Towards a Research Agenda for Understanding and Managing Uncertainty in Self-Adaptive Systems

Danny Weyns[1,2], Radu Calinescu[3], Raffaela Mirandola[4], Kenji Tei[5],
Maribel Acosta[6], Amel Bennaceur[7], Nicolas Boltz[8], Tomas Bures[9], Javier Camara[10],
Ada Diaconescu[11], Gregor Engels[12], Simos Gerasimou[3], Ilias Gerostathopoulos[13],
Sinem Getir Yaman[3], Vincenzo Grassi[14], Sebastian Hahner[8],
Paola Inverardi[15], Dimitri Van Landuyt[1], Rogerio de Lemos[16], Emmanuel Letier[17], Marin Litoiu[18],
Lina Marsso[19], Angelika Musil[1,20], Juergen Musil[20], Genaina Nunes Rodrigues[21], Diego Perez-Palacin[2],
Federico Quin[1], Patrizia Scandurra[22], Antonio Vallecillo[10], Andrea Zisman[7]

[1]KU Leuven; [2]Linnaeus University; [3]University of York; [4]Politecnico di Milano;
[5]Waseda University; [6]Ruhr University Bochum; [7]The Open University;
[8]Karlsruhe Institute of Technology; [9]Charles University, Prague; [10]Universidad de Málaga;
[11]Télécom Paris, IP Paris; [12]Paderborn University; [13]Vrije Universiteit Amsterdam;
[14]Università di Roma; [15]Gran Sasso Science Institute; [16]University of Kent;
[17]University College London; [18]York University; [19]University of Toronto;
[20]CDL-SQI, TU Wien; [21]University of Brasília; [22]University of Bergamo

danny.weyns@kuleuven.be, radu.calinescu@york.ac.uk, raffaela.mirandola@polimi.it, ktei@aoni.waseda.jp

ABSTRACT

Despite considerable research efforts on handling uncertainty in self-adaptive systems, a comprehensive understanding of the precise nature of uncertainty is still lacking. This paper summarises the findings of the 2023 Bertinoro Seminar on Uncertainty in Self-Adaptive Systems, which aimed at thoroughly investigating the notion of uncertainty, and outlining open challenges associated with its handling in self-adaptive systems. The seminar discussions were centered around five core topics: (1) agile end-to-end handling of uncertainties in goal-oriented self-adaptive systems, (2) managing uncertainty risks for self-adaptive systems, (3) uncertainty propagation and interaction, (4) uncertainty in self-adaptive machine learning systems, and (5) human empowerment under uncertainty. Building on the insights from these discussions, we propose a research agenda listing key open challenges, and a possible way forward for addressing them in the coming years.

1. INTRODUCTION

Modern software systems are expected to operate effectively and without interruption in uncertain real-world conditions. However, engineering such systems is highly challenging and poses an important rising topic of research in software engineering [40]. Possible causes of uncertainties include changes in the operational environment, dynamics in the availability of resources, and variations in user goals. A prominent approach to mitigate uncertainties is self-adaptation [78, 28, 93, 95] that is typically realized using a feedback loop. The feedback loop collects additional data about the uncertainties during operation, and uses these data to resolve uncertainties, to reason about the system, and to make decisions about how to reconfigure or adjust it to satisfy the user goals under changing conditions—or to degrade gracefully if necessary.

Over the past several years, researchers have started studying the notion of uncertainty in the context of self-adaptation. Different sources of uncertainty in self-adaptive systems have been identified, related to the system itself, the system goals, the execution context, and human aspects [75, 37, 69, 56]. An example of a concrete source of uncertainty within the system itself is incomplete knowledge, e.g., some parts of the system are missing at development time and may need to be added at runtime. An example of uncertainty related to system goals is a change of goals that may be induced by new customer needs, new regulations, or new market rules. An example of uncertainty in the execution context is imperfect monitoring due to noise in sensors, which are not ideal devices. An example of uncertainty related to human aspects is human behavior (e.g., interaction with the system), which can diverge from the expected behavior.

Researchers have studied multiple strategies for ”taming” uncertainty. One prominent approach is the use of first-class modeling concepts for specifying the goals of self-adaptive systems that are subject to uncertainties [98, 86]. Such modeling concepts typically relax the goals of self-adaptive systems under certain conditions. Making goal models based on such relaxed goals runtime entities allows a feedback loop system to take into account the uncertainties when making adaptation decisions. Another pioneering approach to mitigate uncertainties is the use of stochastic models with parameters that represent uncertainties [17, 31, 22]. These models are updated at runtime using actual observations of the uncertain parameters. The feedback loop system can then use the up-to-date models to reason about the system and make informed adaptation decisions. A typical example of such an approach is runtime quantitative verification, which uses Markovian models and model checking at runtime for analyzing quality properties (such as performance and reliability) and selecting adaptation configurations that achieve the required goals. To increase the scalability of this approach, several researchers have recently studied the use of statistical verification techniques [58]. Another approach explored by researchers is the use of controllers based on control theory to realize runtime adaptation [38, 83]. The task of the controller is to ensure that the output of the system follows a set-point goal while reducing the effects of uncertainty, which appear as disturbances, or as noise in variables or imperfections in the models of the system or environment used to design the controller.

While researchers have devoted substantial efforts to handling uncertainty in self-adaptive systems, a comprehensive understanding of the precise nature of uncertainty is lacking. This refers to what uncertainty is. Without a proper understanding of the notion of uncertainty, the research community is doomed to study approaches for mitigating uncertainty in an ad-hoc, case-by-case manner. This refers to how uncertainty can be mitigated in a principled and systematic way. As an example, the focus of research in self-adaptation so far has primarily been on parametric uncertainties, i.e., the uncertainties related to the values of unknown model parameters such as request arrival rates, component failure probabilities, or cost of operations. Little research has been devoted to so-called structural uncertainties, i.e. uncertainties related to the inability to accurately model real-world phenomena. Structural uncertainties may manifest themselves as model inadequacy, model bias, model discrepancy, etc. To tackle this problem, techniques from other fields may provide a starting point. E.g., in health economics, techniques such as model averaging and discrepancy modeling have been used to deal with structural uncertainties.

This paper reports the results of the Bertinoro seminar on Uncertainty in Self-Adaptive Systems that was organized at the University Residential Center of Bertinoro in Italy from 11 to 15 June 2023.[1] The overall goal of the seminar was to investigate thoroughly the what and how of uncertainty, and to outline a research agenda for addressing the challenges identified by this investigation over the coming years. Having a well-defined framework of what uncertainty is and how it affects self-adaptive systems will provide a foundation for future research efforts and lay the basis for developing reusable engineering solutions. One of the most prominent examples is the self-driving vehicle industry: autonomous vehicles must be prepared to face a multitude of uncertainties while driving. Massive efforts are devoted to this problem and any solution must be trustworthy, which requires a solid understanding of the uncertainties, and systematic and reusable solutions for handling them.

2. APPROACH

Motivated by the urgent need for a holistic and systematic discussion of the numerous challenges associated with tackling uncertainty, the seminar involved a group of 30 active leading researchers from different communities pursuing research on the causes, effects, and potential solutions of uncertainty. This resulted in a unique environment to discuss the central aspects of uncertainty, generate new insights and knowledge on uncertainty, and identify the key challenges for a future research agenda.

The seminar was organized into several parts. In preparation for the seminar, the participants were asked to complete a short survey to provide input on their interests and expectations for the seminar. The data collected from this survey was used to set the scene at the start of the seminar. Then, four participants were invited to provide 30-minute talks, introducing different research areas and perspectives on the notion of uncertainty.

After the first half day, we organized a plenary discussion to identify topics for in-depth analysis in breakout groups. This discussion was driven by the input obtained from the pre-seminar survey and the discussions during the morning. As a result, five groups were formed respectively on the topics: (1) agile end-to-end handling of uncertainties in goal-oriented self-adaptive systems, (2) managing uncertainty and risk for self-adaptive systems, (3) uncertainty propagation and interaction, (4) goal-driven self-adaptive machine learning systems, and (5) human empowerment under uncertainty.

The major part of the seminar consisted of breakout groups with short plenary reports to steer the work forward and align the activities of the breakout groups as needed. On the last day, a plenary session was organized to summarize the meeting findings and outline future actions for the participants and their research communities.

[1]https://www.ceub.it/events/event/bertinoro-meeting-on-uncertainty-in-self-adaptive-systems-usas/

3. PRE-SEMINAR SURVEY

Before the seminar, the organizers invited the participants to complete a short online survey to provide input on their interests and expectations for the seminar. Concretely, the participants were asked to answer two open questions: (1) Please list your research interests related to the topic of the seminar, and (2) Please let us know what you expect to gain from attending the seminar. From a coding of the answers (29 participants completed the survey), we identified three dimensions of input.

The first basic dimension is the themes of interest, which refers to what the participants are interested in for the seminar. Within this dimension, we identified 9 concrete themes: the notion and types of uncertainty (12 occurrences), machine learning/artificial intelligence and uncertainty (9), humans and uncertainty (5), goals, requirements, and uncertainty (7), security, privacy, safety, and ethics under uncertainty (8) representation and quantification of uncertainty (7), propagation and interaction of uncertainty (3), mitigating uncertainty (15), and verification/validation and uncertainty (8).

The second dimension focuses on the methodological viewpoint, i.e., how participants want to learn from the seminar. We identified three foci: discuss and enhance understanding (13), exposure to diverse perspectives (6), and explore future challenges (8).

Lastly, the third dimension was about the output format, i.e., the format to consolidate the insights and knowledge obtained from the seminar. We identified two formats: establish an agenda for future research (5), and establish (project) collaborations (7).

4. SHORT INVITED INTRODUCTORY TALKS

To give the participants a good overview of the current landscape of research on uncertainty in self-adaptive systems and help clarify the commonalities and differences in opinions between research areas, four speakers were invited to give short talks. Amel Bennaceur talked about adapting systems to the uncertainty of user requirements, preferences, and values. Ada Diaconescu introduced the participants to essential concepts for self-adaptation under uncertainty. Paola Inverardi proposed to rely on what we know (in contrast to focusing on what we do not know). She attempted to characterize (Self-)adaptive/evolving systems under uncertainty. Finally, Antonio Vallecillo explained the representation of aleatory and epistemic uncertainty in software models.

4.1 Adapting Systems to the Uncertainty of User Requirements, Preferences, and Values

One approach to deal with uncertainty is delaying some decisions until more knowledge is available. The talk presented three examples to support this argument.

The first example focuses on requirements adaptation for the case of meal planning to reduce food waste [13]. Delaying requirement adaptation until runtime, when the availability of the resources is known, allows for constraining the search space for achievable requirements.

The second example focuses on a framework to deal with the inherent uncertainty of users' value [11]. It is common for users to gain a better understanding of their values as they experience, reflect, and learn more about them. The framework enables users to (i) represent, instantiate, and monitor their values and behavior; (ii) understand mismatches between stated values and their observed behavior; and (iii) recommend ways to align users' values and behavior. The approach was illustrated for the domain of food consumption which is rich in values and regularly undergoes reflection and debate.

The third example focuses on the uncertainty of human behavior and preferences arguing for collaborations with other disciplines such as social psychology to represent and reason about human behavior and preferences [44]. An adaptive software architecture was presented that enables cooperation between humans and autonomous systems by leveraging social identity. The social identity approach establishes that group membership can explain and drive human behavior and cooperation. By reasoning on groups during operation, we limit the number of cooperation strategies the autonomous system needs to explore. The approach was illustrated for a search-and-rescue scenario in which a rescue robot optimizes evacuation by delaying the decision about the cooperative humans at runtime.

