Dealing with belief uncertainty in domain models

LOLA BURGUEÑO, Open University of Catalonia, Spain
PAULA MUÑOZ, ITIS Software, Universidad de Málaga, Spain
ROBERT CLARISÓ, Open University of Catalonia, Spain
JORDI CABOT, ICREA, Open University of Catalonia, Spain
SÉBASTIEN GÉRARD, CEA List, France
ANTONIO VALLECILLO, ITIS Software, Universidad de Málaga, Spain

There are numerous domains in which information systems need to deal with uncertain information. These uncertainties may originate from different reasons such as vagueness, imprecision, incompleteness or inconsistencies; and, in many cases, they cannot be neglected. In this paper, we are interested in representing and processing uncertain information in domain models, considering the stakeholders' beliefs (opinions). We show how to associate beliefs to model elements, and how to propagate and operate with their associated uncertainty so that domain experts can individually reason about their models enriched with their personal opinions. In addition, we address the challenge of combining the opinions of different domain experts on the same model elements, with the goal to come up with informed collective decisions. We provide different strategies and a methodology to optimally merge individual opinions.

CCS Concepts: • **Software and its engineering → Software notations and tools**; **Model-driven software engineering**; **System description languages**.

Additional Key Words and Phrases: Information systems, software, domain models, uncertainty, belief, belief fusion, consensus, subjective logic, vagueness, decision-making.

ACM Reference Format:
Lola Burgueño, Paula Muñoz, Robert Clarisó, Jordi Cabot, Sébastien Gérard, and Antonio Vallecillo. 2024. Dealing with belief uncertainty in domain models. 1, 1 (February 2024), 34 pages. https://doi.org/10.1145

1 INTRODUCTION

Information systems in domains such as archaeology, history, geography, biology, Internet of Things (IoT) or Artificial Intelligence (AI) have to deal with data that is sometimes missing, inaccurate, vague or even inconsistent due to unreliable sources, lack of knowledge, or imprecise information [12, 38]. In information systems, domain modeling is an important activity where the key concepts of a domain are captured. Therefore, the uncertainty that is intrinsic to a domain must be captured when experts build their models.

Current proposals for dealing with uncertain information in models tend to explicitly associate a *degree of uncertainty* to the affected instances, which can come from statistical analysis [5], trust on the sources [38], or even beliefs [6]. One particular type of degree of uncertainty can be captured by the so-called degree of confidence. Degrees of confidence are normally expressed either as a probability (also called *credence* [13]) or as a fuzzy quantifier (*certain*, *probable*, *possible*, etc.) [63].

In addition to representing uncertainty about the information, several proposals such as [1, 38] also enable reasoning about these enriched instance models (also called object models, or object

Authors' addresses: Lola Burgueño, lburguenoc@uoc.edu, Open University of Catalonia, Spain; Paula Muñoz, ITIS Software, Universidad de Málaga, Spain; Robert Clarisó, Open University of Catalonia, Spain; Jordi Cabot, ICREA, , Open University of Catalonia, Spain, jordi.cabot@icrea.cat; Sébastien Gérard, CEA List, France; Antonio Vallecillo, av@lcc.uma.es, ITIS Software, Universidad de Málaga, Spain.

2024. XXXX-XXXX/2024/2-ART $15.00
https://doi.org/10.1145

diagrams in UML [44] terminology) — see also [63] for a survey of works that explicitly represent uncertainty in software models.

However, these proposals also present some shortcomings. In particular, they only permit a single degree of uncertainty to be expressed on a model element (also called instance), which does not allow separate users or statistical inference systems to assign different degrees of uncertainty to the same instances. Most importantly, they do not allow the different degrees of uncertainty coming from different sources to be combined. This is desirable, for instance, in cases in which various experts (users, stakeholders, modelers, etc.) express their beliefs on model elements and need to merge them in order to reach agreements, or at least compromises, about the final instance model.

This proposal aims at addressing these issues. First, we use Subjective logic [28] to express the opinions of the domain experts. Each opinion is composed by four values that are defined on an instance model element. These are: (1) the prior probability of the opinion (which is an objective uncertainty – also called initial degree of uncertainty); (2) the degree of *belief*; (3) the degree of *disbelief*; and (4) the degree of *uncertainty* (in this case, this is a subjective uncertainty that reflects up to what extent the domain expert is unsure and does not have any belief or disbelief). All these degrees are real numbers in the range [0..1]).[1] Second, we show how more than one opinion can be assigned to one element without altering the domain model definition. To accomplish this task, in this paper, we have used the UML notation [44]. We represent domain models by means of class diagrams, and have defined a UML profile to enrich instance model elements with the opinions held by the domain experts. We have also created a set of operations to deal with opinions (e.g., propagate their associated uncertainties through operations such as `and`, `or` or `xor`). Third, we use the Subjective logic fusion operators to combine the opinions from different domain experts into a consensus agreement, or at least into a single compromise opinion. Our proposal has been implemented as a MagicDraw plugin.[2] We also present the formal semantics of our proposed UML profile in OCL [43].

We demonstrate our approach with several applications from different domains, which have served us to both compare our proposal with related works and to validate it.

