Journal on Software and Systems Modeling manuscript No.

Uncertainty representation in software models: A survey

Javier Troya · Nathalie Moreno · Manuel F. Bertoa · Antonio Vallecillo

Received: date / Accepted: date

Abstract This paper provides a comprehensive overview and analysis of research work on how uncertainty is currently represented in software models. The survey presents the definitions and current research status of different proposals for addressing uncertainty modeling, and introduces a classification framework that allows to compare and classify existing proposals, analyse their current status and identify new trends. In addition, we discuss possible future research directions, opportunities and challenges.

Keywords Software models · Uncertainty · Modeling languages · UML · Systematic literature review

1 Introduction

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. Software models were initially conceived to design and develop general Information Technology (IT) systems, such as financial applications, enterprise databases or component-based systems, and have proven to be excellent artefacts for representing the basic structure and behavior of these systems. However, we now live in the age of cyber-physical systems (CPS), smart applications and the Internet of things (IoT), which require some forms of interaction with the physical world. This imposes new requirements on software models to deal with the essential aspects of these kinds of systems, such as self-adaptation [39], integration with embedded systems [187], management of physical quantities [211], or uncertainty handling [133,147,193].

In this paper we are concerned with the general challenge of representing uncertainty in software models. Uncertainty is an inherent property of any system that operates in a real environment or that interacts with physical elements or with humans. Uncertainty can be due to different factors, such as imprecision in the measuring tools; lack of knowledge about the system or its environment; imperfect, incorrect, incomplete or vague information; unreliable data sources or communication networks; numerical approximations; distinct, even conflicting, interpretations of the same evidences by separate parties; unforeseen, emergent or unpredictable behavior; or the inability to determine whether particular events have occurred or not [163]. The purpose of explicitly representing uncertainty is twofold: a software engineer who represents or simulates a system needs to capture the relevant characteristics of uncertainty in a suitable way, while a systems engineer analyses uncertainty to try to remove it, reduce it or mitigate its effects [185].

Scientists and engineers already know how to deal with uncertainty in its many forms (objective, subjective, epistemic, aleatory) [219], using different approaches such as mathematical and numerical models [189], probabilities [145,146], Fuzzy set theory [205,233], variability analysis [210] or risk assessment [202], among others. Uncertainty has also been an active research field in

Javier Troya
SCORE Lab, I3US Institute, Universidad de Sevilla, Seville, Spain. jtroya@us.es

Nathalie Moreno
ITIS Software, Universidad de Málaga, Malaga, Spain. moreno@lcc.uma.es

Manuel F. Bertoa
ITIS Software, Universidad de Málaga, Malaga, Spain. bertoa@lcc.uma.es

Antonio Vallecillo
ITIS Software, Universidad de Málaga, Malaga, Spain. av@lcc.uma.es

computer science and database systems, with specialized venues and scientific journals devoted to this topic. However, software models are still falling short for explicitly representing uncertainty and effectively dealing with it [133, 147]. For example, primitive datatypes normally neglect measurement uncertainty in numerical values, and cannot adequately cope with physical units; the underlying Boolean semantics of UML and OCL assumes crisp decisions, ignoring the case for predicates that may be partially true; software designs are usually set in stone and difficult to adapt to unforeseen situations; the behavior of the environment is assumed to be known in advance; interpretations of models are assumed to be objective and unique, and so on.

The situation began to change in the early 2000s, when the modeling community realized that uncertainty needed to be embraced as a first-class entity in software models, especially in domains such as the IoT [151], Industry 4.0 applications [187, 232] or in Intelligent Transportation Systems [141, 165]. Otherwise, software models were not faithful enough to the systems they represented, and the advantages of using models to understand systems and predict their behavior could be lost [147, 174].

Research on uncertainty representation in software models has significantly grown since then. Several groups are working on different types of uncertainty, using various notations and formalisms. The OMG also proposes partial solutions in the fields of real-time systems and system engineering, with the UML Profile for MARTE [72] and the SysML notation [71], respectively. They allow modelers to specify probability distributions in property values to represent measurement uncertainty, and also in activity diagrams to represent stochastic behavior. Nevertheless, these notations cannot deal with other types of uncertainty, such as Design or Belief uncertainty. The OMG is also working on a metamodel for the precise specification of uncertainty (PSUM) [193], based on a previous uncertainty model for the domain of cyber-physical systems [121].

With regard to the representation of uncertainty, the majority of these proposals, including those from the OMG, have been developed independently of each other, mainly with specific applications in mind, and without a common conceptual reference framework. This leads to problems such as conflicting terminology, contradictory notations and redundant work, which can be misleading for software engineers who need to deal with the representation of uncertainty in their software models.

In this context, a clear and complete picture of the current research on uncertainty representation in software models would provide very valuable benefits to the software (modeling) engineering community. In particular, it would help to identify overlapping proposals that address the same type of uncertainty but using different (and potentially complementary) approaches; gaps in unexplored areas that would require further research; application domains that would benefit from the results of existing proposals currently implemented in other domains; and, finally, new requirements for software modeling notations and tools in order to tackle the new challenges posed by cyber-physical, IoT, and human-centered systems.

With these goals in mind, we wanted to identify (1) the existing proposals on this topic; (2) the different types of uncertainty currently addressed by these works; (3) the notations and formalisms used to represent it; (4) the application domains where uncertainty in software models is used and how the proposals are applied in practice; (5) the software development phase in which they are used, and (6) the types of analyses that each proposal allows (simulation, visualization, verification, etc.), as well as their level of tool-support.

This paper provides a comprehensive overview and analysis of research work on uncertainty representation in software models, which aims to answer these questions. The paper presents a classification framework that allows to compare and classify existing proposals, analyse their current research status and identify new trends. In addition, the survey identifies some key areas of research that need to be addressed, as well as some of the main challenges ahead.

The structure of this document is as follows. Next, Sect. 2 provides a brief introduction to uncertainty, mainly in the context of software models. We also describe an uncertainty classification framework that will be used to categorize existing proposals. Then, Sect. 3 describes how the literature review has been conducted, including the data sources consulted, the inclusion/exclusion criteria, quality assessment and the data extraction process. It also poses the six research questions that will guide our study, which are responded in detail in sections 4 to 8. Sect. 9 discusses other relevant aspects of the survey, such as the distribution of research around topics, their coverage, or the common publication venues. We also discuss there some limitations and threats to the validity of the study. Finally, Sect. 10 describes what we consider the main challenges and opportunities for this research, and Sect. 11 concludes the paper.

2 Preliminaries

Uncertainty is “the quality or state that involves imperfect and/or unknown information. It applies to predictions of future events, estimates, physical measurements, or unknown properties of a system” [163].

Uncertainty can be classified according to different criteria and parameters, and it also depends on the specific application domain. The community has not yet agreed on a common classification of uncertainty, despite the many existing proposals (see Sect. 2.1), probably because of the diversity of application domains and the particular nature of each one. This section presents the main classifications of uncertainty currently used in several domains, and proposes one for the general problem of representing uncertainty in software models. It will allow us to classify the surveyed research works.

2.1 Existing classifications of uncertainty

The primary classification of uncertainty divides it into aleatory and epistemic uncertainty [167,189]. *Aleatory uncertainty* refers to the inherent probabilistic variability or randomness of a phenomenon. For example, deciding the result of rolling a die. This type of uncertainty is irreducible, in that there will always be variability in the underlying variables [163]. *Epistemic uncertainty* refers to the lack of knowledge we have about the system (modeled or real) or its elements. For instance, the confidence we have on the actual occurrence of a modeled event, or the current location of the Lost Ark. This type of uncertainty is reducible, in that additional information or knowledge may reduce it.

Two related terms, not to be confused with uncertainty, are randomness and stochasticity [221]. *Randomness* refers to the unpredictable variation over time and/or space, which does not follow any pattern; i.e., given complete knowledge of all previous outcomes, it is not possible to predict the next one. In contrast, *Stochasticity* refers to the variations of processes over time and space, which are mathematically describable by some probability distributions. Thus, randomness implies aleatory uncertainty when we try to predict the outcome of a process, whilst stochasticity can be considered a kind of epistemic uncertainty because the process outcomes can be approximated, thus taming their unpredictability.

We can also distinguish between *Objective* and *Subjective* uncertainties. The former refers to phenomena or concepts whose value, existence or nature are independent of any observing agent; the latter depends on the particular observation or reasoning by each agent [193].

In [219], the authors present an interesting summary of uncertainty taxonomies in various fields, including social sciences, physics, computational modeling and simulation, and several engineering disciplines. A specific uncertainty classification for the design and development of complex systems is proposed in that paper, too, which distinguishes between ambiguity, epistemic, aleatory, and interaction uncertainty. Epistemic uncertainty is in turn subdivided into model-form, phenomenological, and behavioral uncertainty. Targeted at design and development of complex systems in general, it has a broader scope than ours and therefore provides a more generic classification framework (covering cases such as linguistic imprecision and approximation). Nevertheless, it misses others that are specific to the representation of uncertainty in software models, such as belief or occurrence uncertainty. In this sense, it can be considered as complementary to ours.

Other uncertainty classifications have also been proposed for specific domains within software development and engineering, mainly in the areas of adaptive systems [73,135,144,180,199,201,223], complex event processing [125], databases [131,156,175,178,195,200,224], requirements engineering [207], and cyber-physical systems [120,121]. All of them try to harmonize the terminology of uncertainty and propose conceptual frameworks that classify different types of uncertainty according to several criteria.

The fact that they were defined for specific application domains enabled them to perform well when classifying existing work within those domains. However, none of them allowed us to adequately cover all the cases that we came across in our literature review. For instance, classifications focusing on software architectures and adaptive systems perfectly cover Design uncertainty, which is a type of uncertainty that is not normally treated in the rest of the domains. In contrast, these classifications do not properly cover subjective interpretations and degrees of belief, i.e., Belief uncertainty, which can be found in the database and cyber-physical systems domains.

2.2 Types of uncertainty

In the end, we decided to use an uncertainty taxonomy that provided a suitable and comprehensive classification of the types of uncertainty identified in our survey, which allowed us to represent all the different approaches, and covered all cases. The classification defines six types of uncertainty, which are described next, together with some of their particular characteristics.

Measurement uncertainty. This is an aleatory uncertainty that refers to a set of possible states or outcomes of a *measurement*, where probabilities are assigned to each possible state or outcome [163]. It is normally associated to the values of model attributes and properties that are represented using primitive datatypes.

– In the case of numeric values, this type of uncertainty is usually represented according to the GUM [163], i.e., expressed by a parameter, associated with the result of a measurement x, which characterizes the

dispersion of the values that could be attributed to the measurand. This can be expressed in different ways, e.g., by means of the standard deviation u of the possible variation of the values of x ($x \pm u$, e.g., 3.5 ± 0.01); using intervals (e.g., $[a..b]$) according to Uniform or Triangular distributions; by means of samples [163]; or by specific probability distributions that describe the estimated dispersion of the true value of x, which is not possible to determine [153]. See [215] for an extensive survey on this topic.
– Measurement uncertainty applies not only to numerical datatypes (Real, Integer), but also to all other primitive datatypes (Boolean, String and enumerations), and their collections. In the case of Boolean values, their uncertainty can be expressed using either real numbers in the range [0..1] that represent *probabilities* (interpreted in Probability theory [146] or Uncertainty theory [176]), *possibilities* (in Fuzzy set theory [205,233]), or *plausibilities* or belief functions (in Dempster-Shafer's theory [213]). Many-valued logics (e.g., Kleene or Priest logics, that include null values [138]) can also be used to represent incomplete information in logic predicates (as in SQL or OCL).
– Uncertainty in Strings can be expressed either by assigning a *confidence* to each character, to the entire string, or to both. By confidence we mean a real number in the range [0..1] or a fuzzy value that measures the degree of belief that the contents of the String are correct. This is common, for instance, in Optical Character Recognition (OCR) systems that extract data from images or from hand-written text. This type of uncertainty measures the reliability of the contents of the String. For example, a confidence of 0.91 in the string "`Hello wor1d`" indicates that we are unsure about at most one of its characters [6].
– Uncertainty in enumerations can be expressed using different mechanisms, such as discrete random variables [145] or fuzzy numbers [154] denoting collections of possible values and their associated probabilities. For example, an automatic classification system for apples could assign the value `{(Braeburn,0.6), (Fuji,0.3),(Cameo,0.1)}` to a given apple, instead of having to choose only one literal.
– Finally, Measurement uncertainty in collections represents the lack of confidence that we have on their contents, too. This is just a generalization of the uncertainty that can be associated to Strings, and treated similarly.

Occurrence uncertainty. This is a kind of epistemic uncertainty that refers to the degree of belief that we have on the actual existence of an entity, i.e., the real entity that a model element represents.

– Normally expressed in software models by assigning a confidence level to individual instances (i.e., objects or links) that refers to the likelihood that such an instance or link exists in reality.
– The confidence level is usually expressed by real numbers in the range [0..1] that represent probabilities (interpreted in Probability theory or Uncertainty theory), possibilities (in Fuzzy set theory), or plausibilities (in the Dempster–Shafer theory).
– Examples include the confidence assigned to events in a stream coming from unreliable sources (where false positives and false negatives are possible [125]), objects identified by automatic recognition systems, or relationships between objects established by automatic recommendation systems.
– Normally, the level of confidence is propagated through the decision rules, enabling the corresponding uncertainty analysis [11].

Design uncertainty. This is a type of epistemic uncertainty that refers to a set of possible decisions or system design options.

– It captures the usual uncertainty that the developer has about the system design, which may be different depending on the conditions the system may face during its operation, and the expected requirements by its intended users. This information is normally unknown during the early analysis phase, but heavily influences the system design. It also covers ambiguous or imprecise requirements regarding the envisioned operating environment, target technologies, market or legal regulations, etc.
– Under design uncertainty, the modeler may decide to maintain all options open and try to defer the decisions to later stages, when some of this epistemic uncertainty is gone. Alternatively, probabilities can be assigned to the design options.
– Normally represented in software models by variability models [216] that specify the possible design options, using different approaches. For example, by using feature or partial models that explicitly describe the possible design options and their probabilities [33,85]. Alternatively, these options can be implicitly specified by sets of conditions or constraints that describe the possible options, as a Constraint Satisfaction Problem (CSP); or by incomplete models that can be completed in different ways, each one representing a possible design alternative which are automatically computed by SAT solvers, or other design space exploration techniques [222].
– Examples include alternative architectural designs of a software system that may need to contemplate three or four architectural levels, or the use of dif-

ferent design patterns or mechanisms depending on conditions not known yet by the modeler.

Behavior uncertainty. A type of epistemic uncertainty that refers to the lack of knowledge about the real behavior of the system or its environment, including the actions that can be performed, the motivations for performing them, the real values of the parameters of these actions, and when and how they can be performed.

– Examples include the behavior of pedestrians and drivers in intersections, environmental weather conditions, or that of unmanned vehicles.
– This kind of uncertainty is also common in approaches such as self-adaptive [21,107] or "uncertainty-aware" systems [39,120], whose operating environments are unknown or may exhibit uncertain behaviors; for example, a robot operating on Mars, or an application whose users may have erratic or random behavior [73,180].
– Uncertain behavior in software models can be represented in different manners, depending on the nature of the unknowns. First, using variability models [216] to specify the possible alternative options and their valid combinations. Second, using declarative approaches based on fuzzy or probabilistic extensions of modal logics, such as temporal [184,230] or deontic [80,206] logics. Third, using operational approaches such as probabilistic statecharts [148,188], sequence diagrams [203], Fuzzy-DEVS [169], Markov chains [214], Fuzzy Petri Nets [177,183], Generalized Stochastic Petri Nets [182], or probabilistic process algebras [73]. Other approaches use partial behavior models [90,101], which are based on modal transition systems [172] that allow modeling incomplete behaviors. Finally, other authors use Bayesian Belief networks [142,196], Causal networks [197,198], or Influence diagrams [158] to specify the decisions, uncertainties and objectives of possible behaviors.

Belief uncertainty. A type of epistemic uncertainty in which a belief agent is uncertain about any of the statements made in the model about the system or its environment (i.e., about how the system has been modeled, or about the system itself).

– Normally represented in software models by assigning a *degree of belief* to model elements such as classes, relationships, attributes, or their values.
– The confidence level assigned in this kind of uncertainty is subjective, i.e., it depends on the individual agent holding the belief. For example, Mary is 95% confident about the precision of the temperature sensor readings; John is 95% sure that the model class Employee indeed represents all valid employees of the real system.
– Belief uncertainty is sometimes called second-order probability or second-order uncertainty in the literature of statistics and economics because it can also be used to quantify other types of uncertainties. For example, Ada is 85% sure that the probability of rain tomorrow is 60%, or Bob is 60% sure that the stock market will lose more than 20% of its investors due to the financial crisis.
– Belief uncertainty is normally expressed by probabilities (interpreted in Probability theory or in Uncertainty theory), possibilities (in Fuzzy set theory), plausibilities (in the Dempster–Shafer theory), or opinions (in Subjective logic [164]).

Spatiotemporal uncertainty. This type of epistemic uncertainty refers to the lack of certainty about the geographical or physical location of a system, its elements or its environment, or about time properties expressed in statements about the system or its environment.

– It is normally represented in software models using different ways, including Fuzzy set theory, many-valued logics (with unknown values), interval arithmetic, discrete random variables or fuzzy numbers (denoting collections of possible values and their associated probabilities), or by means of rough sets [231]. A formalism called Object-Fields [140] is also used to represent this type of uncertainty in software models.
– Examples include timestamps of events produced by sources that have different clocks, models of typhoon or storms trajectories, the geographic locations of historical cities and archaeological sites, or the dates of particular historical events (e.g., "*circa* 700 B.C.", or "during the late Jurassic period") [63].
– While it might be thought that this type of uncertainty could be represented by the Measurement uncertainty of the attributes representing times or geographical coordinates, its nature is different. Measurement uncertainty expresses possible variations of a measured value, and is of an aleatory and objective nature (due, for example, to the precision of the GPS system in the expression of the position of an object when it is given by its latitude and longitude). However, Spatiotemporal uncertainty also implies vagueness and incompleteness, and is of epistemic and subjective nature (e.g., when stating that an archaeological site is located "somewhere" in Northern Europe). This is why this type of uncertainty has required its own dimension in the classification.

2.3 Relationship with other uncertainty classifications

As mentioned in Sect.2.1, several authors have proposed other classifications of uncertainty for different application domains. Some of them had a wider scope than ours (e.g., [219], which targets the design and development of complex systems, in general), whilst others were aimed towards concrete domains such as adaptive systems, complex event processing or databases, and hence had different scopes.

This section defines the terminology mappings between our dimensions and those defined by other taxonomies in order to improve the understanding and interoperability of our proposal.