4.2 Essential Concepts for Self-adaptation under Uncertainty

Self-adaptive systems must observe relevant aspects of themselves and their environment and change themselves in the face of external and internal variability so as to reach various goals, which may also change. Such variability may feature diverse levels of uncertainty, ranging from value changes of known variables, through updates of known components, and all the way to the occurrence of unknown events, which the system is ill-equipped to observe, represent or handle (i.e. unknown unknowns). Numerous approaches, both overlapping and complementary, have been proposed over the last decades to handle such cases. These include techniques for online learning, self-evolution, and self-integration; dynamic discovery and definition of modeling meta-types; and management of conflicts, stability, and convergence.

This talk relied on the vast body of research in the aforementioned areas to distill several generic concepts, which can be reused for facilitating the development of self-adaptive systems operating under uncertainty. The key concepts identified include:

- *Goal-orientation*: the objectives that a self-adaptive system strives for must be specified explicitly and formally, allowing the system to self-evaluate with respect to them. This is essential when unexpected changes may push the system to adapt beyond anything that was predicted at design time.
- *Innovative self-adaptation*: the self-adaptation logic must be able to undergo open-ended meta-adaptation or evolution. This allows system adaptation to unforeseen situations, especially when the goals themselves may change (e.g. introducing a new type of objective, never seen before).
- *Dynamic meta-models*: the system's ontology for modeling or representing itself and its environment must be able to evolve dynamically, as new types of events occur (e.g. discovering a new type of resource, never seen before).
- *Multi-scale abstraction*: the system's ability to represent itself and its environment at increasing abstraction levels, interconnected by feedback loops, maintains its viability when the number of observable resources and dynamic changes increases. Multi-scale structures help to encapsulate subsystems, rendering them more independent from each other and hence avoiding chain-propagation of disturbing events. Only changes that necessitate coordination across the entire system should penetrate to top scales.
- *Self-explanation*: the system's ability to explain its self-adaptive actions to users (at various expertise levels) and to other systems allows it to include users, improve debugging and increase trust in the open-ended self-adaptive process.

4.3 If We Do Not Know, Let Us Rely on What We Know

Software systems today operate in the presence of different (unpredictable) context variability dimensions, namely: heterogeneity of the environment and changing user needs. Self-adaptive systems provide means to adjust their behavior in response to changes in the Self and in the Context. Operating on the Self dimension may imply operating on the whole body of software, as represented by the whole set of artifacts that characterize the development and operation of the system (e.g., new requirements) Operating on the Context dimension means taking into consideration everything in the operating environment that may affect the system properties and behavior. Both dimensions may be sources of uncertainties when dealing with unforeseen context variations that may require switching towards an unanticipated system variant that satisfies a new requirement (at runtime). The talk introduced an approach to characterize evolving systems vs adaptive one that stress the notion of core functionalities, the known, ones that needs to be preserved through the changes and are immutable, as opposed to added functionalities that may be added to respond to a specific context or user's need changes. Although these evolutions are unknown the variations in the system do not happen in a vacuum rather, they need to cope with a set of functional and non-functional constraints that can help reduce the uncertainties. These constraints may come from the requirements, the software architecture, the programming languages, and the runtime support. These constraints can be used in an assume-guarantee fashion to reduce the level of uncertainty.

4.4 Representing Aleatory and Epistemic Uncertainty in Software Models

A fundamental characteristic of software models is their ability to represent the relevant characteristics of the system under study, at the appropriate level of abstraction. For cyber-physical systems, smart applications, and the Internet of Things, this requires some form of interaction with the physical world. Uncertainty is an inherent property of any system that operates in a real environment or that interacts with physical elements or with humans. Unfortunately, the explicit representation, management, and analysis of uncertainty have not received much attention from the software modeling community. This talk briefly introduced two classes of uncertainty, namely measurement and belief uncertainty, and how to represent and deal with them using software models and modeling tools. More specifically, an extension of standard UML data types was introduced to deal with measurement uncertainty, and the use of subjective logic to represent belief uncertainty in software models. These approaches were illustrated with examples, such as an extension of knowledge graphs to deal with subjective opinions, and a UML profile to enrich models with individual opinions and help stakeholders reach a consensus.

5. AGILE END-TO-END HANDLING OF UNCERTAINTIES IN GOAL-ORIENTED SELF-ADAPTIVE SYSTEMS

5.1 Motivation

Nowadays software applications have to be viewed as comprehensive end-to-end systems, which serve the needs of users and their businesses, are realized by networks of components, and are deployed in an uncertain runtime environment. This integrated view on systems is often termed *BizDevOps* (Business - Development - Operations) [50]. There is a growing need to adapt these software end-to-end systems in the field with minimal human intervention and with minimal or no interruption of their services. This is where self-adaptation comes into place. In order to be able to adapt to diverse context conditions (like different user profiles or infrastructures) and in response to changes in the system itself and in its requirements/goals [71], these systems have to be designed as self-adaptive systems (SAS).

To realize these comprehensive self-adaptive end-to-end systems in a reliable and resilient way, tight integration of all aspects is needed. This means that all involved stakeholders as well as all managed artefacts have to be aligned. These are in particular on one side stakeholders who are acting e.g. as a user, developer, tester, or operator, and on the other side models like e.g. context models, goal models, architecture models, or deployment models. Thus, managing (in-)consistencies, communication, and interrelations during the lifetime of a steadily evolving end-to-end system becomes a challenging task. This becomes even more ambitious as quite a number of uncertainties might exist. These range from uncertain goal-oriented requirements for the system on the user side to uncertain context and infrastructure conditions, the behavior of humans interacting with the system, the system itself that is managed, the feedback loops, and the runtime models used to realize the self-adaptive behavior on the running system side.

In this section, we will discuss these challenges of managing uncertainties in an end-to-end view of systems. We refer to existing work and identify open research questions on the goal-oriented evolution management of such systems. In particular, as in [49] we aim at exploring the role of runtime models as a dynamic knowledge base that abstracts useful information about the system. Such models can be augmented with information available at design-time and with information collected at runtime as a means to cope with uncertainty by using, for example, distance metrics for quantifying it [26, 27] along an enduring evolution process.

Motivating example. To illustrate our ideas we use a simple *smart home app* example. The high-level goal of this example system is to provide high comfort to the user at home continuously by adapting/evolving the system at runtime. This high-level goal can be realized by achieving more concrete goals like continuously maintaining a constant warm temperature. This concrete goal again can be realized by a combination of more detailed goals, e.g., goals that specify what "warm" temperature means in specific rooms like a living room or a kitchen. Finally, the most detailed specified goals need to be achieved by taking actions, so-called operationalization, such as controlling the temperature by opening/closing a window (manual action) or using a heater (automatic action). Already in this simple example, we can identify a set of different factors at the user's home that influence achieving the goals at runtime. Such sources of uncertainties in the environment/context are, e.g., the number and type of the existing rooms, house structure, existing resources able to influence the temperature, temperatures per room that are preferred by the user, user presence, or the outside temperature. Since these factors are uncertain at design time, it must be started with an initial version of the goal model that is specified, implemented, and deployed. But this goal model needs to be adapted/evolved during runtime as soon as the knowledge about the environment/context gets available, e.g., in the form of adding new goals or changing goals.

We framed our work into the following main research question: How to support agile end-to-end handling of uncertainties in goal-oriented self-adaptive systems?

5.2 Related Work

We drew inspiration from existing works and frameworks that influenced our vision. These include, among others, works related to the evolution/adaptation of goal-oriented requirements, and layered architectures to deal with changing goals.

Awareness Requirements (AwReqs) [86, 85] specify either the degree of success or the degree of failure where requirements are acceptable or can be tolerated. AwReqs were extended with control-theoretic information concerning control variables and indicators to allow the synthesis of controllers by selecting a new variant of the system's goal model, and/or new values for its control variables. *RELAX* [98, 41] supports the explicit expression of environmental uncertainty in requirements. As an extension, AutoRELAX automatically generates RELAXed goal models and specifies fuzzy logic function boundaries to the goal's satisfaction criteria. Tradeoffs between minimizing the number of RELAXed goals and maximizing delivered functionality are then performed. *FLAGS* [8] generalizes the basic features of the *KAOS model* [33] (i.e., refinement and formalization) and adds the concept of adaptive goal. These goals define the countermeasures that one must perform if one or more goals are not fulfilled satisfactorily. Each countermeasure produces changes in the goal model.

The need to deal with hierarchies of control loops and layers to support both vertical and horizontal modularization in self-adaptive systems is widely recognized, but the challenging problem of how to evolve goals and assumptions at runtime has been investigated to a lesser extent. The reference architecture MORPH [15] comprises a three-layer architecture model that makes an explicit distinction between the functionality of the managing system that is responsible for realizing the adaptation goals in the current situation (change management) and the functionality that is responsible for selecting adaptation goals over time when the conditions of the system change (goal management). ActivFORMS [97] offers a change management layer comprising formally verified feedback loop models that are directly executed and on top of that a goal management layer that offers basic support for on-the-fly changing adaptation goals and updating of verified feedback loop models. Other related lines of recent research that focus on systems subject to uncertainty and online goal changes include self-evolution [94], self-improving system integration [10], and self-development [67].

5.3 Novelty and Envisioned Approach

A central problem to cope with is the "tension" between the different types of *uncertainty* affecting the three cornerstones of the considered scenario (*Biz*, *Dev*, *Ops*), which can be identified according to the following schema:

- *Requirement uncertainty* (*Biz*): concerns the definition of the set $\mathcal{O}$ of "observables" that are best suited to express the business goals, and the range of values for each of them that can be qualified as "acceptable"; from the *Dev/Ops* viewpoint, the greater this uncertainty, the larger the space of possible design and operational solutions that can be explored;

- *Design/architectural uncertainty* (*Dev*): includes possible alternative identification of functions/agents (and interactions among them) that can affect elements of $\mathcal{O}$, and the possible conflicts that could emerge from the interactions among these functions/agents, which could affect the achievement of the required values for the elements of $\mathcal{O}$;

- *Operation/implementation uncertainty* (*Ops*): includes imprecisions in the estimated/predicted values for elements of $\mathcal{O}$ and how these imprecisions affect the assessment of the "acceptability" of a given implementation and the identification of possible corrective (adaptation, evolution) actions.

To resolve this tension, we advocate for a *comprehensive* and *systematic* approach that goes beyond existing partial or case-specific solutions. Underpinning this approach are the runtime *models* we use to represent the knowledge and related uncertainty we have about the three elements (*Biz*, *Dev*, *Ops*) mentioned above.

Figure 1 exemplifies our approach, which refines the *twin peaks model* [29] by explicitly distinguishing the two *Dev* and *Ops* "peaks" (besides the *Biz* peak concerning requirements). As in [29], we envision a continuous co-evolution of these three peaks and their associated models, through a series of fine-grained iterations across all of them, to achieve the goal of developing software architectures that are stable, yet adaptable, in the presence of modifications or refinements of the knowledge we have about any of the three peaks. This will allow systematic and agile end-to-end handling of the knowledge and uncertainty concerning the whole system. The co-evolution process of the three peaks generally initiates from the *Biz* (requirements) peak for newly developed systems but could initiate equally well at the other peaks in case of projects involving modifications to existing systems. Key points towards the achievement of this vision are:

- The definition of *propagation links* between models, to support a coherent knowledge transfer and models evolution as new knowledge gets available;

- The identification of suitable *solution patterns* for the management of conflicts among corrective action targeting different elements of $\mathcal{O}$, aimed at bringing them toward what is considered as an acceptable value;

- The adoption of methodologies aimed at proactively supporting the acquisition of new knowledge to be embedded in models and propagated between them;

- The definition of a Reference Architecture for agile goal–oriented self-adaptive systems, which includes components supporting the points listed above;

- The extension of the Reference Architecture above via a *multi-scale structure*, for scalability purposes. Here, the functions listed above are performed in parallel, at multiple abstraction levels; and interrelated via various abstraction (bottom-up) and reification (top-down) data flows.