This paper is structured as follows. Next, Sect. 2 presents the context of our work, introduces an illustrative example to motivate our proposal and states our research questions. Sect. 3 introduces the background of our work. Then, Sect. 4 presents our proposal, including the UML profile and its formalization in OCL. Section 5 describes the methodology we propose to use our approach, i.e., the process and guidelines that should be followed to capture and represent beliefs in domain models and to reason and operate with them. In Sect. 6, we describe the tool that we have developed to support our approach and, in Sect. 7, we validate it through different application scenarios and an empirical study with users. Finally, Sect. 8 relates our work to similar approaches and Sect. 9 concludes with an outline of future work.

2 MOTIVATION AND RESEARCH QUESTIONS

2.1 A motivating example

To illustrate the problem and introduce our proposal, we will use an example of a system whose users receive information from sensors and other external data sources that are not completely reliable or may contain inaccuracies, and therefore have an associated uncertainty.

Let us suppose a smart house whose rooms are equipped with sensors that measure their temperature and humidity. According to their manufacturers' information, the accuracy of the temperature

[1] We explain all the concepts in more detail in Section 2 with an example.

[2] https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/

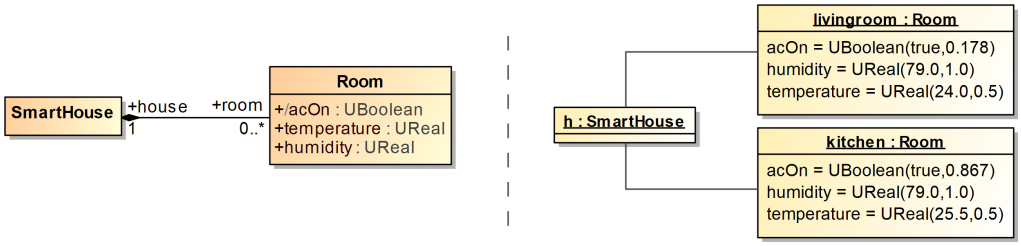


Fig. 1. A class diagram of the Smart House system (left) and an object diagram showing one house and two of its rooms (right).

and humidity sensors are ±0.5 degrees and ±1.0%, respectively. Note that this uncertainty is objective and does not come from personal (and, therefore, subjective) impressions or experiences. A control device in the room decides whether the Air Conditioning (AC) system should be turned on or off depending on the values of the sensors' measurements. If the temperature is higher than 25 degrees or the humidity in the room is higher than 80%, the air conditioning will be automatically turned on. If this condition is not met, it should be off.

The diagrams in Fig. 1 show the domain model of the system as a UML class diagram (left), and an object model that describes a house and two of its rooms (right). The condition that establishes when the air conditioning is turned on or off is given by the following OCL expression:

```
acOn : UBoolean derive = self.temperature >= 25.0 or self.humidity >= 80.0
```
(1)

Note the use of the UML extended datatypes `UReal` and `UBoolean` that represent Real and Boolean values enriched with uncertainty [3]. Type `UReal` extends type `Real` with measurement uncertainty, enabling users to represent and operate with values like 3.0 ± 0.01, which is simply written as `UReal(3.0,0.01)`. Type `UBoolean` implements the probabilistic logic extension to Boolean logic that uses Bayesian probabilities to represent the likelihood of a value to be true. A `UBoolean` value is a pair (b, c) where b is a Boolean value (i.e., *true*, *false*) and c is a real number in the range [0..1], representing the confidence that b is certain by means of a probability.

The extended type system takes care of propagating the corresponding uncertainty through the OCL expressions. In the particular case of the values of attribute `acOn`, the uncertainties are propagated through the operators `>=` and `or`. Assuming a threshold of 0.5, where a `UBoolean(true,c)` with `c>=0.5` corresponds to the Boolean *true* and `c<0.5` corresponds to the Boolean *false*, in view of the object model shown in Fig. 1, the AC system of the `kitchen` will be switched on while that of the `livingroom` will be turned off. Once again, note that this derived uncertainty also has an objective nature since it is based on facts and derived from them.

Unlike this scenario, there are many situations in which the users of such systems also associate a subjective uncertainty that coexists with the objective uncertainty of the sources. For example, let us assume that one of the house occupants, Bob, does not trust the temperature readings of the kitchen because he knows that the sensor is close to the oven and therefore the measurements are occasionally unreliable. Bob would add some subjective uncertainty to the objective results of the sensor's measurements. This type of subjective uncertainty assigned to the same sensor by different users can vary, depending on their personal history, experiences and convictions—e.g., their individual level of trust in the sensor manufacturer. Thus, other occupants may have different opinions, or trust, on the sensors' readings which will consequently affect their decisions on whether to switch on or off the AC system of a room. This type of uncertainty is typically called *second-order probability* or *second-order uncertainty* in the literature of statistics and economics,

and needs to be captured, explicitly represented, propagated, and taken into account in order to make informed decisions about the system and the actions to take.

To further complicate matters, in most systems, users do not exist isolated, but exchange information and interoperate with each other to achieve their goals. In our example, if Ada, Bob, and Cam are occupants of the house and they hold different beliefs about the reliability of the sensors readings, how can they reach a consensus about when to turn the AC system on or off?

2.2 Research Questions

To address these issues in an orderly and systematic manner, this paper aims at answering the following research questions:

RQ1 How can stakeholders capture and explicitly represent their opinions about the instances of domain models?

RQ2 How can stakeholders reason about the opinions specified in the instance models and make informed decisions?

These questions will be answered in sections 4.1, 4.2 and 4.3 after some necessary background is first introduced.