Measurement uncertainty is also called *Attribute uncertainty* by other authors [125,131,175,225]. Other proposals distinguish between uncertainty in the values of numerical attributes (and also call it *Attribute uncertainty*) and the confidence assigned to Strings and collections (*Content uncertainty*). However, the term *Content uncertainty* is also used in other proposals to refer to the degree of belief that a belief agent has in the content of a belief statement [120,121,193]. To avoid terminology conflicts, we grouped all *objective* uncertainties of the values of model attributes (numerical or not) in the Measurement uncertainty dimension, while subjective uncertainty was classified in the Belief or Spatiotemporal uncertainty dimensions.

Occurrence uncertainty is also called *Existential* uncertainty [131,225], *Record* uncertainty [175] or *Tuple* uncertainty [195] in the database domain, although the meaning in all the cases is the same.

Design uncertainty is also referred to as *Model uncertainty* in other contexts [223].

Some classifications also group Measurement and Occurrence uncertainties under the names *Data uncertainty* [125] or *Content uncertainty* [121]. We found our separation useful for discriminating between the different nature of these two types of uncertainty. Moreover, we wanted to avoid the term *Content uncertainty* because we realized it referred to quite different uncertainties in different classifications.

Other authors also separate the Spatiotemporal dimension into *Geographical location uncertainty* and *Time uncertainty* [63,120,121,125]. However, we preferred to unite them under the same dimension, as it is usually done in the Geographical Information Systems (GIS) domain [212]. Also, because they share some specific characteristics.

Finally, other authors define a separate dimension: *Environment uncertainty* [121,180], also referred to as *Context uncertainty* [144,223]. This type of uncertainty embraces a variety of uncertainties originating from environment circumstances, such as different or unknown execution contexts, the boundaries and surroundings of the system to be modelled, or the behavior of external actors interacting with the system. By analysing the existing proposals that deal with this type of uncertainty in software models, we concluded that it comprises two main aspects. The first one affects the *Design* of the system in order to accommodate to unknown, vague or changing user or environmental requirements. The second one focuses on the execution of the system at runtime, to cope with uncertain *behaviors* from external actors or from the environment itself. In all cases, the system and its environment could be treated as the two flip sides of the same coin, and thus we realized that this type of uncertainty could be perfectly covered with the Design and Behavior uncertainty dimensions.

Similarly, the uncertainty aspects of the system *Requirements* and goals, which was another dimension defined in other taxonomies [144,180,201], could be easily integrated into other dimensions of our classification depending on whether they refer to the system Design or Behavior because they also affected them. In fact, we came to this conclusion about the *Environment* and *Requirements* dimensions because we initially defined them, but as soon as we started classifying the proposals, these dimensions were never used in isolation but together with others. It was then that we began to realize the overlapping nature of these dimensions and decided to include them in others, with satisfactory results.

3 Research questions and review method

To gather the papers related to uncertainty representation in software models, we followed a systematic and structured method inspired by the guidelines of Kitchenham [166] and Webster et al. [226]. We also took inspiration from surveys on related topics, such as formal verification of static software models in MDE [150], model transformation design patterns [171] and the execution of UML models [137].

3.1 Research questions

The first step is to state the research questions that will guide our study. These are the following.

- **RQ1**. What proposals exist right now to represent uncertainty in software models?
- **RQ2**. What type(s) of uncertainty does each proposal address?
- **RQ3**. What notations and formalisms are used to represent each type of uncertainty?
- **RQ4**. In which domains are the different proposals applied?
- **RQ5**. To which phase of the software life cycle does each proposal apply?
- **RQ6**. Besides the representation of uncertainty, do they allow any kind of analysis? Are they supported by tools?

By answering them, we get an overview of the existing proposals, the type of uncertainty they address and how they represent it. The questions also relate to the way these representations are used, in particular the application domains in which they are used, how and when they are applied in them, and also to their maturity, i.e., whether the existing proposals provide any additional added value (e.g., uncertainty analysis, tool support) to software engineers beyond the mere representation of uncertainty in their models. In short, these questions correspond to the *who*, *which*, *how*, *where*, *when*, and *what*, respectively.

The rest of this section is devoted to answer RQ1. The others are answered in sections 4 through 8.

3.2 Inclusion and exclusion criteria

We scrutinized the existing literature (up to January 2020, when we conducted the literature review) looking for papers having as focus *the representation of uncertainty in software models.* We focused on works dealing with the explicit modeling of uncertainty aspects of systems and their environments using software modeling notations (general-purpose notations such as UML, SysML or OCL; extensions of UML such as Fuzzy UML and other UML profiles; or Domain-Specific Languages such as ConML, FAME, FUSION, or RELAX, to mention a few).

We excluded works according to different criteria. First of all, those works whose goal was not to represent or specify systems, but to query or analyse them. Examples include the extensions to SQL or to Event Processing Languages to deal with uncertainty in databases or event streams, respectively. Examples of excluded papers in this line are [209,186], which deal with probabilistic databases and uncertainty in complex event systems, respectively. Similarly, we did not consider the numerous works dealing with model-based performance or reliability engineering of software systems, which enrich software models with the information required for evaluation using different notations (e.g., UML Profiles such as SPT [190], MARTE [72] or DAM [129]), transform the enriched model to a formal and mathematical model supporting the evaluation (e.g., Queueing Networks [161,217], Probabilistic Process Algebras [157], Stochastic Petri Nets [182,183], Fault Trees [143] or Markov chains [214]) and evaluate the performance or reliability of the system using the tools available for the formal model [139,127]. The interested reader can consult already existing corresponding surveys about these topics, such as [131,156,175,195,200,224] in the context of databases, [125] in the context of complex event processing, or [127,128,168,160] on model-based performance or reliability engineering of software systems. Second, we excluded papers that only describe transformations between notations representing uncertainty that are semantically equivalent. However, we did include those works proposing transformations if the representation in the target domain has a specific purpose related to uncertainty analysis, and it is shown in the paper. For instance, we included [117,118], which transform from Fuzzy UML models to Fuzzy Description Logics and to Fuzzy Ontologies, because they use the representation in the target domain for uncertainty analysis purposes. Third, we did not consider works proposing representations of uncertainty that do not use software modeling notations; for example those that use mathematical models [204], programming languages [136], programming libraries [152,173], or use lower-level modeling notations, such as partial Kripke structures [132]. Finally, we excluded papers that provide classifications of the different kinds of uncertainty applied to specific application domains, such as [135,144,180,199,201,207], or that do not explicitly represent uncertainty [124].

3.3 Data sources

The first step for collecting the research works relevant to this study is to identify the digital libraries and electronic databases to use as sources. Apart from their popularity, and inspired by [150], we considered a strong requirement for the selected digital libraries the possibility to perform batch retrievals of bibliographical references. This allowed us to use reference managers for handling the papers. In particular, we used JabRef[1] and Zotero,[2] which are both open source. Having all the papers loaded into a reference manager reduces processing time compared to having to look at each paper on the corresponding digital library website. In addition, both reference managers can perform quality checks on the references to detect duplicate entries or missing details, so we were able to save time and ensure the quality of the supporting data.

After analyzing several digital libraries, we selected the ones displayed in Table 1. All of them provide advanced search engines.

3.4 Search strategy and paper selection

Table 1 displays the number of papers obtained from each digital library. We used the different engines for applying the query *(vagueness OR imprecis* OR uncertain* OR accura* OR fuzz*) AND ("software model"*

[1] `http://www.jabref.org/`
[2] `https://www.zotero.org/`

Table 1 Search engines and number of studies retrieved.

Digital Library	Studies
ACM	586
DBLP	422
Elsevier	173
IEEE Xplore	389
Scopus	1,403
SpringerLink	224
Web of Science	788
Total	3,985

OR uml OR "modeling language") on the papers' metadata, i.e., title, abstract and keywords. Note that we did not only try to search for *uncertainty*, but also for other terms sometimes used in the same context, such as *vagueness*, *imprecision*, *accuracy* and *fuzziness*. Also note that there are other design languages apart from UML that can be used to model uncertainty. Moreover, our search query uses wildcards. Given that not all search engines allow this type of search, we tried to match the search terms as closely as possible when queries with wildcards were not supported. In addition, not all engines allow to search in the title, abstract and keywords of the papers. For example, with SpringerLink we could only search in the papers' title and main text.

We generated one *bib* file per search engine. SpringerLink only produces *CSV* files, so we had to use Zotero to obtain a *bib* file with the papers [179]. Then, we consolidated all references together in one single *bib* file that was loaded into JabRef. We started with an initial set of 3,985 papers, as shown in Fig. 1, which displays an overview of the selection process. We used JabRef and Zotero to remove duplicates, obtaining 3,131 papers after this first step.

The next step was to apply the inclusion and exclusion criteria presented above to filter the documents from the set of 3,131 papers. For this task, they were split among the four authors of this article, who screened them and decided whether they should be initially selected or not. We discarded 3,008 documents that did not match the inclusion criteria of the survey, maintaining a pool of 123 papers.

Then we carefully read these 123 papers and extracted the relevant information, as explained in Sect. 3.5. 43 of these papers were discarded in this phase, too, because when we read the papers in detail, we realized that some of them did not meet the inclusion criteria (the contributions of some of the papers were sometimes difficult to classify).

At this stage we also performed a process of backward snowballing [227], considering the related works described in the selected papers. The purpose of this process is to identify further papers that had not been retrieved by any search engine or that were discarded by

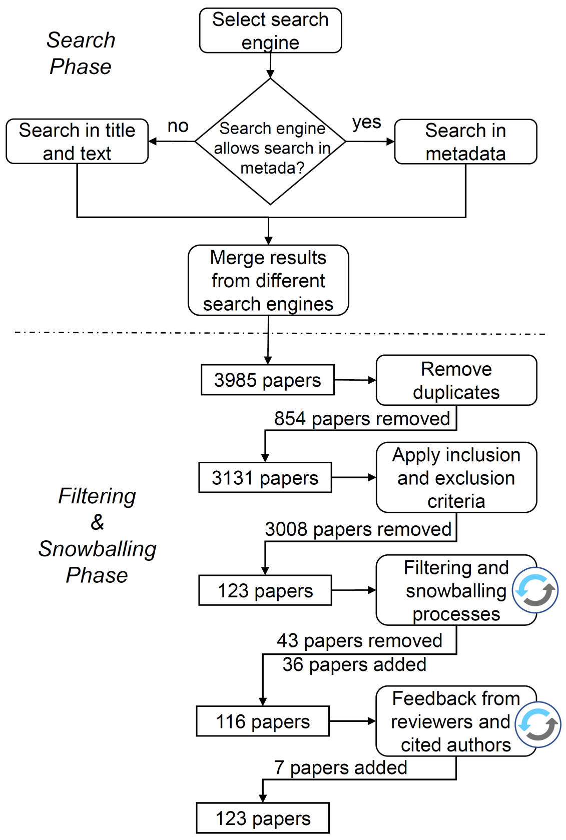


Fig. 1 Overview of the selection process.

mistake in the initial filtering step. 36 new papers were added after this process, making a total of 116 papers. Other three papers were suggested by the reviewers of the paper. Finally, before producing the final version we sent the paper to those whom we cited, to check our comments for accuracy and omission. This also provided one final stage in the systematic trawling of the literature for relevant work, which resulted in 4 more papers added to the final list after checking their suggestions and performing a filtering and snowballing process on them. This set of 123 selected papers will be referred to as *primary studies*, and are listed separately at the beginning of the References section. They can also be consulted online [220].

3.5 Data extraction

All 123 primary studies were carefully analyzed to answer our research questions. For each work, we extracted the following information: the types of uncertainty addressed; the logic or formalism and modeling notation used to represent the uncertainty; the software life cycle phase where the proposal is used; the kinds of analyses that the proposal enables as well as the tool support it

Table 2 Types of uncertainty and papers that deal with them (#Papers > 123 because many papers address more than one).

Type of uncertainty	**#Papers**	**Primary studies**
Behavior	37	[2, 5, 15, 17, 20, 21, 22, 29, 30, 35, 36, 37, 38, 39, 46, 54, 67, 68, 71, 72, 73, 77, 78, 82, 87, 90] [94, 98, 100, 101, 107, 108, 110, 115, 119, 120, 121]
Belief	36	[3, 4, 12, 16, 26, 41, 44, 45, 55, 56, 57, 58, 59, 60, 61, 79, 80, 87, 88, 91, 93, 95, 96, 97, 99] [111, 112, 113, 114, 117, 118, 119, 120, 121, 122, 123]
Design	31	[5, 9, 10, 17, 20, 23, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 39, 42, 47, 49, 52, 67, 81, 83, 84, 85, 86] [94, 107, 108, 115]
Measurement	50	[1, 6, 7, 8, 13, 14, 24, 26, 40, 43, 45, 47, 48, 53, 54, 55, 56, 57, 58, 59, 60, 61, 65, 66, 69, 70, 71, 72] [74, 75, 76, 87, 89, 91, 92, 95, 97, 102, 109, 110, 111, 112, 113, 114, 117, 118, 119, 120, 121, 123]
Occurrence	33	[8, 11, 26, 44, 45, 48, 51, 55, 56, 57, 58, 59, 60, 61, 69, 81, 87, 91, 92, 93, 95, 97, 99] [111, 112, 113, 114, 117, 118, 119, 120, 121, 123]
Spatiotemporal	11	[18, 19, 50, 62, 63, 64, 103, 104, 105, 106, 116]

Table 3 Types of uncertainty addressed per paper.

Types of uncertainty	**#Papers**
Behavior	17
Behavior, Belief, Measurement, Occurrence	4
Behavior, Design	12
Behavior, Measurement	4
Belief	10
Belief, Measurement, Occurrence	19
Belief, Occurrence	3
Design	17
Design, Measurement	1
Design, Occurrence	1
Measurement	18
Measurement, Occurrence	4
Occurrence	2
Spatiotemporal	11
Total	**123**

provides; the application domains in which the proposal is used, and the case studies carried out.

Primary studies were read by at least two different authors to avoid misinterpretations or misclassifications. To facilitate the process, we filled in a data extraction form for each primary study in a shared spreadsheet.

Summary. To answer **RQ1**, about the existing proposals for representing uncertainty in software models, we identified **123 primary studies** that range a 20-year time period (Jan. 2001 – Jan. 2020).

4 Types of uncertainty addressed

The previous section was intended to respond to RQ1 by identifying the existing proposals to represent uncertainty in software models. This section aims at answering RQ2: *What type(s) of uncertainty does each proposal address?*

To respond to this question we will use the classification framework introduced in Sect. 2.2, which defines six types: Behavior, Belief, Design, Measurement, Occurrence and Spatiotemporal. Tables 2 and 3 summarize our findings.

First, Table 2 identifies the papers that deal with each type of uncertainty. The total number adds up to 198 because 48 of the 123 papers deal with more than one type of uncertainty. We can see how there is a fairly uniform distribution of papers around dimensions, except for the Spatiotemporal uncertainty that has received less attention from the software modeling community so far, with only eleven papers. In contrast, Measurement uncertainty has received the highest attention, with 50 papers that deal with it.

Table 3 refines this information, describing the types of uncertainty addressed by each paper. Of the 123 surveyed papers, 75 focus on only one type of uncertainty, while 48 papers deal with more than one: 25 deal with two types, 19 with three, and 4 proposals deal with four types of uncertainty.

The remainder of this section will describe the types of uncertainty shown in Table 3, using a few illustrative examples that describe their most representative features, and how they have been modelled using some of the surveyed proposals. The order in which they are described below aims at improving readability, and therefore we will not strictly follow the order in which they appear listed in Table 3.

Belief uncertainty. Ten papers focus precisely on this type on uncertainty, using different approaches. For example, paper [122] adds degrees of belief to UML Use Case diagrams, using different options and logics: Plausibility, Possibility and Probability. A Fuzzy extension to OCL is proposed in [96] to enable the use of fuzzy variables within OCL constraints to combine requirements on meta-model level with domain heuristics. In [88], fuzzy and probabilistic information is used to represent Belief uncertainty in the UML class diagrams. Paper [12] allows belief agents to assign a degree of belief to model statements and elements (e.g., attribute values, constraints or OCL expressions in general) using probabilities, while paper [41] annotates models with subjective and temporal information. In turn, papers [3, 4] propose a collaborative conceptual model development framework, that uses Subjective logic for merging viewpoint models developed by different users, and offers various tools for reasoning and negotiating over the models and hence formally building and measuring consensus among the viewpoints. Other three papers deal with the specification of trust in software models. First, paper [16] defines Trust Transition Diagrams to specify the trust of each agent and how it evolves, using Fuzzy set theory. Then, papers [79,80] deal with trust management and risk analysis, using extensions of the STAIRS probabilistic sequence diagrams [77].

Measurement uncertainty. Proposals to represent this type of uncertainty generally define extensions of UML that enrich values of attributes with measurement uncertainty. Most of the papers use probability distributions [6,7,13,14,24,43,65,74,89,102,109] or Fuzzy set theory [66,75]. Two other papers use Probabilistic Relational Models (and extension of Bayesian Belief networks) to represent uncertain values [1,53]. Finally, other two papers deal with imprecise OCL constraints [40, 76]. We have also considered here the OMG proposals for modeling metrics (SMM [70]), real-time systems (MARTE [72]), or systems engineering (SysML [71]), which allow to add standard deviations or probability distributions to property values to represent measurement uncertainty. These annotations can also be used in activity diagrams to represent stochastic behaviors, combining these two types of uncertainty, Behavior and Measurement, as proposed as well in [54,110].

Occurrence uncertainty is treated in isolation in two papers. The first one [51] deals with cardinality constraints, which are assigned a degree of certainty on the objects they hold, while the second paper [11] allows modelers to assign a degree of certainty to the model instances themselves. The rest of the proposals combine Occurrence with other types of uncertainty.

Belief, Measurement and Occurrence uncertainties. There is a significant number of papers that use Fuzzy set theory to represent uncertainty in software models, combining Belief, Measurement and Occurrence uncertainties. For example, a UML Profile (FAME) [44, 45] was proposed for adding fuzzy information to UML model elements. However, most of the works on expressing uncertainty using Fuzzy set theory employ Fuzzy UML [58,59,60,91] or Fuzzy Entity-Relationship (ER) models [55,57], sometimes combined with Description logic [26,56,61,117,118,123] or with Fuzzy probabilities [111,112].

Fuzzy UML follows the initial ideas by Zvieli and Chen [234] and distinguishes three levels of fuzziness in UML models. The first one assigns an attribute or a class a degree of belief, which in Fuzzy set theory corresponds to a real number between 0 and 1 expressing a degree of membership. The second level is related to whether some instances are indeed instances of a class, for which an additional attribute (μ) in the class indicates a membership degree of the instance. The third level refers to the fuzziness of the value of an attribute, indicated by a keyword FUZZY in front of the attribute name. They correspond, respectively, to what we have called Belief, Occurrence and Measurement uncertainty

Note that other proposals that use Fuzzy set theory for representing uncertainty in software models employ plain XML, defining specialized schemata. They normally cover several types of uncertainties, such as Belief, Measurement and Occurrence uncertainty [58,95,97,113, 114]; or Belief and Occurrence [93,99]. The main goals of using an abstract syntax such as the one provided by XML are to enable the notation-independent representation of uncertainty and to facilitate the interoperability among modeling notations that use Fuzzy set theory for representing uncertainty. Therefore our decision to include them in this survey.

