Specifying Uncertainty in Use Case Models in Industrial Settings

Man Zhang[1], Tao Yue[1,2], Shaukat Ali[1], Bran Selic[1]
[1]Simula Research Laboratory
[2]University of Oslo
{man, tao, shaukat, bselic}@simula.no

Oscar Okariz[3], Roland Norgren[4], Karmele Intxausti[5]
[3]ULMA Handling Systems,[4]Future Position X, [5]Ikerlan
ookariz@manutencion.ulma.es, roland.norgren@fpx.se,
KIntxausti@ikerlan.es

Abstract— **Latent uncertainty in the context of software-intensive systems (e.g., Cyber-Physical Systems (CPSs)) demands explicit attention right from the start of development. Use case modeling—a commonly used method for specifying requirements in practice, should also be extended for explicitly specifying uncertainty. To this end, we extend the Restricted Use Case Modeling (RUCM) methodology and its supporting tool to specify uncertainty as part of system requirements. Such uncertainties include those caused by insufficient domain expertise of stakeholders, disagreements among them, and known uncertainties pertaining to assumptions about the environment of the system. The extended RUCM, called U-RUCM, inherits the features of RUCM, such as automated analyses and generation of models, to mention but a few. Consequently, U-RUCM provides all the key benefits offered by RUCM (i.e., reducing ambiguities in requirements), but in addition, it allows specification of uncertainties with the possibilities of reasoning and refining existing ones and even uncovering unknown ones.**

We evaluated U-RUCM in the context of the U-Test project [1], with two industrial CPS case studies. Evaluation results showed that, with U-RUCM, we were able to get a significantly better and more precise characterization of the uncertainties involved compared to RUCM. This suggests that U-RUCM is an effective tool for dealing with uncertainty in requirements engineering. We present our experience, lessons learned, and future challenges based on the two industrial case studies.

Index Terms—Use Case Modeling, Belief, Uncertainty.

I. INTRODUCTION

The problem of uncertainty in software-intensive systems such as Cyber-Physical Systems (CPSs)), is familiar to the requirements engineering community. However, it has not been adequately addressed and, therefore, it lacks both methodological and tool support in both the literature and in practice. In their well-known use case modeling book, Bittner and Spence [2] pointed out that it is important to take the time to fill out missing areas and drill down into uncertainty. Uncertainty can be due to diverse causes, such as insufficient domain expertise or lack of information. Given the significant increases in the complexity of modern CPS and the diversity of the environments in which they are deployed, it is becoming critical to address uncertainty up front; that is, right from the start of development. This includes not only uncertainties about the requirements, but also uncertainties about its assumed operating environment.

Uncertainty in requirements has been studied in the context of dynamically adaptive systems in the presence of environmental uncertainty [3, 4]. Several goal-driven solutions [5, 6] have been proposed to handle uncertainty in similar contexts. Partial model-based solutions (e.g., [7, 8]) have been developed to support early requirements and architecture decision making. However, after conducting a literature review, we did not find any use case modeling methodology that explicitly handles uncertainty. Having such a methodology is important since use case modeling is a commonly used technique for specifying requirements in practice [7]. In our view, because uncertainty is a common phenomenon in requirements engineering, it is best to *address it explicitly* by identifying, qualifying, and, where possible, quantifying uncertainty.

The need for such a methodology arose in the context of the EU project U-Test, which focused on testing CPSs under uncertainty. The first key step in this project was to collect use cases with known uncertainties for two industrial CPSs and their environments. This was done with three industrial partners (Future Position X[1], ULMA[2] and Ikerlan[3], which are among the authors of this paper). The ultimate aim was to use these use cases as the starting point to create test-ready models to support automated testing of CPSs under uncertainty. To this end, we first introduced the RUCM methodology to our industrial partners and then extended it to enable specification of uncertainties. This led to the design of the U-RUCM methodology, which, to the best of our knowledge is the first use case modeling methodology that explicitly addresses uncertainty.

As noted, U-RUCM is based on a practical use case modeling solution, called Restricted Use Case Modeling (RUCM) [9, 10]. RUCM was initially proposed by Yue et al., for reducing inherent ambiguity in textual *Use Case Specifications* (UCSs) and to enable automated generation of UML models [10]. Later on, RUCM was extended to address various industrial challenges, including requirements-based testing [11] and use-case based requirements inspection [12]. RUCM and its extensions have been used to address industrial challenges from various domains (e.g., telecommunication [11, 13, 14], automotive [15]).

To structure and specify uncertainties in use case models, two templates were proposed for specifying *Belief Use Case Specifications* (BUCS) and uncertainties. A BUCS annotates the UCS with uncertainty information, including the source of uncertainty, the degree (measurement) of uncertainty, the risk of uncertainty, etc., as perceived by stakeholders and based on available evidence. Such models can be automatically generated as instances of a formal U-RUCM metamodel.

Evaluation of U-RUCM based on the two industrial case studies revealed that, with U-RUCM, we were able to

[1] fpx.se/geo-sports/
[2] www.ulmahandling.com/en/
[3] www.ikerlan.es/

significantly improve on the characterization and understanding of uncertainties in the requirements (up to 306% and 512% for the two case studies) compared to base RUCM. In this paper, we summarize practical lessons learned in the course of this evaluation and also discuss future challenges.

The rest of the paper is organized as follows: Section II presents the background, followed by the related work (Section III). The U-RUCM templates and keywords are explained in Section IV, followed by its formalization (Section V). The tool support and recommended methodology are given in Section VI. Section VII reports user experience, evaluation, lessons learned and future challenges. We conclude the paper in Section VIII.

II. Background and Running Example

A. U-Model

To help us understand the nature of uncertainty in the general context of software engineering, in our previous work [16] we developed a conceptual model called *U-Model* to define uncertainty and its associated concepts. The *U-Model* was developed based on an extensive review of existing literature on uncertainty from several disciplines including philosophy, healthcare and physics, and two industrial case studies from the two industrial partners of the U-Test project. Some of the *U-Model* concepts were further extended for supporting Model-Based Testing (MBT) of CPSs under uncertainty. More specifically, we developed the Uncertainty Modeling Framework [17] for supporting MBT of CPSs, which contains a UML profile, called the UML Uncertainty Profile (UUP), for specifying and measuring uncertainties as part of UML models. UUP was derived based on the U-Model. *UncerTum* has been successfully used for discovering unknown uncertainties [17, 18] and generating test cases [19].

The U-Model takes a subjective approach to representing uncertainty. This means that uncertainty is modeled as a state (i.e., worldview) of some agents (called *BeliefAgent*s), who, for whatever reason, do not have complete and fully accurate knowledge about some subjects of interest. In the U-Model, a *Belief* is an abstract concept, but it can be expressed in concrete form via one or more explicit *BeliefStatement*s (a concrete and explicit specification of some *Belief* held by a *BeliefAgent* about possible phenomena or notions belonging to a given subject area). Uncertainty (i.e., lack of confidence) represents a state of affairs whereby a *BeliefAgent* does not have full confidence in a belief that it holds. This may be due to any number of factors: lack of information, inherent variability in the subject matter, ignorance, or even due physical phenomena such as the Heisenberg uncertainty principle. While *Uncertainty* itself is an abstract concept, it can be quantified by a corresponding *Measurement*, which expresses in some concrete form the subjective degree of uncertainty that the agent ascribes to a *BeliefStatement*. As the latter is a subjective notion, a *Measurement* should not be confused with the degree of validity of a *BeliefStatement*. Instead, it merely indicates the level of confidence that the agent has in a statement.