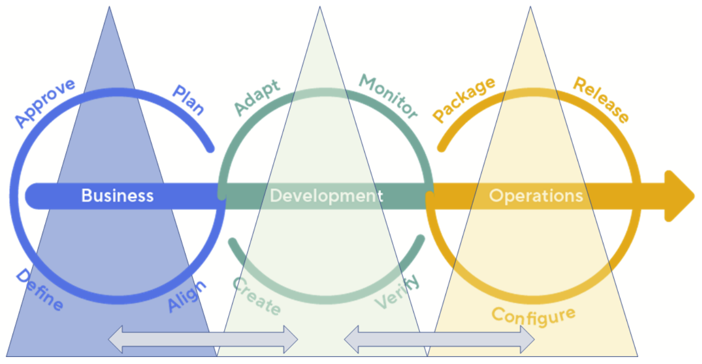

Figure 1: High-level schema of a BizDevOps approach to goal-oriented evolution management and uncertainty mitigation.

5.4 Research Challenges

Some of the essential research challenges to overcome for achieving the above end-to-end approach to uncertainty handling in agile goal-oriented self-adaptive systems include:

- *Managing conflicts* between goals and/or between the system implementation that aims to achieve those goals. Here, conflict management includes conflict detection and handling.

- Handling *goal evolution*, which may be due to e.g., changes in user preferences, conflicts with other goals, lack of system resources or transformations in the running environment.

- Handling the *integration of new components*, previously unknown to the system (e.g. so as to help achieve new goals).

- *Ensuring the coherence of hybrid uncertainty management* solutions, combining fully-automated and human-in-the-loop approaches. This also implies ensuring the overall coherence of the resulting system behavior.

- Ensuring the *scalability* of end-to-end uncertainty management in large self-adaptive systems, undergoing frequent unpredictable changes. Multi-scale approaches can be explored here – i.e. structuring the system into quasi-independent scales (i.e. abstraction levels), with each scale hiding unnecessary details from the scale above.

- *Tracking and limiting uncertainty propagation* across the system, e.g. via encapsulation and abstraction. This concerns both horizontal propagation (i.e. across interrelated system components) and vertical propagation (i.e. between different system scales, or abstraction levels).

- Ensuring the *stability and convergence* of the multi-scale self-adaptive process (i.e. vertical cross-scale self-adaptations, or 'yoyo' process).

- Maintaining a sufficiently accurate model of the system's *stakeholders* (i.e. role models) and execution environment

- Providing *human-comprehensible explanations* to human users, depending on their expertise and required detail level. The system's self-explanatory functions must also be able to self-adapt/evolve so as to follow the system's self-adaptation/evolution.

6. MANAGING UNCERTAINTY RISKS FOR SELF-ADAPTIVE SYSTEMS

6.1 Motivation

In addition to the *what* and *how*, conducting a comprehensive investigation of uncertainty also requires focusing on the *who* and *why*. This observation motivates the need for a problem-driven stakeholder-centered perspective on uncertainty identification, analysis, and management in self-adaptive systems.

To that end, we consider an actor A as *uncertain* about X when A believes that X can have more than one possible value. In other words, A perceives the true value of X as unknown. It's important to note that uncertainty is defined from A's perspective of knowledge, rather than the ground truth. We introduce the concept of *uncertainty risk* as the negative outcome that can arise due to uncertainty about X, whereas a *risk* is an undesirable event whose occurrence is uncertain [91, 57]. A risk can be quantified by estimating its likelihood and severity.

To provide a holistic view of uncertainty management, we raised the research question: How to support stakeholder-driven uncertainty management and risks in self-adaptive systems?

Motivating example. For illustration, we consider an adaptive helper robot for an elderly person. Our particular focus is on enhancing the robot with self-adaptation capabilities to adapt the features of the robot to deal with the cognitive and physical evolution of the elderly person. As an example, consider an elderly person as an actor that interacts with the helper robot. The elderly can be uncertain about the ability of the self-adaptive system to accurately measure his or her happiness. The uncertainty risk here is the potential ignorance or even negative effects of the self-adaptive helper robot that optimizes its behavior for the happiness metric without accurately capturing the elderly's true happiness.

6.2 Related Work

Over the past decade, multiple research teams have provided comprehensive classifications and taxonomies for uncertainty in self-adaptive systems. We highlight a number of characteristic efforts.

Esfahani and Malek [37] classified sources of uncertainty in self-adaptive software according to several dimensions: simplifying assumptions, model drift, noise, parameters in future operation, human in the loop, objectives/goals, decentralization, context, and the cyber-physical nature of systems.

Perez-Palacin and Mirandola [75] presented a taxonomy with three dimensions for uncertainty modeling: the location of uncertainty (context of the model, model structure, input parameters), the level of uncertainty (degree of knowledge from 0 to 4th order), and the nature of uncertainty (epistemic, aleatory).

Mahdavi-Hezavehi et al. [69] classified uncertainty in self-adaptive systems along five dimensions: the location (the place in which uncertainty emerges in the self-adaptive system), the sources of uncertainty (model, adaptation functions, goals, environment, resources, managed system), the nature of uncertainty (epistemic, aleatory), the level/spectrum (scale the uncertainty is specified, statistical or scenario-based), emerging time (design or runtime).

A recent survey with experts in the field of self-adaptation [56] highlighted the sources researchers and engineers considered in self-adaptive systems, the methods used to tackle uncertainty in concrete applications, and the impact of uncertainty on non-functional requirements.

In summary, existing work has come a long way in identifying different facets of understanding uncertainties in self-adaptive systems and mitigating these uncertainties. Yet, despite the multitude of taxonomies, representations, mitigation methods, etc. it is not always clear what software engineering problems need to be tackled when managing uncertainty. In particular, we lack knowledge on involving key stakeholders in this process, both domain stakeholders and engineers of the self-adaptive system.

6.3 Uncertainty Risk Management Framework

Driven by the concerns of both domain stakeholders and engineers, we considered a reusable framework for systematically identifying and managing risk uncertainties, as shown in Fig 2. The framework comprises the following interactive steps:

- Identify and classify uncertainties: leveraging stakeholder-specific concerns to systematically identify and group uncertainties by determining the knowledge gaps or required information needed to address each concern effectively.
- Identify uncertainty risks: determining the concerns that may be negatively affected due to the identified knowledge gaps.
- Analyze uncertainty risks: assessing whether there are any conflicts or redundancies between the identified risks, as it is possible that certain uncertainty risks may no longer pose a significant threat when considered together.
- Identify potential risk management techniques: identify existing applicable risk management techniques;
- Select techniques: Select the most suitable techniques for managing the identified risks.

6.4 Approaches for Resolving Uncertainty Risk

We highlight two possible approaches that may support stakeholders and engineers to realize the different steps of the uncertainty risk management framework. We start with presenting the 5Ws and 1H framework that supports eliciting uncertainties from stakeholders (both domain stakeholders and engineering stakeholders) on the one hand and identifying risk management techniques on the other hand. Then we present the uncertainty heatmap that supports identifying and analyzing uncertainty risks.

5Ws and 1H. Using the common information-gathering and problem-solving 5Ws and 1H framework, we can identify uncertainties (with the 5Ws) and potential risk management techniques (with the 1H) as follows.

(1) WHO: The uncertain actor can be any entity involved in the system, including users, system owners, system designers, coders, testing and validation teams, system administrators, the adaptive system itself, and even malicious actors.

(2) WHAT: The uncertainties can pertain to various aspects, such as the state of the world, the state of the machine, the user's current goal, or the desired properties of the world as defined by stakeholders [75].

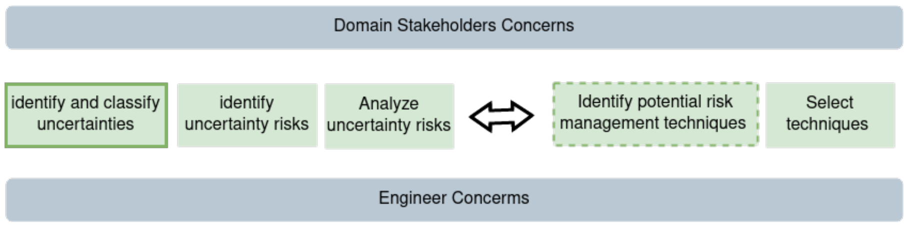
Domain Stakeholders Concerns
identify and classify uncertainties
identify uncertainty risks
Analyze uncertainty risks
Identify potential risk management techniques
Select techniques
Engineer Concerms

Figure 2: Uncertainty risk management framework

(3) WHERE: The uncertain phenomena can be located either in the world or within the machine itself. For example, it could be in the elderly person's house (in his world).

(4) WHEN: Uncertainties can arise at different times, including design time, runtime, or during quality assurance processes.

(5) WHY: Understanding the uncertain actor's objectives and the decisions they need to make can provide insights into the nature of the uncertainties. WHYs are the actor goals affected by the uncertainty. The risks are derived from these goals; they are risks that the goals are not satisfied.

And (1) HOW: Actors dealing with uncertainty, including the uncertain actor or other involved actors, can employ several strategies [21]. Strategies may include ignoring the uncertainty, acquiring more data, mitigating the consequences of uncertainty, or even delaying decisions until more information is available.

Examples The first actor from the motivating example could be the elderly person (WHO). The elderly is uncertain about the robot helper's understanding of what happiness means (WHAT). The uncertainty can arise when the robot helper runs (WHEN). If the elderly's goal is to stay happy, the risk is that this may not be the case (WHY). To ensure that the robot helper understands what the elderly's happiness means, it can adjust the weight assigned to happiness in its decision-making process (HOW).

A second actor from the example could be the adaptive helper robot (WHO) who is uncertain about the validity of the happiness metric (WHAT) within the machine (WHERE) at runtime (WHEN). The robot faces this uncertainty because it needs to make adaptive decisions that optimize happiness (WHY). To address this uncertainty, the engineer may revise the metric with a domain expert, employ multiple metrics for triangulation, or avoid overfitting by using a "good enough" optimization approach (HOW).

Uncertainty Heatmap. Given the information gathered using the 5Ws and 1H framework, one can now systematically identify the negative outcome that can arise due to uncertainty, i.e., risks. This will help the selection process of the most applicable risk management technique (in the "HOW"). Specifically, for each actor's uncertainty (WHO and WHAT), we can identify the risks by exploring the impact of lacking knowledge affecting the end actor goal (WHY) at different times (WHEN) and in different places (WHERE). One possible approach to accomplish this is by producing an uncertainty heatmap. A heatmap lists the concerns of stakeholders in a table along with the knowledge required to deal with these concerns. The required knowledge may be classified into three classes: the knowledge is directly available, the knowledge can be collected at runtime by the system, and the knowledge is inaccessible. Each of these classes expresses a potential level of severity of risk associated with the concerns that can be visually represented in the heatmap with warmer colors (from green to red). Through interaction with the stakeholders and by grouping related concerns with similar patterns, the uncertainty risks can be determined. This provides the basis for managing the uncertainty risks.

As an example, Table 1 shows a small excerpt of an uncertainty heatmap for the motivating example.

Concerns	Knowledge		
	Available	Collect	Inaccessible
happiness	history	current feedback	what impacted the decrease/ increase of happiness
medical duties	history	status context	
safety	...	...	mental state elderly
...	...	...	...

Table 1: Simple excerpts of an uncertainty heatmap

For instance, to deal with the concern of happiness, historical data is available that can be directly used to deal with the concern. Additional knowledge about the happiness of the elderly person may be collected by the system during operation through interaction with the elderly. Yet, the impact of the feedback provided by the elderly on happiness is internal to the person and is not accessible. To deal with the concern of medical duties, the earlier behavior of the elderly person may be available and additional knowledge may be collected about the status of medication. For the safety concern, the mental state of the elderly person may not be accessible.