Three other proposals [8,69,92] combine Occurrence with Measurement uncertainty, expressing vagueness in attribute values, class instances and association links, based on Fuzzy set theory and using either UML or their own modeling notations. Occurrence and Measurement uncertainty is explicitly represented in [48] using probability distributions to achieve Risk and Cost analysis under uncertainty.

Example 1. To illustrate the representation of Belief, Occurrence and Measurement uncertainties, let us use an `Alarm` class (see Fig. 2) that represents the devices typically used in buildings, industrial plants and other facilities to detect fire. Objects of this class have three attributes: an identifier `ID`, e.g., the device serial number; the current `temperature` of the room, as read by a sensor embedded in the device; and whether there is `smoke` in the room of not, as detected by another internal sensor. First, sensors' readings may have some Measurement uncertainty, in terms of precision; for example, the temperature sensor may have a *precision* of 0.1 degrees Celsius, or the smoke sensor may have a *sensitivity* of 85%. Second, the Alarm device itself may be working or not, so we need to assign a *confidence* to it (Occurrence uncertainty). Finally, external belief agents may assign a *degree of belief* (e.g., 0.7) to the temperature readings of the alarm, indicating their subjective trust on that sensor (Belief uncertainty).

Figure 2 shows the `Alarm` class modeled using several proposals for representing these three types of uncertainty. First, the alarm is shown in (a) in plain UML, i.e., without any uncertainty. Then, Fig. 2(b) uses the UML uncertain datatypes proposed in [6,11, 12]. Measurement uncertainty is specified by means of

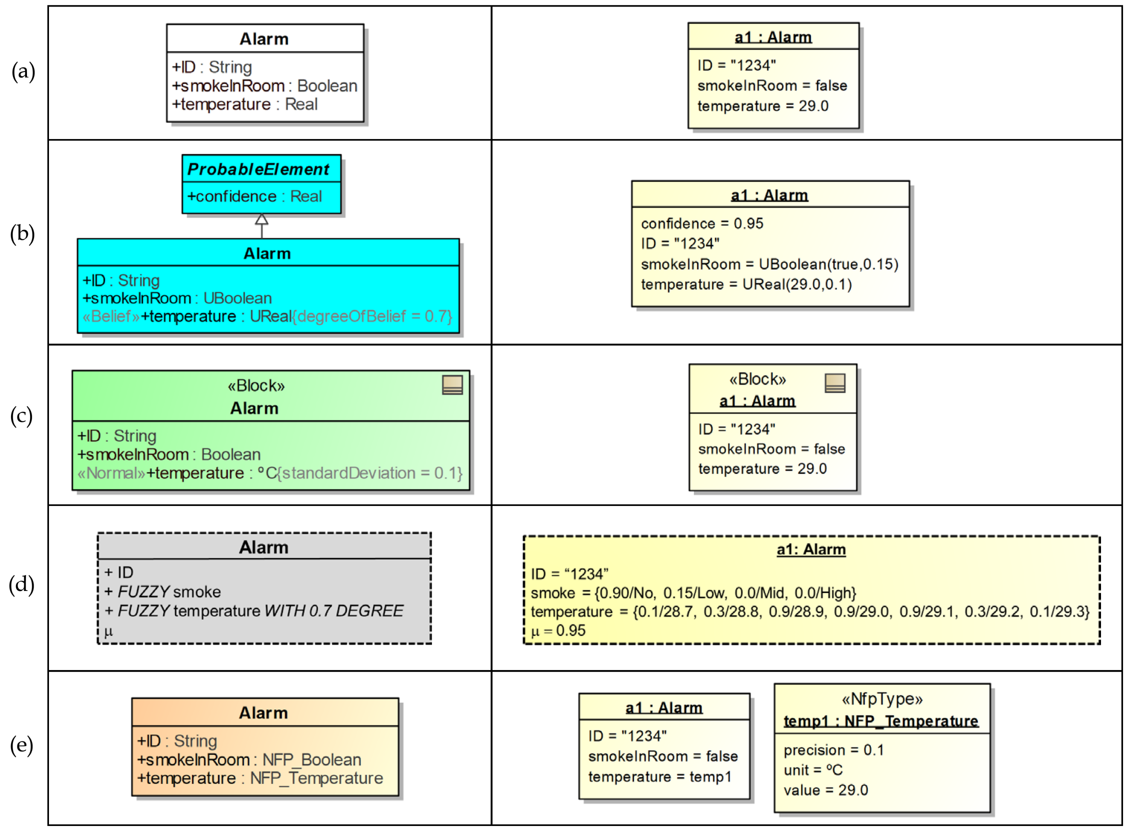


Fig. 2 Different proposals for modeling the `Alarm` class and one of its instances: (a) without uncertainty; (b) using UML uncertain datatypes [6]; (c) with SysML [71]; (d) with Fuzzy UML [59]; and (e) with the UML MARTE profile [72]

the primitive datatypes `UReal` and `UBoolean`, which allow modelers to specify the required precision and sensitivity in the sensor readings; Occurrence uncertainty is modeled by an additional attribute (`confidence`) inherited from abstract class `ProbableElement`; and the Belief uncertainty is modeled using stereotype «`Belief`» that enables the specification of degrees of belief to model elements. Fig. 2(c) shows the system modeled with SysML [71], which only supports Measurement uncertainty. It does so by assigning probability distributions to the possible values of measures, which are modeled as Valuetypes: in this case, the values of the attribute `temperature` follow a Normal distribution with standard deviation 0.1. One advantage of this proposal is that units can also be specified in Valuetypes (Celsius, in this case). Fig. 2(d) displays a model of the system using Fuzzy UML [58, 59]. This notation supports the specification of the three types of uncertainty: Measurement uncertainty is specified by declaring the two sensing attributes as `FUZZY`; Ocurrence uncertainty is specified by the μ attribute; and the Belief uncertainty on the `temperature` attribute is modeled using the expression "`WITH 0.7 DEGREE`" (in Fuzzy UML, individual degrees of belief held by different agents about the same model element cannot be specified). We could not find any modeling tool that supported the Fuzzy UML notation, unfortunately, despite the large amount of papers that describe it. Finally, Fig. 2(e) shows a model of the system specified with the MARTE UML profile [72]. Similar to SysML, MARTE only supports the specification of Measurement uncertainty. In MARTE, this is accomplished by means of the `precision` attribute of *NFP Types*. These are extensions of datatypes that incorporate precision, units and other properties to the values of attributes that represent physical quantities [72]. In the example, attribute `temperature` has been defined using `NFP_Temperature` NFP type, and therefore a `precision` can be specified.

Behavior uncertainty. Seventeen papers focus on the representation of this type of uncertainty, in different ways. Seven of them make use of probabilistic state machines [22, 38], probabilistic sequence diagrams [36, 77, 78, 82], Markov chains [2], or stochastic temporal logic [15]. Two of them use Fuzzy State Machines [46] or Fuzzy DEVS [37] to represent the stochastic behavior of the system objects. One paper [73] proposes a probabilistic extension of the π-calculus to specify the uncertain behavior of the system environment. Two other papers use some form of variability to deal with uncertain behavior, depending on the application domain: component

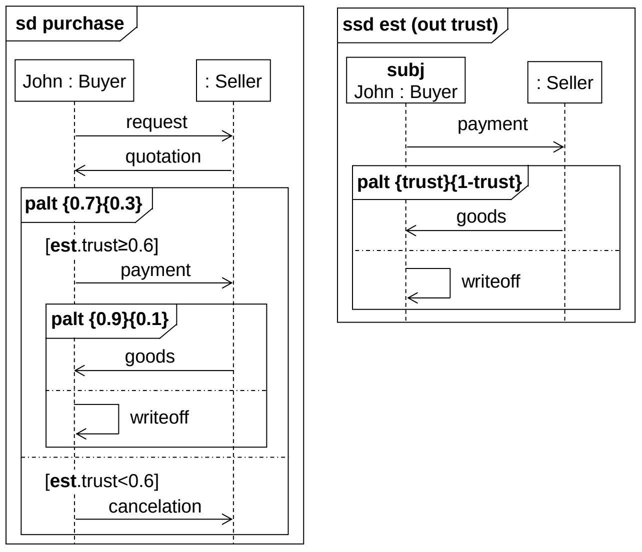


Fig. 3 STAIRS probabilistic sequence diagrams representing uncertain behavior and trust [80].

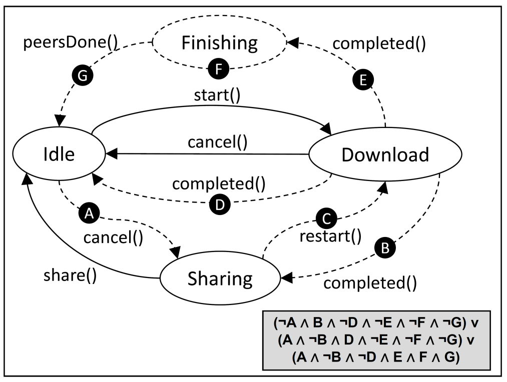


Fig. 4 Uncertain design of the state machine of a distributed file sharing application [33].

interfaces with variability are used in the development of component-based systems in [100], while the selection of the best adaptation to use is proposed in [21] within the Rainbow project to deal with uncertain behavior in self-adaptive systems. Partial behavioral models [90,101] allow the specification of incomplete behaviors using modal-transition systems (MTS) [172]. The work [101] extends the semantics of scenarios expressed in, e.g., sequence diagrams, to produce MTS, while [90] extends Harel's Live Sequence Charts [155] also with MTS semantics. Finally, one paper [68] provides a formal semantics for Fuzzy Communication Diagrams in terms of Fuzzy Petri Nets, and other [98] also uses Fuzzy Petri Nets to simulate a system specified in Fuzzy UML.

Example 2. Figure 3 displays subjective sequence diagram, as defined in [80], which is one of the ways in which uncertain Behavior and Belief can be represented in models. In addition to the probabilistic choices declared in the `palt` blocks, the behavior of the system is also determined by the subjective probability estimate `x` of agent `John` to *trust* the seller, and pay before the goods are delivered. Behavior, trust and risk analyses can be carried out based on such specifications.

Design uncertainty. Most papers that address this type of uncertainty use variability models to describe the possible design alternatives, using different approaches, such as partial models or design space exploration techniques. Partial models are used in general [31,33,34,83,84] or applied to particular domains where uncertainty is unavoidable, such as requirements engineering [85], model evolution [86], software product lines [10,23,32], Web engineering [9], or collaborative modeling [28]. Design space exploration techniques are more common in approaches that deal with the uncertainty of model transformations [25,27,52]. The uncertainty in this case is a consequence of writing non-deterministic programs, i.e., programs that do not have a general consistency restoration procedure or that fail in being deterministic. Some authors also assign probabilities to the different design options in order to conduct various types of analysis, particularly in the field of enterprise architectures to analyse their quality properties [49], or evaluate their evolution [42].

Example 3. To illustrate one way to represent Design uncertainty in models, let us use the case study described in [31,33], and shown in Fig. 4. It corresponds to a simple peer-to-peer file sharing application whose behavior is specified by a UML State Machine. Because of the user's vague requirements, six alternative behaviors must be considered simultaneously during the early design phases, until the user decides what (s)he wants. The 6 alternatives are described in the diagram shown in Fig. 4, using the compact notation defined in [33] to specify partial models. The dashed lines represent decisions not yet made, and the "*may*" formula determines the valid combinations of the possible options. The reasoning about the system behavior considering all possible alternatives is supported by the tool MU-MINT [31].

Behavior and Design uncertainties. Twelve papers deal with both Behavior and Design uncertainties. They all focus on the analysis of behavioral requirements of the system under the presence of uncertainty, mostly within the field of self-adaptive systems. They use different approaches, such as feature or variability models [39,94], Fuzzy set theory [5,20,30,115], Fuzzy branching temporal logic [107,108], stochastic Petri nets [35], or even Machine Learning techniques (Model Trees Learning) [29]. Finally, another paper of this group [17] uses partial models and probabilities to deal with uncertain behavior in assurance cases in the Automotive domain,

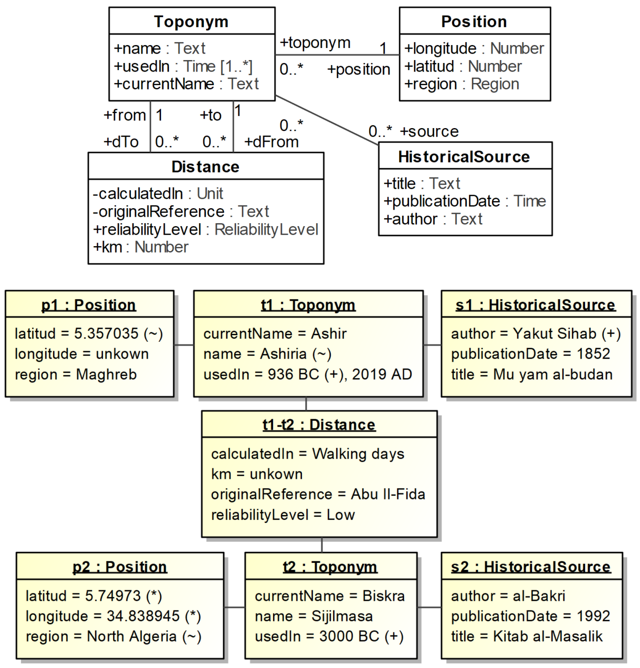


Fig. 5 ConML classes (above) and objects (below) representing information with Belief and Spatiotemporal uncertainty.

while partial models are used in [67] to handle uncertain environments.

Design and other uncertainties. One proposal [47] combines Design and Measurement uncertainty in the domain of business modeling to conduct profitability analysis. It assigns probability to alternative designs and uses probability distributions to describe the variability of the attributes' values. With the same goal, feature models with fuzzy weights that denote fuzzy priorities of the possible alternatives are also used [81] to represent Design and Occurrence uncertainty.

Spatiotemporal uncertainty. Eleven papers address Spatiotemporal uncertainty, using different notations. Six of them [18,19,50,64,106,116] use Fuzzy UML or other fuzzy notations. Three of them [103,104,105] use a formalism called Object-Fields to represent this type of uncertainty in software models. Two other papers [62, 63] use intervals and coarse-grained datatypes to specify vagueness and accuracy in time and space information.

Example 4. To illustrate the use of Belief and Spatiotemporal uncertainty and its representation in software models, we will use one of the existing proposals [63], which defines a UML-like language called ConML for this purpose. The system shown in Fig. 5 is extracted from a project from Digital Humanities, called DICTOMAGRED [181], aimed at extracting information about the location of toponyms in North Africa as they appear in historical sources. The upper part of Fig. 5 shows some of its main classes, and the lower part displays a few objects. We can see that special datatypes `Number`, `Unit`, `Time` can accept vague or even `unknown` values, and attributes values can be qualified with subjective *degrees of belief*: (`*`) stands for sure, (`+`) means probable, (∼) possible, (`-`) improbable, and (`!`) impossible. In addition, a `ReliabilityLevel` can be associated to `Distance` objects to indicate the confidence that we have on the information they provide.

Behavior, Belief, Measurement and Occurrence uncertainties. Four proposals deal with these four types of uncertainty. They correspond to papers that describe overarching proposals for representing the various types of uncertainty that may simultaneously happen in certain domains, such as software architectures [87] or cyber-physical systems [119,120,121]. They usually support several options for representing each type of uncertainty using different formalisms, leaving it up to the user to decide which one to use.

Summary. **RQ2** queried about the type of uncertainty addressed by each proposal. As shown in Table 2, the 6 types of uncertainty are covered by the primary studies in a balanced way, although a slightly higher treatment of Measurement uncertainty is observed (25% of the papers), and lower (5%) in the case of Spatiotemporal uncertainty. In turn, 75 papers addressed only one type of uncertainty; the rest treated two (25), three (19) or even four (4).

5 Logics, formalisms and notations

This section describes the logics, notations and formalisms used in the papers, with the goal of answering RQ3: *What notations and formalisms are used to represent each type of uncertainty?*

First, Table 4 lists the logics and formalisms used by the different proposals. The first column defines the keyword used to identify them, while column two lists alternative notations used. For example, Probabilistic Automaton refers to those approaches that use Probabilistic or Stochastic State Machines, or Probabilistic or Stochastic Sequence Diagrams. The last two columns indicate the number of papers that use the formalism and the type of uncertainty represented, respectively.

Table 5 indicates in detail the combination of logics and formalisms that the primary studies use, depending on the type of uncertainty targeted. It extends the information shown in Table 3, identifying the logics and formalisms used in the different proposals.

Notations used. Another aspect of interest is the notations used by the proposals for representing uncertainty.

Table 4 Logic/Formalism used (#Papers>123 because many papers use more than one formalism).

Keyword	**Includes**	**#Papers**	**Type of Uncertainty Represented**
Possibility	Fuzzy logic	43	Belief, Measur., Occur., Spatiotemp.
Probability	Credence, Confidence	22	Belief, Design, Measur., Occur., Spatiotemp.
Plausibility	Belief Functions, Entropy	7	Belief, Occur., Spatiotemp.
Bayesian Belief Networks	Causal Networks	3	Belief, Measurement
Fuzzy Description Logic		6	Belief, Measurement, Occurrence
Fuzzy Probability		4	Belief, Measurement, Occurrence
Subjective Logic		2	Belief
Prob. Distributions	GUM	16	Measurement
Many-valued logics	Null values	4	Measurement
Intervals		4	Measurement, Spatiotemp.
Discrete Random Vars.		3	Measurement, Spatiotemp.
Fuzzy Numbers		2	Measurement
Probabilistic Automaton	Prob. State Machines, Prob. Seq. Diagrams	16	Behavior
Fuzzy Temporal Logic		2	Behavior
Fuzzy Automaton	Fuzzy State Machines, Fuzzy Seq. Diagrams	2	Behavior
Fuzzy Petri Nets		2	Behavior
Prob. Process Algebras		1	Behavior
Stochastic Petri Nets		1	Behavior
Modal Transition Systems		2	Behavior
Variability Models	Partial Models, Feature Models	26	Design, Behavior
Design Space Exploration	CSP, ASP	3	Design

Table 5 Combinations of Logics/Formalisms used in each paper (#Papers=123).