B. Restricted Use Case Modeling (RUCM)

RUCM encompasses a use case template and 26 restriction rules for specifying textual UCSs [35]. RUCM aims to be easy to use, to reduce ambiguity and improve understanding, and to facilitate automated analysis. Results of two controlled experiments support these expectations [34, 35]. A RUCM UCS has one basic flow and, optionally, one or more alternative flows. An alternative flow always depends on a condition occurring in a specific step of another flow (referred to as the reference flow). We classify alternative flows into three types: A *specific* alternative flow refers to a specific step in the reference flow; a *bounded* alternative flow refers to more than one step (consecutive or not) in the reference flow; a *global* alternative refers to any step in the reference flow. For specific and bounded alternative flows, a RFS (Reference Flow Step) section specifies one or more (reference flow) step numbers. For example, as shown in Appendix A, the use case has one basic flow, called *Normal*. The specific alternative flow of *DetectIntrusion* branches out from step 10 of the basic flow *Normal*, as indicated by keyword RFS and plus "Normal 10". The global alternative flow *CallPolice* is triggered whenever the branching condition "The Alarm has been triggered for more than 5 minutes". The bounded alternative flow of FailOnEnablingMonitoring refers to steps 5-7 of the basic flow *Normal*, as indicated by "URFS Normal 5-7". URFS is a new keyword introduced to U-RUCM and will be discussed in Section IV.

RUCM defines a set of keywords to specify sentences that involve conditional logic (IF-THEN-ELSE-ELSEIF-ENDIF), concurrency (MEANWHILE), condition checking (VALIDATES THAT), and iteration (DO-UNTIL). UCMeta is a metamodel that can be used to formalize textual RUCM to facilitate automated analyses and generation of UML analysis models [9, 20]. UCMeta is specified using the OMG's standard Meta-Object Facility (MOF) [21], while the formalization of RUCM models to UCMeta instances is done automatically, as described in [9].

Since RUCM was initially proposed by Yue et al. [22] in 2009, multiple extensions have been proposed. A restricted test case modeling methodology was presented in [9] to automatically generate executable test cases. In [23], Wang et al. also presented another RUCM-based approach to automatically generate test cases from use case models. Both of these approaches have been evaluated using real industrial case studies. Wu et al. [24] extended RUCM for specifying safety requirements in the domain of safety-critical systems. The authors of [25] presented an approach for facilitating feature-oriented requirements validation in the context of automotive systems, where RUCM was used to specify system features.

C. Running Example

We illustrate U-RUCM using a modified version of the SafeHome case study provided in [26]. The SafeHome system implements various security and safety features in smart homes, including intrusion detection, fire detection, and flooding. One of the key functionalities of the system is that a homeowner activates the monitoring function of the system, which continuously checks for intrusions until it is

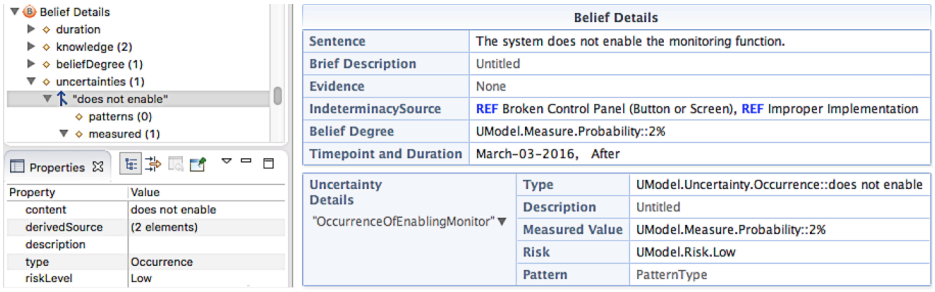


Figure 1. Specifying an Uncertainty of *FailOnEnablingMonitoring* step *A1*

explicitly disabled. During monitoring, any occurrence of an intrusion should be detected, immediately followed by sending an intrusion notification to the homeowner and the activation of an alarm. The corresponding use case for this example, named *Monitor Windows and Doors*, is shown in Appendix A.

III. Related Work

Runtime detection, monitoring, reasoning, and managing of requirements, generally referred to as being *requirements-aware*, is necessary in self-adaptive systems [27], due to inherent changes of operational environments and contextual uncertainties. For this purpose, RELAX [3, 4] – a representative requirements specification and reasoning solution – was proposed to support development of requirements for dynamically adaptive systems with environmental uncertainty. RELAX consists of a set of keywords, which are classified into *Modal* operators (e.g., SHALL), *temporal* operators (e.g., BEFORE), *ordinal* operators (e.g., AS MANY) and *uncertainty* operators (e.g., ENV). Uncertainty factors aim to indicate where a relaxation of requirements is warranted and, therefore, adaptive behavior is needed. Based on a structured natural language based notation, the authors also proposed a methodology for relaxing SHALL statements with the RELAX keywords. In addition, RELAX requirements can be formalized using fuzzy logic and reasoning can be performed, when needed.

RELAX has been also integrated with goal-modeling notations (i.e., KAOS [28]) to allow for fuzzy goals [3]. Along the same line, Luciano et al. [5] proposed FLAGS for enabling the specification of adaptive goals, which are of two types: *crisp* goals (specified via linear temporal logic) and *fuzzy* goals (specified using a fuzzy temporal logic). Chen et al. [29] proposed a goal-driven self-optimization framework to handle three different types of uncertainty in goal models: *contribution*, *preference,* and *effect* uncertainty. ReAssuRE was proposed by Welsh et al. [6] to attach *claims* to softgoal contribution links of goal models, with the aim to record the rationale for selecting a goal realization strategy when the optimum choice is uncertain. Later on, Ramirez et al. [30] integrated ReAssuRE with RELAX to assess the validity of claims at runtime, for dealing with environmental uncertainty in dynamic adaptive systems.

Compared to these goal-based approaches, U-RUCM is more generic, as it is not targeting dynamic adaptive systems in particular. Second, U-RUCM is built on a use case modeling methodology, such that it can naturally facilitate the specification of uncertain alternative scenarios. Furthermore, U-RUCM also enables the specification of various types of uncertainties (e.g., *Time*, *Occurrence*), more precise characterization of uncertainties with information such as *Pattern*, and the ability to quantify uncertainties in different ways (e.g., *Probability*, *Fuzziness*). Currently, U-RUCM has a dedicated template for specifying uncertainty. In the future, it would be useful to investigate using keywords (similar to RELAX) to reduce the effort in specifying uncertainties.