The uncertainty heatmap may give engineers a systematic approach to exploring what additional data the self-adaptive system may or may not be able to collect to reduce some uncertainty risk.

Uncertainty risk management challenges. The framework described above can help identify, analyze, and manage uncertainty risks. However, further study and expansion are necessary to help self-adaptive system engineers. We list a set of key challenges:

- Develop a systematic approach to identify and classify uncertainty risks. 5Hs and 1H is one possible approach that may be useful. Yet, the framework may need to be complemented with more rigorous and scalable approaches.
- Develop a systematic approach to analyze uncertainty risks. The uncertainty heatmap is one possible approach. Yet, the practical applicability of the heatmap in terms of granularity and scalability needs to be further investigated.
- Create reusable engineering solutions to manage uncertainty risks and identify possible gaps in existing techniques to resolve uncertainty risks.

7. UNCERTAINTY PROPAGATION AND INTERACTION

7.1 Motivation

Sources of uncertainty in adaptive systems are rarely independent and their interactions can affect the achievement of system goals in subtle and often unpredictable ways [25]. Hence, management of *uncertainty interactions* (UIx) must be considered as a first-class systems development problem, e.g., by representing them explicitly in models and making analysis and planning activities aware of them.

To devise systematic approaches that can enable system developers to reason about and mitigate the effects of uncertainties, including their interactions, there is a need to address challenges that concern both the modeling and representation of UIx, as well as their analysis and quantification.

So far, we only have notations available to represent different types of uncertainty [88], but not their interaction [24, 25]. Being able to represent different types of UIx that affect relevant system properties remains largely an open problem.

7.2 Related Work and Challenges

Although there is existing work on how uncertainties propagate in areas that are not limited to computer science (such as mechanical engineering [46]), these approaches are mostly focused on *homogeneous* uncertainties, i.e., uncertainties that are similar in nature. Some examples include the propagation of measurement uncertainty [61], propagation of belief uncertainty based on probability theory [34], *possibilities* (in Fuzzy set theory [77, 100]), *plausibilities* or belief functions (in Dempster-Shafer's theory [80]), or subjective logic [62]. Propagation of design uncertainty has also been treated through design space variability exploration techniques [19, 45, 84, 90].

In contrast, the propagation of *heterogeneous* uncertainties has received little attention. This is partly due to the lack of systematic approaches that enable the rigorous treatment (e.g., representation, analysis, mitigation) of common uncertainty interactions in the area of self-adaptation (e.g., measurement uncertainty vs model abstraction). Devising such approaches will enable assessing the impact of the emerging effects of combined uncertainties upon relevant system properties, contributing to the engineering of more robust and resilient adaptive systems. Yet, representing different types of UIx remains a major challenge.

This challenge entails not only categorizing the different classes of interactions that can be found in the context of an adaptive system but also devising appropriate notations and patterns to represent them and enable their automated analysis and mitigation. In particular, some of the requirements for these notations are that they should be able to capture how uncertainty propagates both *horizontally* (i.e., at the same level of abstraction), as well as *vertically* (i.e., across different levels of abstraction, for instance going from the managed subsystem to the managing subsystem [35], and *vice versa*). Representing some classes of uncertainty, such as those of a crosscutting nature that entails uncertainty in timing aspects of the system, or epistemic uncertainty (e.g., due to the incompleteness of information in models), is particularly challenging and demands special attention.

Once the means to appropriately represent UIx are available, there will be a need to leverage such representations and analyze how uncertainty propagation and interaction affect relevant system properties. An important challenge in this context is that analyses of different classes of uncertainty tend to produce results that are often qualitatively different, precluding their meaningful integration. For instance, the statistical techniques used to analyze measurement uncertainty typically produce confidence intervals or credibility intervals, whereas the probabilistic modeling 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. To make things even more challenging, uncertainty propagation can spread to the entire system, and this is something that can add unnecessary complexity to reasoning upon the effects of UIx. Hence, there is a need to provide scoping mechanisms to limit the analysis of uncertainty propagation only to areas of the system that might have an actual impact on system properties.

7.3 Vision

To address the challenges in representation and reasoning about UIx, there is a need for notations and analysis techniques able to capture and analyze heterogeneous sources of uncertainty, enabling engineers to trace them back to the properties on which they have an impact. Data Flow Diagrams (DFD) [36] have been successfully used to trace uncertainty through software system models while analyzing confidentiality [51] and also for the propagation of uncertainty [52]. UML activity diagrams [73] have also been extended with uncertainty information [48, 42]. Furthermore, within the systems safety analysis domain, error propagation analysis has been traditionally employed during the early stages of systems engineering to understand how error can propagate by leveraging system architectural representations [1]. Although quite useful, these notations and analysis techniques are not enough to address the challenges posed by UIx. Driven by the state of practice, however, we posit that leveraging key elements of such data flow and architectural representations in the context of self-adaptive systems can, not only enhance our understanding of how uncertainty propagates both horizontally and vertically at different levels of abstraction but also about how uncertainty interactions (either homogeneous or heterogeneous) can influence the satisfaction of system properties.

Different stakeholders have diverse levels of knowledge and concerns about the multiple system uncertainties. Moreover, the existence of uncertainties spreads along multiple entities in the self-adaptation process. Therefore, an approach that represents uncertainties and their interaction in a flat, holistic model can quickly become cumbersome. On the contrary, hierarchical modeling notations naturally allow information abstraction.

Building on these concepts and focusing on the effect of the propagation of uncertainty in the flow and processing of information, we aim at defining *Uncertainty Flow Diagrams (UFD)* to capture and reason about the impact of uncertainty interactions on system properties. To realize this vision, we propose:

- Operationalizing available artifacts in self-adaptive systems based on the MAPE-K closed-control feedback loop [63, 35, 92, 59] that employ architecture-centric descriptions to reason about system adaptations. For instance, adaptation triggers captured as architectural invariants in Rainbow [43] can be traced back to the component and connector properties on which they depend in architectural descriptions within the knowledge base. These, in turn, can be related to values retrieved by the monitoring infrastructure from sensors embedded at the managed system level.
- Developing modeling notations and formalisms, including abstract syntax and concrete syntax instances, that enable

the definition of UFDs in a structured and formal way. Using this modeling technology, system developers will be able to define UFDs by exploiting architecture-centric descriptions of their target systems and the outputs from item 1 above. To date, we envisage a UFD notation that copes with a) hierarchical modeling, b) data typing, c) uncertainty from the process actions or operations, and d) uncertainty in the data.

- Leveraging model transformation techniques to automatically translate between uncertainty flow diagrams and different formalisms that can enable the analysis of uncertainty interaction. Such formalisms will vary, depending on the types of uncertainties involved. For instance, Bayesian Networks is a promising candidate to analyze belief or conditional dependencies between system components, whereas stochastic Petri Nets [9] can be used to analyse uncertainty in concurrent probabilistic real-time systems.
- Addressing the necessary integration of multiple analysis techniques. This will involve exploring the use of compositional verification, including, but not limited to assume-guarantee [66, 74], as well as direct integration of heterogeneous analysis mechanisms that enable the analysis of system behavior under combined sources of uncertainty (e.g., design uncertainty and stochastic behavior [18, 23]). This piece of work can build on previous results in data flow analysis, e.g., making use of data flow constraints [53].

8. UNCERTAINTY IN SELF-ADAPTIVE MACHINE LEARNING SYSTEMS

8.1 Motivation

Recent advances in machine learning (ML) have had a transformative effect on both the managed components and the feedback loops of self-adaptive systems. The integration of ML elements (for perception, user interaction, etc.) into the managed components of these systems has enabled important new types of applications. As an example, ML techniques for real-time object detection (RTOD) are essential for applications ranging from autonomous driving to automated passport control. As another example, natural language processing (NLP) is at the core of applications including virtual personal assistants, and customer service chatbots. The adoption of ML within the feedback loops of self-adaptive systems [16, 47, 76, 96] has been equally impactful, not least by augmenting these loops with new sensor fusion and fast adaptation-strategy selection capabilities.

More often than not, these uses of ML allow self-adaptive systems to handle uncertainty. For instance, RTOD enables autonomous cars to identify the unknown objects in their vicinity, and ML classifiers can suggest suitable adaptation strategies in scenarios where the merits of the alternative strategies available are difficult to establish. At the same time, ML can never be 100% accurate, and therefore its use within self-adaptive systems is in itself a source of uncertainty that may need to be mitigated through the use of (additional) feedback loops.

This dual role of ML as an uncertainty mitigation tool *and* a source of uncertainty poses major challenges for the development, deployment, and operation of self-adaptive systems with ML components, and requires significant changes to the way in which these systems are engineered.

Motivating example. To illustrate these challenges—and a methodology we advocate for addressing some of them—we use a (self-adaptive) assistive-care robot. Inspired by the autonomous assistive-care application developed by the recent ALMI project [3, 4, 54], this self-adaptive system uses a TIAGo PAL Robotics social robot to help an early-stage Alzheimer's sufferer with the preparation of a simple pasta meal. To accomplish this task, the TIAGo robot uses a combination of ML/AI techniques. These techniques include NLP to receive commands from its user, RTOD to detect the objects around it (including meal ingredients such as a box of spaghetti and a can of pasta sauce), a deep-learning classifier to perceive the state of the user (rested or tired), and an AI planner to select suitable paths for its movement around the user's apartment. Feedback loops are required to ensure that the NLP is adapted in line with any changes in the user's voice (e.g., because the user may have a cold that affects their voice, or is tired and speaking slowly), that the RTOD is adjusted in line with lighting conditions, that navigation waypoints rendered unusable by clutter are no longer selected by the AI planner, etc.

8.2 Vision

As a possible way to tackle the challenges of developing self-adaptive systems that contain ML components, we envision an iterative development process consisting of five steps (Figure 3). We describe each of the steps in the subsections below.

The key idea of the process is to systematically identify uncertainties (often coming out of ML) and to address them via self-adaptation (often employing ML). Furthermore, the process also specifically looks at the uncertainties that emerge through the interaction of feedback loops. The steps of the process are repeated until no new uncertainties are identified.

The process leads to a system architecture that consists of multiple feedback loops, some of which may control another feedback loop or a group of feedback loops. Contrary to the traditional view of self-adaptive systems as consisting of a non-adaptive base and the controller (which may have been introduced later), we assume that the adaptive ML systems are built as greenfield projects with feedback loops present already from the beginning.

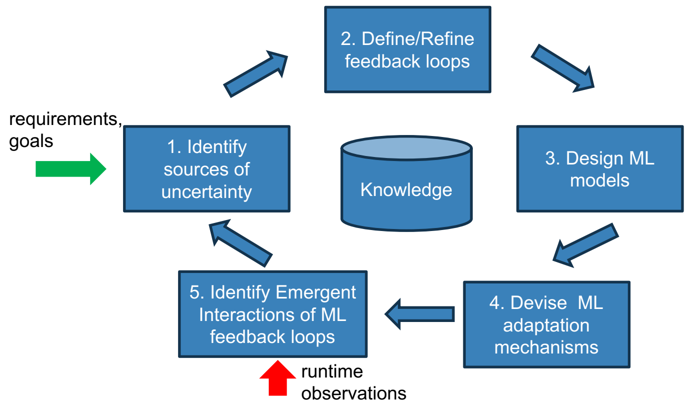
requirements, goals
1. Identify sources of uncertainty
2. Define/Refine feedback loops
3. Design ML models
4. Devise ML adaptation mechanisms
5. Identify Emergent Interactions of ML feedback loops
Knowledge
runtime observations

Figure 3: Goal-driven self-adaptive MLS development process.

