Addressing the Uncertainty Interaction Problem in Software-intensive Systems: Challenges and Desiderata

Javier Cámara
jcamara@uma.es
ITIS Software, University of Málaga
Spain

Radu Calinescu
radu.calinescu@york.ac.uk
University of York
United Kingdom

Betty H.C. Cheng
chengb@msu.edu
Michigan State University
USA

David Garlan
Bradley Schmerl
{garlan,schmerl}@cs.cmu.edu
Carnegie Mellon University
USA

Javier Troya
Antonio Vallecillo
{jtroya,av}@uma.es
ITIS Software, University of Málaga
Spain

ABSTRACT

Software-intensive systems are increasingly used to support tasks that are typically characterized by high degrees of *uncertainty*. The modeling notations employed to design, verify, and operate such systems have increasingly started to capture different types of uncertainty, so that they can be explicitly considered when systems are developed and deployed. While these modeling paradigms consider different sources of uncertainty individually, these sources are rarely independent, and their interactions affect the achievement of system goals in subtle and often unpredictable ways. This vision paper describes the problem of *uncertainty interaction* in software-intensive systems, illustrating it on examples from relevant application domains. We then identify key open challenges and define desiderata that future modeling notations and model-driven engineering research should consider to address these challenges.

CCS CONCEPTS

• **Software and its engineering → System description languages**;

KEYWORDS

Uncertainty interaction, Modeling notations, Patterns

ACM Reference Format:
Javier Cámara, Radu Calinescu, Betty H.C. Cheng, David Garlan, Bradley Schmerl, Javier Troya, and Antonio Vallecillo. 2022. Addressing the Uncertainty Interaction Problem in Software-intensive Systems: Challenges and Desiderata. In *ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22), October 23–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3550355.3552438

MODELS '22, October 23–28, 2022, Montreal, QC, Canada

ACM ISBN 978-1-4503-9466-6/22/10…$15.00
https://doi.org/10.1145/3550355.3552438

1 INTRODUCTION

Systems in which software influences the design, construction, operation, and evolution of the system in an essential manner (i.e., *software-intensive systems*) are widely used to support tasks across key domains that include manufacturing, communications, transportation, health and entertainment. Many of these systems are characterized by high degrees of *uncertainty* induced by multiple sources that include complex interactions among physical elements and software (e.g., in cyber-physical systems), humans in the loop, lack of control over third-party components and services (which may reside in the cloud), and machine learning-enabled components [15]. Given the multiple sources of uncertainty that may appear at different levels of abstraction, different temporal frequencies, with a spectrum of impact, an emergent challenge is how to handle interaction among these disparate sources and types of uncertainty. This paper identifies challenges posed by the uncertainty interaction problem (UIP) and proposes key desiderata for the model-driven engineering community to address these challenges.

A growing body of work addresses the problem of mitigating the effects of uncertainty in software-intensive systems by employing constructs that enable engineers to explicitly capture different types and sources of uncertainty in models [1, 2, 7, 8, 12, 14, 18, 22–25, 27, 29–31]. While these modeling paradigms consider different sources of uncertainty individually, these are rarely independent, and their interactions can affect the achievement of system goals in subtle and often unpredictable ways. Sometimes, these uncertainty interactions can be captured using state-of-the-art modeling notations in an ad-hoc manner for specific systems and types of interaction. However, these approaches are error-prone and labor-intensive. To improve the resilience and reliability of future software-intensive systems, engineers need systematic approaches to represent and manage uncertainty interactions.

This paper posits that modeling plays a central role in managing uncertainty interactions, which should be considered as a first-class systems development problem. Explicitly capturing interactions in patterns that can be reused to represent and manage uncertainty interactions across systems, using models at run time to detect potential interactions among different sources of uncertainty, or facilitating the interoperability of different modeling paradigms to

enable compositional analysis of the effects of uncertainty interactions on the satisfaction of system requirements are examples of how future modeling paradigms can foster more efficient development paradigms of higher quality software-intensive systems.

To achieve this vision, the modeling community needs to address questions such as: What modeling patterns can be developed to represent different types of uncertainty interactions? How can modeling notations enable better compositional analysis of system properties subject to uncertainty interactions? What types of automated analysis can be applied to models to detect interactions? How can models at runtime be used to manage UIPs as the operational context changes? In this vision paper, we discuss some of the challenges that arise from these questions related to the representation, analysis, mitigation, and exploration of uncertainty interactions through the lens of modeling. Previously, we identified UIP as a problem from a self-adaptive systems perspective [10]. In contrast, this vision paper considers UIP from a purely modeling perspective, stating the need for a paradigm shift in the modeling of the numerous software-intensive systems affected by this problem. We then map –for the first time– the high-level challenges from [10] to concrete modeling challenges and use them to distill a set of desiderata for new modeling paradigms that the MDE community needs to work on in order to tackle the UIP problem.