Uncertainty is also considered as an important factor that complicates early requirements definition and decision making. Salay et al. [7] proposed the *MAVO* annotations for modeling uncertainty in requirements engineering models, based on the concept of *partial models* (with their properties checked as *True*, *False* or *Maybe*) [31]. The *MAVO partiality* annotations consist of: *May partiality* (indicating that an element should exist in the model), *Abs partiality* (indicating that an element is a collection of elements), and *Var partiality* (indicating that it is unclear if an element should be merged with others). Famelis and Santosa [8] proposed to use colored Entity-Relation models for explicitly capturing the MAVO partiality, as well as *Points of Uncertainty*, a concept representing a specific decision about which there is uncertainty. Compared to these partial-model solutions, U-RUCM is systematically derived from the U-Model, which extends beyond partial models and, therefore, supports a richer means for specifying and modeling uncertainty. In addition, U-RUCM is integrated with RUCM, which enables the specification of uncertain alternative scenarios.

Uncertainty can hinder organizations in making strategic decisions due to, for example, uncertain stakeholders' goals and priorities. In this context, uncertainty is defined as the lack of knowledge of the consequences of decision alternatives. Letier et al. [32] proposed ways for reasoning about uncertainties to support early requirements and architecture decision analysis. Uncertainties are represented as probability distributions, while Monte-Carlo simulations are used for simulating the impact of alternative decisions. That paper, however, does not provide a solution for uncertainty specification and elicitation. Similarly, Esfahani et al. [33] proposed GuideArch, a framework for quantitative exploration of the architectural solution space under uncertainty, which is based on fuzzy mathematical methods for reasoning about uncertainty. Although these works support means for reasoning, simulation, and exploration in the presence of uncertainty, none of them propose an uncertainty specification and modeling solution in the context of requirements engineering.

IV. U-RUCM Templates and Keywords

As noted in Section II.B, the RUCM methodology has two key distinguishing features: specifying UCSs with the RUCM template, and applying the RUCM restrictions

(including the keywords) to guide the way in which users use natural language to specify the control flows of UCSs. U-RUCM extends the RUCM template and proposes two U-RUCM templates and introduces two new keywords.

One of the U-RUCM templates is for specifying BUCSs as shown in TABLE I. The BUCS template inherits the key heading fields of the RUCM template, such as *Use Case Name* and *Brief Description*, which are not shown in the table due to space limitation. In addition, U-RUCM introduces six new fields to indicate: 1) who specifies the BUCS (*Belief Agent(s)*), 2) when it was specified and the length of time during which the belief agent(s) holds the belief (*Time Point and Duration*), 3) the degree to which the belief agent(s) believes the specification (*Belief Degree*), 4) a set of indeterminacy sources that resulted in the uncertainties of the BUCS (*Indeterminacy Source(s)*), 5) evidence used to support this BUCS and its belief and uncertainty elements (*Evidence*), and 6) the precondition of the UCS on which the belief and uncertainties are founded (*Belief Precondition*).

As discussed in Section II, each RUCM UCS has one and only one basic flow and, optionally, three types of alternative flows. U-RUCM extends each type of event flows by 1) introducing a *belief degree*, which measures the degree to which the belief agent(s) believe a specific flow, 2) introducing a new keyword, *URFS* (Uncertain Reference Flow Step(s)), from which an alternative flow of events branches out, 3) providing the capability to annotate sentences in steps of flows and postconditions with belief and uncertainty information, and 4) introducing the new concept of *alternative steps* to enable the specification of uncertainties for alternative steps across flows of events. Note that for the case of a global alternative flow, a condition for branching from any step in the flow, should be specified via the *Belief Branching Condition* field. In Section V, we formally define each of these fields and concepts.

We have developed an editor for U-RUCM (Section VI). The running example shown in Appendix A is specified with the editor. The use case has one basic flow (called *Normal*) and several alternative flows (as shown in Appendix A). The complete specification is provided in [34] for reference.

The other U-RUCM template is for specifying uncertainty. An example is shown in Figure 1. This template has fields for identifying indeterminacy sources and evidence, and for specifying uncertainty properties such as *Type*, *Pattern*, and *Measured Value*. More details see Section V.

In addition to the standard RUCM keywords (Section II.B), REF is newly introduced for specifying associated indeterminacy sources and evidence so that they can be referenced in multiple places within a BUCS. For example, the field *Indeterminacy Source(s)* lists all the defined indeterminacy sources of the specification, each of which is referenced with REF (Appendix A, C1 and C2). Like the RUCM RFS keyword, the URFS keyword is used for associating an alternative flow with the steps in its reference flow. However, RFS is used for branching out from a reference flow step under a clear condition. For example, in Appendix A the alternative flow *DetectIntrusion* uses RFS to indicate that it branches from step 10 of the basic flow (*Normal*). In contrast, URFS is provided to associate uncertainties across flows of events. For example, in the running example (Appendix A, E1 and E2), the URFS keyword is applied to *FailOnEnablingMonitoring* to show that, in an uncertain unknown condition, it is possible that the sequence of alternative sentences A1 and A2 can "replace" step 2 of the alternative flow *DetectIntrusion*.

V. U-RUCM FORMALIZATION

The formalization of U-RUCM is realized via integration of an extended version of UCMeta and the U-Model. The full formalization is captured in a distinct metamodel (called *BeliefUCMeta*), which is provided in [34] for reference. Due to space limitation, in this section we highlight only the key elements of the metamodel.

A. Top-level Belief Element and Classifier

BUElement, which specializes *UseCaseElement* of UCMeta), is the root of all other elements of BeliefUCMeta (Figure 2).

TABLE I. The U-RUCM Template for Belief Use Case Specifications (BUCSs)

Belief Agent(s)	One or more agents who hold belief about this BUCS.	
Time Point and Duration	The time point when the BUCS is specified and the duration in which the belief agent(s)'s belief on the BUCS holds.	
Belief Degree	The degree to which the belief agent(s) believe the BUCS.	
Indeterminacy Source(s)	The set of indeterminacy sources related to this BUCS.	
Evidence	Evidence to support this BUCS, and its contained belief and uncertainty elements.	
Belief Precondition	Belief agent(s)' belief on the precondition, which describes what should be true before the use case is executed.	
Belief Basic Flow (*Belief degree*)	Specifies the main successful path, also called "happy path".	
	Steps (numbered)	A set of ordered belief sentences.
	Belief Postcondition	Belief agent(s)' belief on what should be true after the basic flow executes.
Belief Specific Alternative Flow (*Belief degree*)	Applies to one specific step of the reference flow.	
	URFS	The reference flow step where the belief agent(s) believe there are uncertainties.
	Alternative Step	An alternative to the reference flow step.
	Steps (numbered)	A set of ordered belief sentences.
	Belief Postcondition	Belief agent(s)' belief on what should be true after the specific alternative flow executes.
Belief Bounded Alternative Flow (*Belief degree*)	Applies to more than one step of the reference flow, but not all of them.	
	URFS	A list of reference flow steps where the belief agent(s) believe there are uncertainties.
	Alternative Steps	A set of alternatives to the reference flow steps.
	Steps (numbered)	A set of ordered belief sentences.
	Belief Postcondition	Belief agent(s)' belief on what should be true after the bounded alternative flow executes.
Belief Global Alternative Flow (*Belief degree*)	Applies to all the steps of the reference flow.	
	Belief Branching Condition	Belief agent(s)' belief on the condition, which describes what should be true when branching from any of the steps of the reference flow.
	Steps (numbered)	The set of ordered belief sentences
	Belief Postcondition	Belief agent(s)' belief on what should be true after the global flow executes.

