections
- **Goal satisfaction probability**: Likelihood of achieving stated objectives
- **Interaction consistency**: The degree of agreement between estimates derived from different models or views of the system. In systems where multiple coupled models coexist, inconsistencies between their uncertainty characterizations can themselves be a signal that the envelope is being approached or violated.

This decomposition emerged from group discussion and should be understood as a preliminary structuring rather than a definitive framework. Crossing the uncertainty envelope boundary does not necessarily imply immediate safety violation. Instead, it indicates that guarantees derived from the current model or belief state may no longer hold. Envelope violations can therefore serve as triggers for adaptation, enabling proactive rather than purely reactive responses.

### Relationship Between Envelopes

We discussed the relationships between several envelope concepts:

1. **Acceptability Envelope**: The mathematical representation of requirement boundaries in the system's state space. This defines the region where the system satisfies its requirements.
2. **Uncertainty Envelope**: The bounds within which our knowledge is trustworthy. This concerns epistemic confidence in our understanding of the system state.
3. **Viability Zone**: A concept from control theory describing whether the system can project a viable path remaining within acceptable bounds given current knowledge.

These concepts are related but distinct: the uncertainty envelope affects our confidence in whether we are within the acceptability envelope, and viability describes our ability to plan paths that remain in acceptable regions.

### Composition and Allocation

One finding concerns the challenge of translating between levels of abstraction. As a first approximation, the problem resembles performance optimization, where reducing any of several contributing factors can improve overall performance: reducing uncertainty at any of several sources can reduce overall decision uncertainty. However, the analogy has important limits. Uncertainty interactions can be more than additive: As our discussion of augmentation and conflict patterns showed, uncertainties may amplify each other non-linearly or even push in opposing directions. This non-additivity makes the composition problem substantially harder than in the performance case, because reducing one source by a given amount does not yield a predictable reduction at the system level. There is no established methodology for determining which uncertainty source to address first, or how bounds at the system level translate back to bounds at the individual uncertainty level. This "reverse calculation" from system requirements to individual uncertainty tolerances remains an open problem.

It is useful to distinguish two logically different tasks. First, the forward problem: given uncertainty descriptions for individual components, determine the induced uncertainty at the composite-system level. Second, the inverse problem: given an admissible uncertainty envelope for the overall system, determine how much uncertainty can be assigned to each component while preserving that envelope. These two problems are not symmetric: propagation upward is an analysis task, while allocation downward is typically a design or certification task, often non-unique and constraint-sensitive.

### Abstraction and Layering

We recognized that uncertainty analysis must proceed at multiple abstraction levels. When analyzing a robot's position uncertainty, we need not model the aerodynamics of the wheels or low-level motor control uncertainties. The system must be decomposed into layers, with uncertainty envelopes defined at each layer, and methods developed for composing these across layers.

### Surrogates and Indirect Observation

A recurring theme in the discussions was the role of surrogate measurements when direct observation of an uncertain quantity is infeasible. In control theory, these concerns are formalized through the concepts of *observability* (whether system states can be inferred from outputs) and *identifiability* (whether model parameters can be uniquely determined from data). The canonical example discussed was tire-on-asphalt grip: the friction coefficient cannot be measured directly during operation, but related observable quantities (wheel slip, lateral acceleration) can serve as proxies. This pattern, using an observable surrogate to estimate a hidden variable, is well established in both control theory (state observers, Kalman filters) and software engineering (runtime monitoring with derived metrics). However, the group noted that introducing surrogates adds a new layer of uncertainty: the mapping from the surrogate to the quantity of interest is itself uncertain, and this "meta-uncertainty" must be accounted for in the overall uncertainty envelope. Whether surrogate-based observation reduces or merely transforms uncertainty depends on the fidelity of the surrogate

relationship, which brings us back to the model error bounds discussed in the uncertainty envelope concept.

## Analysis

The workflow in Figure 1 starts by identifying uncertainty sources for the system and adding them to a list. In a second step, the list is expanded by adding tuples of uncertainty sources that interact. Next, we need to determine which of the tuples in the list—including 1-tuples—require mitigation. To have flexibility, we defined this as a two-step process. First, there is a qualitative assessment of the impact of the uncertainty tuples. Those deemed irrelevant are removed from the list. The tuples that remain require quantification, which is done in the second step. Those that have a small quantified impact are removed from the list. The final product is a list of uncertainty tuples that require mitigation.