2 EXPLORING UNCERTAINTY INTERACTION

In this section, we explore the uncertainty interaction problem in the context of Znn.com, which is a simple news service that uses MAPE-K [16] to deal with varying workloads through different tactics [26]. As Figure 1 illustrates, Znn.com has multiple servers (some of which are inactive), a load balancer, and a database, which are monitored by the MAPE loop to update the models of the system and environment to make informed decisions. To accommodate changing workloads, the self-adaptive layer executes different tactics such as activating servers, enabling a CAPTCHA, or changing the quality of the contents served, among others.

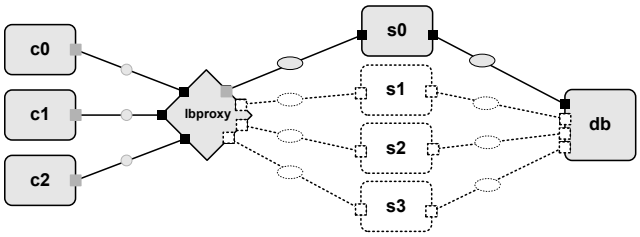

Figure 1: Znn.com system architecture.

2.1 Sources of Uncertainty

Let us consider a situation in which Znn.com receives a spike in workload with a high request arrival rate and has to decide which of the tactics to trigger (if any), in order to continue to satisfy system goals. We may face different uncertainties in such a situation:

2.1.1 Model. The abstraction level of environment and system properties (e.g., coarse-grained discretization of numerical variables like the request arrival rate) of models in the knowledge base of the self-adaptive layer of Znn.com are potential sources of uncertainty.

2.1.2 Adaptation Functions. The exact outcome of executing an adaptation tactic (e.g., activating a server) is unknown, in terms of precise improvements on throughput or response time. Sensing is also imperfect so measurements taken, e.g., at the load balancer to gauge the request arrival rate, may be imprecise (averaging windows are typically employed to mitigate quick fluctuations). The time that it takes to activate a server is also subject to uncertainty and can fluctuate depending on environment conditions.

2.1.3 Goals. In Znn.com, dependencies among goals are not captured explicitly. Instead, the selection of adaptations to satisfy extra-functional goals (i.e., cost and user experience optimization, security) is driven by utility functions that do not clarify under what conditions security has priority over cost, and vice versa.

2.1.4 Environment. The evolution of the request arrival rate (e.g., whether it is going up, down, or remains stable) can be predicted in some cases, but only to some degree of certainty, e.g., using a time series predictor [20]. This is important for anticipating usage peaks when more resources may be needed. The nature of the access of clients to the system (i.e., whether they are legitimate clients or bots attempting to perform a DoS attack) is unknown. We need this information to decide if preventive measures such as the use of CAPTCHAs, are worth the inconvenience to most users.

2.1.5 Resources. Servers may fail, and considering their expected failure rate may provide more realistic estimates when sizing the system. However, predictions based on high uncertainty can increase the number of servers required, thereby unnecessarily increasing the overall costs. Similarly, the availability of additional servers that can be activated to spread the workload may only be known with some degree of certainty, because they may not be available at all times. Their performance can also vary, introducing new sources of uncertainty when predicting system performance.

2.1.6 Managed system. This uncertainty is caused by the complexity and dynamicity of the managed system, which hinders the estimations of its behavior. The system and its parts may also evolve, incorporating new elements whose behavior was not considered when the system was designed. These parts may also fail or behave in erratic or unexpected manners, e.g., the performance of the database can degrade, suffer attacks or intermittent failures.

2.2 Uncertainty Interaction

We are interested in the cases where two or more uncertainties interact. We next describe in detail an example of such situations and their effects. In the MAPE-K adaptation loop of Znn.com, the analysis stage determines if the triggering of adaptations is required, for instance, when an invariant is violated. One typical example is the invariant stating that the current system response time r should always be below a threshold R_{max}. When this invariant is violated, the interaction of the following sources of uncertainty, among others, might affect the correct operation of the analysis:
Adaptation functions. Sensing of the response time property could yield values within some range $[\hat{r}_{min}, \hat{r}_{max}]$ that contains the ground truth value r. If the observed value is above threshold ($\hat{r} \geq R_{max}$), but the real value is not ($r < R_{max}$), this will lead to a false positive that will trigger an adaptation planning and execution cycle when it is not really needed, increasing the cost of operating

the system without need. In the symmetric case ($\hat{r} < R_{max} \leq r$), the adaptation cycle will not be triggered even if it is really needed, causing an unnecessary degradation of performance.

Model abstraction. The coarse-grained discretization of the response time property in the managed system model can also result in undesired adaptation triggers or the lack of required adaptations when the discretized value r_d of the property is on the other side of the threshold, when compared to the ground truth value r. Hence, if we discretize with a granularity of $\mu = 20ms.$, response times measured where $|r - R_{max}| \leq 20$ could be problematic.

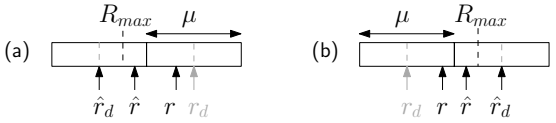

Figure 2: Model-Sensing interaction: (a) causing spurious adaptation (b) preventing required adaptation.