Keyword	**#Papers**	**Type of Uncertainty**	**Primary Studies**
Fuzzy Automaton	2	Behavior	[37,46]
Fuzzy Petri Nets	1	Behavior	[68]
Possibility, Fuzzy Petri Nets	1	Behavior	[98]
Probabilistic Automaton	7	Behavior	[2,15,22,38,77,78,82]
Probabilistic Automaton, Intervals	1	Behavior	[36]
Probabilistic Process Algebras	1	Behavior	[73]
Modal Transition Systems	2	Behavior	[90,101]
Variability Models	2	Behavior	[21,100]
Plausibility	1	Behavior, Belief, Measur., Occur.	[87]
Plausibility, Possibility, Probability	3	Behavior, Belief, Measur., Occur.	[119,120,121]
Possibility	4	Behavior, Design	[5,20,30,115]
Probability, Variability Models	1	Behavior, Design	[17]
Fuzzy Variability Models	3	Behavior, Design	[29,39,94]
Temporal Logic	2	Behavior, Design	[107,108]
Variability Models, Design Space Exploration	1	Behavior, Design	[67]
Stochastic Petri Nets	1	Behavior, Design	[35]
Prob. Distributions, Probabilistic Automaton	4	Behavior, Measurement	[6,7,43,74]
Possibility	1	Belief	[96]
BBN, Possibility	1	Belief	[16]
Fuzzy Probability	1	Belief	[88]
Plausibility, Possibility, Probability	1	Belief	[122]
Probability	2	Belief	[12,41]
Probabilistic Automaton	2	Belief	[79,80]
Subjective Logic	2	Belief	[3,4]
Possibility	11	Belief, Measurement, Occurrence	[45,55,57,58,59,60,91,95,97,113,114]
Fuzzy Probability	2	Belief, Measurement, Occurrence	[111,112]
Possibility, Fuzzy Description Logic	6	Belief, Measurement, Occurrence	[26,56,61,117,118,123]
Possibility	3	Belief, Occurrence	[44,93,99]
Variability Models	12	Design	[9,10,25,28,31,32,33,34,52,83,84,85]
Probability, Variability Models	3	Design	[23,42,49]
Variability Models, Design Space Exploration	2	Design	[27,86]
Intervals, Variability Models	1	Design, Measurement	[47]
Possibility	1	Design, Occurrence	[81]
BBN, Discrete Random Vars.	1	Measurement	[53]
BBN, Probability	1	Measurement	[1]
Fuzzy Numbers, Many-valued logics	1	Measurement	[75]
Probability	4	Measurement	[40,66,76,109]
Prob. Distributions	6	Measurement	[13,14,24,65,70,102]
Prob. Distributions, Many-valued logics	1	Measurement	[89]
Probability, Prob. Distributions	4	Measurement	[54,71,72,110]
Possibility	2	Measurement, Occurrence	[8,69]
Fuzzy Probability, Fuzzy Numbers	1	Measurement, Occurrence	[92]
Probability, Prob. Distributions, Variab. Models	1	Measurement, Occurrence	[48]
Probability	1	Occurrence	[11]
Possibility	1	Occurrence	[51]
Possibility	6	Spatiotemporal	[18,19,50,64,106,116]
Probability	1	Spatiotemporal	[103]
Discr. Random Vars., Intervals, Many-val. logics	2	Spatiotemporal	[62,63]
Plausibility, Possibility	2	Spatiotemporal	[104,105]

Table 6 Notations used (#Papers>123 because some papers use more than one notation).

Notation used	#Papers	%
DSL (new)	32	22.22%
DSL (existing)	23	15.97%
UML	32	22.22%
UML Profile	11	7.64%
OCL	10	6.94%
Fuzzy UML	22	15.28%
Fuzzy XML	7	4.86%
Other	7	4.86%
Total	144	100.00%

Table 6 shows this information. We can see how more than one third of the proposals (38.19%) use Domain-Specific Languages (DSL). Almost half of them are new languages, defined either in an ad-hoc manner to represent uncertainty (e.g., ConML [63] or the one supported by MU-MINT to represent partial models [31]), or as extensions of other DSLs. For example, UIMFL (Uncertain IFML) [9] is an extension of OMG's Interaction Flow Modeling Language, (IMFL) [192]. Other proposals use existing DSLs, because they already provide some features to represent uncertainty (e.g., SysML [71]) or simply use their basic functionality (e.g., KAOS [170]).

Almost 30% of the papers use UML, sometimes combined with OCL, to represent uncertainty. All these papers propose extensions to the standard notations by adding new datatypes or novel languages features. Eleven proposals use UML Profiles, which are the UML mechanism for extending the language in an orderly and semantics-preserving manner. A few of them use standard UML profiles, such as UTP (the UML testing profile [191]); while other proposals define new profiles, such as UUP [119], or DAM [87].

Around 20% of the papers (29) use Fuzzy UML [58], or Fuzzy XML [99]. Interestingly, these notations do not seem to be supported by model editors (see Sect. 8). Probably this is why their application within the field of model-based software engineering seems to be rather limited and mostly at the theoretical level, with no primary study reporting any industrial application of these notations.

Finally, a few proposals use other modeling notations, namely, Fuzzy DEVS [169], the Object-Field model [140], or fUML [194].

Summary. **RQ3** was interested in knowing the notations and formalisms proposed to represent each type of uncertainty. Our study shows a wide variety of formalisms used to support the model-based representation and analysis of uncertainty. They depend largely on the type of uncertainty being addressed and the type of analysis of interest (Tables 4 and 5). The notations used to represent uncertainty in software models are also quite diverse: approximately 38% use DSLs, 37% use UML, OCL or UML profiles, and 20% use Fuzzy UML/XML.

6 Application domains

This section describes the main application domains where uncertainty has been reported to be used, with the aim of answering RQ4: *In which domains are the different proposals applied?*

6.1 Case Studies

Examples, scenarios or case studies are often used to clarify the contents of the papers, illustrate the proposals, or evaluate them. The way this resource is used differs from paper to paper. Depending on the depth with which the example has been developed, as well as its closeness to the real world, we have classified the case studies according to the following categories:

– **Small Fragment**: Short example that shows a simple piece of a system. These examples are easy to understand, and usually known to the reader because they represent prototypical cases, such as a printer, a bank account, or a simple client-server application. They are described and developed in a very brief way, focusing on the theoretical concepts to be illustrated, and avoiding superfluous details.
– **Synthetic Case Study**: An example that describes a system, partly inspired in a real-life scenario that contains uncertainty, but made up to illustrate some particular aspect of interest and to ignore the rest. Despite their simplifications, these case studies allow the reader to effectively evaluate how the proposal would work in a real situation. Examples related to robotics, automotive applications, adaptive systems and real-time systems are commonly used in this category.
– **Real Case Study**: A case study based on a real application. These examples, unlike the previous ones, are analyzed and developed in depth. They are usually closely linked to research projects or challenges that arise in industrial environments and are brought to the academic world by an industry partner. Many of these case studies come from the Systems Engineering domain.
– **None**: Although it is not frequent, we have also found in the literature pure theoretical proposals that are not illustrated by any example.

Table 7 shows the number of papers classified in each category. From the 123 primary studies, 31 proposals (25.2%) have been validated in real settings and describe real case studies.

Table 7 Types of case studies used in the papers.

Type of Case Study	#Papers
Real	27
Synthetic	45
Synthetic plus Real	3
Small fragment	44
Small fragment plus Real	1
None	3
Total	**123**

6.2 Application domains

To classify the application domains where the primary studies have been carried out, we have used the standard ACM Computer Classification System (CCS) [126], descendeding to the second level of its hierarchy. We have identified 12 CCS categories, which are detailed below. Table 8 shows the primary studies carried out in each one, highlighting in a separate column those that have been applied in real applications.

1. **Enterprise computing**: mainly related to architecture frameworks (component-based architectures) and enterprise architectures (DoDAF). 5% of the papers (i.e., 6 out of 123) fall into this category and most of them use non-real case studies. Papers are dated between 2007 and 2019.
2. **Arts and Humanities**: application of IT in these fields. Used in 4 proposals, mostly in the last 3 years, and using both real and hypothetical cases.
3. **Dependable and fault-tolerant systems and networks**, including self-adaptive and context-responsive systems. Thirteen papers fall into this category, five of them applied in real case studies.
4. **Embedded and cyber-physical systems**: modeling uncertainty in this application domain is attracting a growing interest as reflected by the increasing number of papers published in the last five years. Moreover, this is the domain where more proposals use real case studies.
5. **Real-time systems**: Three papers fall into this application domain type, which includes case studies directly related to the industrial sector.
6. **Collaborative and social computing**: These are human-centered computing proposals that allow multiple parties to collaborate on textual and graphical models. Only one paper falls in this category, using a real case study.
7. **Data management systems**: database applications constitute 10% of our primary studies and tend to be the oldest ones. They are not normally reported in real scenarios.
8. **Cross-computing tools and techniques**: papers in this generic category of information systems present general approaches to represent uncertainty in software models, but without being oriented towards any specific application domain. Due to their generic nature, most papers in this category (that represents 23.6% of the papers) deal with non-real case studies.
9. **Information systems applications**: papers in this category include those dealing with uncertainty environmental applications such as energy, air quality, or land desertification, as well as geographic information systems or systems with spatial-temporal constraints. 12 papers address these issues, many of them dealing with real case studies.
10. **World Wide Web**: dedicated to systems that model uncertainty in the context of web applications and services. Four primary studies fall into this category, mostly applied in non-real case studies.
11. **Formal methods and theory of security**: these are papers dealing with aspects of privacy, security, trust and risk analysis. It only contains two papers, none of them based on real cases.
12. **Software notations and tools**, including context-specific languages, system description languages, or transformations between models. It accounts for 17.8% of the proposals, most of them based on non-real case studies.

Summary. **RQ4** investigated the domains in which the different proposals are applied. The results show three main fields of application. Firstly, we have the cyber-physical systems, real time and adaptive systems domains where uncertainty can play a key role, which together gather 31 proposals (25%), most of which are used in industrial projects. Secondly, we have general Information Systems applications, including enterprise computing, arts and humanities, collaborative computing, data management systems, as well as WWW and security applications, in which uncertainty is addressed in 41 proposals (33%). Finally, 51 proposals (41%) are applied in supporting applications for cross-computing tools and techniques, and software notation and tools. The use of real cases decreases in these last domains, where proposals are demonstrated with synthetic applications or simple examples.

7 Software Development Phase

This section is devoted to answer RQ5: *To which phase of the software life cycle does each proposal apply?* For this, we have used the traditional phases defined in the software engineering literature. They are agnostic with

Table 8 Types of application domains (according to the ACM classification system) and papers that deal with them.

Type of application domain	**#Papers**	**Real**	**Synthetic**	**Small Fragment**	**None**
Enterprise Computing	6	[87]	[22,26]	[42,49,73]	
Arts and Humanities	4	[62,63,64]		[41]	
Dependable and fault-tolerant systems and networks	13	[20,30,67,107,108]	[15,29,44,45] [66,115]	[5]	[21]
Embedded and cyber-physical systems	15	[39,54,89,110,119] [120,121,122]	[10,13,17,35,37] [38,74]		
Real-time systems	3	[2,36]		[72]	
Collaborative and social computing	1	[28]			
Data management systems	12		[114]	[8,55,56,57,58,59] [60,75,112,113]	[111]
Cross-computing tools and techniques	29		[3,4,6,7,11,12] [14,31,34,40,65] [78,85,96,100]	[9,33,43,51,61,76] [77,82,86,92,102] [118,123]	[88]
Information systems applications	12	[18,19,98,104,105] [106]	[46,90,101,103]	[50,116]	
World Wide Web	4	[53]	[117]	[16,91]	
Formal methods and theory of security	2		[79,80]		
Software notations and tools	22	[47,48,52,69]	[23,68,84,94,95] [97,99]	[1,24,25,27,32,70] [71,81,83,93,109]	
Total	123	31	45	44	3

respect to the development methodology used. We list them here only to clarify the terminology used.

– Requirements **Analysis**, including the elicitation and analysis of requirements.
– Software **Design**, including both the development of high-level conceptual models of the system, and the production of detailed design models. Model transformations can also be defined in this phase to conduct model analysis (horizontal transformations to other semantic domains) or to generate implementations (vertical model transformations).
– **Verification**, i.e., formal evaluation of the models to check whether they capture the customer needs and requirements, including simulation and other types of analyses, such as model checking, theorem proving, or light-weight formal methods.[3]
– **Implementation**, i.e., generation of the implementation code and the rest of the artifacts needed to execute the system. It can be partly automated in case of model-driven development, or manual.
– **Testing** (aka *validation*) of the system, i.e., the process of checking that the final product meets the specifications, which in this context includes, e.g., model-based generation of test cases and test scripts.
– **Operation**, including the deployment of the system in the real environment and its operational use.

[3] Verification and validation are two controversial terms, to which different people assign different meanings. In this context we will use the meanings defined and used in ISO standards [162].

Table 9 Software Life Cycle Phase targeted by each paper.

SLC Phase	**#Papers**
Analysis	8
Analysis, Design	7
Analysis, Operation	4
Analysis, Design, Operation	1
Design	35
Design, Implementation	11
Design, Operation	1
Design, Testing	2
Design, Verification	43
Testing	4
Verification	7
Total	**123**

– **Maintenance**: evolution of any of the artefacts or documentation associated to the software system.

Table 9 shows the software life cycle phases where the proposals are applied, together with the number of papers that target them. This is graphically depicted later in Fig. 10.

In turn, Table 10 lists the papers that target each phase. The total number adds up to 193 because many papers target more than one phase, as shown in Table 9. As expected, most of the proposals (100) target the Design phase because this is where software models are most commonly used in practice.

Proposals addressing the Analysis phase (20) normally aim at capturing the system requirements when they are unknown, incomplete, vague, imprecise or inconsistent. Most proposals provide reasoning support for

Table 10 Software life cycle phases targeted (#Papers>123 because many papers target more than one phase)

SLC Phase	#Papers	Primary studies
Analysis	20	[5,16,17,20,26,29,30,31,35,39,45,71,72,85,94,107,108,115,121,122]
Design	100	[1,2,3,4,6,7,8,9,10,11,12,13,14,15,16,18,19,21,22,23,24,25,27,28,31,32] [33,34,37,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,55,56,57] [58,59,60,61,62,63,64,65,66,67,69,70,71,72,73,74,75,77,78,79,80,81] [82,83,86,84,88,89,91,92,93,95,96,97,99,100,102,103,104,105,106,109,110] [111,112,113,114,116,118,120,121,122,123]
Verification	50	[1,2,3,4,6,7,10,11,12,13,14,15,22,23,32,37,38,40,43,46,47,48,49,51,56,63,64] [67,68,73,74,75,78,79,80,82,84,90,87,96,98,100,101,102,103,105,110,117,118,123]
Implementation	11	[19,53,57,58,59,93,95,97,99,113,114]
Testing	6	[36,54,76,89,119,120]
Operation	6	[21,29,30,35,39,115]

analysing the uncertainty in the system requirements, as we shall later see in Sect. 8.

Some proposals (35) only do Design, but other (43) also propose some sort of Verification and analysis on the produced models including, e.g., consistency, satisfiability, risk or performance analysis. Seven proposals focus mainly on verification. These analyses, as well as the tool support they provide, will be later described and discussed in Sect. 8.

Ten out of the eleven works that deal with the Implementation phase provide solutions for generating persistent representations of uncertainty values, using either XML schemata [19,58,93,95,97,99,113,114] or database tables [57,59]. As mentioned above, these artefacts are very valuable not only for storage and persistence purposes, but also to achieve interoperability among different modeling representations. Another paper [53] shows how to generate code and database artifacts to produce a functional prototype of a Web application, as well as the uncertainty model to allow the propagation of evidence and the knowledge inference.

Six proposals target the Testing phase of the software development cycle, by showing how test cases can be automatically generated for the system under test, considering uncertainty. Most of them also provide some type of analysis on the generated artefacts, such as test coverage, consistency, correctness, etc.

Proposals targeting the Operation phase usually correspond to systems that use models@runtime [130] to implement self-adaptive features, such as requirements or behavior. Thus, the system can self-adapt to changes in the requirements, the environment, or the actual usage of the system during the operation phase.

Finally, we found no modeling proposal targeting the Maintenance phase considering uncertainty.

Summary. **RQ5** inquired about the software development phase in which the primary studies are applied. As expected, the explicit representation of uncertainty in software models is proposed in those phases where software models play a prominent role. In particular, the Design phase is the one with most proposals (100), followed by the Verification phase where the software engineers check that the models with uncertainty meet the users' requirements (50).

8 Analysis provided and tool support

The last research question, RQ6, investigates whether, in addition to the representation of uncertainty, the proposals allow some kind of analysis on the models produced, and the level of tool support they provide.

8.1 Types of analysis

In Sect. 7 we identified 50 proposals that targeted the Verification phase of the software life cycle, providing some sort of analysis on the models. In addition, 16 of the 20 papers targeting the Analysis phase also provide some sort of analysis, which makes a total of 66 papers (roughly 54% of the primary studies of our survey) that are able not only to represent uncertainty, but also to conduct some kind of analysis on the models endowed with uncertainty. The rest of the papers simply proposed notations for the representation of uncertainty, but gave no hint as to the types of analysis that could be performed with these representations. Table 11 lists these types of analyses, as well as the papers that support each one. The number of papers in Table 11 adds up to 82 because some of the 66 papers provide more than one type of analysis. The categories in Table 11 include different analysis types, which are described next.

First, **Requirement Analyses** include the study of the consistency, refinement, satisfiability or traceability when requirements are vague [45], changing [5,29, 31,115] or partial [84,85], as well as operations for uncertainty mitigation [20,107,108], or fuzzy requirements elicitation [122].

Table 11 Type of analysis supported by primary studies.

Type of Analysis	#Papers	Primary Studies
Req. Analysis	12	[5,20,29,31,35,45,84] [85,107,108,115,122]
Queries	15	[6,7,11,12,13,14,40,43] [46,63,64,74,75,96,102]
Design Space Explor.	4	[10,30,32,47]
Correctness	11	[1,3,4,39,51,56,82,100] [117,118,123]
Capability Analysis	1	[26]
Cost Analysis	2	[23,48]
Performance Analysis	3	[46,49,110]
Risk Analysis	5	[23,48,79,80,94]
Reliability Analysis	4	[4,22,68,87]
Safety Analysis	1	[17]
Trust Analysis	2	[79,80]
Temporal Analysis	8	[2,15,38,67,73,78,90,101]
Simulation	12	[6,7,11,12,13,14,15] [37,40,90,98,101]
Visualization	2	[103,105]
Total	**82**	

Queries refer to the possibility of querying the model about the level of uncertainty of the complete system or that of some of its model elements. This includes supporting *uncertainty analysis*, i.e., the propagation of uncertainty throughout operations [163], and also *sensitivity analysis*, i.e., how the uncertainty in the output of a model or system can be divided and allocated to different sources of uncertainty in its inputs [208].

By **Correctness** we mean the analysis of the consistency, satisfiability or faithfulness [174] of the model specifications once they are endowed with uncertainty.

Design space exploration refers to systematic analysis and pruning of all possible design choices based on parameters of interest.