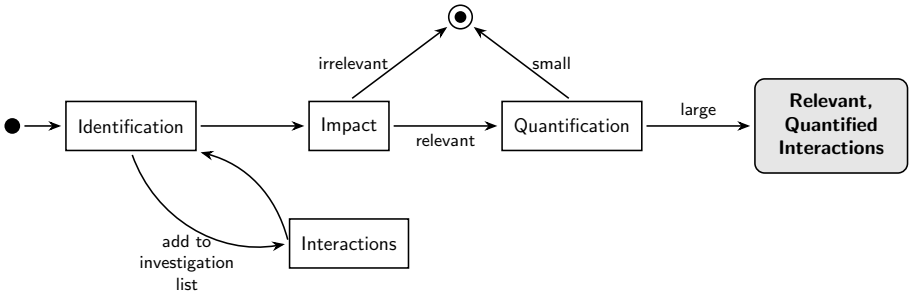


Figure 1: Analysis workflow for identifying, evaluating, and quantifying relevant uncertainty interactions.

The starting point of the formalization is the Jackson and Zave's model of a system satisfying the requirements under a number of assumptions [8]. In this conceptual framework, requirements engineering is characterized as a logical relationship between three sets of statements: the *requirements* $R$, the *domain assumptions* $A$, and the *system* (specification) $S$. The logical relationship can be expressed as:

$$(A, S) \models R$$

We can assume that each of these sets is composed of individual elements, such as:

$$(\{a_1, \ldots, a_n\}, \{s_1, \ldots, s_m\}) \models \{\rho_1, \ldots, \rho_l\}$$

meaning that the requirements are entailed by the conjunction of the domain assumptions and the system properties. In other words, if the assumptions about the environment hold and the system behaves according to its specified properties, then the requirements will be satisfied. Domain assumptions, in this context, represent properties of the environment that are *assumed to hold* independently of the system, such as physical laws, user behavior patterns, or operational constraints.

This formulation provides a useful foundation for reasoning about uncertainty in general, as well as uncertainty interactions. Indeed, uncertainty may

arise from incomplete knowledge about the environment, environmental changes, nondeterministic system behavior, or partially known system properties.

After performing the step in the workflow that identifies uncertainty sources, elements of $A$ and $S$ that are affected by uncertainty can be explicitly annotated. We denote such elements with a question mark (?) to indicate that their truth, value, or behavior may vary or may not be fully known.

$$(\{a_1, \ldots, a_j{}^?, \ldots, a_n\}, \{s_1, \ldots, s_k{}^?, \ldots, s_m\}) \models \{\rho_1, \ldots, \rho_l\}$$

Each identified uncertainty source can therefore be associated with elements of $A$ or $S$. Interactions between uncertainties can be understood as combined deviations that affect the validity of the entailment relation. To analyze combined deviations, we parameterize each assumption $a_j{}^?$ and system specification clause $s_k{}^?$ that is subject to uncertainty with a vector of parameters $\bar{\delta}$ capturing the factors that influence the uncertainty. This yields the parameterized uncertainty sources $a_j(\bar{\delta}^a_j)$ and $s_k(\bar{\delta}^s_k)$. Collectively, the parameters of all uncertainty sources form a vector $\Delta$ representing a point in the *uncertainty space*.

For each requirement $\rho_i$, we then define a function $v_{\rho_i}(\Delta)$ that measures the *degree of violation* of the requirement as a function of the uncertainty parameters. Intuitively, this function quantifies how far the system behavior deviates from satisfying the requirement $\rho_i$ under a given combination of uncertainty parameter values. Following the Signal Temposal Logic robustness convention, values $v_{\rho_i}(\Delta) < 0$ indicate that the requirement is violated to a certain degree, values $v_{\rho_i}(\Delta) >= 0$ indicate that the requirement is satisfied.

Notice that $v_{\rho_i}$ can be interpreted as a signed distance function with respect to the satisfaction boundary of requirement $\rho_i$ in the uncertainty space. In particular, values above 0 correspond to points inside the satisfaction region, while negative values correspond to points in the violation region. The magnitude of $v_{\rho_i}(\Delta)$ therefore provides a measure of how far a given uncertainty configuration is from the satisfaction boundary of the requirement.