These two sources of uncertainty can interact in more than one way. In the situation illustrated in Figure 2(a), the ground truth value of the response time r is observed with some error. Both r and the observed value $\hat{r}$ are below the threshold R_{max}. Dashed gray lines represent the values that the variable can take in the discrete abstraction of the system model. Any real value observed in the surrounding box is snapped to that discrete value. Even if the observed value of the variable $\hat{r}$ contains some error, this error on its own would not be enough to trigger an undesired adaptation. However, due to the discretization of the observed response time values, $\hat{r}$ will be snapped to $\hat{r}_d$, which is above R_{max}, triggering a spurious adaptation. Note that without the error induced by observation, the discretization process on its own would not have been enough to trigger this adaptation, given that r would have been snapped to r_d, which is below R_{max}. Alternatively, Figure 2(b), shows the case in which both the ground truth and the observed value of the response time are above threshold R_{max}. In this case, the discretization process snaps the value of $\hat{r}$ to $\hat{r}_d$, preventing the triggering of adaptation in a situation in which it would have been required. Analogously to the situation described in (a), neither the discretization process nor the observation error on their own would have been enough to prevent the execution of the required adaptation. Instead, it is the compound effect of both sources of uncertainty that causes the undesired situation.

2.2.1 Exploring uncertainty interaction through modeling. To assess the impact of uncertainty interaction on the satisfaction of system goals, we formalize some of the elements of Znn.com in a high-level model of the system that we analyze using the probabilistic model checker PRISM [17]. The model includes processes that capture the behavior of different system components (Figure 3), which take turns at each execution step of the scenario:

(1) ***System***. Models the managed system and includes a variable that corresponds to the ground truth value of the response time r, which is generated at each execution step according to an arbitrary function that depends on the elapsed time t in the scenario.

(2) ***Sensor***. Models the behavior of the sensor that captures the response time and contains a single variable $\hat{r}$ (observed value of r). At each execution step, the sensor takes the value of r and generates a value of $\hat{r}$ that contains some error according to a normal distribution of mean r and standard deviation controlled by an error parameter of the model designated by κ.

(3) ***Discretization***. Captures the behavior of the abstraction process that takes place when the values of the variables observed on the continuous spectrum have to be rounded to their closest values on the discrete grid that can be represented by system models. Concretely, this process contains a variable $\hat{r}_d$, which is obtained from the observed value of response time $\hat{r}$ according to the function:

$$disc(\hat{r}) = \arg\min_{x \in [\mathbb{R}]_\mu} (|\hat{r} - x|), \tag{1}$$

where $[\mathbb{R}]_\mu = \{x \in \mathbb{R} \mid x = i\mu,\ i \in \mathbb{Z},\ \alpha \leq x \leq \beta\}$ is the discrete set of values that the response time variable r can take, μ is a parameter that controls the granularity of the discretization (smaller μ means higher model fidelity), and $[\alpha, \beta]$ is the range of the variable.

(4) ***Adaptation***. Based on the discretized value of the observed response time $\hat{r}_d$, the adaptation process decides if adaptation should be triggered (i.e., if $\hat{r}_d \geq R_{max}$). If that is the case, the value of the ground truth response time variable r is reduced for the next execution step. In our model, we simply assume that this reduction corresponds to cutting by half the existing response time.

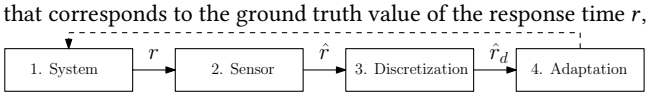

Figure 3: Uncertainty interaction model structure.

The main objective of Znn.com is maximizing accrued utility, which we assume to have a value inversely proportional to system response time throughout execution (in reality, multiple variables contribute to utility, such as cost and fidelity of the contents served to clients, but we abstract them away for brevity). To determine the impact of uncertainty interaction on utility maximization, our model includes a reward structure that computes the accrued error over the execution of the scenario between the utility computed based on the ground truth, and the discretized observed values of the response time (r and $\hat{r}_d$, respectively):

$$\Delta U \equiv \int_o^T |r - \hat{r}_d| \tag{2}$$

To understand how the uncertainty due to model abstraction and imperfect sensing interact, we analyzed four variants of a scenario of duration $T = 20$ time units, where the range of the response time variable r is $[\alpha, \beta] = [0, 200]$ ms., and the threshold for adaptation is $R_{max} = 40$ ms. These four variants correspond to the following cases: (i) no uncertainty, in which there is full model fidelity ($\mu = 1$) and no sensor observation error ($\kappa = 0$), (ii) sensing uncertainty, in which $\mu = 1$ and $\kappa = 20$, (iii) model abstraction uncertainty, in which $\mu = 30$ and $\kappa = 0$, and (iv) combined sensing and model abstraction uncertainty, where $\mu = 30$ and $\kappa = 20$.