1. Identify sources of uncertainty. The goal of the first step is to identify the sources of uncertainties. Uncertainties can be related to both the environment of the system to be developed and the system itself. This is a common first step in the development of self-adaptive systems. This step receives as input a number of goals and requirements that the system should satisfy, e.g. "the system shall adapt to the emotional state of the user" or "the system should use dark mode in low-light environments" (green arrow in Figure 3). For the identification of different types of uncertainties, existing taxonomies can be employed [56]. It is important to highlight that:

- Addressing some of the identified uncertainties may require the use of ML components
- Introducing ML components that takes place in steps 2-5 leading back to step 1 to complete a loop (Figure 3) may in turn introduce more uncertainties that need to be identified, analyzed, and addressed.

A possible source of uncertainty in our motivating example is the human user, whose needs may not be known upfront. The robot should then use natural language generation and processing to identify the user's needs. As another example, the state of the user (tiredness, anxiety level, etc.) is also not known a priori and can change at runtime. A possible solution here is to use an ML classifier to ascertain the user's state based on a video feed.

2. Define/Refine feedback loops. In this step, new feedback loops should be introduced in the system or existing feedback loops may be refined to deal with the uncertainties identified in step 1. For each feedback loop, the triggers (inputs) and actions (outputs) need to be specified. It is also important to specify any context dependencies or assumptions related to a loop – e.g. it might be the case that the feedback loop should only be active under certain conditions (this allows for building hierarchies of loops). Finally, the role of AI/ML methods and models within each feedback loop (e.g. analysis of images for classifying user emotions, reducing the adaptation space via classification of applicable configurations, etc.) needs to be clearly identified.

As an example, a feedback loop in our motivating example could be that the robot uses reduced vocabulary or starts talking slowly or more empathetically if the user is detected to be tired or confused. In this loop, an ML classification model can be used for the identification of the user state (e.g. tired).

3. Design ML models. This step is specific to the development of the ML components of the system that are involved in the feedback loops specified in step 2. For each ML model, its Operational Design Domain (ODD), its input data, potential features, and outputs should be identified. Data pipelines should be designed to gather and pre-process the ML input data. Importantly, in this step, the designers of the system also need to decide on the granularity of the ML models used in a loop – possible options are one per user, one per application, and one per application domain.

In our motivating example, a possible ML model is a user state classifier that obtains images by processing the video feed of the robot, applying certain filters (e.g. to only consider images where a user is detected), and predicting states such as "high tiredness", "high stress", etc. For training such classifiers, a training set with labeled images needs to be provided. Finally, in such a case, a possible design decision is to train and use a single model for all users (contrary to training e.g. one model per user).

4. Devise ML adaptation mechanisms. In this step, the pipelines for training and re-training the ML models specified in step 3 should be implemented and tested. Apart from training the initial version of the models, special focus should be put on allowing the system to evolve its ML models with new data, by connecting the model training and model serving pipelines and if possible, allowing the retraining to happen at runtime. Monitoring mechanisms should also put in place to detect data drifts and trigger model re-training processes. This step considers techniques within the AutoML umbrella – e.g. hyperparameter optimization – as well as techniques within the MLOPS domain – e.g. model versioning, model update, ensemble learning, switching models used for prediction in the running system, etc.

In our motivating example, the pipelines for training, versioning, serving, monitoring, and automated re-training of the user state ML model (e.g. "high tiredness", "high stress", etc.) should be implemented and tested.

5. Identify Emergent Interactions of ML feedback loops. In this final step of the envisioned iterative process, interactions between the feedback loops identified in step 2 and the way they were addressed in steps 3 and 4 need to be analyzed. Here, techniques for analyzing the propagation of uncertainty across loops (Section 7) can be employed. It is important also to focus on possible interactions that may have not manifested yet but can happen be the result of the long-term operation of ML components (e.g. concept drift in one model resulting in a decrease of performance not only of that model but also of other models relying on its results). This step receives inputs from the running system and the deployed models therein (red arrow in Figure 3).

As an example, in our motivating scenario, a possible interaction may happen if the user state classifier starts to wrongly classify users as always tired. Then, the feedback loop would be activated resulting in the robot speaking slowly. This in turn may lead to the user becoming more tired, hence destabilizing the feedback loop.

8.3 Open Challenges

Even when not life-critical, ML systems may have high consequences and if we are to use ML in our businesses, at doctor's offices, on our roads, or in our homes, we need to build ML systems that precisely specify (and satisfy) the requirements of their stakeholders. However, specifying requirements for ML systems remain more a craft than a science as these systems are often specified based on optimization and efficiency measures rather than well-specified quality requirements that relate to stakeholders' needs. The important questions to be asked here are:

- How can we relate stakeholders' requirements to ML configuration parameters such as features, accuracy, and precision?
- How can we integrate the specification of ML components into existing requirement models?

Even when if ML systems are made to work perfectly in experimental settings, they often fail when deployed in real-world settings [32]. There are multiple ways in which ML systems can fail either through adversarial/malicious behavior or due to faults [64]. This will require systematic methods for assessing and assuring their quality to gain confidence in their correctness [55], to quantify the uncertainty introduced by their ML components [20], and to identify deficiencies [65].

While building good models is important, many organizations now realize that more work needs to be done to put them into practical use, from data management to deployment and monitoring.

MLOps aims to support the end-to-end pipeline by unifying ML system development (Dev) and ML system operation (Ops). However, in production deployments, performance can degrade due to unforeseen changes, e.g., concept drift (where the function mapping changes) or data drift (where the input distribution changes). Therefore, adaptation is necessary to enable ML systems to continually improve and avoid, withstand, recover from, and evolve to changes, faults, failure, and adversity, i.e. to be *resilient* [5]. This leads to the following question:

- How to ensure continuous improvement and adaptation of ML systems?

Considering the role of ML testbeds to allow practitioners to learn, test, and evaluate their approaches, it is paramount to have testbeds that truly represent the challenges of ML systems in real-world settings. Thus, the key question here is:

- How can we create ML systems testbeds that reflect the complexity of deployment in the real world?

One of the most pressing issues facing ML systems is ensuring that they are developed in a socially ethical way for maximizing public good [72, 68]. However, while significant work has been achieved in defining what social and professional values should be embedded in ML systems, critically important new research is still needed to investigate how such systems can be engineered responsibly to embody these values that affect society, business, and the environment and how these values can be maintained at runtime in the presence of data and concept drift.

ML practitioners are able to create applications that push the boundary of what is possible but are not always able to foresee the potential consequences (good or bad) of the applications they create. Just like they take responsibility for identifying and fixing the bugs in their code, significant responsibility lies with practitioners to take responsibility for the values embedded into their software. Training ML systems practitioners with this mindset can start by bringing real-world examples into the training environment, to show how the abstract concepts we learn play out in reality. Closing the gap requires answering the following questions:

- How can we implement value-rich self-adaptive ML systems?
- How do we empower developers to adopt responsible software engineering practices when developing self-adaptive ML systems?

9. HUMAN EMPOWERMENT UNDER UNCERTAINTY

9.1 Motivation

Technology plays an increasingly important and impactful role on our everyday lives and society at large. We live in a software-intensive world in which the behavior of people and society is influenced by the software used in both positive and negative ways.

Nowadays, society is experiencing a world that is fully connected; data is available to anyone, anywhere, at any time; and the behavior of people and society is constantly influenced by this connectivity. Therefore, the existing software-intensive world needs to consider *human empowerment* while engineering and using these software systems; i.e., humans need to be in power and control of their own life and beliefs, independently of the technology they use. We refer to *humans* as individuals, groups of people, and society in general. More specifically, we consider that humans engaging in system interactions may be motivated by personal and individual needs, values, and ethical norms [12, 39, 70, 79, 81]. These aspects may be shared among larger communities, regardless of whether these community relations have been regulated. Finally, nations, regions, and societies can be considered as types of community given that these have the additional power to set rules and regulations to impose the ethical concerns for which there is broader societal agreement.

The unmistakeable trend toward extensive automation leads to greater societal fears about autonomy in self-adaptive socio-technical systems (SASTS). Whereas these systems are still controlled by humans — moderators, decision-makers, data scientists, operators — there is a growing concern about (i) these systems optimizing goals that are not in line with societal and human values at stake, (ii) algorithmic bias, and (iii) inability to take into account human values and ethics' concerns [30]. These concerns and uncertainty impede the overall evolution, development, and acceptance of the next generation of SASTS[2].

We discuss the potential upturn and challenges of increasing the role of human empowerment in SASTS in the presence of *uncertainties.* We concentrate on uncertainties with respect to changes in the system; changes in humans' values, needs, and desires; and changes in regulations in which the systems operate. The research problem we focus on raises the following key challenges:

Complex decision-making How can a SASTS accommodate and reconcile the diverse needs, values, and ethics of different humans, which are often not at all fully satisfiable, while also attaining other important systems goals?

Capturing stakeholder goals How can humans express their needs, values, and ethics in such a way that they can be interpreted and taken into account in the complex decision-making of SASTS?

Stakeholder engagement How can humans be sufficiently informed about complex decision-making (transparency and awareness), and how can humans intervene when necessary (empowerment)?

Evolution & change How can a SASTS deal with uncertainties of human values, ethics, and needs, and how can a SAST change while remaining respectful of values, ethics, and needs?

Motivating example. In order to illustrate, consider a software system that implements the queuing logic for dealing with medical prescriptions. In its implementation, the system enforces a 'first-come-first-served scheme', which may be motivated by its designer as the fairest and correct approach. However, such an approach fails to take into account (i) whether specific customers waiting in the queue may be willing to freely forego their position to more vulnerable other customers; (ii) the nature of the individual requests, e.g. one customer may want to buy a significant

[2]Stricter regulation (e.g., GDPR, AI Act, etc) is a direct response to such concerns. However, its development is slow and after the fact, and such initiatives are aimed at limiting excesses and addressing problems that have already become reality. In addition, regulation may also become overly restrictive, impeding positive evolution, innovation, and advancement of society.

part of the medication stock to resell it later, to the disadvantage of other customers waiting in line; or (iii) the medical urgency or necessity behind the request. This simplified example shows the need for systems to not just take into account economical (maximize profit) or technical (to optimize throughput, reduce queue lengths) system goals, but also human values such as fairness, compassion, privacy, respectfulness [60], transparency and explainability [7].

9.2 Vision

We envision a situation where both humans and SASTS can change independently, without any awareness, request for permission to do so, or being concerned about the consequences of these uncertainties. In this case, it is important to empower humans to support the consequences of systems' changes and to provide mechanisms to the systems to deal with uncertainties regarding humans in terms of their needs, values, and ethics concerns. A key design element is to be able to define the scope of autonomy of the SASTS. That is, which are the decisions that the system can take autonomously and which are the decision that can be objects of adaptation/negotiation due to human values and ethical concerns when the system interacts with a human. In [60] the notion of digital ethics as introduced by Floridi [39] is suggested as a way to help draw the line of system's autonomy w.r.t. user's autonomy. Digital ethics is divided into two parts: hard ethics which are defined in terms of laws and established societal rules, e.g. GDPR, and soft ethics which represent the moral preferences of each single user. The system needs to comply with hard ethics and can host soft ethics to enable decisions that have an ethical implication whether at the societal or personal level. However, each user interacting with the system is equipped with her soft ethics, i.e., human values. When the system and the user interact, the user's soft ethics should be able to manifest and influence the system's behavior accordingly. Referring to the queuing system example, the compassionate user shall be able to leave her position to others and update the default system's soft ethics.

In order to carry on the envisaged interactions, we propose an architecture that aims to empower humans and distributes the responsibility by including two main components, namely Connector and Mediator, as explained below.