Other types of analyses include **Capability**, **Performance**, **Risk**, **Cost**, **Safety**, **Trust** or **Reliability** analysis. Although a few of them are carried out in the Requirements Analysis phase, the majority of the proposals conduct these analyses on the Design models.

Temporal analysis focuses on the study of the temporal properties of the system, be they real time properties or not. For the latter, some modal logics are generally used.

Twelve proposals allow the **Simulation** of the models endowed with uncertainty, using different notations such as Fuzzy-DEVS [37], Fuzzy Petri Nets [98], or action languages associated to modeling tools such as SOIL [134], the textual action language provided by the USE modeling tool [149].

Finally, two studies focus on **Visualization** of the Spatiotemporal information with uncertainty, using Object-Fields [140].

8.2 Tool support

We also wanted to know the level of tool support that the 123 proposals provided. For this purpose we differentiate between the following three different types of tools.

– **Model editors**, to specify the models and represent the associated uncertainty;
– **Analysis tools**, to carry out the analysis tests that the proposals provided; and
– **Supporting tools**, in charge of performing intermediate tasks, such as the transformations between the outputs of the model editors and the analysis tools.

First, Model editors are essential tools in model-based engineering, where models are software artifacts that must be able to be manipulated by computers, through model transformations or through other tools.

Table 12 shows that only 61 of the 123 proposals (49.6%) claimed any tool support for specifying their models. Tool support ranged from dedicated editors developed as Eclipse plugins, as stand-alone tools, or as extensions of existing model editors. Eleven proposals defined a UML Profile, so any UML modeling tool with support for Profiles could be used to edit these models.

The remaining 62 proposals (50.4%) remained at a theoretical level. Even when this topic is recent, we find such a number somewhat excessive for any engineering discipline that aims to provide useful results for practitioners and industrial developments. Besides, only 22 of the 61 proposals with a priori tool support for model edition provided a link to access the tool or UML Profile mentioned in the article, which means that only 18% of the 123 proposals could actually be modeled with tools.

Table 12 also shows the number of proposals that provided some sort of Verification/Analysis on the models endowed with uncertainty, as identified in the previous section. Of these 66 proposals, only 37 provided some form of tool support for implementing or carrying out these analysis, including queries, model verification, simulation, temporal analysis, etc. The type of tool varied according to the proposal: analysis tools developed adhoc for the proposal, external tools used to perform the analyses, libraries developed to perform the tests, etc. Of those that provided some type of analysis tool support, only 15 indicated a working link to use the proposal. The others did not mention any reference to a website or URL from which the analysis toolkit could be accessed, downloaded or used; mentioned that their tools were still in a prototype stage and not ready for public use; or provided a link that did not work.

The right side of Table 12 shows the breakdown of tool support according to the type of uncertainty

Table 12 Tool support provided by the primary studies.

	123 (100%)	Design	Meas.	Belief	Occur.	Behav.	Spatiot.
Proposals with tool-supported editors	61 (50%)	19	25	13	9	23	3
With working link to the editing tool a	22 (18%)	7	11	1	2	6	0
Proposals providing model analysis/evaluation	66 (54%)	20	20	14	10	28	4
Proposals supported by analysis tools	37 (30%)	10	13	4	3	14	2
With working link to the analysis tool	15 (19%)	6	7	1	2	6	0

handled by the different proposals. There is no particular bias towards any of them. Something that is interesting to note is that more than half of the proposals with tool support have been published in the last 5 years (see Fig. 11 in Sect. 9), which represents a positive trend. Even so, the number of proposals with tool support remains very low.

Summary. **RQ6** investigated the level of tool support provided by the different proposals. Firstly, only 61% of the proposals provide tool-supported model editors with the ability to describe the models with uncertainty, a somewhat disappointing result for an engineering discipline. Fortunately, the situation is improving in recent years with the emergence and adoption of Software Language Engineering tools and techniques. Secondly, only half of the primary studies that proposed analysis methods were supported by tools, and only a third made these tools available to the community. Once again, we see a recent trend to make tools and other software artifacts available to the community so that other researchers and practitioners can use them in their applications. In turn, the types of analysis provided by primary studies widely differ (see Table 11), with four main groups: requirement analysis, model queries, quality analysis (such as correctness, cost, reliability, trust, safety, etc.), and simulation and visualization.

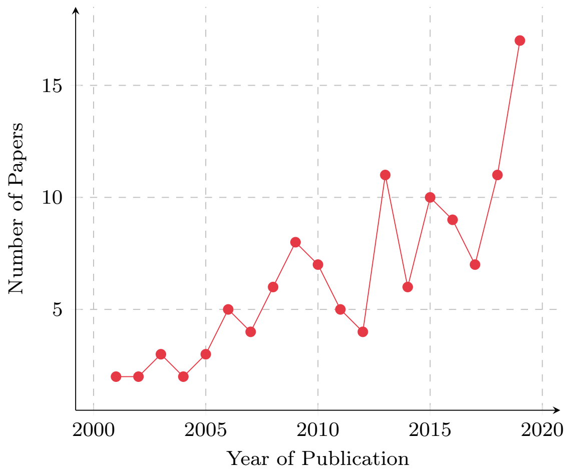


Fig. 6 Number of papers per year of publication.

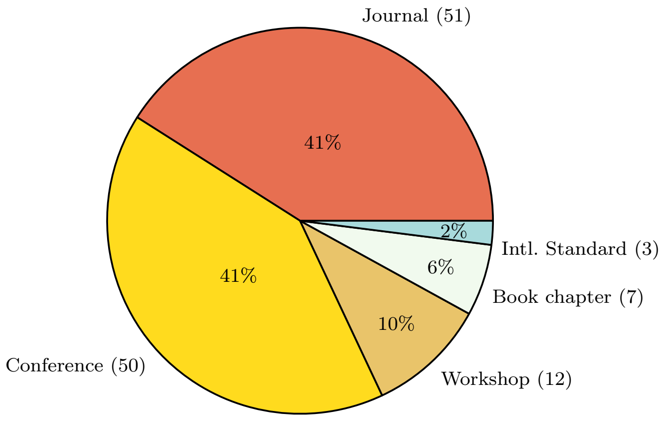


Fig. 7 Distribution of papers per publication type.

9 Analysis of literature review

9.1 Timeline and publication trends

Figure 6 shows the number of primary studies according to the year of their publication. We can see how there is an increasing trend throughout the years.

9.2 Publication venues

Figure 7 shows the distribution of papers according to the type of publication. Figure 8 refines this information by showing the evolution over the years of the different publication venues where the primary studies have appeared. We use the 5-year average, which is more informative and visually understandable than the individual years' figures, because it absorbs the peaks that some particular years present (this is normal when dealing with publications in journals, which may suffer delays that cause artificial accumulations in some years).

9.3 Research distribution across uncertainty types

Figure 9 shows a pie chart with distribution of primary studies published for each type of uncertainty. This information is refined in Table 13, which shows the number of papers published every year for each type of uncertainty. We can see an increasing trend in the number of publications, which is similar in all types of uncertainty.

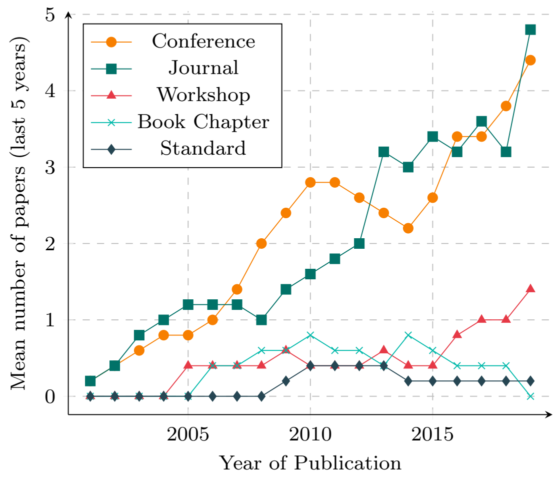


Fig. 8 5-year average of published papers according to the type of publication.

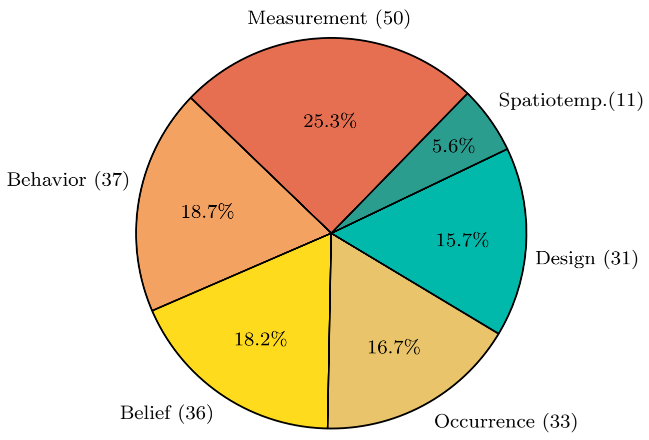


Fig. 9 Distribution of papers per type of uncertainty.

9.4 Software Development Phase

Figure 10 displays the number of primary studies that are applied in each software life cycle phase, which were described in Table 10. As mentioned in Sect. 7 when discussing these results, they correspond to how general software models are used in these phases, too.

9.5 Tool support

Finally, Fig. 11 shows the 5-year average number of primary studies supported by tools. Again, the 5-year average is used to display the data because this information allows to see the trends in a more informative and visually understandable manner than showing the individual years' numbers. We can clearly see in the chart how the number of primary studies that provide tool support steadily increases, with a significant growth in the last five years.

Table 13 Number of papers that address each type of uncertainty per year of publication.

Year	Beh.	Bel.	Des.	Mea.	Occ.	Spa.	Total
2001	0	0	0	1	1	1	3
2002	0	2	0	1	2	0	5
2003	0	1	2	0	2	0	5
2004	0	0	0	2	2	0	4
2005	0	2	0	1	1	1	5
2006	1	2	0	3	2	1	9
2007	2	1	0	2	1	0	6
2008	2	3	0	1	0	0	6
2009	4	2	2	4	1	0	13
2010	4	2	2	2	1	1	12
2011	3	2	1	2	2	0	10
2012	0	2	1	3	2	0	8
2013	4	4	5	3	3	0	19
2014	1	2	2	2	4	0	11
2015	2	1	4	4	1	1	13
2016	2	2	3	6	2	0	15
2017	3	1	3	1	1	2	11
2018	3	3	2	5	3	1	17
2019	6	4	3	7	2	3	25
Jan'20	0	0	1	0	0	0	1
Total	**37**	**36**	**31**	**50**	**33**	**11**	**198**

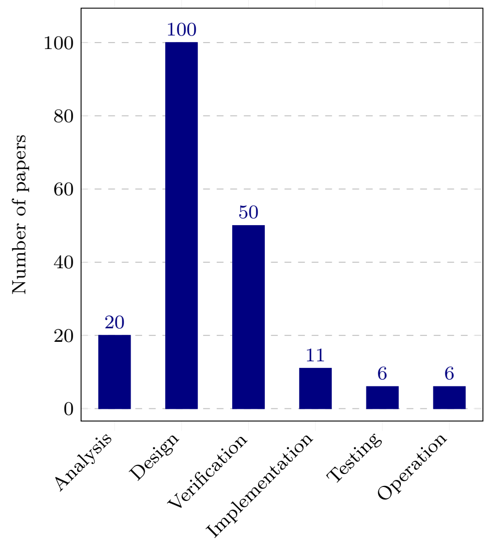


Fig. 10 Number of papers for each software life cycle phase.

10 Discussion

The previous sections aimed to give a picture of the state of the art of the explicit representation of uncertainty in software models and its evolution since it started, in the early 2000s. This section examines some of the trends, opportunities and challenges of this topic. The threats to the validity of our study are also presented.

10.1 Maturity level

From our survey we can see how the community has explored many different options, using alternative notations and underlying logics, and applied them in diverse

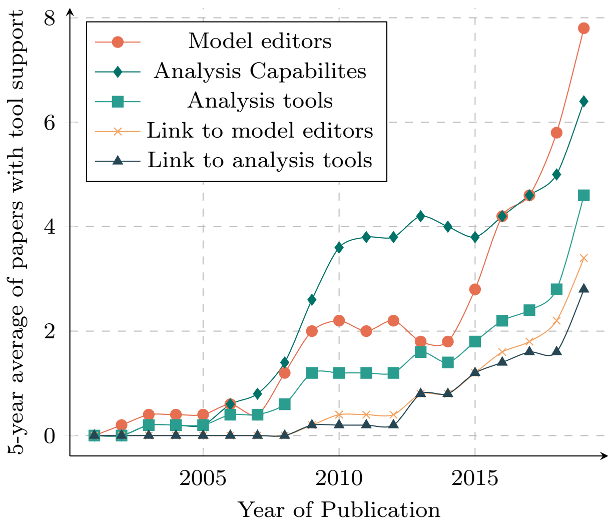


Fig. 11 5-year average number of primary studies supported by tools.

domains and case studies. From our point of view, efforts so far have been mostly of an exploratory nature, trying to investigate the different types of uncertainty that occur in these domains, how to represent them explicitly in software models, and how to reason about the system with these models once they are endowed with uncertainty.

This basically corresponds to the *Initial* level of any Maturity Model [159]. This level is characterized by heroic and individual efforts, normally undocumented and driven in an ad-hoc, uncontrolled, chaotic and reactive manner. Tool support is scarce and no agreed guidelines and methods are in place. This is the situation identified by our survey. Now we need to move forward, advancing through the following levels.

In next maturity level, called *Repeatable*, the process is documented sufficiently such that repeating the same steps may be attempted, possibly with consistent results. In this case, precise guidelines need to be in place to help software engineers to identify the types of uncertainty that can affect their application domains, suggest the best notations to use depending on the required level of analysis and tool support, and count on methods to implement them in practice. Although not rigorous yet, the process of modeling uncertainty should be documented, predictable and supported by engineering tools.

In the third maturity level, called *Defined*, the community should have agreed on a series of standard processes and tools for modeling uncertainty, which should be widely known and have a supporting history of successful (and failing) applications and validation in a range of situations. They should also be subject to some degree of improvement over time.

The final two phases, *Capable* and *Efficient*, respectively require that the representation of uncertainty in software models be quantitatively managed according to agreed metrics, and that it be subjected to deliberate optimization and improvement processes.

Our survey makes it clear that we are still at the *Initial* maturity level. However, we have also seen a recent trend to apply these proposals in real case studies, from various application domains. The results and lessons learned from these experiences are fundamental to start working towards the next level of maturity, through the *development of guidelines, methods and tools for the representation and analysis of uncertainty in software models*. In particular, *tools* are essential in any engineering discipline. Without them we can once again disappoint the software industry with promises that will not help them solve the real problems they have, this time about managing uncertainty as a first-class citizen of their analyses, designs and developments.

Reaching this next maturity level is probably the greatest challenge that uncertainty modeling currently faces. Note that it is also important that we consolidate the second level before we can address some of the issues that fall under the third one, such as standardization.

10.2 Trends and Opportunities

Our survey shows an increasing number of research efforts dedicated to uncertainty modeling, which means that there is a vibrant community interested in the topic. Next we describe the opportunity envisioned, possible next steps and the importance of industry.

Opportunity. The growing number of application domains where uncertainty needs to be explicitly represented in software models demonstrates an increasing interest in this area, particularly in certain fields such as self-adaptive systems, the IoT or cyber-physical applications. Such a momentum could be effectively used to leverage the research power of the modeling community, which has an outstanding opportunity to work in collaboration with other physics and engineering disciplines, and with the industry, on this topic.

Next steps. So far, the efforts have been mostly made in a disorganized and disorderly manner, mainly to explore modeling alternatives. Perhaps this is a good time for the modeling community to try to organize their efforts to achieve concrete objectives, useful for the industry and, as mentioned before, that can help reach a higher maturity level. In particular, (i) reaching a consensus on a taxonomy of types of uncertainty, (ii) identifying the most appropriate notations for representing them depending on the situation, and (iii) providing useful tool support, would constitute the most sensible next steps.

Industry-driven. It is very important that we start to be driven by industrial requirements, in addition to the mainly scientific or research initiatives that have guided us so far. In this way, we will be able to identify the key problems to be addressed, as well as the main requirements and limitations of our proposed solutions. Industry's current interest in this issue, as demonstrated by the number of real case studies used in the primary studies, should be the driving force behind our efforts.

10.3 Threats to validity of our study

According to Wohlin et al. [228], there are four basic types of validity threats that can affect the validity of our study. We cover each of these in the following.

10.3.1 Conclusion Validity

Threats to the conclusion validity are concerned with the issues that affect the ability to draw correct conclusions and whether the survey can be repeated. In order to mitigate these threats, we have made available the set of selected primary studies [220], from which papers are easily accessible, so that the experiment can be repeated and the results verified. Another possible threat to conclusion validity is that of *publication bias* [218]. This threat refers to works rejected by reviewers or editors, as well as works not submitted or published by their authors, as they might be considered non-significant. Such papers might alter the results of our study. However, we decided not to include them for two reasons: the difficulty of finding them, and their possible lack of quality since they have not undergone the scientific filter of a peer review process. In this respect, the potential advantage of including them could be jeopardised by the possible disadvantages they might entail.

10.3.2 Construct Validity

These threats are concerned with the relationship between theory and what is observed. According to Woling et al. [228], construct validity threats are related with those issues that might arise during research design. One of the threats is known as the *mono-method bias*, which is related to the use of one single metric in the papers analysis. In our survey, we have mitigated this threat by studying the primary studies from several dimensions and drawing conclusions for each one. As explained in Section 3.1, such dimensions correspond to the *who*, *which*, *how*, *where*, *when* and *what* of all primary studies. Another type of threat is known as *confounding constructs* as well as *levels of constructs* [228]. In our study, this threat has to do with the fact that a particular proposal could be categorized in more than one specific dimension. In fact, a specific paper may deal with more than one type of uncertainty (cf. Sect. 4). To mitigate this threat, we have considered all the types of uncertainty that are addressed in all primary studies when drawing figures and conclusions, and not just the most representative one as is being done in some other mapping studies [229].

10.3.3 Internal Validity

These threats are related to the factors that could affect the results of our evaluation. In the case of internal validity, they also influence the process of selecting the papers. According to Wohlin et al. [228], we should consider *publication selection* and *instrumentation.*

The main factors influencing the publication selection process are keywords, digital libraries, language of publication and time frame. In terms of instrumentation, it is mainly related to the venues considered by the digital libraries used. Our selection process is explained in Sect. 3.2. Our goal was to mitigate the internal validity threats by avoiding overly restrictive decisions. For example, the keywords used are sufficiently generic, which is supported by the fact that we started with an initial set of 3,131 documents after removing duplicates (see Fig. 1). In addition, we use seven different digital libraries, as shown in Table 1, which minimizes the risk of having left out many important papers. Furthermore, we trust that these seven digital libraries include all venues where papers on the subject of uncertainty representation in software models are published, since they all return papers in the field of software engineering. Of course, some papers may have been overlooked because they did not explicitly mention any of the selected keywords (uncertainty, vagueness, imprecision, etc.). Thus, before submitting the final version, we sent the paper to those whom we cited, to check our comments for accuracy and possible omissions. This also provided one final stage in the systematic trawling of the literature for relevant work. We aimed for completeness by trying to include all the papers on the subject of the study, as explained above, but no survey of extensive literature can ever claim to be complete.