This interpretation allows us to reason about requirement robustness with respect to uncertainty. Configurations of parameters $\Delta$ that lie close to the threshold represent situations in which small deviations in uncertainty parameters may cause the requirement to become violated. However, interactions between uncertainty sources may still be relevant even when they do not lead to requirement violations. In particular, certain combinations of uncertainty parameters may significantly influence the value of $v_{\rho_i}(\Delta)$ while remaining within the satisfaction region. Such interactions may reveal *sensitivity* of the system to particular combinations of uncertainties and therefore represent valuable information for system understanding, maintenance, and future evolution under changing environmental conditions.

To capture such effects, the analysis can also consider how $v_{\rho_i}(\Delta)$ varies with respect to the uncertainty parameters. Measures such as covariance, sensitivity, or other dependence metrics can be used to identify regions of the uncertainty space where the combined variation of parameters produces significant changes in the violation function. Sudden changes, spikes, or strong correlations in $v_{\rho_i}(\Delta)$ may indicate the presence of interactions between uncertainty sources, even if the requirement remains satisfied.

## Mitigation

The discussions in the Mitigation Group led to several preliminary findings regarding the mitigation of uncertainty interactions in Gen-AI based software-intensive systems.

A first finding is the recognition that uncertainty interactions represent a central challenge in Gen-AI based systems, where multiple uncertainty sources may dynamically appear simultaneously across different components and layers of the system architecture. While traditional software systems already exhibit uncertainties arising from environmental variability, sensing inaccuracies, or incomplete system models, Gen-AI based architectures introduce additional sources such as stochastic inference processes, hallucinations, variability in reasoning outputs, context window limitations, and evolving internal memory states. These uncertainties may interact in ways that amplify their effects on system behavior, making it insufficient to treat them independently.

A second finding concerns the importance of explicitly reasoning about uncertainty interaction categories when analyzing and mitigating uncertainty. The interaction categories previously identified in the literature (i.e., dominance, augmentation, and conflict) provide a useful conceptual basis for understanding how uncertainties influence each other and how their combined effects may affect system goals [2, 3]. The characteristics of these interaction categories may also have an effect on the application of mitigation strategies.

A third finding emerging from the discussions is the potential role of the mitigation function as a conceptual mechanism for structuring mitigation decisions. The mitigation function was discussed as a way to relate the characteristics of interacting uncertainties to potential mitigation strategies and their expected effects on system quality attributes. Although still conceptual and under development, this idea provides a promising direction for organizing mitigation knowledge and supporting reasoning about mitigation trade-offs in Gen-AI based systems and software-intensive systems in general.

A fourth observation is that mitigation decisions are strongly influenced by system architecture and lifecycle considerations. Effective mitigation strategies depend not only on run-time mechanisms but also on design-time decisions that determine the system's monitoring capabilities, controllable parameters, available mitigation tactics, and operational constraints.

Finally, from the examples provided by the participants of the seminar, the discussions highlighted that mitigation strategies in Gen-AI based systems are likely to take the form of architectural and operational tactics. Examples discussed during the seminar included techniques such as routing requests across models, managing context representations, improving monitoring and observability, augmenting system knowledge, or combining multiple models [17, 9].

# Identified issues and future directions

## Understanding

### CH-U1: Cross-Formalism Translation

A challenge is translating uncertainty representations across different formalisms. Probability distributions, intervals, fuzzy sets, and qualitative descriptors each capture different aspects of uncertainty, but there is no methodology for consistent translation between them. This is problematic when different system components use different uncertainty representations, and their interactions must be analyzed.

This challenge is compounded by the fact that different system components are typically designed and maintained by different engineering teams, each with their own modeling conventions. As recent surveys on uncertainty representation in cyber-physical systems have shown [14], the field suffers from a "jungle of terminology" where different frameworks use overlapping or contradictory terms for the same concepts, and identical terms for different concepts. Establishing consistent cross-formalism translation therefore requires not only technical mechanisms for converting between representations, but also community-wide agreement on the semantics of uncertainty categories, a goal to which this seminar contributes.