Connector An element close to the human. The Connector manages the representations of the individual human values w.r.t. the interaction with the system. This element is highly trusted and needs to be controlled by the human. It engages in negotiation and communication on the human's behalf.

Mediator An element placed in between the human and the system. The Mediator makes complex trade-offs that take into account hard ethics, as well as values (soft ethics) from the humans and requirements of the system.

As shown in Figure 4, when interacting with the *System*, the *Human* only interacts with her *Connector*. The *Connector* in turn interacts with the *Mediator*, which interacts with the *System*. Through our envisioned architecture, the responsibility of aligning with values and ethics is distributed across different components and, therefore, it is not only the system's responsibility. Models of the human (including information about his or her values and ethics) are maintained by the *Connector* and *Mediator* components. Moreover, we envision that separate models should be built for each distinct system. The *System* does not store models of the human, but rather interacts with the *Mediator* to learn about and factor in the human's values and ethics.

Dealing with uncertainties. We make a distinction between two types of uncertainties: uncertainties concerning changes to the system, and uncertainties concerning changes to the human. In case of the system changes and no longer aligns with the human's values and ethics, the Mediator and Connector are still in place to empower the human; i.e., providing the human with additional means to interact with the system[3] in a way that is compliant with his or her own values and ethics. As human values and ethics change, individuals can adjust their values and ethics by using the Connector, or the Mediator, regarding ethics, e.g. through legal change.

Figure 4 also shows a *Third Party Auditor*, which checks, inspects, and verifies the *Human*, *Connector*, *Mediator*, and *System*. This auditor is required to inspect whether someone within the interaction is misbehaving. This is especially important in case of adaptation of either the *Connector*, *Mediator*, or *System* in our envisioned architecture. Part of our vision is also aligned with Art. 12 of the GDPR [89]. We envision that the system should be auditable, act in a transparent way, and be able to justify certain behavior or decisions. Checking these requirements is also part of the responsibility of the *Third Party Auditor*.

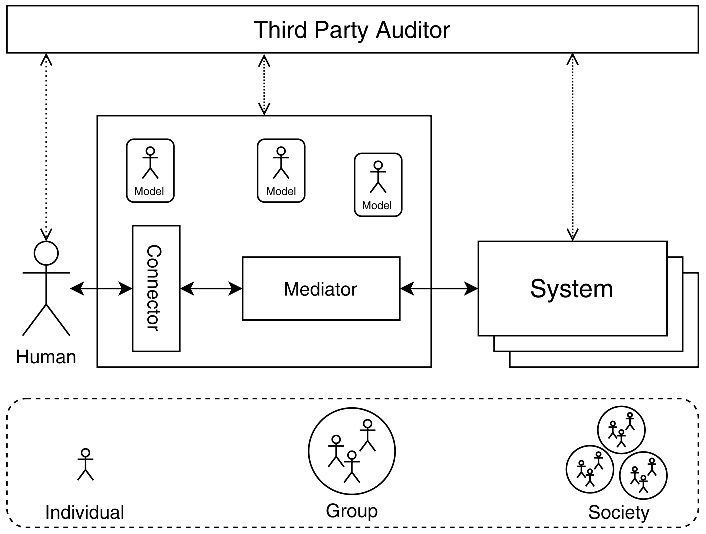

Figure 4: Vision of human empowerment with regards to systems.

9.3 Open Challenges

There are several open challenges when dealing with human empowerment under uncertainties for SASTS. We categorize the open challenges into three areas: system, human, and the general realization of the vision.

System. Regarding the system, we identify the operationalization of human values and ethics to be a significant challenge [12, 82]. Despite some advances in the literature (e.g. [87, 99]), the operationalization of human values, as defined in [82] — *the process of identifying human values and translating them to accessible and concrete concepts so that they can be implemented, validated, verified, and measured in software*, is still an open problem both during design time and run time of the systems [12, 87]. This is even more prominent under uncertainty situations. There are difficulties in modeling values and ethics and accounting for context-awareness. Especially the flexibility in human values

[3]For instance, the *Connector* should be open to interventions initiated by the human.

(soft ethics) needs to be considered and distinguished correctly between human's preferences and his or her values[4]. Additionally, we see, that human values need to be made tangible, ensuring that the system understands them and can adapt according to their changes. Moreover, uncertainty also lies in the future evolution of regulations and the representation of the hard ethics in a matter that is comprehensive to the system [14]. The impacts of system adaptations on the system and humans are uncertain and need to be handled. Uncertainty also arises regarding the representation of human values, their potentially higher rate of change, and their potential conflicts and ambiguities.

Humans. Regarding humans, empowering individuals, groups and society is a challenge [2, 6]. In addition to the current lack of means to empower people, there is uncertainty about the effectiveness of future methods and whether they will actually be utilized. Furthermore, we see that when providing means to empower humans, potential opportunities for abuse of the system arise. This in turn produces uncertainty in dealing with potential adversarial humans. It is necessary to support negotiation between humans and the system when interacting. However, negotiation in this setting requires a mutual understanding of both values and ethics. Also, there is uncertainty regarding the outcome of the negotiation process, especially in the presence of conflicts.

Vision Realization. In terms of the realization of our vision described in Section 9.2, the development and operationalization of the system, connectors, and mediators are open challenges. Negotiation protocols that handle the previously described uncertainties need to be established. The negotiation protocols need to consider transactional properties of negotiation outcomes and to enable re-negotiation after changes in the human or system. In this process, the system should be able to inform humans about its changes and be aware of possible reluctance. Additionally, it is crucial that these negotiation protocols can deal with conflicts that may arise due to incompatibilities between a human's ethics and the system's goals. Balancing the system's involvement with connectors and mediators is also challenging. We highlight that there should always be a default of zero harm to humans, a fallback policy that may involve minimal system involvement or even require blocking it altogether. Therefore, the system should remain open, aware, and adaptive to mediators.

10. TOWARDS A RESEARCH AGENDA

Leveraging the open challenges identified in the different discussion groups as explained in sections 5 to 9, we outline now a possible agenda for future research as shown in Figure 5.

We identified two main lines for future research: understanding uncertainty and managing uncertainty respectively.

Understanding Uncertainty. While substantial efforts have been devoted on understanding the notion of uncertainty in self-adaptive systems, the discussions pointed out that a holistic perspective is required to understand uncertainty that goes beyond the classic taxonomies and classifications in terms of sources and types of uncertainties. To obtain such a holistic understanding, research is required to better understand the "who" and "why" of uncertainty. Furthermore, the discussions lead to the insight that uncertainties are heterogeneous and crosscut business, design and implementation, and operation. This calls for research on modeling heterogeneous uncertainties and notations and formalisms for the propagation of uncertainties and the interaction between uncertainties. We also need a deeper understanding of the relationships between uncertainties and risks. Finally, as machine learning is penetrating many software systems today, we need to better understand how machine learning can help resolve uncertainties, but on the other hand also how machine learning components may introduce new types of uncertainty.

Managing Uncertainty. A key insight from the discussions is that managing uncertainty is a lifelong process, meaning uncertainty needs to be considered throughout the lifetime of a system, from inception to operation and evolution. We identified two main needs for future research on managing uncertainty: the need for an end-to-end approach, and the need for a stakeholder/human-centered perspective.

In contrast to traditional uncertainty management approaches that focus on specific facets of systems or specific phases of the life cycle, the discussions pointed out that managing uncertainty requires an end-to-end approach. Such an approach put several challenges on the table. It requires new methods to model and analyze uncertainties throughout the different phases of the life cycle. Evidently, this involves methods to represent and manage the propagation and interaction of uncertainties within and between different phases. To effectively deal with uncertainties, research is required to develop knowledge and reusable methods to assess and manage uncertainties. Given the increasing role and use of machine learning in software systems, an important research challenge is to develop methods to manage the impact of learning, as well as learning methods that help mitigate uncertainty. Any methods developed to identify, analyze, and manage uncertainties within and across different phases of a system's lifetime should be scalable and able to deal with system evolution that will increasingly occur during operation.

Whereas traditional approaches for managing uncertainty put the emphasis on the technical aspects, the discussions pointed out the need for future research that put stakeholders and humans in the center. Hence, an important challenge is to develop knowledge and methods for the identification and analysis of uncertainty and associated risks that take a problem-driven stakeholder-centered perspective. Since stakeholder goals and preferences are dynamic and change over time, any methods used for dealing with uncertainty should take these dynamics into account. Research is required that considers human values and empowerment as first-class citizens. Dealing with the tension of values of individuals, groups, and society poses another difficult challenge for future research. Last but not least, new methods are required that enable the machine to explain to stakeholders how it makes decisions, and vice versa, new methods are needed that empower humans to express their needs and preferences to the machine.

All together, tackling these challenges will require a multi-years concerted effort of multiple research teams. Multi-disciplinarity within and across these teams will be key to obtaining the fundamental knowledge and engineering know-how that is required to realize reusable solutions that will work in practice.

Acknowledgements

We express our sincere appreciation to the staff of the Bertinoro research centre for their commitment in organising the practicalities of the seminar. The financial support by Christian Doppler Research Association, the BMDW and the National Foundation for Research, Technology and Development is also acknowledged.

[4]Including the deduction of the motives behind human decisions.

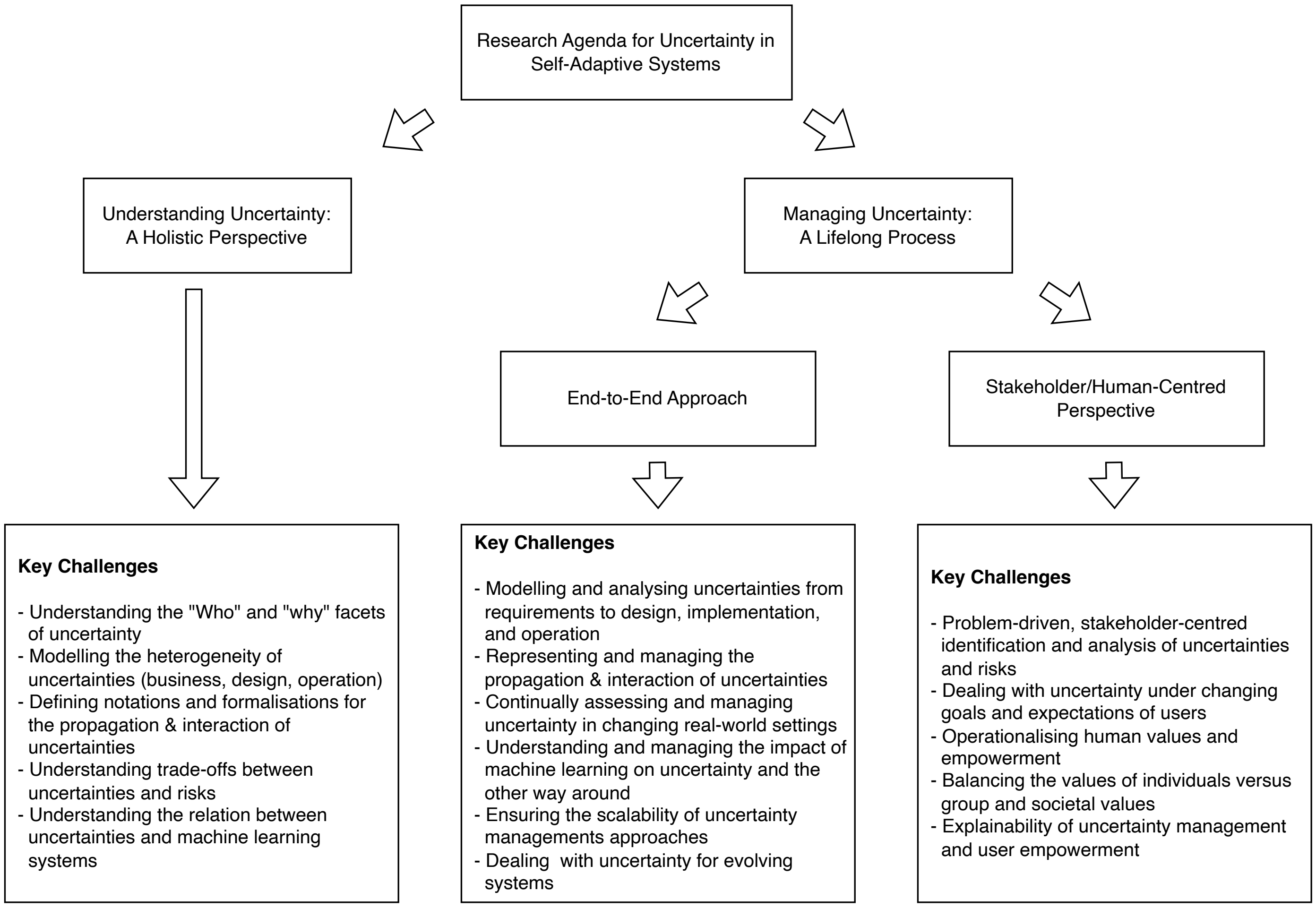
Research Agenda for Uncertainty in Self-Adaptive Systems
Understanding Uncertainty: A Holistic Perspective
Managing Uncertainty: A Lifelong Process
End-to-End Approach
Stakeholder/Human-Centred Perspective
Key Challenges
- Understanding the "Who" and "why" facets of uncertainty
- Modelling the heterogeneity of uncertainties (business, design, operation)
- Defining notations and formalisations for the propagation & interaction of uncertainties
- Understanding trade-offs between uncertainties and risks
- Understanding the relation between uncertainties and machine learning systems
Key Challenges
- Modelling and analysing uncertainties from requirements to design, implementation, and operation
- Representing and managing the propagation & interaction of uncertainties
- Continually assessing and managing uncertainty in changing real-world settings
- Understanding and managing the impact of machine learning on uncertainty and the other way around
- Ensuring the scalability of uncertainty managements approaches
- Dealing with uncertainty for evolving systems
Key Challenges
- Problem-driven, stakeholder-centred identification and analysis of uncertainties and risks
- Dealing with uncertainty under changing goals and expectations of users
- Operationalising human values and empowerment
- Balancing the values of individuals versus group and societal values
- Explainability of uncertainty management and user empowerment