Results of the analysis (Figure 4) show that, as expected, the accrued difference in utility ΔU for the execution of scenarios of different durations without uncertainty ($\mu = 1, \kappa = 0$) is always 0, given that the ground truth, observed values, and observed discretized values always coincide (i.e., $\forall t_{\{0..T\}}, r(t) = \hat{r}(t) = \hat{r}_d(t)$). The case with model uncertainty ($\mu = 30, \kappa = 0$) shows that ΔU now increases progressively with the duration of the scenario as multiple instances of systems states in which $\hat{r}_d \neq \hat{r}$ are given,

reaching a maximum of $\Delta U = 217$ when $T = 20$. The case in which uncertainty is due to imperfect sensing ($\mu = 1, \kappa = 20$) shows that ΔU also increases over time in states in which $\hat{r} \neq r$, reaching a value of $\Delta U = 380$ when $T = 20$. This form of uncertainty has an impact that is in this case ≈ 1.75 times that of model uncertainty. Finally, the case in which we have both sensing and model uncertainty ($\mu = 30, \kappa = 20$) shows that impact on ΔU also accrues progressively, but at a higher rate, compared to any of the other uncertainties considered individually. The impact on ΔU reaches a maximum of 584, which is ≈ 1.53 times the impact of sensing uncertainty, and ≈ 2.69 times the impact of model uncertainty. These results indicate that different sources of uncertainty can compound and have a notable impact on the achievement of system goals.

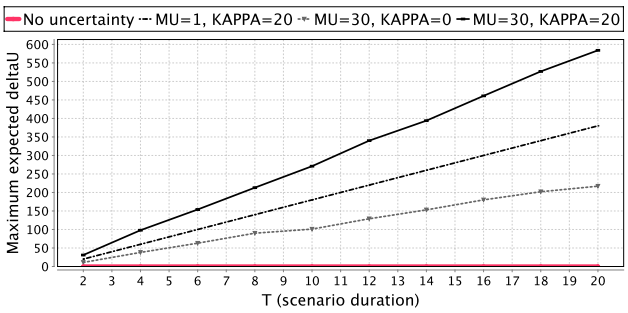

Figure 4: Impact of 2-way Model-Sensing uncertainty interaction on accrued system utility in Znn.com.

2.2.2 Other examples of uncertainty interaction. Beyond the example discussed in this section, there are multiple other sources of uncertainty that can interact in software-intensive systems. These interactions can involve an arbitrary number of uncertainty sources, but we present only pairwise interactions for illustration purposes:

Time and measurement uncertainty. The uncertainty in the time that spans between the triggering of an adaptation like activating a server in Znn.com, which can take between seconds and minutes, and the instant in which its effects take place, can interact with the imperfect sensing of system variables. A delayed or anticipated triggering of the adaptation due to error in response time measurement can compound with adaptation latency, causing the system to execute adaptations that are not required yet, or not needed anymore after a transient change.

Model abstraction and physical environment. An autonomous service robot operating in a healthcare facility is tasked with navigating between two locations and 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 of 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, when 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 remaining battery and presence of people can allow re-planning that might still lead to arriving at the target location. However, the same situation with uncertainty in the remaining battery, or the presence of humans who can delay the progress of the robot through a corridor can lead to generating a plan based on unrealistic estimates prone to fail.

2.2.3 Limitations. The exercise of building the PRISM model used to explore our example already points to some limitations that current modeling notations exhibit when it comes to capturing and reasoning about uncertainty interactions. The first one is the inability to represent variables with an arbitrary precision. Probabilistic model checkers like PRISM that enable exhaustive analysis can only handle discrete variables, so representing uncertainty due to discretization processes is cumbersome and possible only in a limited number of cases. This also applies to time, which has to be represented in a discrete fashion. Although some formalisms like Probabilistic Timed Automata (PTA) and Continuous-Time Markov Chains (CTMC) support capturing continous time, their ability to capture time uncertainty is quite limited. Uncertainty due to structural variability is also a limitation of this model, which cannot systematically represent alternative system structures that satisfy non-trivial structural invariants, such as those that can be imposed by notations such as OCL [21]. Furthermore, all variables in the model had to be captured at the same level of abstraction, thus leaving out uncertainties that might propagate and interact with uncertainty sources at other levels of abstraction.

3 CHALLENGES

Based on these examples and others, we next identify a set of challenges related to the modeling, analysis, mitigation, and exploration of uncertainty in software-intensive systems.

3.1 Modeling Challenges

Providing means to model uncertainty interactions is key to analyzing, mitigating and exploring the effects of such interactions. We identify the following challenges in this context.

M1. Improving current modeling notations to represent and manage uncertainty. Although the representation of uncertainty in software models is improving [27], we are still far from having sufficiently good solutions. For example, explicitly representing the uncertainties identified in Section 2—not to mention their interactions—is not possible with most of the modeling notations currently available. We need more expressive modeling notations to represent and manage uncertainty at different levels of abstraction.

There is an ongoing effort within the OMG to develop a metamodel for the precise specification of uncertainty in software models (PSUM) [22], which could provide the required set of concepts and relations to represent uncertainty. Although this OMG proposal could serve to enrich current modeling notations with such concepts, it may remain at a too high level of abstraction to capture the nuances in some uncertainties—such as some of the ones described in Section 2. Moreover, in addition to representing uncertainty, it is just as important to be able to manage, propagate, and reason about it. Links to the right tools for uncertainty analysis (propagation, estimation, bounding, management) are also required.