3 BACKGROUND

3.1 Subjective logic

A particular type of subjective uncertainty, called *Belief Uncertainty* [45, 63], occurs when a user is not sure about the truth of a statement, i.e., a Boolean predicate. Belief uncertainty is normally expressed by assigning a *degree of belief* (or *confidence*) that represents how sure the user is about the truth of the statement. Several extensions to the Boolean logic enable dealing with Belief uncertainty, including probability theory [14, 19], possibility theory (based on fuzzy logic [52, 70]), plausibility (a measure in the Dempster-Shafer theory of evidence [56]), and uncertainty theory [33]. These proposals assign different probabilities to propositions, rather than truth values, and probability formulas replace truth tables.

However, these probabilistic extensions to binary logic present some limitations. In particular, users are able to express their degrees of belief using it, but they cannot express ignorance, i.e., when the modeler is uncertain about the probability that she has to assign to a logic predicate or to a Boolean attribute. For example, when the user has total ignorance about some statement x, it might be preferable to say "I don't know" than assigning x a confidence of 0.5. A probability of 0.5 would mean that x and $not(x)$ are equally likely, which does not represent ignorance since it is already quite informative [28]. In general, forcing users to set probabilities to express their opinions could lead to unreliable conclusions [40]. This is when Subjective logic comes into play.

Subjective logic, by Audun Jøsang [27, 28], is a type of probabilistic logic that explicitly takes uncertainty into account. Subjective opinions[3] express beliefs and disbeliefs about the truth of propositions under degrees of uncertainty.

Let x be a Boolean predicate. A binomial *opinion* about the truth of x is defined as a quadruple $\omega_x = (b_x, d_x, u_x, a_x)$ where:

- b_x (*belief*) is the degree of belief that x is true.
- d_x (*disbelief*) is the degree of belief that x is false.
- u_x (*uncertainty*) is the degree of uncertainty about x, i.e., the amount of uncommitted belief.
- a_x (*base rate*) is the prior probability of x (i.e., the objective probability).

[3]It may seem that the term *subjective opinion* is redundant as the word *opinion* already implies subjectivity, but opinions expressed with Subjective logic are called as such in the literature. The authors of this paper internally discussed the terminology and decided to stick to the literature.

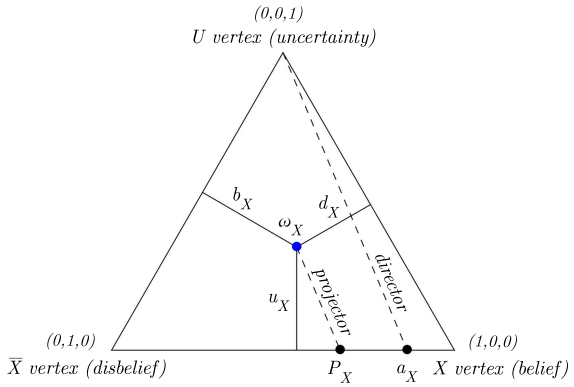


Fig. 2. Graphical representation of a binomial opinion using Barycentric coordinates [28].

These values satisfy $b_x + d_x + u_x = 1$, and $b_x, d_x, u_x, a_x \in [0, 1]$. Opinions where $b_x = 1$ or $d_x = 1$ are called *absolute* opinions, and are equivalent to the Boolean values *true* and *false*, respectively. An opinion where $b_x + d_x = 1$ (hence $u_x = 0$) is a *dogmatic* opinion which is equivalent to a traditional probability. If $b_x + d_x < 1$, we have an *uncertain* opinion which expresses a degree of uncertainty. Finally, if $b_x + d_x = 0$ (i.e., $u_x = 1$) we have a *vacuous* opinion that expresses total uncertainty.

Intuitively, the *base rate* of an opinion represents the *objective* probability that can be assigned to the statement using *a priori* evidences or statistical estimates, whilst the other elements of the tuple represent the *subjective* degrees of belief, disbelief and uncertainty about the statement assigned by the user/expert (whom we will call *belief agent*, or simply *agent* for short). Thus, regardless of the value of the prior probability, different agents can express their subjective opinions about the statement, including their degree of uncertainty.

Opinions can be represented on an equilateral triangle using Barycentric coordinates as shown in Fig. 2 [28]. A point inside the triangle represents a (b_x, d_x, u_x) triple. Vertices at the bottom represent absolute opinions, and the vertex at the top represents the vacuous opinion ($u_x = 1$). Dogmatic opinions belong to the base line ($u_x = 0$), and correspond to probabilities. The base rate a_x, or prior probability, is shown along the base line, too.

The *projected probability* (or *projection*) of an opinion is defined as $P_x = b_x + a_x u_x$. Graphically, it is formed by projecting the opinion ω_x onto the base, parallel to the base rate director line. The projected probability permits combining the objective and subjective values into one single probability, which modifies the prior base rate according to the opinion of the agent.

Note that the benefits of Subjective Logic as opposed to Kleene logic is that Kleene logic does not include base rates and therefore probability projections cannot be derived.

In addition to the traditional logical operators (`and`, `or`, `implies`, etc.) used to combine the opinions of the same agent about different truth statements, Subjective logic implements *fusion* operators for combining the subjective opinions of different agents about the same statement. The goal of these fusion operators is to produce a single opinion that better reflects the collection of opinions, or is closer to the truth than each opinion in isolation. This is of paramount importance for permitting collaborative modeling and enabling cooperative work between the agents that need to reach agreements on how to proceed, such as in fleets of unmanned vehicles or in flocks of drones.