BeliefClassifier (Figure 3) is an abstract metaclass, introduced to support classification of a set of BUCS elements (e.g., *BeliefPrecondition*, *BeliefFlowOfEvents*), to which belief and uncertainty information can be attached. *BeliefClassifier* has three attributes: *isUncertainty : Boolean* (a derived attribute to indicate if a belief classifier has an associated uncertainty), *isComposite : Boolean* (indicating if a belief classifier can be decomposed into finer belief elements), and *beliefDegree : Measurement* (formalizing the degree to which a belief agent believes in a belief classifier). More detail on measurements is provided in Section E below.

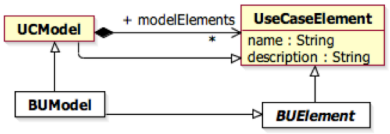


Figure 2. Root Elements of BeliefUCMeta

B. Belief Use Case Specification

BeliefUseCaseSpecification specializes *BeliefClassifier* and *UseCaseSpecification* of UCMeta. In general, *BeliefUseCaseSpecification* is a concrete specification of a use case specified by a *BeliefAgent*. It includes information about the agent's confidence that the use case will execute as specified. The BUCS template of U-RUCM (TABLE I) corresponds to this metaclass. An instance of *BeliefUseCaseSpecification* is created for each BUCS and serves as the container of other belief-related elements (e.g., *BeliefPrecondition*, *BeliefFlowOfEvents*, *BeliefSentence*). In particular, a BUCS includes specifications of *IndeterminacySource*s and *Evidence* relevant to the use case. The attribute *nature* of *IndeterminacySource* is typed with *IndeterminacyNature* defined by five enumeration literals representing five possible indeterminacy sources. These can be referenced (via REF) by other belief elements such as *Uncertainty*. Such references are formalized as instances of *IndeterminacyKnowledge* and *EvidenceKnowledge* of BeliefUCMeta (Figure 3).

C. Belief Flow Of Events

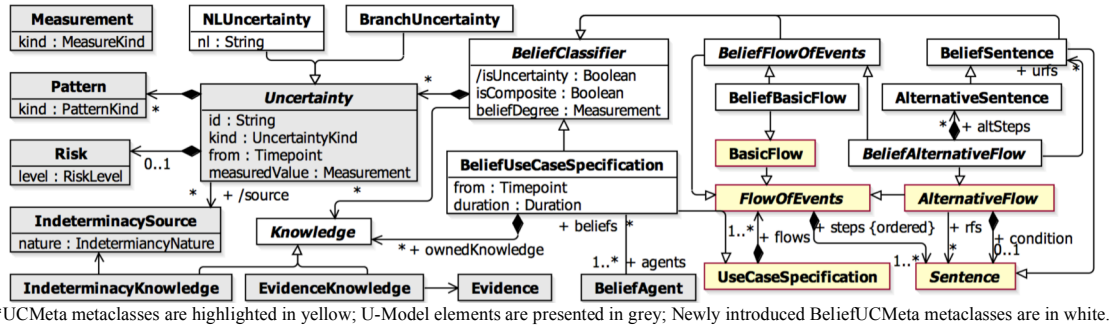
BeliefFlowOfEvents extends *FlowOfEvents* of UCMeta (Figure 3). Hence, it inherits the three types of alternative flows: *Specific*, *Bounded* and *Global*, which are not shown in the figure due to space limitations. Alternative flows can also be differentiated based on whether RFS or URFS is used to refer to one or more steps of a reference flow. RFS is used only when the branching condition and the system behavior under this condition are fully clear to the belief agent and can, therefore, be specified as a definite alternative flow. For example, the *DetectIntrusion* flow in Appendix A branches out from step 10 of the basic flow under the condition that "the windows and doors are open". In contrast, URFS is used when the belief agent is not fully confident about a certain piece of system behavior or condition (represented by one or more steps in a flow). In principle, an URFS alternative flow should always be linked to one or more indeterminacy sources. If such indeterminacy sources are known, U-RUCM provides a way to specify them (see Section B). For example, the *FailOnEnablingMonitoring* flow is defined as the belief agent is not fully confident that "the system enables the monitoring function" (step 5 of the basic flow, Appendix A) will actually occur.

In summary, U-RUCM provides five different ways of specifying alternative flows: *Global* with RFS, *Bounded* with RFS, *Specific* with RFS, *Bounded* with URFS and *Specific* with URFS. Global alternative flows are not recommended to be combined with URFS, as it is often impossible to be unclear about every single step of a reference flow. For any URFS alternative flow, there should be at least one alternative sentence specified in the alternative flow, which "replaces" the referenced steps of the reference flows of events defined in the URFS statement. Any RFS alternative flow should not contain any alternative step. Note that alternative sentences (which are ordered as sequential steps) are defined at the beginning of an URFS alternative flow, followed by a sequence of regular belief sentences.

D. Belief Sentence

All sentences in a BUCS are belief sentences, which are *formalized* as *BeliefSentence* elements (specializing the *Sentence* concept of UCMeta). In RUCM and UCMeta, sentences are classified into *simple*, *complex* and *special* sentences. Consequently, U-RUCM and BeliefUCMeta classify belief sentences into *BeliefSimpleSentence*, *BeliefComplexSentence* and *BeliefSpecialSentence* (not shown in Figure 3 due to space constraints).

A belief simple sentence is an atomic belief statement, which, from the sentence structure perspective, is composed of only one subject and one predicate. Belief simple sentences can appear in action steps of flows, preconditions, postconditions, and other fields of a BUCS. Belief complex sentences are sentences with the following RUCM keywords applied: IF-THEN-ELSE-ELSEIF-THEN-ENDIF for conditions, DO-UNTIL for iterations, MEANWHILE for concurrency, and VALIDATE THAT for validation. A complex sentence can consist of one or more belief simple sentences. Belief special sentences are sentences involving keywords RESUME STEP, ABORT, RFS, EXTENDED BY, INCLUDE and URFS (newly introduced in U-RUCM).



*UCMeta metaclasses are highlighted in yellow; U-Model elements are presented in grey; Newly introduced BeliefUCMeta metaclasses are in white.
Figure 3. BeliefUCMeta – the metamodel formalizing U-RUCM

In U-RUCM, we also introduce *AlternativeSentence*, which extends *BeliefSentence* (Figure 3). Alternative sentences only appear in URFS alternative flows as action steps (Section C). In the current implementation of the U-RUCM editor (Appendix A), alternative sentences (steps) are denoted with A1, A2, etc. An alternative sentence can be any type of belief sentence: simple, complex or special.