### CH-U2: The Chicken-and-Egg Problem

In uncertainty interaction analysis, one faces a prioritization challenge: which uncertainties are relevant to analyze? The set of potentially interacting uncertainties is large, but computational resources for analysis are finite. We need heuristics or methods for identifying which uncertainty combinations merit analysis, but these methods themselves require some understanding of how uncertainties interact.

### CH-U3: Composition and Decomposition Methodology

While we established that uncertainty envelopes are needed at multiple abstraction levels, we lack methods for (1) decomposing system-level uncertainty requirements into component-level uncertainty budgets, (2) composing component-level uncertainty characterizations into system-level guarantees, and (3) determining how tightly coupled the uncertainty envelopes at different levels must be.

This decomposition challenge is not unique to uncertainty; it mirrors well-known difficulties in compositional verification and assume-guarantee reasoning. However, uncertainty adds a complicating dimension: the "contracts" between layers must themselves be expressed in terms of uncertainty bounds, specifying what uncertainty a component assumes from its environment and what bounds it guarantees to its clients, and violating such a contract does not produce a binary failure but rather a gradual degradation of guarantees. Methods from robust control (e.g., structured singular value analysis) handle this for specific mathematical formulations, but no general methodology exists for the heterogeneous, multi-formalism models typical of software-intensive systems.

### CH-U4: Measurable versus Non-Measurable Uncertainties

Some uncertainties can be quantified through measurement or statistical estimation; others resist quantification (e.g., uncertainty about whether a model structure is correct, uncertainty about unknown unknowns). Current frameworks handle measurable uncertainties better than non-measurable ones. Methods are needed for incorporating non-quantifiable uncertainties into analysis frameworks.

### CH-U5: Normative Requirement Integration

The distinction between normative and quantitative requirements raises questions: How should systems be architected to handle both types? When do edge-case handling mechanisms get triggered? How do we avoid conflicts between normative constraints and optimization objectives? This area requires further investigation, potentially drawing on communities (ethics, law, safety engineering) beyond software engineering and control theory.

### CH-U6: Envelope Violation Response

Responding to envelope violations involves three distinct challenges: detecting that a violation has occurred, analyzing its cause and severity, and selecting an appropriate mitigation. When an uncertainty envelope is violated (indicating that guarantees may no longer hold), what should the system do? Options include increasing sensing frequency to reduce uncertainty, triggering adaptation to move to more robust operating points, alerting human operators, or entering safe degraded modes.

The appropriate response likely depends on the type and severity of envelope violation, but guidelines are lacking.

### CH-U7: Tooling and Practical Application

While our discussions advanced conceptual understanding, tooling for uncertainty interaction analysis remains limited. Future work could develop domain-specific modeling languages for uncertainty representation, analysis tools for uncertainty propagation and interaction, pattern libraries for uncertainty interaction scenarios, and integration with existing model-driven engineering toolchains.

A concrete step toward better tooling would be to extend existing model-driven engineering frameworks with first-class support for uncertainty annotations and propagation. Standards such as the OMG's Precise Semantics for Uncertainty Modeling (PSUM) provide a foundation, and recent work on unified conceptual models for uncertainty in cyber-physical systems [14] offers a harmonized terminology that could underpin such tooling. The group discussed the potential for pattern libraries that capture recurring uncertainty interaction scenarios together with their known mitigation strategies, analogous to design pattern catalogs in software engineering.

### CH-U8: Community Building

Bridging the control theory and software engineering communities requires continued collaboration. The terminology and intuitions differ, and mutual under-

standing developed during this seminar should be maintained through collaboration, joint publications, and future meetings. Concrete problems for future work include: compositional uncertainty across heterogeneous formalisms, inverse uncertainty budgeting from system-level envelopes to component-level tolerances, detection and management of envelope violations, and benchmark case studies that make competing notions of uncertainty interaction comparable across domains.

## Analysis

Although the discussions were engaging and productive, they also highlighted several challenges that remain to be addressed.

#### CH-A1: Completeness of uncertainty sources

The envisioned workflow assumes that uncertainty sources affecting the elements of $A$ and $S$ can be identified. In practice, however, uncertainty sources may be difficult to enumerate exhaustively, particularly when they arise from evolving environments or incomplete domain knowledge.

#### CH-A2: Explosion of the uncertainty space

The step that expands the list of parametric uncertainties into interacting elements may lead to a combinatorial explosion as the number of identified uncertainties grows. High-order interactions can quickly become infeasible to analyze exhaustively.