Each fusion operator is suitable or not depending on the particular situation, i.e., if the agents are willing to compromise or they will not change their minds, or if we want to give more relevance to the most confident opinions, i.e., those with less uncertainty mass. The fusion operators that we have implemented in our proposal are described in Sect. 4.3.

To represent and operate with Subjective logic values, in [40], we defined the new type `SBoolean` that extends type `UBoolean`. A `SBoolean` value is defined by the quadruple (b, d, u, a) that represents the corresponding opinion in Subjective logic. The embedding of a `UBoolean` value $x = (true, c)$ into type `SBoolean` is achieved by assigning the opinion $w_x = (c, 1 - c, 0, c)$ to x. Considering the embedding of type `Boolean` into `SBoolean`, we have that Boolean values `true` and `false` correspond, respectively, to the `SBooleans` $(1, 0, 0, 1)$ and $(0, 1, 0, 0)$.

Examples of the use and application of Subjective logic in models represented with UML/OCL can be found in [40] and Section 7. In addition, in this paper we have extended type `SBoolean` by adding the implementation of the fusion operators, as described in Sect. 4.3.

3.2 Extending UML: UML Profiles

The existing modeling proposals that enable the representation of uncertain information are divided between those that use domain-specific languages (DSL), e.g., ConML [37], and those that use general-purpose modelling languages such as UML or its extensions, e.g., Fuzzy UML [24, 58].[4]

In general, the definition of domain-specific languages provide users with compact and specialized notations very close to the languages used by the domain experts [20]. Although initially they suffered from the lack of associated tools, current Software Language Engineering practices have enabled the easy development of editors, model simulators and other analysis tools that make their adoption in industry feasible and practically achievable [62, 67]. However, they still suffer from two main drawbacks. First, each DSL requires its own dedicated knowledge and skills, so users are forced to get familiar with a varied plethora of notations, editors, supporting tools, etc. Secondly, most of them live in silos with limited interoperability capabilities, including model exchange and cross-tool analyses between them.

In contrast, general purpose modeling languages such as UML provide generic modeling notations able to represent a wide rage of systems, addressing in this way some of the aforementioned drawbacks of DSLs. Furthermore, UML was defined with an extension mechanism that enables users to create specializations of the language, by means of UML Profiles [44]. Profiles allow extending the UML metamodel with new elements, and do so in a conservative and compatible manner, i.e., without breaking the semantics of existing UML constructs. Even though UML profiles may not provide such an elegant, compact and tailored notations as specialized DSLs, they can be handled by commercial UML tools and interoperate with other UML models.

UML profiles are defined in terms of two basic mechanisms: stereotypes and tagged values. *Stereotypes* define the required extensions to the corresponding UML constructs (i.e., metaclasses). For example, we can extend the elements defined in a domain model with additional properties, such as the (un)certainty of their values. A *tagged value* is an additional meta-attribute that is attached to a metaclass of the domain model extended by a profile. Tagged values have a name and a type, and are associated to a specific stereotype. For example, if we define an stereotype named «`UncertainElement`», we may have an associated tagged value named `beliefs`, that represents the individual beliefs of several agents about the value of the attribute. Graphically, tagged values are specified as attributes of the stereotype.

In this paper, we define a UML profile for representing degrees of belief about those instance model elements that users are uncertain about. This way, any UML object model can be enriched with this type of information without altering the class diagram (domain model). In addition, we will show how this profile is formalized and how the belief uncertainties are propagated through the operations using the standard OCL language.

[4] See [63] for a survey of the existing proposals for representing uncertainty in software models, including belief uncertainty.

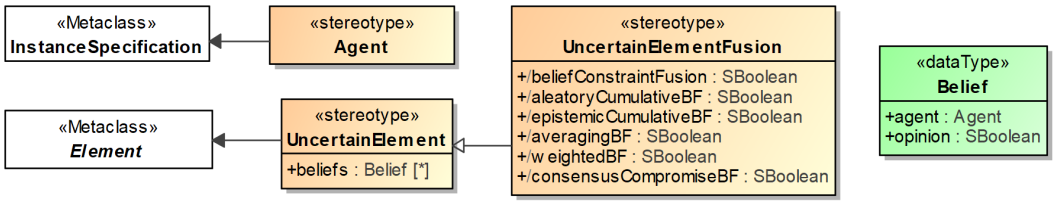


Fig. 3. The Belief Uncertainty UML Profile for Subjective Opinions.

4 EXPRESSING SUBJECTIVE STATEMENTS IN DOMAIN MODELS

This section describes our proposal, and how to represent and operate with belief uncertainty expressed with subjective opinions in domain models. First, Sect. 4.1 describes the Belief uncertainty UML profile we have defined to assign subjective opinions to elements in UML. Then, Sect. 4.2 shows the formalization of our profile using OCL. Finally, Sect. 4.3 discusses the fusion operators we implemented to merge individual opinions from different agents and, this way, allow them to reach agreements.

4.1 The Belief Uncertainty UML Profile

To enable users to express their opinions about particular model elements we have created the UML profile that Fig. 3 shows. Let us explain its elements.