M2. High-level modeling. Beyond representing the individual uncertainties a system might be subject to and their sources, it would be desirable to indicate at a very high level of abstraction what

uncertainty interactions will likely take place among specific uncertainty sources. However, models at too high level of abstraction are typically unable to capture and manage the individual uncertainties with the required degree of precision to faithfully represent the real systems. Hierarchies of related models that capture uncertainties at different levels of abstraction could be an option to explore.

M3. Uncertainty interaction modeling. Different uncertainty sources are normally represented using different notations and at possibly different levels of abstraction, which may jeopardize reasoning about their interactions. It is important to be able to represent *how* (and not only *if* as in Challenge M2) these uncertainty sources are likely to interact. Different situations might occur when two or more uncertainties interact. For instance, in the physical world we see how uncertainties cancel each other out or at least offset each other's effects, or in Section 2.2 we have seen examples of the effect of two uncertainty sources when interacting. Identifying the most suitable representation and underlying logic to capture the interactions and possible emergent uncertainties is a difficult challenge, given that no uniform notation able to represent all types of uncertainties, and their interactions, seems possible. In addition, models at different levels of abstraction can be generally very difficult to combine (cf. [28]), which may hinder the representation and reasoning about the combination of the uncertainties that each of them describes, and of their possible interactions.

3.2 Analysis Challenges

Developed independently by multiple research communities, the wide range of methods for the analysis of uncertainty differ significantly in the types of models, techniques and results with which they operate. Several major challenges for the analysis of uncertainty interactions stem from these differences.

A1. Integration of uncertainty analyses. The analyses of different classes of uncertainty yield results that are often qualitatively different, precluding a meaningful integration. As an example, the statistical analysis used to analyse measurement uncertainty typically produces confidence intervals (in frequentist statistics) or credibility intervals (in Bayesian statistics), while the probabilistic modelling techniques used to establish dependability and performance properties of software systems operate with point estimates of the probabilities of transition between pairs of system states.

A2. Complex relationships between uncertainty analyses. Because of the nonlinear dependencies between different types of uncertainty, selecting the right levels of precision for each individual analysis is challenging. The conservative analysis of each source of uncertainty can lead to a combined overall result that cannot guarantee compliance with the dependability, performance, and resource use of the system because it is overly conservative. While analyses carried out more precisely avoid this limitation, they require the acquisition of more data, larger models, and significantly higher computational overheads and latencies.

A3. Multiple changes in uncertainty. Changes—whether gradual or sudden—in the uncertainty characteristics (e.g., probabilities) associated with a single source of uncertainty are already difficult to handle in the analysis of software systems. Detecting changes due to multiple sources of uncertainty and reapplying the analyses required to establish their combined impact on a software-intensive system necessitates techniques for compositional and incremental analysis that are largely unavailable today.

3.3 Mitigation Challenges

Beyond the ability to model and analyze uncertainty interaction, an important challenge is to be able to reason about ways to mitigate — or reduce – the impact of uncertainty interaction [9, 19]. Such mitigations can take the form of actions that explicitly target sources of uncertainty and their interaction by allocating resources to narrow the degree of uncertainty [9]. For example, increasing the number of active sensors on a robot can decrease uncertainty in the position of the robot. However, given that there are costs associated with such reductions, a key question to answer is whether the increase in expected system utility is justified by those costs. This leads to a significant challenge for modeling: namely, being able to predict the consequences of uncertainty reduction and to make tradeoffs between the costs and benefits of doing so.

When carrying out uncertainty reduction it may be possible to exploit some common modeling patterns of uncertainty interaction to reason about appropriate mitigation actions. Examples of modelling challenges to support such inference include at least the following three forms of interaction and mitigation.

Mt1. Dominance. It may be that one form of uncertainty dominates (or subsumes another) and that by reducing the dominant uncertainty dimension necessarily reduces the other. For example, uncertainty about a robot's location that is reduced through turning on a headlight might completely dominate the uncertainty associated with a robot's intrusiveness.

Mt2. Augmentation: Some uncertainties may be highly correlated with each other (but neither dominating the other), so that uncertainty reduction of one type may positively affect reduction of another. For example, uncertainty about a robot's location may be augmented by uncertainty about the safety of the robot, and that by reducing location uncertainty (e.g., turning on a spotlight in a dark area) the other uncertainty is also partially reduced. Models should be able to help determine the degree of such correlation.

Mt3. Conflicts. Reduction of uncertainty of one form may increase uncertainty in another form. For example, devoting resources to reduce uncertainty about robot location, may remove resources from another aspect of a robot's behavior, increasing its uncertainty. For example adding sensors to improve localization may increase uncertainty about a robot's power reserve. Models of uncertainty reduction should expose and allow reasoning about such tradeoffs.

3.4 Exploration Challenges

Given the range of uncertainty sources, the different types of uncertainty interaction, and the potentially negative consequences of uncertainty interaction, it becomes increasingly important to have effective proactive uncertainty interaction exploration capabilities. Taking a model-driven engineering approach to UIP provides a number of opportunities with respect to scalability and automated analysis capabilities. For UIP exploration, however, the aforementioned challenges become accumulative, with an added complexity posed by the human in the loop. We identify three key challenges.