Figure 5: Overview of research agenda for uncertainty in self-adaptive systems

11. REFERENCES

[1] Walid Abdelmoez, DM Nassar, Mark Shereshevsky, Nicholay Gradetsky, Rajesh Gunnalan, Hany H Ammar, Bo Yu, and Ali Mili. Error propagation in software architectures. In *Proc. of METRICS'04*, pages 384–393. IEEE, 2004.

[2] Costanza Alfieri, Paola Inverardi, Patrizio Migliarini, and Massimiliano Palmiero. Exosoul: ethical profiling in the digital world. In *HHAI*, 2022.

[3] ALMI assistive-care robot video, 2023. `https://youtu.be/VhfQmJe4IPc`.

[4] Ambient Assisted Living for Long-term Monitoring and Interaction (ALMI), 2023. Project website. `https://www.york.ac.uk/assuring-autonomy/demonstrators/safe-robots-assisted-living/`.

[5] Jesper Andersson, Vincenzo Grassi, Raffaela Mirandola, and Diego Perez-Palacin. A conceptual framework for resilience: fundamental definitions, strategies and metrics. *Computing*, 103:559–588, 2021.

[6] Marco Autili, Davide Di Ruscio, Paola Inverardi, Patrizio Pelliccione, and Massimo Tivoli. A software exoskeleton to protect and support citizen's ethics and privacy in the digital world. *IEEE Access*, 7:62011–62021, 2019.

[7] Nagadivya Balasubramaniam, Marjo Kauppinen, Kari Hiekkanen, and Sari Kujala. Transparency and explainability of ai systems: ethical guidelines in practice. In *Requirements Engineering: Foundation for Software Quality: 28th International Working Conference, REFSQ 2022, Birmingham, UK, March 21–24, 2022, Proceedings*, pages 3–18. Springer, 2022.

[8] Luciano Baresi, Liliana Pasquale, and Paola Spoletini. Fuzzy Goals for Requirements-Driven Adaptation. In *RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010*, pages 125–134. IEEE Computer Society, 2010.

[9] Falko Bause and Pieter S Kritzinger. *Stochastic petri nets*, volume 1. Vieweg Wiesbaden, 2002.

[10] K. Bellman, A. Diaconescu, and S. Tomforde. Special issue on "self-improving self integration". *Future Generation Computer Systems*, 119:136–139, 2021.

[11] Amel Bennaceur, Diane Hassett, Bashar Nuseibeh, and Andrea Zisman. Values@ runtime: An adaptive framework for operationalising values. In *International Conference on Software Engineering: Software Engineering in Society*, 2023.

[12] Amel Bennaceur, Diane Hassett, Bashar Nuseibeh, and Andrea Zisman. Values@runtime: An adaptive framework for operationalising values. In *Proceedings of the 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, ICSE-SEIS '23, New York, NY, USA, 2023. Association for Computing Machinery.

[13] Amel Bennaceur, Andrea Zisman, Ciaran McCormick,

Danny Barthaud, and Bashar Nuseibeh. Won't take no for an answer: resource-driven requirements adaptation. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 77–88. IEEE, 2019.

[14] Nicolas Boltz, Leonie Sterz, Christopher Gerking, and Oliver Raabe. A model-based framework for simplified collaboration of legal and software experts in data protection assessments. *INFORMATIK 2022*, 2022.

[15] Victor Braberman, Nicolas D'Ippolito, Jeff Kramer, Daniel Sykes, and Sebastian Uchitel. An extended description of morph: A reference architecture for configuration and behaviour self-adaptation. In *Software Engineering for Self-Adaptive Systems III. Assurances*, pages 377–408, Cham, 2017. Springer International Publishing.

[16] Ricardo Diniz Caldas, Arthur Rodrigues, Eric Bernd Gil, Genaína Nunes Rodrigues, Thomas Vogel, and Patrizio Pelliccione. A Hybrid Approach Combining Control Theory and AI for Engineering Self-Adaptive Systems. In Shinichi Honiden, Elisabetta Di Nitto, and Radu Calinescu, editors, *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2020.

[17] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic qos management and optimization in service-based systems. *IEEE Transactions on Software Engineering*, 37(3), 2011.

[18] Radu Calinescu, Milan Ceska, Simos Gerasimou, Marta Kwiatkowska, and Nicola Paoletti. Designing robust software systems through parametric markov chain synthesis. In *Proc. of ICSE'17*, pages 131–140. IEEE Computer Society, 2017.

[19] Radu Calinescu, Milan Češka, Simos Gerasimou, Marta Kwiatkowska, and Nicola Paoletti. Efficient synthesis of robust models for stochastic systems. *Journal of Systems and Software*, 143:140–158, 2018.

[20] Radu Calinescu, Calum Imrie, Ravi Mangal, Genaína Nunes Rodrigues, Corina Păsăreanu, Misael Alpizar Santana, and Gricel Vázquez. Discrete-event controller synthesis for autonomous systems with deep-learning perception components. *CoRR*, abs/2202.03360, 2023.

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

[22] Radu Calinescu, Danny Weyns, Simos Gerasimou, Muhammad Usman Iftikhar, Ibrahim Habli, and Tim Kelly. Engineering trustworthy self-adaptive software with dynamic assurance cases. *IEEE Transactions on Software Engineering*, 44(11):1039–1069, 2018.

[23] Javier Cámara. HaiQ: Synthesis of Software Design Spaces with Structural and Probabilistic Guarantees. In Kyungmin Bae, Domenico Bianculli, Stefania Gnesi, and Nico Plat, editors, *Proc. of FormaliSE@ICSE'20*, pages 22–33. ACM, 2020.

[24] Javier Cámara, Radu Calinescu, Betty H. C. Cheng, David Garlan, Bradley R. Schmerl, Javier Troya, and Antonio Vallecillo. Addressing the uncertainty interaction problem in software-intensive systems: challenges and desiderata. In *Proc. of MODELS'22*, pages 24–30. ACM, 2022.

[25] Javier Cámara, Javier Troya, Antonio Vallecillo, Nelly Bencomo, Radu Calinescu, Betty H. C. Cheng, David Garlan, and Bradley R. Schmerl. The uncertainty interaction problem in self-adaptive systems. *Softw. Syst. Model.*, 21(4):1277–1294, 2022.

[26] Matteo Camilli, Angelo Gargantini, Patrizia Scandurra, and Catia Trubiani. Uncertainty-aware exploration in model-based testing. In *14th IEEE Conference on Software Testing, Verification and Validation, ICST 2021, Porto de Galinhas, Brazil, April 12-16, 2021*, pages 71–81. IEEE, 2021.

[27] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. Taming model uncertainty in self-adaptive systems using bayesian model averaging. In Bradley R. Schmerl, Martina Maggio, and Javier Cámara, editors, *International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2022, Pittsburgh, PA, USA, May 22-24, 2022*, pages 25–35. ACM/IEEE, 2022.

[28] Betty HC Cheng et al. Software engineering for self-adaptive systems: A research roadmap. In *Software engineering for self-adaptive systems*. Springer, 2009.

[29] Jane Cleland-Huang, Robert S. Hanmer, Sam Supakkul, and Mehdi Mirakhorli. The twin peaks of requirements and architecture. *IEEE Software*, 30(2):24–29, 2013.

[30] European Commission. European group on ethics in science and new technologies, statement on artificial intelligence, robotics and 'autonomous' systems. *Publications Office*, 03 2018.

[31] J. Cámara, W. Peng, D. Garlan, and B. Schmerl. Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation. *Science of Computer Programming*, 167:51–69, 2018.

[32] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *The Journal of Machine Learning Research*, 23(1):10237–10297, 2022.

[33] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-Directed Requirements Acquisition. *Science of Computer Programming*, 20(1):3–50, 1993.

[34] Bruno de Finetti. *Theory of Probability: A critical introductory treatment*. John Wiley & Sons, 2017.

[35] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, pages 1–32. Springer, 2013.

[36] Tom DeMarco. *Structure analysis and system specification*, pages 255–288. Springer, 1979.

[37] Naeem Esfahani and Sam Malek. 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*. Springer, 2013.

[38] Antonio Filieri, Henry Hoffmann, and Martina Maggio. Automated design of self-adaptive software with control-theoretical formal guarantees. In *36th International Conference on Software Engineering*. ACM, 2014.

[39] Luciano Floridi. Soft ethics and the governance of the digital. *Philosophy & Technology*, 31:1–8, mar 2018.

[40] Stefano Forti, Uwe Breitenbücher, and Jacopo Soldani.

Trending topics in software engineering. *SIGSOFT Softw. Eng. Notes*, 47(3):20–21, jul 2022.

[41] Erik M. Fredericks, Byron DeVries, and Betty H. C. Cheng. AutoRELAX: Automatically RELAXing a Goal Model to Address Uncertainty. *Empir. Softw. Eng.*, 19(5):1466–1501, 2014.

[42] Luis Enrique García-Fernández and Mercedes Garijo. Modeling strategic decisions using activity diagrams to consider the contribution of dynamic planning in the profitability of projects under uncertainty. *IEEE Trans. Engineering Management*, 57(3):463–476, 2010.

[43] David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley R. Schmerl, and Peter Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54, 2004.