#### CH-A3: Definition of filtering criteria

The workflow relies on an initial qualitative filtering step to discard uncertainty tuples that appear irrelevant. However, defining consistent criteria for such qualitative judgments can be challenging. Different analysts may reach different conclusions about the relevance of certain tuples.

#### CH-A4: Definition of the "degree of violation"

The formalization assumes that for each requirement $\rho_i$ it is possible to define a function $v_{\rho_i}(\Delta)$ that quantifies the degree of violation. In practice, however, such functions are often implicit or only partially observable, as they may depend on complex system behavior or simulation outputs. Making these functions fully explicit and analyzable can therefore be challenging.

#### CH-A5: Identification of meaningful interaction metrics

While interactions may be detected through regions where $v_{\rho_i}(\Delta) > 0$ or through sensitivity and covariance analyses, determining appropriate metrics to characterize the strength and significance of interactions remains an open problem. Different metrics may reveal different types of interactions.

#### CH-A6: Robustness under evolving requirements and environments

The analysis assumes a fixed set of requirements and domain assumptions. In practice, both requirements and environmental conditions may evolve over time, potentially altering the structure of the uncertainty space and the interactions between uncertainties.

#### CH-A7: Interpretability of analysis results

Even when interactions are automatically detected through analysis of $v_{\rho_i}(\Delta)$, interpreting the results and translating them into actionable mitigation strategies may be challenging for system designers and operators.

### Mitigation

The group made progress on the identification of some important challenges and open research questions during the discussions.

#### CH-M1: Uncertainty interaction characterization in Gen-AI based Systems

One major issue concerns the lack of systematic methods for characterizing uncertainty interactions in modern Gen-AI based systems. Although the community has introduced the concept of the Uncertainty Interaction Problem and identified interaction categories, there is limited guidance on how to identify, model, and reason about such interactions in complex Gen-AI based architectures [**?**, 21].

#### CH-M2: Guidance to select mitigation strategies

A second issue relates to the absence of practical guidance for selecting or defining mitigation strategies. While many potential mitigation techniques exist, system architects need a way to determine which tactics, or combination thereof, are appropriate for a given uncertainty interaction scenario.

#### CH-M3: Architectural complexity of Gen-AI based Systems and Uncertainty Interactions

A third challenge arises from the increasing architectural complexity of Gen-AI based software systems. These systems often combine multiple models, external services, data sources, and orchestration mechanisms. As a result, uncertainties may propagate across components and layers of the system architecture.

#### CH-M4: Trade-off evaluation

The evaluation of mitigation strategies and their impact on system qualities remains a challenge. Since mitigation actions may introduce trade-offs across quality attributes such as performance, reliability, cost, safety, or usability, systematic approaches are needed to assess these trade-offs.

Future directions identified by the group include:

- Developing models and taxonomies of uncertainty interactions tailored to Gen-AI based software-intensive systems.
- Refining the concept of the mitigation function to support systematic reasoning about mitigation decisions and quality trade-offs.
- Identifying and organizing mitigation tactics that address different uncertainty interaction scenarios.
- Investigating architectural patterns and design strategies that support uncertainty mitigation across the system lifecycle.
- Developing evaluation approaches to assess mitigation strategies with respect to system quality attributes and operational constraints.

# References


[1] Burton, S., Herd, B.: Addressing uncertainty in the safety assurance of machine-learning. Frontiers in computer science **5**, 1132580 (2023)

[2] Cámara, J., Calinescu, R., Cheng, B.H.C., Garlan, D., Schmerl, B., Troya, J., Vallecillo, A.: Addressing the uncertainty interaction problem in software-intensive systems: Challenges and desiderata. In: Proceedings of the 25th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS 2022). ACM (2022). https://doi.org/10.1145/3550355.3552438

[3] Cámara, J., Troya, J., Vallecillo, A., Bencomo, N., Calinescu, R., Cheng, B.H.C., Garlan, D., Schmerl, B.: The uncertainty interaction problem in self-adaptive systems. Software and Systems Modeling **21**(4), 1277–1294 (Aug 2022). https://doi.org/10.1007/s10270-022-01037-6, `https://doi.org/10.1007/s10270-022-01037-6`