E. Uncertainty

In U-RUCM, we classify uncertainties into two basic types: *NLUncertainty* and *BranchUncertainty*. An instance of *NLUncertainty* refers to a Part of Speech (PoS) (e.g., noun, verb) of a belief sentence, about which a belief agent lacks confidence. For example, one instance of *NLUncertainty* in alternative sentence *A1* of the *FailOnEnablingMonitoring* flow (Appendix A) shows that the belief agent is only 2% confident about the occurrence of the event. This is captured by "*enables*", which is the predicator verb of the sentence. For example, the uncertainty is the *Occurrence* type of uncertainties (formalized as one of the literals of enumeration *UncertaintyKind* of BeliefUCMeta) which is not shown in Figure 3 due to space limitation).

A set of branches can be derived from a BUCS, systematically by following certain strategies [11]. Each of these derived branches represent a straight path from the precondition of the specification all the way to a postcondition of a flow of events. This can be considered as all-possible execution paths of a use case and therefore the occurrence of a particular path is an uncertainty. Such an uncertainty is an instance of *BranchUncertainty* with the *Occurrence::UncertaintyKind* kind. Since such branches can be automatically generated, measurements of the branch uncertainties of the belief specification can be automatically calculated when needed, if and only if uncertainties of the belief sentences of the specifications are specified.

F. Measurement

In U-RUCM, there are two types of metrics: one is for measuring the belief degree of a belief classifier and the other is about measuring uncertainty. A measurement can take different kinds of measures such as *Probability*, *Vagueness* and *Ambiguity*, which are formalized as enumeration *MeasureKind* of BeliefUCMeta (not shown in Figure 3 due to space limitation). Note that, in U-RUCM, all measurements are subjective. This is because, at the requirements level, domain experts specify measurements based on their experience, knowledge, and even preference, as opposed to basing them on available hard data.

VI. Tool Support and Methodology

A. Tool Support

BUCSs are specified in the U-RUCM editor, which is implemented in a modeling framework, called the *Lightweight Modeling Framework* (LMF [35])). This framework implements functionality similar to those of the Eclipse Modeling Framework (EMF), but with a lightweight design with the aim of reducing tight coupling with Eclipse (so as to facilitate easier porting to other platforms). LMF has two editors: a reflective model editor and a metamodel editor. The LMF Reflective Editor is a simple model the LMF metamodel reflection mechanism.

BUCSs specified with the editor can be automatically formalized into instances of BeliefUCMeta concepts. In the past, we have developed the transformation from RUCM to UCMeta, based on natural language processing techniques [9]. The transformation from U-RUCM to BeliefUCMeta is just an extension of the transformation from RUCM to UCMeta. The formalized specifications can be directly used for performing different kinds of analyses and generations of other artifacts when needed.

We have made a video to demonstrate the U-RUCM editor and the formalization from U-RUCM to BeliefUCMeta, along with the metamodel of BeliefUCMeta, UCMeta and U-Model in [16] for references.

B. Methodology

Although the U-RUCM concepts can be used in many different ways, in this section, we recommend one methodology based on our own experience. It starts with the creation of a use case model, specifying the actors, use cases, and relationships among them. The belief agents in this case are the requirements engineers who capture the information, including indeterminacy sources, evidence, and uncertainty degrees from the various stakeholders. Of course, it is always possible to revisit the initial specifications subsequently should new evidence or indeterminacy sources be uncovered.

When the overall context of a use case model is established, one can start to develop a BUCS for each use case. There is actually no particular order for specifying primary and secondary actors, belief agents, etc. We recommend a sequence for guiding users through the process that proceeds from simple tasks to more complicated ones. Specifying flows of events is the most challenging task, as it requires a lot of careful analysis, discussions, and design. The process is always iterative.

When specifying alternative flows, global alternative flows are often used to specify exceptions and behaviors crosscutting all the steps of a reference flow. The key task here is to identify the proper branching condition. If one needs to refer to one or more (but not all) steps of a reference flow, specific or bounded alternative flows can be created. As discussed in Section V.C, U-RUCM provides five different ways of specifying alternative flows and some constraints (e.g., alternative sentences only appear in URFS alternative flows) should be applied when using U-RUCM in this aspect. In our current implementation of the editor, we have enforced these constraints so that chances of violating them are eliminated. By definition, URFS and RFS are different and therefore should be applied in different situations, as discussed in Section V.C. We highly recommend using URFS to identify uncertain alternative flows only after the entire structure of flows of events (using RFS) of a BUCS is defined.

Each flow of events consists of a set of steps, which are specified as belief sentences. For each belief sentence, one should refer to one or more relevant indeterminacy sources and evidence, based on which, one can define the belief

degree and associated uncertainties. As discussed in Section V, uncertainties can be more precisely characterized with pattern and risk information and measured in different ways. In practice, it is not always possible to obtain and enter all of this information at once. So, a rule of thumb is to first identify as many uncertainties as possible and only then refine them with more detailed information.

VII. User Experience -- Evaluation

A. Case Studies and Overall Objective of Evaluation

One of the two industrial case studies involved the Automated Warehouse (AW) from ULMA Handling Systems. These complex systems serve to monitor, control, and manage warehouses for goods of different types, such as food and beverages and textiles. Each handling facility (e.g., crane, conveyor) forms a physical unit, and together they are dedicated to one handling system application (e.g., storage).

The second industrial case study used the Geo Sports (GS) system from Future Position X. This system measures the performance of an individual or a team as well as the conditions of athletes over a sustained period of time in actual game environments (e.g., a soccer field). The measurements are made continuously and in real time using geo-position sensors during both training and competition. Our case study involved bandy, a form of ice hockey played predominantly in Northern Europe and Russia. To the best of our knowledge, this project was the first to monitor sports on ice using sensors.

Our overall objective of the evaluation was to assess, in an industrial setting, whether U-RUCM was effective in terms of facilitating the development of use case models with the explicit focus on uncertainty.

B. Context and Execution of Evaluation

At the start of the project, base RUCM was introduced to both industrial partners (i.e., ULMA and FPX) as a means for identifying and specifying the initial versions of their uncertainty requirements. Subsequently, two onsite workshops (one for each partner) were conducted to further refine the collected requirements. The resulting RUCM models were refined iteratively during the process of developing the U-Model [16]. As shown in column *RUCM Model* of TABLE II, a total of 20 use cases for each case study were designed, 93 RUCM event flows were produced (52 for AW and 41 for GS). These flows of events were later refined into belief flows and alternative flows with URFS applied. In total, the RUCM model for AW had 229 sentences, while the GS model had 256. About uncertainties specified in the RUCM models, 33 (for AW) and 26 (for GS) sets of steps of flows of events describing alternative scenarios were considered as involving uncertainties. These were later refined into alternative sentences and instances of *NLUncertainty* in the U-RUCM models.