E1. Exploring Multiple Sources of Uncertainty. With multiple modeling notations to capture the different types of uncertainty,

each of which captures different levels of abstraction, temporal granularity, and precision, the corresponding but disparate uncertainty analysis techniques make it difficult to explore different scenarios of uncertainty interaction. As such, in addition to overcoming the analysis challenges (i.e., ***A1-A3***), UIP exploration requires the ability to understand tradeoffs between different configurations of uncertainty sources, which will be limited by the respective modeling expressiveness and analysis techniques. How can the different uncertainty models incorporate and process historic data, synthetic data (e.g., adversarial, diverse known unknown environment, probabilistic), to provide predictive capabilities? For example, how can frameworks, such as MODA [11] be leveraged and customized to provide data-centric exploration of heterogeneous models to explore uncertainty interaction scenarios. In Section 2.2.1, we illustrated how formal analysis techniques can be used to detect uncertainty interaction, but we are also interested in more open-ended exploration – how can the disparate uncertainty models be integrated for exploration in search-based frameworks, such as game theory, probabilistic analysis, and general "what if?" scenarios?

E2. Executable Environments. Many of the sources of uncertainty require dynamic, behavioral analysis. For uncertainty interaction exploration purposes, executable models are invaluable. "What if? scenario exploration requires a broad range of configuration capabilities to handle the different sources of uncertainty and range/types of data. Digital Twin frameworks have largely focused on system functionality and performance, but how they can be revised to explore uncertainty interaction explicitly remains a challenge.

E3. UIP Visualization. Exploring uncertainty interaction will involve humans in the loop to both configure uncertainty interaction model-based exploration (e.g., identify relevant uncertainty models, identify relevant data, identify appropriate uncertainty analysis techniques), but also to interpret the resulting exploration results. As such, informative and interactive visualization techniques are needed to support uncertainty interaction exploration tools.

4 DESIDERATA

We build upon the challenges identified to pin down desiderata for modeling notations that can help to address the UIP and, ultimately, to provide better system assurances.

D1. Uncertainty Patterns. One way to address challenge ***M2*** is to develop a set of high-level patterns that account for the different uncertainty sources. These patterns should be able to express the uncertainty sources and interactions that can have an influence on the system, and appropriately combine them when required. Standardisation efforts, such as the PSUM, can help lay the groundwork for describing such patterns in a precise and uniform way. Such patterns could also provide uncertainty mitigation actions in the form of model transformations or analyses for reducing uncertainty, addressing mitigation challenges ***Mt1-Mt3***.

D2. Hierarchy of domain-specific modeling languages (DSMLs). Uncertainty sources and interactions can be specified in models either in an intrusive or non-intrusive way. The former approach requires extending the models' syntax and semantics with notations to represent uncertainties. A non-intrusive approach can be addressed with the use of external DSMLs that contain references to the system's models. Indeed, the uncertainty patterns mentioned above can be specified with appropriate high-level DSMLs, addressing challenge ***M2***. In fact, we can think of a hierarchy of DSMLs for uncertainty representation. High-level DSMLs would allow to indicate the likely occurrence of uncertainty sources and their interactions through the instantiation of patterns. Then, lower-level DSMLs must allow to specify how each uncertainty pattern is to be materialized in each specific system. The lower-level DSMLs would address challenge ***M3***. These DSMLs must allow describing how each uncertainty source might affect the system and modeling how the interaction of the uncertainty sources can affect the system, addressing challenge ***A1***. All uncertainty sources and interactions should be defined at the same level of abstraction.

The amount of information available during modeling can vary, so we need to be able to define uncertainty sources and interactions with different level of detail—this is akin to the concept of *multi-fidelity* in the field of digital twins [3]. This emphasizes the importance of a hierarchy of DSMLs: the user must employ the DSMLs at the appropriate level of abstraction. Hierarchical modeling languages can help to address challenge ***A2*** by reasoning at multiple levels about uncertainty mitigation by providing both abstract and concrete operations and ways to reason about cross-model interactions, where the impact of uncertainty reduction of one type can impact positively or negatively other types of uncertainty.

D3. Robustness. The analysis of uncertainty models should yield results that are not affected significantly by small changes and/or small estimation errors in the model parameters. This is required in order to limit the frequency with which the models need to be re-analysed after changes (challenge ***A3***) and to support "what if" scenario exploration (challenge ***E2***). Advances in confidence-interval model checking [5, 6] and the synthesis of robust software system designs [4] satisfy this desideratum for a single source of uncertainty, but cannot yet handle the uncertainty interactions.

D4. Reusability. Having a library of high-level patterns and non-intrusive DSMLs as described above, these should be reusable across different systems and models. This involves challenges ***M1***, ***M2***, and ***M3***. Such reusability also applies to adressing challenges across across analysis, mitigation, and exploration.