First, the stereotype «Agent» is used to indicate those entities that hold opinions, which can be domain experts, users, modelers or any other stakeholder. The stereotype «UncertainElement» is used to mark any element of an object model as uncertain, and enables the assignment of degrees of belief to it. For this, the datatype Belief represents a pair (agent:Agent, opinion:SBoolean) that describes the agent holding an opinion, and the opinion held by that agent. A set of beliefs can be associated to an uncertain element, which allows different agents to express their opinions about the same model element. The stereotype «UncertainElementFusion» extends «UncertainElement» by adding a set of derived tags. Each one computes the corresponding fusion operator on the list of opinions stored in the tag beliefs of the «UncertainElement» stereotype (see Sect. 4.3).

This profile is used to assign degrees of belief to instances, to links and to attributes' values. In the former case, the degree of belief expresses the opinions of the agents about the actual occurrence (or existence) of the instance [4]. The same holds for links (i.e., association instances), which express the agents' degrees of belief about the actual existence of the relationship that the link represents in the system. When applied to attributes' values, an opinion must be interpreted as the opinion that the agent has about the truthiness of the value of the attribute (not about whether the attribute should exist or not). In the case of derived attributes, whose values are automatically computed by OCL expressions, their opinions will be automatically derived by propagating the agents' opinions on the operands through the expression's operators. Thus, agents do not need to explicitly assign opinions to derived attributes.

When an opinion is assigned to a link, the existence of the two related objects is assumed. This means that an opinion on a link represents a *conditional* opinion (in Subjective logic terms) hence the independence between the opinions about the objects and about the existence of the link between them is achieved. This simplifies the interpretation of the object model and also the calculations when propagating the belief uncertainty, since the opinion on the links and those on the related objects are *independent*, and thus they can be simply combined with "and" operators.

Figure 4 illustrates the use of the UML profile in the case of the Smart House system described in Sect. 2.1. This object model shows a room, bedroom1, annotated with the opinions of three occupants

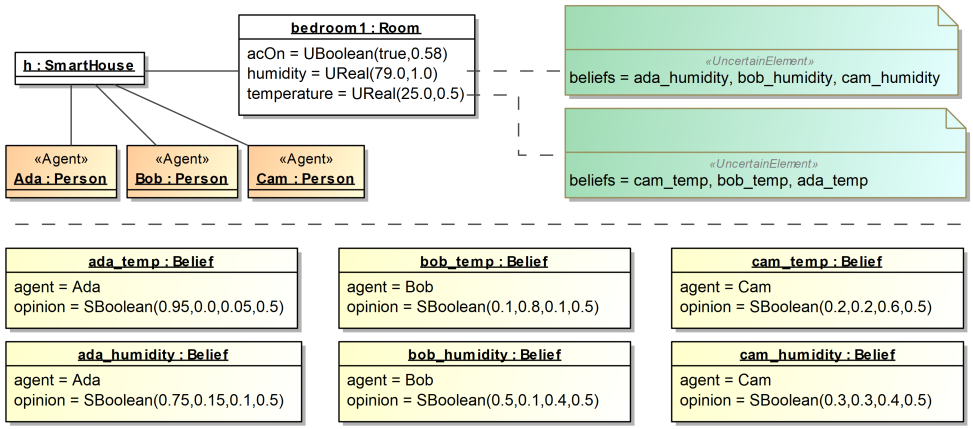


Fig. 4. The Smart House object model of bedroom1, annotated with belief uncertainty.

of the house, who hold different opinions on the `temperature` and `humidity` values of this object. Namely, Ada is quite confident in her opinions, Bob does not believe in the temperature sensor and partially trust on the humidity readings, and Cam is not sure about any of the sensors. Note that all of them use the same base rate. This is because we are giving an opinion about the correctness of the value. Since we do not have any previous evidence, it is equally probable for it to be wrong or right. The fact that agents hold opinions on the operands of the `acOn` expression imply a derived opinion on the value of the derived attribute `acOn`, which is calculated by propagating the opinions through the expression.

Although this object model allows agents expressing opinions, we still need to specify how these opinions are propagated through the OCL expressions defined in the model, in particular, in the expression that computes the derived attribute `acOn` to enable the agents to make decisions.

4.2 Semantics of the Belief Uncertainty UML profile in OCL

This section describes the semantics of the UML profile in OCL. We do this by representing in plain UML the elements of the profile and by specifying in OCL their behavior. While the use of profiles enables users to enrich UML models without altering the original models, these are not available in plain UML. As such, this representation of the profile in plain UML enables as a side-effect, the use of our approach on tools that do no have UML profiling capabilities. Therefore, in this section, we: *i)* explain how the original UML class diagrams are extended to store opinions; and *ii)* provide a high-level specification that describes in detail the expected behavior of the propagation of the subjective opinions' through the association links, methods and logical expressions.

Our starting point is the extension of the basic UML and OCL datatypes with uncertainty, in particular the `UBoolean` [3] and `SBoolean` [40] datatypes. In previous works, we introduced and implemented these datatypes in a Java library and we used it to extend the USE tool [22], see [40, 46].

The strategy to explicitly assign opinions to the object model elements will depend on the type of element (instance object, attribute value or link). We will use a systematic approach that enables the automatic generation of the corresponding UML/OCL specification from any input UML instance model annotated with the belief uncertainty profile described in Sect. 4.1.

First, the stereotype «`Agent`» will be represented by an abstract class with the same name, from which all entities that hold beliefs will inherit.

Second, the datatype `Belief` will be represented by a `Tuple(agent:Agent,opinion:SBoolean)` that contains its possible values.