After that, we conducted a questionnaire-based survey (derived from the RUCM models) to collect data to quantify the identified uncertainties. During this non-trivial process, which involved collecting uncertainty information, deriving and verifying the U-Model, and involving multiple stakeholders, RUCM was deemed adequate for the purpose of providing initial data. It captured uncertainty requirements at a relatively coarse-grained level.

Once the U-Model was finalized, we developed U-RUCM, which integrates the U-Model and RUCM. U-RUCM was then used to refine the RUCM models developed by the industrial partners. Results of this are presented in Section C.

C. Results

All the initial RUCM models developed by the industrial partners were refined using U-RUCM to capture all the identified uncertainties. Descriptive statistics of the resulted U-RUCM models are reported in the *Result* columns of TABLE II. The table shows how many elements were added, modified, and removed during the process of moving from RUCM to U-RUCM for the two case studies.

Recall that U-RUCM realizes the *Uncertainty* concept of U-Model by four concrete means: 1) *NLUncertainty* for belief sentences, 2) *BranchUncertainty* for possible executions of UCSs from the beginning to end, 3) uncertainties in flows of events (captured via URFS and alternative flows), and 4) uncertainties in sentences across flows of events (captured as alternative sentences). We applied these four U-RUCM mechanisms systematically by following the guidelines described in Section VI and then carefully examined all the specified BUCSs to further refine the RUCM models.

As shown in TABLE II, we refactored the design of the RUCM use case model of AW by merging three use cases describing similar scenarios into one, which led to the deletion of 2 use cases (as shown in the table). We also added 2 use cases to the RUCM model of GS as the result of the refactoring, as these two use cases can be invoked (via the include relationships) by multiple use cases.

TABLE II also indicates that 3 uncertainties in the AW

TABLE II. Descriptive Statistics of the RUCM Models, U-RUCM Models and Refinements

AW							GS					
Key RUCM Elements		Refinement			U-RUCM Model	Key U-RUCM Elements	U-RUCM Model	Refinement			Key RUCM Elements	
		Added	Modified	Removed				Added	Modified	Removed		
Use Case	20	0	20	2	**18**	Use Case	**23**	3	20	0	20	Use Case
FlowOfEvents	52	18	23	4	**66**	(Belief) Flow Of Events	**71**	31	21	1	41	FlowOfEvents
Sentence	229	63	56	16	**276**	(Belief) Sentence	**348**	97	46	5	256	Sentence
		45	0	0	**45**	Alternative Sentence	**76**	76	0	0		
Uncertainty	33	32	18	3(1,2)$^{+}$	**62**	(NL)Uncertainty	**70**	48	15	4 (2,2)$^{+}$	26	Uncertainty
		72	0	0	**72**	BranchUncertainty	**89**	89	0	0		
		43		0	**43**	URFS	**48**	48	0	0		
		23	0	0	**23**	Indeterminacy Source	**18**	18	0	0		

(n,m)$^{+}$ -- n is the number of uncertainties removed due to refactoring; m is the number of uncertainties that are changed to indeterminacy source.

RUCM model were removed and 4 from the GS model. This was because: 1) We optimized the design of the use case model by removing duplicated uncertainties (e.g., those describing improper wearing of positioning devices, which is the same for either indoor or outdoor games), 1 from AW and 2 from GS; 2) We identified uncertainties from the RUCM models that are actually indeterminacy sources (e.g., long distance between a positioning device and the satellites is an indeterminacy source, which can lead to the failure of locating the satellites with insufficient resolution nature), 2 for AW and 2 for GS.

The uncertainty-specific concepts *Indeterminacy Source*, *Alternative Sentence,* and *BranchUncertainty* were only introduced in U-RUCM. Consequently, there were no corresponding elements in the RUCM models. After carefully going through details of the RUCM models using steps described previously, we derived a total of 23 indeterminacy sources for AW and 18 for GS, 45 alternative sentences for AW and 76 for GS. Furthermore, we discovered 32 instances of *NLUncertainty* for AW and 48 for GS. These turned out to be cases of "unknown knowns" for our industrial partners, that is, tacit knowledge that was not explicit initially. This activity led to the addition of 18 belief flows of events for AW and 31 for GS, 72 new branch uncertainties for AW and 89 for GS, and 43 additional alternative flows with URFS applied for AW and 48 for GS.

In summary, the total numbers of the instances of metaclasses *NLUncertainty* and *BranchUncertainty* of U-Model, populated for each of the industrial case studies are 62+72=134 for AW and 70+89=159 for GS. When comparing this with their corresponding "rough" RUCM models, we conclude that, by using U-RUCM, we were able to significantly enhance the extent and precision of modeling uncertainties in requirements (i.e., (134-33)/33=306% for AW and (159-26)/26=512% for GS). This suggests that U-RUCM is an important improvement in dealing with uncertainty in requirements engineering.

In the U-Test project, we also developed test ready models [17, 18] (represented as UML class diagrams and state machines) using the *UncerTum* (see Section II.A). The test ready models were used to generate test cases, which were then executed successfully in actual systems [19]. There are clear correspondences between the scenarios and uncertainties defined in the test ready models and the ones defined in the U-RUCM models, since the U-RUCM models were used as the input (uncertainty requirements) for developing the test ready models. This gives us a high degree of confidence in the quality of the derived U-RUCM models.

D. Experience, Lessons Learned and Future Challenges

Identifying common uncertainties. From the GS case study, we noted that human behavior was the key indeterminacy source of uncertainties, due to incorrect interactions with the system. For the AW case study, on the other hand, uncertainties and indeterminacy sources centered mainly on the data communications between control units and their controlled devices. From these types of observations, we can conclude that it is possible in principle to identify common sources and types of uncertainties that occur in a given domain or even across domains. This knowledge can be then used to define *reusable* uncertainty specifications and their corresponding behaviors.

Learning about uncertainty by applying U-RUCM. In the past, we experienced that one can learn how to better design use case models by using RUCM. This is why RUCM is used as a teaching method for requirements engineering and software engineering courses both at the undergraduate and graduate levels[4]. Similarly, based on the results of this project, we surmise that it is possible to gain more precise and more direct understanding of both uncertainty and indeterminacy sources by using U-RUCM.

Automated, scalable and systematic reasoning. For more effective coping with uncertainty, automated/semi-automated reasoning about uncertainty and indeterminacy sources can certainly be helpful. This is because, for any non-trivial system, a use case model might be large and may contain a large number of potentially inter-related uncertainties. From our experience during the initial phases of our study when we were not using U-RUCM, we learned that unassisted human reasoning tends to be time-consuming and unsystematic. This is why we chose a more formal approach when developing U-RUCM – via the BeliefUCMeta metamodel – which provides a formal foundation for future, automated reasoning techniques.

Specializing U-RUCM. RUCM can be specialized for different purposes and for different domains. For example, in another research project we are developing a version of RUCM specifically for real-time systems. In such cases, the standard RUCM template and keywords are extended to allow the specification of time constraints. These are also subject to uncertainty. Based on that, we anticipate that U-RUCM will also need to be extended for specific domains.