D5. Compositionality. This is another feature that the DSMLs should have. In order to more effectively use a hierarchy of reusable DSMLs, these should be defined in a compositional fashion. Compositionality would enable us to address challenges such as ***M3*** by facilitating the description of interactions of uncertainty from multiple sources and at different levels of abstraction, and ***A3*** by enabling the compositional and incremental analyses of qualitatively different aspects of the system (e.g., uncertainty due to structural variability vs. stochastic behavior).Compositional reasoning should also be extended to understanding how uncertainty mitigations (***Mt1-Mt3***) performed on one model affect uncertainties in another.

D6. Configurability. In order to support model exploration for different combinations and instantiations of uncertainty sources, the models and their respective analysis techniques should be independently and correlated configurable. With configurability, the reusable patterns can be better leveraged for both the models and the applicable analysis techniques to address interaction detection and mitigation, thereby addressing ***E1-E3*** and ***Mt1-Mt3***. For example, the different sources of uncertainty for the robot can be

configured (i.e., instantiated) to detect and explore different types of uncertainty interaction that go beyond pairwise interaction.

D7. Executability. To facilitate analysis and exploration, model executability is an important property. When combined with configurability, uncertainty interaction exploration can be dynamically observed, better informing the developer about how to mitigate the interaction problem(s), addressing both ***E1-E3*** and ***Mt1-Mt3***.

5 CONCLUSIONS

This paper described the *Uncertainty Interaction Problem*, focusing on it from a model-driven engineering perspective. The motivation has been illustrated with a detailed example in the context of an autoscaling news website infrastructure. We have outlined a set of challenges that concern the representation, analysis, mitigation, and exploration of interactions among uncertainties, as well as desiderata for future modeling notations to tackle those challenges.

Future work will involve building a repository of patterns to capture uncertainty interactions and their corresponding solutions. We will leverage the template from Gamma et al [13] and extend it to include fields specific to capture UIP concerns, including the sources of uncertainty, the level(s) of abstraction in which interaction occurs, effects of interaction and properties of interest.

We believe that this is an important area that deserves the attention of the modeling community, and that research in this direction will pave the way towards more holistic approaches that enable the construction of safer and more resilient software-intensive systems.

ACKNOWLEDGEMENTS

Work supported by the AAIP project 'Ambient Assisted Living for Long-term Monitoring and Interaction Integration', the European Commission (FEDER) and Junta de Andalucía projects MBT-I4A (P20-00067-FR) and EKIPMENT-PLUS (P18-FR-2895), by the Spanish Government (FEDER/MICINN – AEI) under project COSCA (PGC2018-094905-B-I00), by FCT through the CMU Portugal Program under Grant SFRH/BD/150643/2020, projects (POCI-01-0247-FEDER-045915, POCI-01-0247-FEDER-045907), by NASA (Award 80NSSC20K1720), and by EPSRC (EP/V026747/1). Cheng's work has been sponsored by NSF (DBI-0939454), Ford Motor Company, General Motors Research, ZF, and the AFRL (FA8750-16-2-0284, FA8750-19-2-0002). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copy-right notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as representing official policies or endorsements of research sponsors.

REFERENCES

[1] Naif Alasmari, Radu Calinescu, Colin Paterson, and Raffaela Mirandola. 2022. Quantitative verification with adaptive uncertainty reduction. *J. Syst. Softw.* 188 (2022).

[2] Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras. 2017. Probabilistic Complex Event Recognition: A Survey. *ACM Comput. Surv.* 50, 5 (2017), 71:1–71:31.

[3] Aitor Arrieta. 2021. Multi-Fidelity Digital Twins: a Means for Better Cyber-Physical Systems Testing? *CoRR* abs/2101.05697 (2021). arXiv:2101.05697

[4] Radu Calinescu, Milan Češka, Simos Gerasimou, Marta Kwiatkowska, and Nicola Paoletti. 2018. Efficient synthesis of robust models for stochastic systems. *J. Syst. Softw.* 143 (2018), 140–158.

[5] Radu Calinescu, Carlo Ghezzi, Kenneth Johnson, Mauro Pezzé, Yasmin Rafiq, and Giordano Tamburrelli. 2016. Formal verification with confidence intervals to establish quality of service properties of software systems. *IEEE Trans Reliab* 65, 1 (2016), 107–125.

[6] Radu Calinescu, Kenneth Johnson, and Colin Paterson. 2016. FACT: A probabilistic model checker for formal verification with confidence intervals. In *Proc. of TACAS (LNCS)*. Springer, 540–546.

[7] Radu Calinescu, Raffaela Mirandola, Diego Perez-Palacin, and Danny Weyns. 2020. Understanding uncertainty in self-adaptive systems. In *IEEE International Conference on Autonomic Computing and Self-organizing Systems*. 242–251.

[8] Javier Cámara, David Garlan, Won Gu Kang, Wenxin Peng, and Bradley R. Schmerl. 2017. *Uncertainty in Self-Adaptive Systems: Categories , Management , and Perspectives.* Technical Report CMU-ISR-17-110. Carnegie Mellon University. http://reports-archive.adm.cs.cmu.edu/anon/isr2017/CMU-ISR-17-110.pdf

[9] Javier Cámara, Wenxin Peng, David Garlan, and Bradley R. Schmerl. 2018. Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation. *Sci. Comput. Program.* 167 (2018), 51–69.