Finally, for each attribute <*X*> stereotyped as «`UncertainElement`» we need to create an additional attribute `_<X>Beliefs:Sequence(Tuple(agent:Agent,opinion:SBoolean))` to store the corresponding tag value `beliefs` defined in the profile.

In our running example, we are interested in assigning opinions to the temperature and humidity sensors of a room. This can be implemented by adding the following two attributes to the class `Room`:

```
_humidityBeliefs:Sequence(Tuple(agent:Agent,opinion:SBoolean)) init:Sequence{}
_temperatureBeliefs:Sequence(Tuple(agent:Agent,opinion:SBoolean)) init:Sequence{}
```

To enable the assignment of the stereotype «`UncertainElement`» to model instances, an attribute `_beliefs:Sequence(Tuple(agent:Agent,opinion:SBoolean))` will be added to the corresponding class. This attribute will store the tag `beliefs` defined in the profile.

To assign opinions to links between objects which are instances of an association *R*, stereotyped as «`UncertainElement`», we reify the association *R* transforming it into an association class, and then a new attribute `_beliefs:Sequence(Tuple(agent:Agent,opinion:SBoolean))` will be added to the association class.

Operation `opinionIn(beliefs:Sequence(Tuple(agent:Agent,opinion:SBoolean)):SBoolean` defined in class `Agent` returns the opinion of the agent in the sequence. In case the agent has not expressed any opinion in that sequence, this query operation returns the value by default, which is the dogmatic *true*, i.e., `SBoolean(1,0,0,1)`..

```
opinionIn(beliefs:Sequence(Tuple(agent:Agent,opinion:SBoolean))):SBoolean =
  if beliefs->select(t|t.agent=self)->isEmpty()
  then SBoolean(1,0,0,1) -- true
  else beliefs->select(t|t.agent=self)->collect(t|t.opinion)->last()
  endif
```

Opinions on derived attributes will be automatically computed by propagating the opinions of the agents on their operands through the operations in the OCL derivation expression.[5] For example, the derived opinion of the agents about the `acOn` attribute, whose value was computed using formula (1), can be calculated as follows:

```
_acOnBeliefs:Sequence(Tuple(agent:Agent,opinion:SBoolean)) derive =
   self.house.occupants->
     iterate(a ; acc:Sequence(Tuple(agent:Agent,opinion:SBoolean))=Sequence{}|
      acc->append(Tuple{agent=a,opinion=
        a.opinionIn(_temperatureBeliefs).applyOn(self.temperature >= 25.0)
        or a.opinionIn(_humidityBeliefs).applyOn(self.humidity >= 80.0)
      }))
```

This expression iterates over the set of house occupants, and for each one it adds their derived opinion. The opinion of each agent is computed by applying their opinion on the atomic Boolean expressions of formula (1), and combining them with the `or` operator that is used in the formula.

The operation `applyOn()` of type `SBoolean` applies a subjective opinion on an `UBoolean` value, producing another subjective opinion whose base rate is the confidence of the argument. This way, we adjust the prior probability of the initial subjective opinion to the actual confidence of the operand of the OCL derivation expression. The resulting opinion will have the same uncertainty as the original opinion, still maintaining the ratio b_x/a_x.

[5]Note that such an opinion refers to the resulting value of the attribute. This is, it is not an opinion on the derivation formula used to derive the attribute since we are not dealing with *design uncertainty* [63] in this work.

Thus, propagating the opinions of the three agents about the temperature and humidity of `bedroom1` (see Fig. 4) through the derivation expression that computes the value of that attribute, defined by equation (1), produces the results shown below. Note that the base rates of all the opinions is the same (0.579) because we assume the three agents use the calculated probability that the AC system needs to be switched on as common evidence for their opinions. Such prior probability corresponds to the probability calculated using the objective information available about the sensors and the current values of the room attributes used in the derivation formula.

```
_acOnBeliefs = Sequence{
     Tuple{agent=Ada,opinion=SBoolean(0.164, 0.662, 0.174, 0.579)},
     Tuple{agent=Bob,opinion=SBoolean(0.171, 0.286, 0.543, 0.579)},
     Tuple{agent=Cam,opinion=SBoolean(0.541, 0.408, 0.051, 0.579)}
 }
```

The projections of the individual opinions are, respectively, 0.265, 0.485, and 0.571.

4.3 Belief Fusion Operators

As part of this work, we have extended the type `SBoolean` with the fusion operators described in [28, 29, 66], which were not present in our previous implementation [40]. These fusion operators are used to combine the subjective opinions of different belief agents about the same statement, with the goal of producing a single opinion that represents a consensus, or represents the truth more faithfully than each individual opinion. Each fusion operator is designed for a specific purpose and scenario. Depending on the situation, the agents (or the person in charge of merging the opinions) need to decide which fusion operator is the most suitable for each particular case.

Let us briefly describe next how they work, and illustrate their behavior by merging three opinions, w_x, w_y and w_z. The merged opinion will be denoted $w = (b, d, u, a)$. The situations in which they are applicable are also discussed.