10.3.4 External Validity

These threats are related to the extent to which it is possible to generalize the findings and conclusions of this study. Since this is a survey of a specific topic, we do not intend to make any generalizations. Our study refers to published papers on the explicit representation of uncertainty in software models, so it cannot be generalized to any closely related field of research.

11 Conclusions

We have provided a comprehensive overview and analysis of the research work on uncertainty representation in software models. The survey presented the current research state of the existing proposals (up to January 2020). It also summarized the definitions, notations and formalisms used to represent uncertainty in software models, as well as the application domains in which existing proposals are used, and how they are applied. Finally, research trends, opportunities and challenges in representing uncertainty in software models were also discussed.

The results of our study show that we are at a stage where the research community should move from an initial level of maturity, focused purely on scientific and exploratory issues, to a more pragmatic level where we can reflect on what we have learned so far to improve current industrial modeling practices. In this respect, we advocate the development of agreed guidelines, methods and tools for the representation of uncertainty in software models that can help software engineers to represent and analyse their systems in a more predictable way, with consistent results and supported by useful tools. After twenty years of research on this topic, we are at a time when the community should analyse current achievements and existing proposals, and try to synthesise notations, guidelines and tools for the industrial representation of uncertainty in software models.

We hope that this survey can help software engineering researchers in that endeavor, and that it will help more researchers contribute to the difficult problem of modeling complex systems in a more faithful and accurate manner.

Acknowledgements We would like to thank the reviewers of the paper for their insightful comments and very valuable suggestions, which helped us significantly to improve this work. Many thanks also to the authors of the cited papers and to other members of the community who kindly provided comments and feedback on earlier drafts of this paper. This work is partially supported by the European Commission (FEDER) and the Spanish Government under projects APOLO (US-1264651), HORATIO (RTI2018-101204-B-C21), EKIPMENT-PLUS (P18-FR-2895) and COSCA (PGC2018-094905-B-I00).

Primary Studies

1. Agli, H., Bonnard, P., Gonzales, C., Wuillemin, P.: Business rules uncertainty management with probabilistic relational models. In: Proc. of RuleML'16, *LNCS*, vol. 9718, pp. 53–67. Springer (2016). DOI 10.1007/978-3-319-42019-6_4
2. Ali, S., Basit-Ur-Rahim, M.A., Arif, F.: Formal Verification of Time Constrains SysML Internal Block Diagram Using PRISM. In: Proc. of ICCSA'15, pp. 62–66. IEEE Computer Society, USA (2015). DOI 10.1109/ICCSA.2015.11
3. Bagheri, E., Ghorbani, A.A.: Experiences on the Belief-Theoretic Integration of Para-consistent Conceptual Models. In: Proc. of ASWEC'08, pp. 357–366. IEEE (2008). DOI 10.1109/ASWEC.2008.4483224
4. Bagheri, E., Ghorbani, A.A.: A belief-theoretic framework for the collaborative development and integration of para-consistent conceptual models. J. Syst. Softw. **82**(4), 707–729 (2009). DOI 10.1016/j.jss.2008.10.012
5. Baresi, L., Pasquale, L., Spoletini, P.: Fuzzy goals for requirements-driven adaptation. In: 2010 18th IEEE International Requirements Engineering Conference. IEEE (2010). DOI 10.1109/re.2010.25
6. Bertoa, M.F., Burgueño, L., Moreno, N., Vallecillo, A.: Incorporating measurement uncertainty into OCL/UML primitive datatypes. Software and Systems Modeling (2019). DOI 10.1007/s10270-019-00741-0
7. Bertoa, M.F., Moreno, N., Barquero, G., Burgueño, L., Troya, J., Vallecillo, A.: Expressing measurement uncertainty in OCL/UML datatypes. In: Proc. ECMFA'18, *LNCS*, vol. 10890, pp. 46–62. Springer (2018). DOI 10.1007/978-3-319-92997-2_4
8. Blanco, I.J., Marin, N., Pons, O., Vila, M.A.: Softening the object-oriented database model: imprecision, uncertainty, and fuzzy types. In: Proc. of NAFIPS'01, vol. 4, pp. 2323–2328 (2001). DOI 10.1109/NAFIPS.2001.944435
9. Brambilla, M., Eramo, R., Pierantonio, A., Rosa, G., Umuhoza, E.: Enhancing flexibility in user interaction modeling by adding design uncertainty to IFML. In: Proc. of FLEXMDE@MODELS'17, *CEUR Workshop Proceedings*, vol. 2019, pp. 435–440. CEUR-WS.org (2017). URL `http://ceur-ws.org/Vol-2019/flexmde_9.pdf`
10. Bucaioni, A., Cicchetti, A., Ciccozzi, F., Mubeen, S., Pierantonio, A., Sjödin, M.: Handling Uncertainty in Automatically Generated Implementation Models in the Automotive Domain. In: Proc. of SEAA'16, pp. 173–180 (2016). DOI 10.1109/SEAA.2016.32
11. Burgueño, L., Bertoa, M.F., Moreno, N., Vallecillo, A.: Expressing confidence in models and in model transformation elements. In: Proc. of MODELS'18, pp. 57–66. ACM (2018). DOI 10.1145/3239372.3239394
12. Burgueño, L., Clarisó, R., Cabot, J., Gérard, S., Vallecillo, A.: Belief uncertainty in software models. In: Proc. of MiSE@ICSE'19, pp. 19–26. IEEE (2019). DOI 10.1109/MiSE.2019.00011
13. Burgueño, L., Mayerhofer, T., Wimmer, M., Vallecillo, A.: Using physical quantities in robot software models. In: Proc. of RoSE@MODELS'18, pp. 23–28. ACM (2018). DOI 10.1145/3196558.3196562
14. Burgueño, L., Mayerhofer, T., Wimmer, M., Vallecillo, A.: Specifying quantities in software models. Information and Software Technology **113**, 82–97 (2019). DOI 10.1016/j.infsof.2019.05.006
15. Cámara, J., Peng, W., Garlan, D., Schmerl, B.R.: Reasoning about sensing uncertainty and its reduction in

decision-making for self-adaptation. Sci. Comput. Program. **167**, 51–69 (2018). DOI 10.1016/j.scico.2018.07.002

16. Chang, E.J., Hussain, F.K., Dillon, T.S.: Fuzzy Nature of Trust and Dynamic Trust Modeling in Service Oriented Environments. In: Proc. of SWS@CCS'05, pp. 75–83. ACM (2005). DOI 10.1145/1103022.1103036
17. Chechik, M., Kokaly, S., Rahimi, M., Salay, R., Viger, T.: Uncertainty, modeling and safety assurance: Towards a unified framework. In: Proc. of VSTTE'19, *LNCS*, vol. 12031, pp. 19–29. Springer (2019). DOI 10.1007/978-3-030-41600-3_2
18. Chen, X., Cheng, H., Wang, H., Li, W.: Fuzzy spatiotemporal object modeling based on UML class diagram. Journal of Intelligent and Fuzzy Systems **33**(5), 2727–2736 (2017). DOI 10.3233/JIFS-169322
19. Chen, X., Yan, L., Li, W., Ma, Z.: Reengineering fuzzy spatiotemporal UML data model into fuzzy spatiotemporal XML model. IEEE Access **5**, 17,975–17,987 (2017). DOI 10.1109/ACCESS.2017.2745540
20. Cheng, B.H.C., Sawyer, P., Bencomo, N., Whittle, J.: A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In: Proc. of MODELS'09, *LNCS*, vol. 5795, pp. 468–483. Springer (2009). DOI 10.1007/978-3-642-04425-0_36
21. Cheng, S.W., Garlan, D.: Handling uncertainty in autonomic systems. In: In Proc. of IWLU@ASE'07. ACM (2007). URL `http://se.cs.toronto.edu/IWLU/papers/Autonomic_Cheng.pdf`
22. Cheung, L., Golubchik, L., Medvidovic, N., Sukhatme, G.S.: Identifying and Addressing Uncertainty in Architecture-Level Software Reliability Modeling. In: Proc. of IPDPS'07, pp. 1–6. IEEE (2007). DOI 10.1109/IPDPS.2007.370524
23. Crossland, R., Williams, J.H.S., McMahon, C.A.: An object-oriented modeling framework for representing uncertainty in early variant design. Research in Engineering Design **14**(3), 173–183 (2003). DOI 10.1007/s00163-003-0039-z
24. D'Emilia, G., Paolone, G., Natale, E., Gaspari, A., Villano, D.D.: Business modeling of a measurement-based context: A methodological process. In: Proc. of ICSOFT-EA'15', vol. 1, pp. 1–8. SciTePress (2015). DOI 10.5220/0005499402690276
25. Diskin, Z., Eramo, R., Pierantonio, A., Czarnecki, K.: Incorporating uncertainty into bidirectional model transformations and their delta-lens formalization. In: Proc. of BX@ETAPS'16, *CEUR Workshop Proceedings*, vol. 1571, pp. 15–31. CEUR-WS.org (2016). URL `http://ceur-ws.org/Vol-1571/paper_9.pdf`
26. Dong, Q., Wang, Z., Zhu, W., He, H.: Capability requirements modeling and verification based on fuzzy ontology. Journal of Systems Engineering and Electronics **23**(1), 78–87 (2012). DOI 10.1109/JSEE.2012.00011
27. Eramo, R., Pierantonio, A., Rosa, G.: Managing uncertainty in bidirectional model transformations. In: Proc. of SLE'15, pp. 49–58. ACM (2015). DOI 10.1145/2814251.2814259
28. Eramo, R., Pierantonio, A., Rosa, G.: Approaching collaborative modeling as an uncertainty reduction process. In: Proc. of COMMitMDE@MODELS'16, *CEUR Workshop Proceedings*, vol. 1717, pp. 27–34. CEUR-WS.org (2016). URL `http://ceur-ws.org/Vol-1717/paper7.pdf`
29. Esfahani, N., Elkhodary, A.M., Malek, S.: A learning-based framework for engineering feature-oriented self-adaptive software systems. IEEE Trans. Software Eng. **39**(11), 1467–1493 (2013). DOI 10.1109/TSE.2013.37
30. Esfahani, N., Malek, S., Razavi, K.: Guidearch: guiding the exploration of architectural solution space under uncertainty. In: Proc. of ICSE'13, pp. 43–52. IEEE Computer Society (2013). DOI 10.1109/ICSE.2013.6606550
31. Famelis, M., Chechik, M.: Managing design-time uncertainty. Software and Systems Modeling **18**(2), 1249–1284 (2019). DOI 10.1007/s10270-017-0594-9
32. Famelis, M., Rubin, J., Czarnecki, K., Salay, R., Chechik, M.: Software product lines with design choices: Reasoning about variability and design uncertainty. In: Proc. of MODELS'17. IEEE Computer Society (2017). DOI 10.1109/models.2017.3
33. Famelis, M., Salay, R., Chechik, M.: Partial models: Towards modeling and reasoning with uncertainty. In: Proc. of ICSE'12, pp. 573–583. IEEE Press (2012). DOI 10.1109/ICSE.2012.6227159
34. Famelis, M., Santosa, S.: MAV-Vis: A notation for model uncertainty. In: Proc. of MiSE@ICSE'13, pp. 7–12. IEEE (2013). DOI 10.1109/MiSE.2013.6595289
35. García-Valls, M., Perez-Palacin, D., Mirandola, R.: Pragmatic cyber physical systems design based on parametric models. J. Syst. Softw. **144**, 559–572 (2018). DOI 10.1016/j.jss.2018.06.044
36. Garousi, V.: Traffic-aware stress testing of distributed real-time systems based on UML models in the presence of time uncertainty. In: Proc. of ICST'08, pp. 92–101. IEEE Computer Society (2008). DOI 10.1109/ICST.2008.7
37. Garredu, S., Bisgambiglia, P.A., Vittori, E., Santucci, J.F.: A New Approach to Describe DEVS Models Using Both UML State Machine Diagrams and Fuzzy Logic. In: Proc. of EMSS'10, pp. 215–221. SCS (2010)
38. Geng, S., Peng, J., Li, P.: Modeling and verification of cyber-physical systems under uncertainty. In: Proc. of ICNC-FSKD'17, pp. 1491–1496 (2017). DOI 10.1109/FSKD.2017.8392986
39. Giese, H., Bencomo, N., Pasquale, L., Ramirez, A.J., Inverardi, P., Wätzoldt, S., Clarke, S.: Living with Uncertainty in the Age of Runtime Models. In: Models@run.time, *LNCS*, vol. 8378, pp. 47–100. Springer (2014). DOI 10.1007/978-3-319-08915-7_3
40. Gogolla, M., Vallecillo, A.: On softening OCL invariants. Journal of Object Technology **18**(2), 6:1–22 (2019). DOI 10.5381/jot.2019.18.2.a6
41. Gonzalez-Perez, C.: Modelling temporality and subjectivity in conml: Short paper. In: Proc. of RCIS'13, pp. 1–6. IEEE (2013). DOI 10.1109/RCIS.2013.6577685
42. Hacks, S., Lichter, H.: A Probabilistic Enterprise Architecture Model Evolution. In: Proc. of EDOC'18', pp. 51–57. IEEE Computer Society (2018). DOI 10.1109/EDOC.2018.00017
43. Hall, B.D.: Component interfaces that support measurement uncertainty. Computer Standards & Interfaces **28**(3), 306–310 (2006). DOI 10.1016/j.csi.2005.07.009
44. Han, D., Yang, Q., Xing, J.: Extending UML for the modeling of fuzzy self-adaptive software systems. In: Proc. of CCDC'14, pp. 2400–2406. IEEE (2014). DOI 10.1109/CCDC.2014.6852575
45. Han, D., Yang, Q., Xing, J., Li, J., Wang, H.: FAME: A UML-based framework for modeling fuzzy self-adaptive software. Information and Software Technology **76**(C), 118–134 (2016). DOI 10.1016/j.infsof.2016.04.014
46. Haroonabadi, A., Teshnehlab, M., Movaghar, A.: A novel method for modeling and evaluation of uncertain information systems. In: Proc. of ICIT'08, pp. 238–243. IEEE Computer Society (2008). DOI 10.1109/ICIT.2008.24

47. Jiménez-Ramírez, A., Weber, B., Barba, I., del Valle, C.: Generating optimized configurable business process models in scenarios subject to uncertainty. Information and Software Technology **57**, 571–594 (2015). DOI 10.1016/j.infsof.2014.06.006
48. Johnson, P., Iacob, M.E., Välja, M., Sinderen, M., Magnusson, C., Ladhe, T.: A Method for Predicting the Probability of Business Network Profitability. Inf. Syst. E-Bus. Manag. **12**(4), 567–593 (2014). DOI 10.1007/s10257-014-0237-4
49. Johnson, P., Ullberg, J., Buschle, M., Franke, U., Shahzad, K.: An architecture modeling framework for probabilistic prediction. Inf. Syst. E-Business Management **12**(4), 595–622 (2014). DOI 10.1007/s10257-014-0241-8
50. Khalfi, B., de Runz, C., Faiz, S., Akdag, H.: Modélisation conceptuelle d'objets géographiques imprécis et multiples : Une approche basée F-Perceptory. In: Proc. of SAGEO'15, *CEUR Workshop Proceedings*, vol. 1535, pp. 297–311. CEUR-WS.org (2015). URL `http://ceur-ws.org/Vol-1535/paper-21.pdf`
51. Koehler, H., Link, S., Prade, H., Zhou, X.: Cardinality Constraints for Uncertain Data. In: Proc. of ER'14, *LNCS*, vol. 8824, pp. 108–121. Springer (2014). DOI 10.1007/978-3-319-12206-9_9
52. Laghouaouta, Y., Laforcade, P.: Dealing with uncertainty in model transformations. In: Proc. of SAC'20, pp. 1595—-1603. ACM (2020). DOI 10.1145/3341105.3373971
53. López-Landa, R., Noguez, J.: PRoModel: a model-driven software environment that facilitates and expedites the development of systems that handle uncertainty. In: Proc. of SpringSim'12, p. 41. SCS/ACM (2012)
54. Ma, T., Ali, S., Yue, T., Elaasar, M.: Testing self-healing cyber-physical systems under uncertainty: a fragility-oriented approach. Software Qual J **27**(2), 615–649 (2019). DOI 10.1007/s11219-018-9437-3
55. Ma, Z.: The fuzzy ER/EER and UML data models, pp. 59–77 (2006). DOI 10.1007/11353270_4
56. Ma, Z., Zhang, F., Yan, L., Cheng, J.: Fuzzy Description Logic and Ontology Extraction from Fuzzy Data Models. In: Fuzzy Knowledge Management for the Semantic Web, vol. 306, pp. 99–156. Springer (2014). DOI 10.1007/978-3-642-39283-2_5
57. Ma, Z.M.: Modeling Fuzzy Information in the EER and Nested Relational Database Models, pp. 123–146. Springer (2006). DOI 10.1007/3-540-33289-8_5
58. Ma, Z.M., Yan, L.: Fuzzy XML data modeling with the UML and relational data models. Data Knowl. Eng. **63**(3), 972–996 (2007). DOI 10.1016/j.datak.2007.06.003
59. Ma, Z.M., Yan, L., Zhang, F.: Modeling Fuzzy Information in UML Class Diagrams and Object-Oriented Database Models. Fuzzy Sets Syst. **186**(1), 26–46 (2012). DOI 10.1016/j.fss.2011.06.015
60. Ma, Z.M., Zhang, F., Yan, L.: Fuzzy Information Modeling in UML Class Diagram and Relational Database Models. Appl. Soft Comput. **11**(6), 4236–4245 (2011). DOI 10.1016/j.asoc.2011.03.020
61. Ma, Z.M., Zhang, F., Yan, L., Cheng, J.: Representing and reasoning on fuzzy UML models: A description logic approach. Expert Syst. Appl. **38**(3), 2536–2549 (2011). DOI 10.1016/j.eswa.2010.08.042
62. Martín-Rodilla, P., Gonzalez-Perez, C.: Representing imprecise and uncertain knowledge in digital humanities: A theoretical framework and conml implementation with a real case study. In: Proc. of TEEM'18, pp. 863–871. ACM (2018). DOI 10.1145/3284179.3284318
63. Martin-Rodilla, P., Gonzalez-Perez, C.: Conceptualization and non-relational implementation of ontological and epistemic vagueness of information in digital humanities. Informatics **6**(2) (2019). DOI 10.3390/informatics6020020
64. Martín-Rodilla, P., Gonzalez-Perez, M.P.F.C.: Qualifying and quantifying uncertainty in digital humanities: A fuzzy-logic approach. In: Proc. of TEEM'19, pp. 788–794. ACM (2019). DOI 10.1145/3362789.3362833
65. Mayerhofer, T., Wimmer, M., Vallecillo, A.: Adding uncertainty and units to quantity types in software models. In: Proc. of SLE'16, pp. 118–131. ACM (2016). DOI 10.1145/2997364.2997376
66. McKeever, S., Ye, J., Coyle, L., Dobson, S.: A context quality model to support transparent reasoning with uncertain context. In: Proc. of QuaCon'09, *LNCS*, vol. 5786, pp. 65–75. Springer (2009). DOI 10.1007/978-3-642-04559-2_6
67. Menghi, C., Spoletini, P., Chechik, M., Ghezzi, C.: A verification-driven framework for iterative design of controllers. Formal Aspects Comput. **31**(5), 459–502 (2019). DOI 10.1007/s00165-019-00484-1
68. Motameni, H., Ghassempouri, T., Nematzadeh, H.: Evaluating the reliability of Communication diagram using Fuzzy Petri net. In: Proc. of ICSEES'11, pp. 520–523. IEEE (2011). DOI 10.1109/ICSESS.2011.5982367
69. Nasiri, R., Moeini, A., Abdollahzadeh, A.: A New Approach Towards Procurement of Software Models Via Distributed Business Models. J. Supercomput. **29**(3), 287–302 (2004). DOI 10.1023/B:SUPE.0000032782.92290.52
70. Object Management Group: Structured Metrics Metamodel (SMM) Specification. Version 1.2 (2018). OMG Document formal/18-05-01
71. Object Management Group: OMG Systems Modeling Language (SysML), version 1.6 (2019). OMG Document formal/19-11-01
72. Object Management Group: UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. Version 1.2 (2019). OMG Document formal/19-04-01
73. Oquendo, F.: Coping with uncertainty in systems-of-systems architecture modeling on the IoT with SosADL. In: Proc. of SoSE'19, pp. 131–136 (2019). DOI 10.1109/SYSOSE.2019.8753842
74. Ortiz, V., Burgueño, L., Vallecillo, A., Gogolla, M.: Native support for UML and OCL primitive datatypes enriched with uncertainty in USE. In: Proc. of OCL@MODELS'19, *CEUR Workshop Proceedings*, vol. 2513, pp. 59–66. CEUR-WS.org (2019). URL `http://ceur-ws.org/Vol-2513/paper5.pdf`
75. Ozgur, N.B., Koyuncu, M., Yazici, A.: An Intelligent Fuzzy Object-Oriented Database Framework for Video Database Applications. Fuzzy Sets Syst. **160**(15), 2253–2274 (2009). DOI 10.1016/j.fss.2009.02.017
76. Packevicius, S., Usaniov, A., Bareisa, E.: Software testing using imprecise OCL constraints as oracles. In: Proc. of CompSysTech'07, pp. 1–6. ACM (2007). DOI 10.1145/1330598.1330726
77. Refsdal, A., Runde, R.K., Stølen, K.: Underspecification, inherent nondeterminism and probability in sequence diagrams. In: Proc. of FMOODS'06, *LNCS*, vol. 4037, pp. 138–155. Springer (2006). DOI 10.1007/11768869_12
78. Refsdal, A., Runde, R.K., Stølen, K.: Stepwise refinement of sequence diagrams with soft real-time constraints. J. Comput. Syst. Sci. **81**(7), 1221–1251 (2015). DOI 10.1016/j.jcss.2015.03.003
79. Refsdal, A., Solhaug, B., Stolen, K.: A UML-based method for the development of policies to support trust