[10] Javier Cámara, Javier Troya, Antonio Vallecillo, Nelly Bencomo, Radu Calinescu, Betty H.C. Cheng, David Garlan, and Bradley Schmerl. 2022. The Uncertainty Interaction Problem in Self-Adaptive Systems. *Softw. Syst. Model.* 21, 4 (2022).

[11] Benoît Combemale, Jörg Kienzle, Gunter Mussbacher, Hyacinth Ali, Daniel Amyot, Mojtaba Bagherzadeh, Edouard Batot, Nelly Bencomo, Benjamin Benni, Jean-Michel Bruel, Jordi Cabot, Betty H. C. Cheng, Philippe Collet, Gregor Engels, Robert Heinrich, Jean-Marc Jézéquel, Anne Koziolek, Sébastien Mosser, Ralf H. Reussner, Houari A. Sahraoui, Rijul Saini, June Sallou, Serge Stinckwich, Eugene Syriani, and Manuel Wimmer. 2021. A Hitchhiker's Guide to Model-Driven Engineering for Data-Centric Systems. *IEEE Softw.* 38, 4 (2021), 71–84.

[12] Naeem Esfahani and Sam Malek. 2013. Uncertainty in self-adaptive software systems. In *SEfSAS II (LNCS)*, Vol. 7475. Springer, 214–238.

[13] Erich Gamma, Richard Helm, Ralph Johnson, Ralph E Johnson, John Vlissides, et al. 1995. *Design patterns: elements of reusable object-oriented software.* Pearson.

[14] Holger Giese, Nelly Bencomo, Liliana Pasquale, Andres J. Ramirez, Paola Inverardi, Sebastian Wätzoldt, and Siobhán Clarke. 2014. Living with Uncertainty in the Age of Runtime Models. In *Models@run.time (LNCS)*, Vol. 8378. Springer, 47–100.

[15] Sara M. Hezavehi, Danny Weyns, Paris Avgeriou, Radu Calinescu, Raffaela Mirandola, and Diego Perez-Palacin. 2021. Uncertainty in Self-Adaptive Systems: A Research Community Perspective. *ACM Trans. Auton. Adapt. Syst.* 15, 4 (2021).

[16] Jeffrey O. Kephart and David M. Chess. 2003. The Vision of Autonomic Computing. *Computer* 36 (2003). Issue 1.

[17] M. Kwiatkowska et al. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. of CAV'11 (LNCS)*, Vol. 6806. Springer.

[18] Sara Mahdavi-Hezavehi, Paris Avgeriou, and Danny Weyns. 2017. *A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements.* Morgan Kaufmann, Chapter 3, 45 – 77.

[19] Gabriel A. Moreno, Javier Cámara, David Garlan, and Mark Klein. 2018. Uncertainty reduction in self-adaptive systems. In *Proc. of SEAMS'18*. ACM, 51–57.

[20] Gabriel A. Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2015. Proactive Self-Adaptation under Uncertainty: A Probabilistic Model Checking Approach *(ESEC/FSE 2015)*. ACM, 1–12.

[21] Object Management Group. 2014. *Object Constraint Language (OCL) Specification. Version 2.4.* OMG Document formal/2014-02-03.

[22] Object Management Group. 2017. *Precise Semantics for Uncertainty Modeling (PSUM) RFP.* OMG Document ad/2017-12-1.

[23] Diego Perez-Palacin and Raffaela Mirandola. 2014. Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation. In *Proc. of ICPE'14*. ACM, 3–14.

[24] Andres J. Ramirez, Adam C. Jensen, and Betty H. C. Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *Proc. of SEAMS'12*. IEEE, 99–108.

[25] Ahmad M. Salih, Mazni Omar, and Azman Yasin. 2017. Understanding Uncertainty of Software Requirements Engineering: A Systematic Literature Review Protocol. In *Proc. of APRES'17*, Vol. 809. Springer, 164–171.

[26] Bradley R. Schmerl, Javier Cámara, Jeffrey Gennari, David Garlan, Paulo Casanova, Gabriel A. Moreno, Thomas J. Glazier, and Jeffrey M. Barnes. 2014. Architecture-based self-protection: composing and reasoning about denial-of-service mitigations. In *Proc. of HotSoS'14*. ACM, 2.

[27] Javier Troya, Nathalie Moreno, Manuel Bertoa, and Antonio Vallecillo. 2021. Uncertainty representation in software models: A survey. *Softw. Syst. Model.* 20, 4 (2021).

[28] Antonio Vallecillo. 2010. On the Combination of Domain Specific Modeling Languages. In *Proc. of ECMFA'10 (LNCS)*, Vol. 6138. Springer, 305–320.

[29] 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 (2003), 5–17.

[30] Man Zhang, Shaukat Ali, Tao Yue, Roland Norgren, and Oscar Okariz. 2019. Uncertainty-Wise Cyber-Physical System Test Modeling. *Softw. Syst. Model.* 18, 2 (2019), 1379–1418.

[31] Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. 2016. Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model. In *Proc. of ECMFA'16 (LNCS)*, Vol. 9764. Springer, 247–264.