Harvesting the benefits of natural language processing techniques. During the process of deriving U-RUCM and performing the two industrial case studies, we noticed that there is an opportunity to further refine *NLUncertainty*, the core concept for representing uncertainties in belief sentences (see Section V.E). The general idea here is to rely on natural language processing techniques to automatically identify grammatical structures (e.g., Subject), PoSs (e.g., Verb), sentence structures (e.g., Subject-Verb-Object), and/or sentence semantics (e.g., an actor sends a request to the system) in belief sentences. Based on this, heuristics can be defined to automatically identify potential uncertainties and/or verify already specified ones in belief sentences. For example, a verb of the predicator of a sentence might have an *Occurrence* uncertainty associated with it. A noun (being the direct object of a simple sentence) might be associated with a *Content* uncertainty.

Reckoning on branch uncertainties. It may be possible to automatically derive values of branch uncertainties (A branch uncertainty indicates a belief agent's confidence in the possibility that the execution of the use case takes this particular branch.) At the very least, branch uncertainties can at least help to 1) identify critical paths to reduce uncertainties or perform risk analyses (if the postcondition

[4] http://www.zen-tools.com/rucm/index.html

that a branch leads to is considered as the consequence of the branch), 2) verify the overall belief degree that a belief agent has in a belief specification, and 3) derive test cases targeting particularly branches with high uncertainty. This is a possible avenue of further research.

Systematically discovering unknown known indeterminacy sources and uncertainties and transforming them into known unknown uncertainties and known known indeterminacy sources. As the experiments showed, it is possible to systematically identify previously unknown known based on already-specified (known) uncertainties and indeterminacy sources. A systematic methodology (ideally with tool support) can be followed to identify more unknown knowns and currently known unknown uncertainties (e.g., by combining already identified uncertainties that are associated with the same part of system behavior).

Transforming U-RUCM models into other downstream artifacts. To maximize the benefit of U-RUCM models, one possibility is to transform them automatically or semi-automatically into other artifacts that need to be developed during system development. For example, U-RUCM models can be transformed into UML state machines via the UUP profile (Section II.A), for supporting MBT of CPSs under uncertainties. This is feasible as RUCM models can be transformed into UML models and test cases (Section II.B).

VIII. SUMMARY

The impact of uncertainty, which is increasingly being recognized as an inherent and crucial property of non-trivial software-intensive systems (e.g., CPSs), needs to be better understood and addressed explicitly in all phases of system development. In particular, it has to be explored and characterized as much as possible during requirements engineering (e.g., elicitation, specification, and verification). Use case modeling is a well-known and commonly applied requirements specification/modeling method in practice. Specifying uncertainty as part of use case models is therefore particularly useful. In this paper, we described a methodology and a corresponding tool (U-RUCM) for helping practitioners to specifying uncertainties in requirements as part of use case models.

U-RUCM originated in the context of the U-Test project (http://www.u-test.eu/), which involved a consortium of nine partners. The initial version of the uncertainty requirements was developed by our industrial partners using the basic RUCM methodology, on which U-RUCM was founded. After refining the RUCM models, by applying the U-RUCM methodology, we successfully identified and specified more than 300% and 500% (previously unknown) uncertainty requirements for the two case studies. The resulting U-RUCM models was used as a reference to develop test ready models for generating executable test cases to test the two industrial applications. As users of U-RUCM ourselves during these “real-world” experiments, we gained invaluable experience about its use and future potential.

REFERENCES

[1] S. Ali and T. Yue, "U-Test: Evolving, Modelling and Testing Realistic Uncertain Behaviours of Cyber-Physical Systems," 2015.

[2] K. Bittner and I. Spence, *Use Case Modeling*: Addison-Wesley, 2003.

[3] B. H. C. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, "A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty," in *Model Driven Engineering Languages and Systems, MODELS 2009*.

[4] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel, "RELAX: a language to address uncertainty in self-adaptive systems requirement," *Requirements Engineering,* 2010.

[5] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy goals for requirements-driven adaptation," in *2010 18th IEEE International Requirements Engineering Conference*, 2010.

[6] K. Welsh, P. Sawyer, and N. Bencomo, "Towards requirements aware systems: Run-time resolution of design-time assumptions," in *Automated Software Engineering (ASE), 2011*.

[7] R. Salay, M. Chechik, J. Horkoff, and A. D. Sandro, "Managing requirements uncertainty with partial models," *Requirements Engineering*, 2013.

[8] M. Famelis and S. Santosa, "MAV-Vis: a notation for model uncertainty," in *Modeling in Software Engineering (MiSE), 2013 5th International Workshop on*, 2013.

[9] T. Yue, L. Briand, and Y. Labiche, "aToucan: an Automated Framework to Derive UML Analysis Models From Use Case Models," *ACM Transactions on Software Engineering and Methodology,* 2015.

[10] T. Yue, L. C. Briand, and Y. Labiche, "Facilitating the transition from use case models to analysis models: Approach and experiments," *ACM Transactions on Software Engineering and Methodology (TOSEM), 2013*.

[11] T. Yue, S. Ali, and M. Zhang, "Applying A Restricted Natural Language Based Test Case Generation Approach in An Industrial Context," in *International Symposium on Software Testing and Analysis (ISSTA)*, 2015.

[12] H. Zhang, T. Yue, S. Ali, and C. Liu, "Facilitating requirements inspection with search-based selection of diverse use case scenarios," in *proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, 2016.

[13] M. Zhang, T. Yue, S. Ali, H. Zhang, and J. Wu, "A Systematic Approach to Automatically Derive Test Cases From Use Cases Specified in Restricted Natural Languages," in *8th System Analysis and Modelling Conference (SAM'14)*, 2014.

[14] T. Yue and S. Ali, "Bridging the gap between requirements and aspect state machines to support non-functional testing: industrial case studies," in *European Conference on Modelling Foundations and Applications*, 2012.

[15] G. Wang, V. Pascucci, A. Goknil, L. Braind, and Z. Lqbal, "Automatic Generation of System Test Cases from Use Case Specifications," presented at the International Symposium on Software Testing and Analysis, Baltimore, Maryland, 2015.

[16] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model," in ECMFA, 2016.

[17] M. Zhang, S. Ali, T. Yue, and R. Norgren, "An Integrated Modeling Framework to Facilitate Model-Based Testing of Cyber-Physical Systems under Uncertainty," 2016.

[18] M. Zhang, S. Ali, T. Yue, and R. Norgren, "Interactively Evolving Test Ready Models with Uncertainty Developed for Testing Cyber-Physical Systems," 2016.

[19] M. Zhang, S. Ali, T. Yue, and M. Hedman, "Uncertainty-based Test Case Generation and Minimization for Cyber-Physical Systems: A Multi-Objective Search-based Approach," 2016.

[20] J. Guo, J. White, G. Wang, J. Li, and Y. Wang, "A genetic algorithm for optimized feature selection with resource constraints in software product lines," *Journal of Systems and Software,* 2011.