[4] Esfahani, N., Malek, S.: Uncertainty in self-adaptive software systems. In: Software engineering for self-adaptive systems II: International seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised selected and invited papers. pp. 214–238. Springer (2013)

[5] Fang, X., Calinescu, R., Paterson, C., Wilson, J.: Presto: Predicting system-level disruptions through parametric model checking. In: Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 91–97 (2022)

[6] Hezavehi, S.M., Weyns, D., Avgeriou, P., Calinescu, R., Mirandola, R., Perez-Palacin, D.: Uncertainty in self-adaptive systems: A research community perspective. ACM Transactions on Autonomous and Adaptive Systems **15**(4), 10:1–10:36 (2021). https://doi.org/10.1145/3487921

[7] Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A framework for proactive self-adaptation of service-based applications based on online testing. In: European Conference on a Service-Based Internet. pp. 122–133. Springer (2008)

[8] Jackson, M., Zave, P.: Deriving specifications from requirements: an example. In: Proceedings of the 17th International Conference on Software Engineering. p. 15–24. ICSE '95, Association for Computing Machinery, New York, NY, USA (1995). https://doi.org/10.1145/225014.225016, `https://doi.org/10.1145/225014.225016`

[9] Jain, H., Pandey, D., Vaidhyanathan, K.: Calm: A self-adaptive orchestration approach for qos-aware routing in small language model-based systems. arXiv preprint (2026)

[10] Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: International School on Formal Methods for the Design of Computer, Communication and Software Systems, pp. 220–270. Springer (2007)

[11] de Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N.M., Vogel, T., Weyns, D., et al.: Software engineering for self-adaptive systems: A second research roadmap. In: Software Engineering for Self-Adaptive Systems II, Lecture Notes in Computer Science, vol. 7475, pp. 1–32. Springer (2013)

[12] Li, J., Zhang, M., Li, N., Weyns, D., Jin, Z., Tei, K.: Generative ai for self-adaptive systems: State of the art and research roadmap. ACM Transactions on Autonomous and Adaptive Systems **19**(3), 1–60 (2024)

[13] Mahdavi-Hezavehi, S., Avgeriou, P., Weyns, D.: A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements. In: Managing trade-offs in adaptable software architectures, pp. 45–77. Elsevier (2017)

[14] Mäkelburg, J., Perez-Palacin, D., Mirandola, R., Acosta, M.: Surveying uncertainty representation: A unified model for cyber-physical systems. arXiv preprint arXiv:2503.23892 (2025), technical University of Munich / Karlsruhe Institute of Technology / Linnaeus University

[15] Moreno, G.A., Cámara, J., Garlan, D., Schmerl, B.: Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In: Proceedings of the 2015 10th joint meeting on foundations of software engineering. pp. 1–12 (2015)

[16] Nascimento, N., Alencar, P., Cowan, D.: Self-adaptive large language model (llm)-based multi-agent systems. In: Proceedings of the IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C 2023). pp. 104–109. IEEE (2023)

[17] Pandey, D., Gupta, V., Singhal, P., Vaidhyanathan, K.: Polaris: Is multi-agentic reasoning the next wave in engineering self-adaptive systems? arXiv preprint (2025)

[18] Perez-Palacin, D., Mirandola, R.: 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. pp. 3–14 (2014)

[19] Ramirez, A.J., Jensen, A.C., Cheng, B.H.: A taxonomy of uncertainty for dynamically adaptive systems. In: 2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). pp. 99–108. IEEE (2012)

[20] Rotmans, J., Sluijs, J., Asselt, M., Janssen, P., Krauss, M.: A conceptual basis for uncertainty management (2003)

[21] Troya, J., Moreno, N., Bertoa, M.F., Vallecillo, A.: Uncertainty representation in software models: A survey. Software and Systems Modeling **20**(4), 1183–1213 (2021). https://doi.org/10.1007/s10270-020-00842-1

[22] Villegas, N.M., Tamura, G., Müller, H.A., Duchien, L., Casallas, R.: Dynamico: A reference model for governing control objectives and context relevance in self-adaptive software systems. In: Software Engineering for

Self-Adaptive Systems II, Lecture Notes in Computer Science, vol. 7475, pp. 265–293. Springer (2013). https://doi.org/10.1007/978-3-642-35813-5$_1$1