management. In: TRUST MANAGEMENT II, vol. 263, pp. 33+. IFIP (2008). DOI 10.1007/978-0-387-09428-1_3
80. Refsdal, A., Stølen, K.: Extending UML sequence diagrams to model trust-dependent behavior with the aim to support risk analysis. Sci. Comput. Program. **74**(1–2), 34–42 (2008). DOI 10.1016/j.scico.2008.09.003
81. Robak, S., Pieczynski, A.: Employing Fuzzy Logic in Feature Diagrams to Model Variability in Software Product-Lines. In: proc. of ECBS'03, pp. 305–311. IEEE Computer Society (2003). DOI 10.1109/ECBS.2003.1194812
82. Runde, R.K., Refsdal, A., Stølen, K.: Relating computer systems to sequence diagrams: the impact of underspecification and inherent nondeterminism. Formal Asp. Comput. **25**(2), 159–187 (2013). DOI 10.1007/s00165-011-0192-5
83. Salay, R., Chechik, M.: A generalized formal framework for partial modeling. In: Proc. of FASE'15, *LNCS*, vol. 9033, pp. 133–148. Springer (2015). DOI 10.1007/978-3-662-46675-9_9
84. Salay, R., Chechik, M., Famelis, M., Gorzny, J.: A methodology for verifying refinements of partial models. J. Object Technol. **14**(3), 3:1–31 (2015). DOI 10.5381/jot.2015.14.3.a3
85. Salay, R., Chechik, M., Horkoff, J., Sandro, A.D.: Managing requirements uncertainty with partial models. Requirements Eng **18**(2), 107–128 (2013). DOI 10.1007/s00766-013-0170-y
86. Salay, R., Gorzny, J., Chechik, M.: Change Propagation due to Uncertainty Change. In: Proc. of FASE'13, *LNCS*, vol. 7793, pp. 21–36. Springer (2013). DOI 10.1007/978-3-642-37057-1_3
87. Sedaghatbaf, A., Azgomi, M.A.: Reliability evaluation of UML/DAM software architectures under parameter uncertainty. IET Software **12**(3), 236–244 (2018). DOI 10.1049/iet-sen.2017.0077
88. Sheng, J., Yan, L., Ma, Z.: Modeling probabilistic data with fuzzy probability measures in UML class diagrams. In: Proc. of IFSA/NAFIPS'19, *Advances in Intelligent Systems and Computing*, vol. 1000, pp. 589–600. Springer (2019). DOI 10.1007/978-3-030-21920-8_52
89. Shin, S.Y., Chaouch, K., Nejati, S., Sabetzadeh, M., Briand, L.C., Zimmer, F.: HITECS: A UML Profile and Analysis Framework for Hardware-in-the-Loop Testing of Cyber Physical Systems. In: Proc. of MODELS'18, pp. 357–367. ACM (2018). DOI 10.1145/3239372.3239382
90. Sibay, G.E., Braberman, V.A., Uchitel, S., Kramer, J.: Synthesizing modal transition systems from triggered scenarios. IEEE Trans. Software Eng. **39**(7), 975–1001 (2013). DOI 10.1109/TSE.2012.62
91. Sicilia, M.A., Diaz, P., Aedo, I., Garcia, E.: Fuzziness in adaptive hypermedia models. In: Proc. of NAFIPS'02, pp. 268–273 (2002). DOI 10.1109/NAFIPS.2002.1018068
92. Sicilia, M.A., Mastorakis, N.: Extending UML 1.5 for Fuzzy conceptual modeling: An strictly additive approach. WSEAS transactions on systems **3**(5), 2234–2239 (2004)
93. de Soto, A.R., Capdevila, C.A., Fernández, E.C.: Fuzzy Systems and Neural Networks XML Schemas for Soft Computing. Mathware and Soft Computing **10**(2–3), 43–56 (2003)
94. Stephenson, Z.R., Attwood, K., McDermid, J.A.: Product-Line Models to Address Requirements Uncertainty, Volatility and Risk, pp. 111–131. Springer (2011). DOI 10.1007/978-3-642-21001-3_8
95. Thomas, O., Dollmann, T.: Fuzzy-EPC Markup Language: XML Based Interchange Formats for Fuzzy Process Models. In: Soft Computing in XML Data Management, vol. 255, pp. 227–257. Springer (2010). DOI 10.1007/978-3-642-14010-5_9
96. Troegner, D.: Combination of fuzzy sets with the object constraint language (OCL). In: Proc. of Informatik'10, *LNI*, vol. P-176, pp. 705–710 (2010). URL `https://dl.gi.de/20.500.12116/19308`
97. Tseng, C., Khamisy, W., Vu, T.: Universal fuzzy system representation with XML. Computer Standards & Interfaces **28**(2), 218–230 (2005). DOI 10.1016/j.csi.2004.11.005
98. Tudoroiu, R., Astilean, A., Letia, T., Neacsu, G., Maroszy, Z., Tudoroiu, N.: Fuzzy UML and petri nets modeling investigations on the pollution impact on the air quality in the vicinity of the black sea constanta romanian resort. In: Proc. of FedCSIS'11, pp. 763–766 (2011)
99. Turowski, K., Weng, U.: Representing and processing fuzzy information - an xml-based approach. Knowl. Based Syst. **15**(1-2), 67–75 (2002). DOI 10.1016/S0950-7051(01)00122-8
100. Ubayashi, N., Kamei, Y., Sato, R.: Modular Programming and Reasoning for Living with Uncertainty. In: Proc. of ICSOFT'18, vol. 1077, pp. 220–244. Springer (2019). DOI 10.1007/978-3-030-29157-0_10
101. Uchitel, S., Brunet, G., Chechik, M.: Synthesis of partial behavior models from properties and scenarios. IEEE Trans. Software Eng. **35**(3), 384–406 (2009). DOI 10.1109/TSE.2008.107
102. Vallecillo, A., Morcillo, C., Orue, P.: Expressing Measurement Uncertainty in Software Models. In: Proc. of QUATIC'16, pp. 15–24 (2016). DOI 10.1109/QUATIC.2016.013
103. Voudouris, V.: Towards a unifying formalisation of geographic representation: the object–field model with uncertainty and semantics. International Journal of Geographical Information Science **24**(12), 1811–1828 (2010). DOI 10.1080/13658816.2010.488237
104. Voudouris, V., Wood, J., Fisher, P.F.: Collaborative geovisualization: Object-Field Representations with Semantic and Uncertainty Information. In: Proc. of OTM'05 Workshops, *LNCS*, vol. 3762, pp. 1056–1065. Springer (2005). DOI 10.1007/11575863_128
105. Voudouris, V., Wood, J., Fisher, P.F.: Capturing and Representing Conceptualization Uncertainty Interactively using Object-Fields, pp. 755–770. Springer (2006). DOI 10.1007/3-540-35589-8_47
106. Wang, Y., Bai, L.: Fuzzy Spatiotemporal Data Modeling Based on UML. IEEE Access **7**, 45,405–45,416 (2019). DOI 10.1109/ACCESS.2019.2908224
107. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.: RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In: Proc. of RE'09, pp. 79–88. IEEE Computer Society (2009). DOI 10.1109/RE.2009.36
108. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.: RELAX: a language to address uncertainty in self-adaptive systems requirements. Requir. Eng. **15**(2), 177–196 (2010). DOI 10.1007/s00766-010-0101-0
109. Xiao, J., Pinel, P., Pi, L., Aranega, V., Baron, C.: Modeling Uncertain and Imprecise Information in Process Modeling with UML. In: Proc. of ICMD'08, pp. 237–240. Computer Society of India/Allied Publishers (2008)
110. Xu, S., Miao, W., Kunz, T., Wei, T., Chen, M.: Quantitative Analysis of Variation-Aware Internet of Things

Designs Using Statistical Model Checking. In: Proc. of QRS'16, pp. 274–285. IEEE (2016). DOI 10.1109/QRS.2016.39

111. Yan, L., Ma, Z.: A Probabilistic Object-Oriented Database Model with Fuzzy Measures, pp. 23–38. Springer, Berlin, Heidelberg (2013). DOI 10.1007/978-3-642-37509-5_2
112. Yan, L., Ma, Z.: A probabilistic object-oriented database model with fuzzy probability measures and its algebraic operations. J. Intell. Fuzzy Syst. **28**(5), 1969–1984 (2015). DOI 10.3233/IFS-141307
113. Yan, L., Ma, Z.: A formal approach for graphically building fuzzy XML model. International Journal of Intelligent Systems **34**(11), 3058–3076 (2019). DOI 10.1002/int.22188
114. Yan, L., Ma, Z.M.: Extending Engineering Data Model for Web-Based Fuzzy Information Modeling. Integr. Comput.-Aided Eng. **20**(4), 407–420 (2013). DOI 10.3233/ICA-130440
115. Yang, Z., Jin, Z., Li, Z.: Modeling Uncertainty and Evolving Self-Adaptive Software: A Fuzzy Theory Based Requirements Engineering Approach. CoRR **abs/1704.00873** (2017)
116. Yazici, A., Zhu, Q., Sun, N.: Semantic data modeling of spatiotemporal database applications. Int. J. Intell. Syst. **16**(7), 881–904 (2001). DOI 10.1002/int.1040
117. Zhang, F., Cheng, J.: Verification of fuzzy UML models with fuzzy description logic. Appl. Soft Comput. **73**, 134–152 (2018). DOI 10.1016/j.asoc.2018.08.025
118. Zhang, F., Ma, Z.M.: Construction of fuzzy ontologies from fuzzy UML models. Int. J. Comput. Intell. Syst. **6**(3), 442–472 (2013). DOI 10.1080/18756891.2013.780735
119. Zhang, M., Ali, S., Yue, T., Norgre, R.: Uncertainty-Wise Evolution of Test Ready Models. Information and Software Technology **87**, 140–159 (2017). DOI 10.1016/j.infsof.2017.03.003
120. Zhang, M., Ali, S., Yue, T., Norgren, R., Okariz, O.: Uncertainty-Wise Cyber-Physical System Test Modeling. Software and System Modeling **18**(2), 1379–1418 (2019). DOI 10.1007/s10270-017-0609-6
121. Zhang, M., Selic, B., Ali, S., Yue, T., Okariz, O., Norgren, R.: Understanding uncertainty in cyber-physical systems: A conceptual model. In: Proc. of ECMFA'16, *LNCS*, vol. 9764, pp. 247–264. Springer (2016). DOI 10.1007/978-3-319-42061-5_16
122. Zhang, M., Yue, T., Ali, S., Selic, B., Okariz, O., Norgren, R., Intxausti, K.: Specifying uncertainty in use case models. Journal of Systems and Software **144**, 573–603 (2018). DOI 10.1016/j.jss.2018.06.075
123. Zhou, B., Lu, J., Wang, Z., Zhang, Y., Miao, Z.: Formalizing Fuzzy UML Class Diagrams with Fuzzy Description Logics. In: Proc. of IITA'09, vol. 1, pp. 171–174 (2009). DOI 10.1109/IITA.2009.97

References

124. Albers, A., Zingel, C.: Extending SysML for engineering designers by integration of the contact & channel - approach (C&C^2-A) for function-based modeling of technical systems. Procedia Computer Science **16**, 353–362 (2013). DOI 10.1016/j.procs.2013.01.037
125. Alevizos, E., Skarlatidis, A., Artikis, A., Paliouras, G.: Probabilistic complex event recognition: A survey. ACM Comput. Surv. **50**(5), 71:1–71:31 (2017). DOI 10.1145/3117809
126. Association for Computing Machinery: ACM Computing Classification System (2012). URL `https://dl.acm.org/ccs`
127. Balsamo, S., Marco, A.D., Inverardi, P., Simeoni, M.: Model-based performance prediction in software development: A survey. IEEE Trans. Software Eng. **30**(5), 295–310 (2004). DOI 10.1109/TSE.2004.9
128. Bernardi, S., Merseguer, J., Petriu, D.C.: Dependability modeling and analysis of software systems specified with UML. ACM Comput. Surv. **45**(1), 2:1–2:48 (2012). DOI 10.1145/2379776.2379778
129. Bernardi, S., Merseguer, J., Petriu, D.C.: Model-Driven Dependability Assessment of Software Systems. Springer (2013)
130. Blair, G.S., Bencomo, N., France, R.B.: Models@run.time. IEEE Computer **42**(10), 22–27 (2009). DOI 10.1109/MC.2009.326
131. Bosc, P., Pivert, O.: Modeling and querying uncertain relational databases: a survey of approaches based on the possible worlds semantics. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **18**(5), 565–603 (2010). DOI 10.1142/S0218488510006702
132. Bruns, G., Godefroid, P.: Model checking partial state spaces with 3-valued temporal logics. In: Proc. of CAV'99, *LNCS*, vol. 1633, pp. 274–287. Springer (1999). DOI 10.1007/3-540-48683-6_25
133. Bucchiarone, A., Cabot, J., Paige, R.F., Pierantonio, A.: Grand challenges in model-driven engineering: an analysis of the state of the research. Software and Systems Modeling **19**(1), 5–13 (2020). DOI 10.1007/s10270-019-00773-6
134. Büttner, F., Gogolla, M.: On OCL-based imperative languages. Sci. Comput. Program. **92**, 162–178 (2014). DOI 10.1016/j.scico.2013.10.003
135. Cámara, J., Garlan, D., Kang, W.G., Peng, W., Schmerl, B.R.: Uncertainty in self-adaptive systems: Categories , management , and perspectives. Tech. Rep. CMU-ISR-17-110, Carnegie Mellon University (2017). URL `http://reports-archive.adm.cs.cmu.edu/anon/isr2017/CMU-ISR-17-110.pdf`
136. Chen, M.: A BDI agents programming language based fuzzy beliefs. In: Proc. of IHMSC'15, vol. 1, pp. 334–337 (2015). DOI 10.1109/IHMSC.2015.170
137. Ciccozzi, F., Malavolta, I., Selic, B.: Execution of UML models: a systematic review of research and practice. Software and Systems Modeling **18**(3), 2313–2360 (2019). DOI 10.1007/s10270-018-0675-4
138. Console, M., Guagliardo, P., Libkin, L.: Propositional and predicate logics of incomplete information. In: Proc. of KR'18, pp. 592–601. AAAI Press (2018)
139. Cortellessa, V., Marco, A.D., Inverardi, P.: Model-Based Software Performance Analysis. Springer (2011). DOI 10.1007/978-3-642-13621-4
140. Cova, T.J., Goodchild, M.F.: Extending geographical representation to include fields of spatial objects. International Journal of Geographical Information Science **16**(6), 509–532 (2002). DOI 10.1080/13658810210137040
141. Dajsuren, Y., van den Brand, M. (eds.): Automotive Systems and Software Engineering – State of the Art and Future Trends. Springer (2019). DOI 10.1007/978-3-030-12157-0
142. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press (2009)
143. Dugan, J.B., Bavuso, S.J., Boyd, M.A.: Dynamic fault-tree models for fault-tolerant computer systems. IEEE Transactions on Reliability **41**(3), 363–377 (1992). DOI 10.1109/24.159800