[44] Carlos Gavidia-Calderon, Amel Bennaceur, Anastasia Kordoni, Mark Levine, and Bashar Nuseibeh. What do you want from me? adapting systems to the uncertainty of human preferences. In Liliana Pasquale and Christoph Treude, editors, *44th IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Results ICSE (NIER) 2022, Pittsburgh, PA, USA, May 22-24, 2022*, pages 126–130. IEEE/ACM, 2022.

[45] Simos Gerasimou, Radu Calinescu, and Giordano Tamburrelli. Synthesis of probabilistic models for quality-of-service software engineering. *Automated Software Engineering*, 25:785–831, 2018.

[46] Roger Ghanem, David Higdon, Houman Owhadi, et al. *Handbook of uncertainty quantification*, volume 6. Springer, 2017.

[47] Omid Gheibi, Danny Weyns, and Federico Quin. Applying machine learning in self-adaptive systems: A systematic literature review. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 15(3):1–37, 2021.

[48] Carlo Ghezzi, Leandro Sales Pinto, Paola Spoletini, and Giordano Tamburrelli. Managing non-functional uncertainty via model-driven adaptivity. In *Proc. of ICSE'13*, pages 33–42. IEEE Computer Society, 2013.

[49] Holger Giese, Nelly Bencomo, Liliana Pasquale, Andres J. Ramirez, Paola Inverardi, Sebastian Wätzoldt, and Siobhán Clarke. Living with uncertainty in the age of runtime models. In Nelly Bencomo, Robert France, Betty H. C. Cheng, and Uwe Aßmann, editors, *Models@run.time: Foundations, Applications, and Roadmaps*, pages 47–100. Springer International Publishing, Cham, 2014.

[50] Volker Gruhn and Clemens Schäfer. Bizdevops: Because devops is not the end of the story. In Hamido Fujita and Guido Guizzi, editors, *Intelligent Software Methodologies, Tools and Techniques*, pages 388–398, Cham, 2015. Springer International Publishing.

[51] Sebastian Hahner, Tizian Bitschi, Maximilian Walter, Tomáš Bureš, Petr Hnětynka, and Robert Heinrich. Model-based confidentiality analysis under uncertainty. In *Proc. of ICSA'23 Companion (ICSA-C)*, pages 256–263, 2023.

[52] Sebastian Hahner, Robert Heinrich, and Ralf Reussner. Architecture-based uncertainty impact analysis to ensure confidentiality. In *Proc. of SEAMS'23*. IEEE/ACM, 2023. Accepted, to appear.

[53] Sebastian Hahner, Stephan Seifermann, Robert Heinrich, Maximilian Walter, Tomáš Bureš, and Petr Hnětynka. Modeling data flow constraints for design-time confidentiality analyses. In *Proc. of ICSA'21 Companion (ICSA-C)*, pages 15–21, 2021.

[54] Jordan Hamilton, Ioannis Stefanakos, Radu Calinescu, and Javier Camara. Towards planning and adaptation of assistive-care robot tasks. In *Fourth Workshop on Formal Methods for Autonomous Systems*, 2022.

[55] Richard Hawkins, Colin Paterson, Chiara Picardi, Yan Jia, Radu Calinescu, and Ibrahim Habli. Guidance on the assurance of machine learning in autonomous systems (AMLAS). *CoRR*, abs/2102.01564, 2021.

[56] Sara M. Hezavehi, Danny Weyns, Paris Avgeriou, Radu Calinescu, Raffaela Mirandola, and Diego Perez-Palacin. Uncertainty in self-adaptive systems: A research community perspective. *ACM Trans. Auton. Adapt. Syst.*, 15(4), dec 2021.

[57] Douglas W Hubbard. *The failure of risk management: Why it's broken and how to fix it.* John Wiley & Sons, 2020.

[58] M. Usman Iftikhar and Danny Weyns. Activforms: Active formal models for self-adaptation. In *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, page 125–134. ACM, 2014.

[59] Didac Gil De La Iglesia and Danny Weyns. Mape-k formal templates to rigorously design behaviors for self-adaptive systems. *ACM Trans. Auton. Adapt. Syst.*, 10(3), sep 2015.

[60] Paola Inverardi. The european perspective on responsible computing. *Commun. ACM*, 62(4):64, mar 2019.

[61] JCGM 100:2008. *Evaluation of measurement data—Guide to the expression of uncertainty in measurement (GUM).* ISO Joint Com. for Guides in Metrology, 2008. `http://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf`.

[62] Audun Jøsang. *Subjective Logic – A Formalism for Reasoning Under Uncertainty.* Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, 2016.

[63] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41–50, 2003.

[64] Ram Shankar Siva Kumar, David O Brien, Kendra Albert, Salomé Viljöen, and Jeffrey Snover. Failure modes in machine learning systems. *arXiv preprint arXiv:1911.11034*, 2019.

[65] Marta Z. Kwiatkowska. Safety verification for deep neural networks with provable guarantees (invited paper). In Wan J. Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, pages 1:1–1:5. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[66] Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Assume-guarantee verification for probabilistic systems. In *Proc. of TACAS'10*, volume 6015 of *LNCS*, pages 23–37. Springer, 2010.

[67] M. Lippi, S. Mariani, M. Martinelli, and F. Zambonelli. Individual and collective self-development: Concepts and challenges. In *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 15–21, 2022.

[68] Qinghua Lu, Liming Zhu, Jon Whittle, and James Bret Michael. Software engineering for responsible AI. *Computer*, 56(4):13–16, 2023.

[69] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns. A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements. In Ivan Mistrik, Nour Ali, Rick Kazman, John Grundy, and Bradley Schmerl, editors, *Managing Trade-Offs in Adaptable Software Architectures*, pages 45–77. Morgan Kaufmann, Boston, 2017.

[70] Gregory R Maio. *The psychology of human values.* Routledge, 2016.

[71] Danilo Filgueira Mendonça, Genaína Nunes Rodrigues, Raian Ali, Vander Alves, and Luciano Baresi. GODA: A Goal-Oriented Requirements Engineering Framework for Runtime Dependability Analysis. *Information and Software Technology*, 80:245–264, 2016.

[72] Jessica Morley, Luciano Floridi, Libby Kinsey, and Anat Elhalal. From what to how: an initial review of publicly available ai ethics tools, methods and research to translate principles into practices. *Science and engineering ethics*, 26(4):2141–2168, 2020.

[73] Object Management Group. *Unified Modeling Language (UML) Specification. Version 2.5*, March 2015. OMG document formal/2015-03-01.

[74] Esteban Pavese, Víctor A. Braberman, and Sebastián Uchitel. Probabilistic environments in the quantitative analysis of (non-probabilistic) behaviour models. In *Proc. of FSE'09*, pages 335–344. ACM, 2009.

[75] Diego Perez-Palacin and Raffaela Mirandola. Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation. In *Proceedings of the International Conference on Performance Engineering, (ICPE'14),Dublin, Ireland*, pages 3–14. ACM, 2014.

[76] Arthur Rodrigues, Ricardo Diniz Caldas, Genaína Nunes Rodrigues, Thomas Vogel, and Patrizio Pelliccione. A Learning Approach to Enhance Assurances for Real-Time Self-Adaptive Systems. In *International Conference on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '18, pages 206–216. ACM, 2018.

[77] Stuart J. Russell and Peter Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall, 3 edition, 2010.

[78] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2), 2009.

[79] Shalom H Schwartz. An overview of the schwartz theory of basic values. *Online readings in Psychology and Culture*, 2(1):2307–0919, 2012.

[80] Glenn Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[81] Mojtaba Shahin, Waqar Hussain, Arif Nurwidyantoro, Harsha Perera, Rifat Ara Shams, John C. Grundy, and Jon Whittle. Operationalizing human values in software engineering: A survey. *CoRR*, abs/2108.05624, 2021.

[82] Mojtaba Shahin, Waqar Hussain, Arif Nurwidyantoro, Harsha Perera, Rifat Ara Shams, John C. Grundy, and Jon Whittle. Operationalizing human values in software engineering: A survey. *ArXiv*, abs/2108.05624, 2021.

[83] Stepan Shevtsov, Danny Weyns, and Martina Maggio. Handling new and changing requirements with guarantees in self-adaptive systems using simca. In *IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2017.

[84] Marco Sinnema and Sybren Deelstra. Classifying variability modeling techniques. *Inf. Softw. Technol.*, 49(7):717–739, 2007.

[85] Vítor E. Silva Souza, Alexei Lapouchnian, and John Mylopoulos. Requirements-Driven Qualitative Adaptation. In Robert Meersman, Hervé Panetto, Tharam Dillon, Stefanie Rinderle-Ma, Peter Dadam, Xiaofang Zhou, Siani Pearson, Alois Ferscha, Sonia Bergamaschi, and Isabel F. Cruz, editors, *On the Move to Meaningful Internet Systems: OTM 2012.* Springer, 2012.

[86] Vítor Estêvão Silva Souza, Alexei Lapouchnian, William N. Robinson, and John Mylopoulos. Awareness Requirements for Adaptive Systems. In Holger Giese and Betty H. C. Cheng, editors, *2011 ICSE Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2011, Waikiki, Honolulu , HI, USA, May 23-24, 2011*, pages 60–69. ACM, 2011.

[87] Beverley Townsend, Colin Paterson, T. T. Arvind, Gabriel Nemirovsky, Radu Calinescu, Ana Cavalcanti, Ibrahim Habli, and Alan Thomas. From pluralistic normative principles to autonomous-agent rules. *Minds Mach.*, 32(4):683–715, 2022.

[88] Javier Troya, Nathalie Moreno, Manuel F. Bertoa, and Antonio Vallecillo. Uncertainty representation in software models: A survey. *Softw. Syst. Model.*, 2021.

[89] European Union. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec. *Official Journal of the European Union*, 59:1–88, 05 2016.

[90] Ken Vanherpen, Joachim Denil, Paul De Meulenaere, and Hans Vangheluwe. Design-space exploration in MDE: an initial pattern catalogue. In *Proc. of CMSEBA@MODELS'14*, volume 1340 of *CEUR Workshop Proceedings*, pages 42–51. CEUR-WS.org, 2014.

[91] David Vose. *Risk analysis: a quantitative guide.* John Wiley & Sons, 2008.

[92] D. Weyns, M. Usman Iftikhar, and J. Söderlund. Do external feedback loops improve the design of self-adaptive systems? a controlled experiment. In *2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 3–12, 2013.

[93] Danny Weyns. *Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective.* Wiley, 2020. ISBN 978-1-119-57494-1.

[94] Danny Weyns and Jesper Andersson. From Self-Adaptation to Self-Evolution Leveraging the Operational Design Domain. In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2023.

[95] Danny Weyns et al. Self-adaptation in industry: A survey. *ACM Transactions on Autonomous and Adaptive Systems*, 18(2), 2023.

[96] Danny Weyns, Omid Gheibi, Federico Quin, and Jeroen Van Der Donckt. Deep learning for effective and efficient reduction of large adaptation spaces in self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems*, 17(1–2), 2022.

[97] Danny Weyns and M. Usman Iftikhar. Activforms: A formally founded model-based approach to engineer self-adaptive systems. *ACM Trans. Softw. Eng. Methodol.*, 32(1):12:1–12:48, 2023.

[98] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty HC Cheng, and Jean-Michel Bruel. Relax: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In *2009 17th IEEE International Requirements Engineering Conference*, pages 79–88. IEEE, 2009.

[99] Sinem Getir Yaman, Charlie Burholt, Maddie Jones, Radu Calinescu, and Ana Cavalcanti. Specification and validation of normative rules for autonomous agents. In *Fundamental Approaches to Software Engineering.* Springer, 2023.

[100] Hans-Jürgen Zimmermann. *Fuzzy Set Theory – and Its Applications.* Springer Science+Business Media, 2001.