- The **Belief Constraint Fusion (BCF)** is used when the agents have committed their choices and will not change their minds, at the potential cost of not being able to reach a consensus. Roughly, it computes the overlapping beliefs (the relative *Harmony*) and the non-overlapping beliefs (the relative *Conflict*) of the opinions to fuse. More precisely, in the case of two opinions, $Har = b_x b_y + b_x u_y + b_y u_x$ and $Con = b_x d_y + b_y d_x$. Then, in the general case the components of the fused opinion are defined as follows:
  $b = Har/(1 - Con)$ $\quad$ $d = 1 - b - u - a$
  $u = u_x u_y/(1 - Con)$ $\quad$ $a = (a_x(1 - u_x) + a_y(1 - u_y)/(2 - u_x - u_y)$
  The divisor $(1 - Con)$ normalises the belief mass and uncertainty, and ensures additivity. The constraint fusion operator is commutative and non-idempotent. Associativity is preserved when the base rate is equal for all agents. The result is undefined in case of totally conflicting opinions.
- The **Consensus & Compromise Fusion (CCF)** preserves the shared beliefs from the sources and transforms conflicting opinions into vague belief. It is suitable for situations in which consensus is sought if it exists, and a vague opinion is acceptable if not. This operator assumes *dependence* between opinions[6] and it is *idempotent* with the vacuous opinion as neutral element. It uses a three-steps process: 1) consensus; 2) compromise; 3) merge. The consensus step determines the shared beliefs and disbeliefs between the two opinions. The compromise step redistributes conflicting residual beliefs and disbeliefs to produce the so-called *compromise* belief. Finally, the merge step distributes that compromise between

[6]Opinions are said to be *dependent* when they are based on the same or dependent evidence; and they are said to be *independent* when they are based on different or independent evidence.

the shared belief and the uncertainty to compute the resulting opinion. This operator is applicable in situations where (possibly non-expert) agents have dependent opinions about the same fact, such as when they are asked to give their opinions in a survey [28].

- The **Averaging Belief Fusion (ABF)** operator assumes that the agents' opinions are *dependent*, and the operator is commutative, idempotent, and non-associative. Not having a neutral element means that every opinion, even a vacuous one, influences the fused result. This is the case, for instance, when all agents observe the same situation at the same time, and all opinions should be taken into account. For example, when a jury tries to reach a verdict after having observed the court proceedings. Basically, this operator computes the average of the opinions by weighing the belief of each opinion with the product of the uncertainty of the rest. In the case of three non-dogmatic opinions, it is expressed as follows:

  $b = (b_x u_y u_z + b_y u_x u_z + b_z u_x u_y)/(u_x u_y + u_x u_z + u_y u_z) \qquad d = 1 - b - u - a$
  $u = 3u_x u_y u_z/(u_x u_y + u_x u_z + u_y u_z) \qquad a = (a_x + a_y + a_z)/3$

  If dogmatic opinions are present, this operator computes the average of the beliefs and disbeliefs of these opinions only, resulting in a dogmatic opinion too [29].
- The **Weighted Belief Fusion (WBF)** operator works as ABF but giving more weight to those opinions with less uncertainty, i.e., the smaller the value of the uncertainty mass, the higher the weight—unlike ABF where all opinions have the same weight even when some of them are very uncertain. This operator has the vacuous opinion as neutral element and, like ABF, it is *idempotent*. To illustrate how WBF works, the following expression computes the WBF of three non-dogmatic opinions[66]:

  $b = (b_x(1-u_x)u_y u_z + b_y(1-u_y)u_x u_z + b_z(1-u_z)u_x u_y/(u_x + u_y + u_z - 3u_x u_y u_z)$
  $d = 1 - b - u - a$
  $u = (3 - u_x - u_y - u_z)u_x u_y u_z/(u_x + u_y + u_z - 3u_x u_y u_z)$
  $a = (a_x(1-u_x) + a_y(1-u_y) + a_z(1-u_z)/(3 - u_x - u_y - u_z)$
- The **Aleatory** and **Epistemic Cumulative Belief Fusion** operators (**ACBF** and **ECBF**, resp.) can be applied when the evidences are *independent*, i.e., the amount of evidence increases when more agents give their opinion. The Aleatory operator (ACBF) is suitable when giving opinions about a variable governed by a frequentist process, such as flipping a coin. In turn, the ECBF is used for facts whose uncertainty is of epistemic nature. For example, when witnesses express their opinions as to whether they saw the accused at the scene of the crime, which when merged produces an opinion as to whether the accused was actually there. When the ECBF operator finds contradictory opinions, it increases the uncertainty of the results. Like the ABF, the ACBF operator calculates the mean of the opinions by weighting the belief of each opinion with the product of the uncertainty of the others. However, it modulates this mean by subtracting the product of the uncertainties. Thus, in the case of three non-dogmatic opinions, their aleatory cumulative fusion can be expressed as follows:

  $b = (b_x u_y u_z + b_y u_x u_z + b_z u_x u_y)/(u_x u_y + u_x u_z + u_y u_z - 2u_x u_y u_z) \qquad d = 1 - b - u - a$
  $u = u_x u_y u_z/(u_x u_y + u_x u_z + u_y u_z - 2u_x u_y u_z) \qquad a = (a_x + a_y + a_z)/3$

  Again, if dogmatic opinions are present, this operator computes the cumulative fusion of these opinions only, resulting in a dogmatic opinion too [29]. ECBF is an extension of the ACBF operator that produces the same opinion but with maximized uncertainty.[7]

The complete descriptions of their behavior can be found in [28], with some extensions in [29, 66]. Their full implementation in Java is available in [8]. This implementation is the one used in the UML profile for computing fused opinions, see Sect. 6.

[7]The *uncertainty-maximized* opinion $\ddot{w}$ of an opinion w is the opinion with maximum uncertainty that still preserves the same projected probability as w [28].