144. Esfahani, N., Malek, S.: Uncertainty in self-adaptive software systems. In: R. de Lemos, et al. (eds.) Software Engineering for Self-Adaptive Systems II, *LNCS*, vol. 7475, pp. 214–238. Springer (2013)
145. Feller, W.: An Introduction to Probability Theory and Its Applications. Wiley (2008)
146. de Finetti, B.: Theory of Probability: A critical introductory treatment. John Wiley & Sons (2017). DOI 10.1002/9781119286387
147. Garlan, D.: Software Engineering in an Uncertain World. In: Proc. of the FoSER Workshop at FSE/SDP 2010, pp. 125–128. ACM (2010). DOI 10.1145/1882362.1882389
148. Gnesi, S., Latella, D., Massink, M.: A stochastic extension of a behavioural subset of UML statechart diagrams. In: Proc. of HASE'00, pp. 55–64. IEEE Computer Society (2000). DOI 10.1109/HASE.2000.895442
149. Gogolla, M., Büttner, F., Richters, M.: USE: A UML-based specification environment for validating UML and OCL. Sci. Comput. Program. **69**(1-3), 27–34 (2007). DOI 10.1016/j.scico.2007.01.013
150. González, C.A., Cabot, J.: Formal verification of static software models in MDE: A systematic review. Information and Software Technology **56**(8), 821–838 (2014). DOI 10.1016/j.infsof.2014.03.003
151. Greengard, S.: The Internet of Things. MIT Press, Cambridge, MA (2015)
152. Hall, B.D.: GTC: The GUM Tree Calculator (2020). URL `https://github.com/MSLNZ/GTC`
153. Hall, B.D., White, D.R.: An Introduction to Measurement Uncertainty. Measurement Standards Laboratory of New Zealand (2018). DOI 10.5281/zenodo.3872590
154. Hanss, M.: Applied Fuzzy Arithmetic, An Introduction with Engineering Applications. Springer (2005)
155. Harel, D., Marelly, R.: Come, Let's Play, Scenario-Based Programming Using LSCs and the Play-Engine. Springer (2003). DOI 10.1007/978-3-642-19029-2
156. Hariri, R.H., Fredericks, E.M., Bowers, K.M.: Uncertainty in big data analytics: survey, opportunities, and challenges. J. Big Data **6**, 44 (2019). DOI 10.1186/s40537-019-0206-3
157. Hermanns, H., Herzog, U., Katoen, J.: Process algebra for performance evaluation. Theor. Comput. Sci. **274**(1-2), 43–87 (2002). DOI 10.1016/S0304-3975(00)00305-4. URL `https://doi.org/10.1016/S0304-3975(00)00305-4`
158. Howard, R.A., Matheson, J.E.: Influence diagrams. Decision Analysis **2**(3), 127–143 (2005). DOI 10.1287/deca.1050.0020
159. Humphrey, W.S.: Characterizing the software process: a maturity framework. IEEE Software **5**(2), 73–79 (1998). DOI 10.1109/52.2014
160. Immonen, A., Niemelä, E.: Survey of reliability and availability prediction methods from the viewpoint of software architecture. Software and Systems Modeling **7**(1), 49–65 (2008). DOI 10.1007/s10270-006-0040-x
161. Islam, F., Petriu, D.C., Woodside, C.M.: Simplifying layered queuing network models. In: Proc. of EPEW'15, *Lecture Notes in Computer Science*, vol. 9272, pp. 65–79. Springer (2015). DOI 10.1007/978-3-319-23267-6_5
162. ISO 9000:2015: Quality management systems – Fundamentals and vocabulary (2015). URL `https://www.iso.org/obp/ui/#iso:std:iso:9000:ed-4:v1:en`
163. JCGM 100:2008: Evaluation of measurement data—Guide to the expression of uncertainty in measurement (GUM). Joint Com. for Guides in Metrology (2008). URL `http://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf`
164. Jøsang, A.: Subjective Logic – A Formalism for Reasoning Under Uncertainty. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer (2016). DOI 10.1007/978-3-319-42337-1
165. Karkhanis, P., van den Brand, M.G.J., Rajkarnikar, S.: Defining the C-ITS reference architecture. In: ICSA'18 Companion, pp. 148–151. IEEE Computer Society (2018). DOI 10.1109/ICSA-C.2018.00044
166. Kitchenham, B.: Procedures for performing systematic reviews. Tech. Rep. TR/SE-0401, Keele University (2004). URL `http://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf`
167. Kiureghian, A.D., Ditlevsen, O.: Aleatory or epistemic? does it matter? Structural Safety **31**(2), 105–112 (2009)
168. Koziolek, H.: Performance evaluation of component-based software systems: A survey. Perform. Evaluation **67**(8), 634–658 (2010). DOI 10.1016/j.peva.2009.07.007
169. Kwon, W.T., Park, N.C., Jung, S.H.J., Kim, T.G.: Fuzzy-DEVS Formalism: Concepts, Realization and Applications. In: Proc. of AIS'96, pp. 227—234 (1996)
170. van Lamsweerde, A.: Requirements Engineering – From System Goals to UML Models to Software Specifications. Wiley (2009)
171. Lano, K., Rahimi, S.K., Tehrani, S.Y., Sharbaf, M.: A survey of model transformation design patterns in practice. Journal of Systems and Software **140**, 48–73 (2018). DOI 10.1016/j.jss.2018.03.001
172. Larsen, K.G., Thomsen, B.: A modal process logic. In: Proc. of LICS'88, pp. 203–210. IEEE Computer Society (1988). DOI 10.1109/LICS.1988.5119
173. Lebigot, E.O.: Uncertainties: a Python package for calculations with uncertainties (2017). URL `https://pythonhosted.org/uncertainties/`
174. Lee, E.A., Sirjani, M.: What good are models? In: Proc. of FACS'18, *LNCS*, vol. 11222, pp. 3–31. Springer (2018). DOI 10.1007/978-3-030-02146-7_1
175. Li, L., Wang, H., Li, J., Gao, H.: A survey of uncertain data management. Frontiers Comput. Sci. **14**(1), 162–190 (2020). DOI 10.1007/s11704-017-7063-z
176. Liu, B.: Uncertainty Theory, 5 edn. Springer (2018). URL `http://orsc.edu.cn/liu/ut.pdf`
177. Looney, C.G.: Fuzzy petri nets for rule-based decision-making. IEEE Trans. on Systems, Man and Cybernetics **18**(1), 178–183 (1988). DOI 10.1109/21.87067
178. Ma, Z.M., Yan, L.: A literature overview of fuzzy conceptual data modeling. J. Inf. Sci. Eng. **26**(2), 427–441 (2010)
179. Maccaferri, L.: Using Zotero to convert Springer Link CSV search result to BibTex format (2017). URL `https://www.leniel.net/2017/06/using-zotero-to-convert-springerlink-full-csv-search-result-to-bibtex-format.html`
180. Mahdavi-Hezavehi, S., Avgeriou, P., Weyns, D.: A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements, chap. 3, pp. 45 – 77. Morgan Kaufmann (2017). DOI B978-0-12-802855-1.00003-4
181. Manzano, M.A., de Felipe-Rodríguez, H., Gago-Gómez, L.: DICTOMAGRED: Diccionario de Toponimia Magrebí (2018). URL `https://dictomagred.usal.es/`
182. Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with Generalized Stochastic Petri Nets. John Wiley and Sons (1995)
183. Marsan, M.A., Conte, G., Balbo, G.: A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. ACM Trans. Comput. Syst. **2**(2), 93–122 (1984). DOI 10.1145/190.191

184. ick Moon, S., Lee, K.H., Lee, D.: Fuzzy branching temporal logic. IEEE Trans. on Systems, Man and Cybernetics, Part B-Cybernetics **34**(2), 1045–1055 (2004). DOI 10.1109/TSMCB.2003.819485
185. Moreno, G.A., Cámara, J., Garlan, D., Klein, M.: Uncertainty reduction in self-adaptive systems. In: J. Andersson, D. Weyns (eds.) Proc. of SEAMS@ICSE'18, pp. 51–57. ACM (2018). DOI 10.1145/3194133.3194144
186. Moreno, N., Bertoa, M.F., Burgueño, L., Vallecillo, A.: Managing measurement and occurrence uncertainty in complex event processing systems. IEEE Access **7**, 88,026–88,048 (2019). DOI 10.1109/ACCESS.2019.2923953
187. Mosterman, P.J., Zander, J.: Industry 4.0 as a cyber-physical system study. Software and Systems Modeling **15**(1), 17–29 (2016). DOI 10.1007/s10270-015-0493-x
188. Novák, P.: Probabilistic behavioural state machines. In: Proc. of ProMAS'09, *LNCS*, vol. 5919, pp. 67–81. Springer (2009). DOI 10.1007/978-3-642-14843-9_5
189. Oberkampf, W.L., DeLand, S.M., Rutherford, B.M., Diegert, K.V., Alvin, K.F.: Error and uncertainty in modeling and simulation. Reliability Engineering & System Safety **75**(3), 333–357 (2002). DOI 10.1016/S0951-8320(01)00120-X
190. Object Management Group: UML Profile for Schedulability, Performance & Time. Version 1.1 (2005). OMG Document formal/05-01-02
191. Object Management Group: UML Testing Profile, version 1.2 (2013). OMG Document formal/13-04-03
192. Object Management Group: Interaction Flow Modeling Language (IFML), version 1.0 (2015). OMG Document formal/15-02-05
193. Object Management Group: Precise Semantics for Uncertainty Modeling (PSUM) RFP (2017). OMG Document ad/2017-12-1
194. Object Management Group: Semantics Of A Foundational Subset For Executable UML Models (FUML), version 1.4 (2018). OMG Document formal/18-12-01
195. Othman, N.A., Eldin, A.S., Zanfaly, D.S.E.: Handling uncertainty in database: An introduction and brief survey. Computer and Information Science **8**(3), 119–133 (2015). DOI 10.5539/cis.v8n3p119
196. Pearl, J.: A probabilistic calculus of actions. In: Proc. of the Tenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI'94), pp. 454–462. Morgan Kaufmann (1994)
197. Pearl, J.: Causality: Models, reasoning and inference. Cambridge University Press (2000)
198. Pearl, J., Mackenzie, D.: The book of why: The new science of cause and effect. Basic Books (2018)
199. Perez-Palacin, D., Mirandola, R.: Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation. In: Proc. of ICPE'14, pp. 3–14. ACM (2014). DOI 10.1145/2568088.2568095
200. Pivert, O., Prade, H.: Handling uncertainty in relational databases with possibility theory - A survey of different modelings. In: Proc. of SUM'18, *LNCS*, vol. 11142, pp. 396–404. Springer (2018). DOI 10.1007/978-3-030-00461-3_30
201. Ramirez, A.J., Jensen, A.C., Cheng, B.H.C.: A taxonomy of uncertainty for dynamically adaptive systems. In: Proc. of SEAMS'12, pp. 99–108. IEEE Computer Society (2012). DOI 10.1109/SEAMS.2012.6224396
202. Rausand, M.: Risk Assessment: Theory, Methods, and Applications. John Wiley & Sons (2013)
203. Refsdal, A.: Specifying computer systems with probabilistic sequence diagrams. Ph.D. thesis, University of Oslo, Norway (2008). URL `https://www.duo.uio.no/handle/10852/9873`
204. Rinderknecht, S.L., Borsuk, M.E., Reichert, P.: Bridging uncertain and ambiguous knowledge with imprecise probabilities. Environmental Modelling and Software **36**, 122–130 (2012). DOI 10.1016/j.envsoft.2011.07.022
205. Russell, S.J., Norvig, P.: Artificial Intelligence, A Modern Approach, 3 edn. Prentice Hall (2010)
206. Sadegh-Zadeh, K.: Fuzzy deontics. In: Soft Computing in Humanities and Social Sciences, *Studies in Fuzziness and Soft Computing*, vol. 273, pp. 141–156. Springer (2012). DOI 10.1007/978-3-642-24672-2_7
207. Salih, A.M., Omar, M., Yasin, A.: Understanding uncertainty of software requirements engineering: A systematic literature review protocol. In: Proc. of APRES'17, *Communications in Computer and Information Science*, vol. 809, pp. 164–171. Springer (2017). DOI 10.1007/978-981-10-7796-8_13
208. Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S.: Global Sensitivity Analysis: The Primer. John Wiley & Sons (2008). DOI 10.1002/9780470725184
209. Saoud, Z., Faci, N., Maamar, Z., Benslimane, D.: A fuzzy-based credibility model to assess Web services trust under uncertainty. Journal of Systems and Software **122**, 496–506 (2016). DOI 10.1016/j.jss.2015.09.040
210. Seely, A.J., Macklem, P.T.: Complex systems and the technology of variability analysis. Critical Care **8**, 367–384 (2004). DOI 10.1186/cc2948
211. Selic, B.: Beyond Mere Logic – A Vision of Modeling Languages for the 21st Century. In: Proc. of MODELSWARD 2015 and PECCS 2015, pp. IS–5. SciTePress (2015). URL `http://cescit2015.um.si/Presentations/KN_Selic.pdf`
212. Senaratne, H., Gerharz, L., Pebesma, E., Schwering, A.: Usability of spatio-temporal uncertainty visualisation methods. In: Bridging the Geographic Information Sciences. Proc. of AGILE'12, Lecture Notes in Geoinformation and Cartography, pp. 3–23. Springer (2012). DOI 10.1007/978-3-642-29063-3_1
213. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press (1976)
214. Shokravi, S., Smith, A.J.R., Burvill, C.R.: Industrial environmental performance evaluation: A markov-based model considering data uncertainty. Environ. Model. Softw. **60**, 1–17 (2014). DOI 10.1016/j.envsoft.2014.05.024
215. da Silva Hack, P., ten Caten, C.S.: Measurement uncertainty: Literature review and research trends. IEEE Trans. Instrumentation and Measurement **61**(8), 2116–2124 (2012). DOI 10.1109/TIM.2012.2193694
216. Sinnema, M., Deelstra, S.: Classifying variability modeling techniques. Inf. Softw. Technol. **49**(7), 717–739 (2007). DOI 10.1016/j.infsof.2006.08.001
217. Smith, C.U.: Performance engineering of software systems. Addison-Wesley (1990)
218. Thornton, A., Lee, P.: Publication bias in meta-analysis: its causes and consequences. Journal of Clinical Epidemiology **53**(2), 207–216 (2000). DOI 10.1016/S0895-4356(99)00161-4
219. Thunnissen, D.P.: Uncertainty classification for the design and development of complex systems. In: Proc. of the 3rd Annual Predictive Methods Conference, Veros Software (2003). URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.128.133`
220. Troya, J., Moreno, N., Bertoa, M.F., Vallecillo, A.: Representing uncertainty in software models: A Survey –

Companion website (2020). URL http://atenea.lcc.uma.es/projects/UncertaintySurvey.html

221. Tumeo, M.A.: The meaning of stochasticity, randomness and uncertainty in environmental modeling. In: Stochastic and Statistical Methods in Hydrology and Environmental Engineering: Volume 2, pp. 33–38. Springer (1994). DOI 10.1007/978-94-011-1072-3_3

222. Vanherpen, K., Denil, J., Meulenaere, P.D., Vangheluwe, H.: Design-space exploration in MDE: an initial pattern catalogue. In: Proc. of CMSEBA@MODELS'14, *CEUR Workshop Proceedings*, vol. 1340, pp. 42–51. CEUR-WS.org (2014). URL http://ceur-ws.org/Vol-1340/paper6.pdf

223. Walker, W., Harremoës, P., Rotmans, J., van der Sluijs, J., van Asselt, M., Janssen, P., Krayer von Krauss, M.: Defining Uncertainty: A Conceptual Basis for Uncertainty Management in Model-Based Decision Support. Integrated Assessment **4**(1), 5–17 (2003). DOI 10.1076/iaij.4.1.5.16466

224. Wang, Y., Li, X., Li, X., Wang, Y.: A survey of queries over uncertain data. Knowl. Inf. Syst. **37**(3), 485–530 (2013). DOI 10.1007/s10115-013-0638-6

225. Wang, Y.H., Cao, K., Zhang, X.M.: Complex event processing over distributed probabilistic event streams. Computers & Mathematics with Applications **66**(10), 1808–1821 (2013)

226. Webster, J., Watson, R.T.: Analyzing the past to prepare for the future: Writing a literature review. MIS Quarterly **26**(2), xiii–xxiii (2002)

227. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proc. of EASE'14, pp. 38:1–38:10. ACM (2014). DOI 10.1145/2601248.2601268

228. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer (2012)

229. Wolny, S., Mazak, A., Carpella, C., Geist, V., Wimmer, M.: Thirteen years of SysML: a systematic mapping study. Software and Systems Modeling **19**(1), 111–169 (2020). DOI 10.1007/s10270-019-00735-y

230. Woltzenlogel Paleo, B.: An expressive probabilistic temporal logic. CoRR **abs/1603.07453** (2016). URL http://arxiv.org/abs/1603.07453

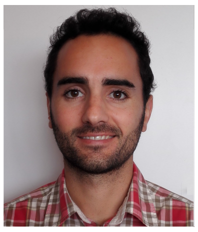
231. Wong, S.K.M., Butz, C.J.: Rough sets for uncertainty reasoning. In: Proc. of RSCTC'00, *LNCS*, vol. 2005, pp. 511–518. Springer (2000). DOI 10.1007/3-540-45554-X_63

232. Wortmann, A., Barais, O., Combemale, B., Wimmer, M.: Modeling languages in industry 4.0: an extended systematic mapping study. Software and Systems Modeling **19**(1), 67–94 (2020)

233. Zimmermann, H.J.: Fuzzy Set Theory – and Its Applications. Springer Science+Business Media (2001)

234. Zvieli, A., Chen, P.P.: Entity-relationship modeling and fuzzy databases. In: Proc. of ICDE'86, pp. 320–327. IEEE Computer Society (1986). DOI 10.1109/ICDE.1986.7266236

Author biographies



Javier Troya is Associate Professor of Software Engineering at the University of Seville, Spain. Before, he was a post-doctoral researcher in the TU Wien, Austria (2013-2015), and obtained his International PhD with honors from the University of Málaga, Spain (2013). His current research interests include Model-based Software Engineering, Software Testing and Software Quality. Contact him at jtroya@us.es, or visit http://www.lsi.us.es/~jtroya/.

Nathalie Moreno received the M.Sc. and Ph.D. degrees in Computer Science from the Department of Computer Science, University of Málaga, Spain, in 2012, where she is currently an Assistant Professor. Her research interests are mainly oriented towards model-driven development. In particular, she focuses on conceptual modeling methodologies, business process modeling, model transformation languages, and uncertainty on complex event processing systems for its application on the Internet of Things.

Manuel F. Bertoa is Assistant Professor at the University of Málaga, Spain. His research interests include software quality, software measurement, and uncertainty modeling. He is a Telecommunications Engineer from the Technical University of Madrid, Spain, and has a Ph.D. in Computer Science from the University of Málaga. He has more than 16 years of experience in international IT companies, Public Administration, and Health Care systems.

Antonio Vallecillo is Professor of Software Engineering at the University of Málaga, Spain, where he leads the Atenea Research Group. His current research interests include Model-based Software Engineering, Open Distributed Processing, and Software Quality. Information about his publications, research projects and activities can be found at http://www.lcc.uma.es/~av.