[21] OMG, "Meta Object Facility (MOF) Core Specification", 2014.

[22] T. Yue, L. Briand, and Y. Labiche, "A Use Case Modeling Approach to Facilitate the Transition Towards Analysis Models: Concepts and Empirical Evaluation."

[23] C. Wang, F. Pastore, A. Goknil, L. Briand, and Z. Iqbal, "Automatic generation of system test cases from use case specifications,"

presented at the Proceedings of the 2015 International Symposium on Software Testing and Analysis, Baltimore, MD, USA, 2015.

[24] J. Wu, S. Ali, T. Yue, J. Tian, and C. Liu, "Assessing the Quality of Industrial Avionics Software: An Extensive Empirical Evaluation," *Empirical Software Engineering,* 2016.

[25] T. Yue, H. Zhang, S. Ali, and C. Liu, "A Practical Use Case Modeling Approach to Specify Crosscutting Concerns: Industrial Applications," 2015.

[26] R. S. Pressman, *Software engineering: a practitioner's approach 7th edition*: Palgrave Macmillan, 2010.

[27] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein, "Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems," in *2010 18th IEEE International Requirements Engineering Conference*, 2010.

[28] A. Van Lamsweerde, *Requirements engineering: from system goals to UML models to software specifications*: Wiley Publishing, 2009.

[29] B. Chen, X. Peng, Y. Yu, and W. Zhao, "Uncertainty handling in goal-driven self-optimization–limiting the negative effect on adaptation," *Journal of Systems and Software,* 2014.

[30] A. J. Ramirez, B. H. C. Cheng, N. Bencomo, and P. Sawyer, "Relaxing Claims: Coping with Uncertainty While Evaluating Assumptions at Run Time," in *Model Driven Engineering Languages and Systems: 15th International Conference, MODELS 2012*.

[31] M. Famelis, R. Salay, and M. Chechik, "Partial models: Towards modeling and reasoning with uncertainty," in *Software Engineering (ICSE), 2012 34th International Conference on*, 2012.

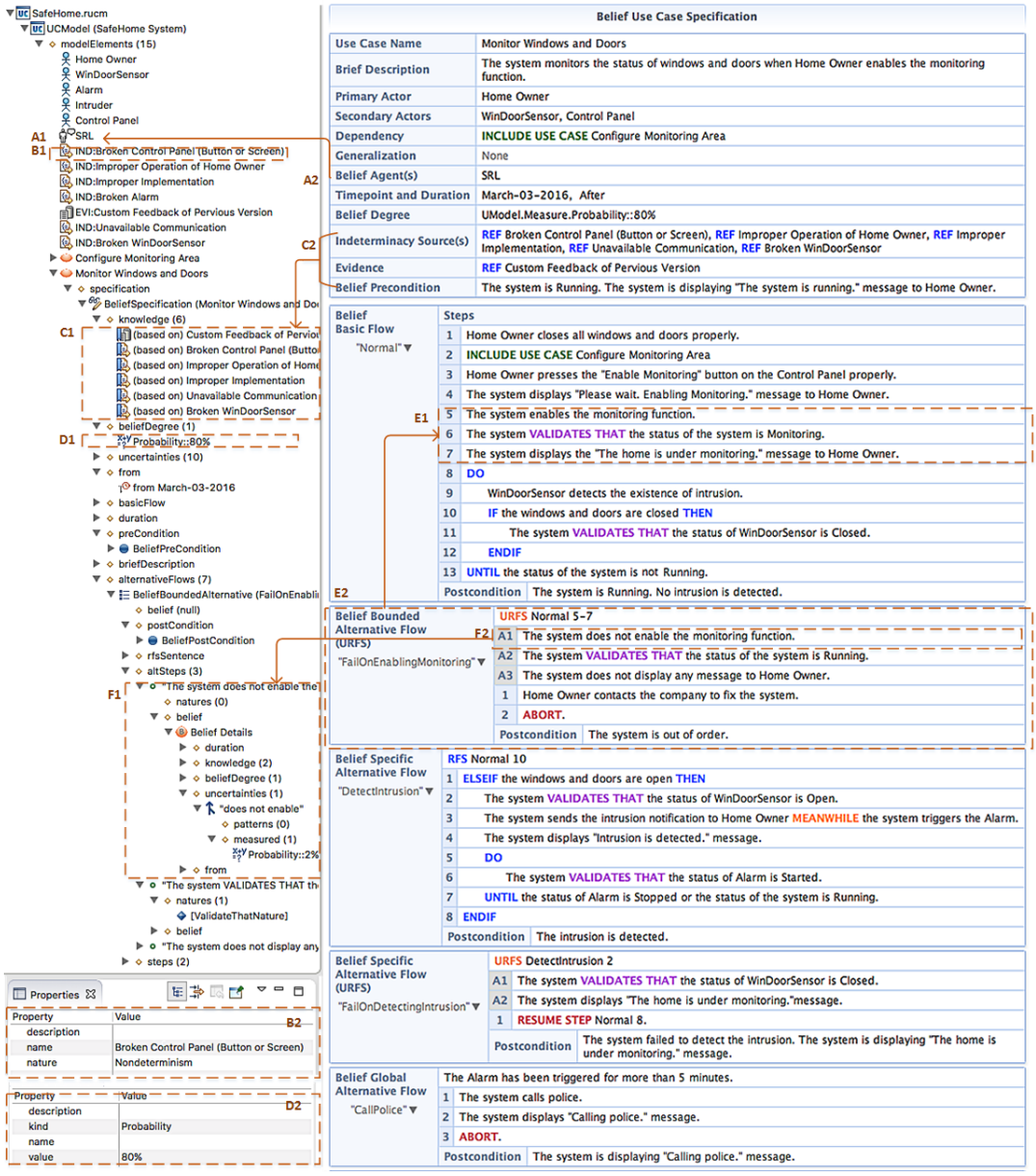
[32] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *Proceedings of the 36th International Conference on Software Engineering*, 2014.

[33] N. Esfahani, S. Malek, and K. Razavi, "GuideArch: guiding the exploration of architectural solution space under uncertainty," in *35th International Conference on Software Engineering (ICSE)*, 2013.

[34] Simula (2016). *U-RUCM: Specifying Uncertainty in Use Case Models*. Available: http://zen-tools.com/rucm/U_RUCM.html

[35] G. Zhang, T. Yue, J. Wu, and S. Ali, "Zen-RUCM: A Tool for Supporting a Comprehensive and Extensible Use Case Modeling Framework," in *Demos/Posters/StudentResearch@ MoDELS*, 2013.

APPENDIX A



**A2 Belief Agent* is formalized into elements shown in *A1*; the properties of *B1(D1) IndeterminacySource(Measurement)* is shown in the property window *B2 (D2)*; *C2* is the set of knowledge that are formalized as elements shown in *C1*; *E2* refers to a set of sentences indicated by *E1*; the alternative sentences indicated by *F2* are formalized into elements shown in *F1*.