	Belief Constraint Fusion (BCF)	Cumulative Belief Fusion ([A&E]CBF)	Averaging Belief Fusion (ABF)	Weighted Belief Fusion (WBF)	Consensus & Compromise Fusion (CCF)
Agents' willingness to compromise	–	✓	✓	✓	✓
Dependence between opinions	✓	–	✓	✓	✓
Vacuous opinion is neutral element	✓	✓	–	✓	✓
Preserve shared beliefs; conflicting opinions turned into vague beliefs	–	–	–	–	✓

Table 1. Fusion operators' properties.

Different opinions can be fused in various ways, each of which reflects how the specific fusion situation needs to be handled. In general, determining the correct fusion operator best suited for a specific situation is challenging. To address this issue, we have identified the following characteristics, which may be helpful when determining the operator to use in each case. These are summarized in Table 1:

- **Willingness to compromise**. The agents agree to find a compromise even in case of totally conflicting opinions.
- **Assumed dependence between opinions**. The agents formed their opinions witnessing the same events at the same time.
- **Vacuous opinion as neutral element (idempotence)**. Fusion of vacuous opinions (i.e., those with uncertainty mass $u_x = 1$) have no effect in the result.
- **Preserve shared beliefs, and conflicting opinions are turned into vague belief**. In order to find a compromise, it is possible to turn conflicting opinions into vague beliefs.

For illustrative purposes, Table 2 shows the result of the application of the different fusion operators on the opinions held by the agents of our running example about the value of attribute `acOn` of object `bedroom1`. Each row shows the belief fusion operator applied to these values, the resulting opinion, and its projected probability. Depending on the fusion operator used, the merged decision about whether the AC system of that room should be switched on or off changes.

Although we have shown in Table 2 the result of all fusion operators for illustrative purposes, note that only CCF and WBF apply to this particular situation. This is the reasoning behind:

- The Belief Constraint Fusion (BCF) cannot be used because a decision must be made about the AC system, which cannot be on for some agents and off for others, so a compromise must be reached in case of conflicting opinions.
- The Cumulative Belief Fusion operators (A-CBF and E-CBF) are not suitable because the opinions are dependent, as all agents observe the same situation at the same time.
- The Averaging Belief Fusion (ABF) operator is also unsuitable because, even if the opinions of the agents are dependent, in our situation vacuous opinions must be discarded as they should have no effect on the final decision.
- The Weighted Belief Fusion (WBF) operator can be used because the opinions are dependent and it may make sense to give more weight to more confident opinions.
- Finally, the Consensus & Compromise fusion (CCF) is applicable because agents must be willing to give in on their opinions to reach agreements.

In summary, the fusion operators that best fit our case are CCF and WBF. Both present a small degree of uncertainty and have a projection above 0.5, which means that the AC system should be switched on (merging the disparate opinions of Ada, Bob and Cam).

Fusion operator	Fused opinion	Projection
Belief Constraint Fusion (BCF)	`SBoolean(0.303, 0.687, 0.010, 0.579)`	`0.309`
Consensus & Compromise Fusion (CCF)	`SBoolean(0.494, 0.489, 0.017, 0.579)`	`0.503`
Aleatory Cumulative Belief Fusion (ACBF)	`SBoolean(0.470, 0.490, 0.040, 0.579)`	`0.493`
Epistemic Cumulative Belief Fusion (ECBF)	`SBoolean(0.000, 0.149, 0.852, 0.579)`	`0.493`
Average Belief Fusion (ABF)	`SBoolean(0.435, 0.454, 0.111, 0.579)`	`0.499`
Weighted Belief Fusion (WBF)	`SBoolean(0.453, 0.454, 0.093, 0.579)`	`0.507`

Table 2. Results of applying the fusion operators on the opinions about the value of slot `acOn` of `bedroom1`.

5 METHODOLOGY

5.1 Guidelines to use our approach

Although any methodology could be used to realize our approach, we devised it within a collaborative and iterative process, with the following steps in mind.

(1) Each agent involved in the design of a domain starts instantiating the domain model. This instance model that represents the system under study might not be the same for all agents: each agent may have independently developed a different model according to the information they have, their sources, and their individual perceptions.
(2) Using the Belief profile defined in Sect. 4.1, agents annotate those instances and links for which they are uncertain about their existence as well as those attributes for which they are uncertain about their actual values. Annotations are expressed in terms of *beliefs*. Model elements with no assigned beliefs are assumed to be dogmatic trues, i.e, `SBoolean(1,0,0,1)`.
(3) The belief uncertainty information is automatically propagated to the derived properties and operations of the model.
(4) Once the agents have their individual instantiations of the domain model enriched with their subjective opinions, they can decide to merge these object models. This process builds a model with the union of all elements and links, and their annotations. Unless otherwise agreed by the agents, by convention, those elements present in the object model of one agent but not in the object model of others will appear in the merged object model, but they will be stereotyped as «UncertainElement» and will have one opinion with value `SBoolean(0,1,0,0)` (i.e., the dogmatic *false*) by each one of agents that did not have it in their individual instance.
(5) After observing each other's opinions in the merged model, the belief agents can reconsider their opinions. This can lead to a repetition of step (2) of this process.
(6) When the combined object model with the aggregated opinions of all the agents is ready, the individual opinions can be merged. To do this, they need to select the appropriate fusion operator(s) that can be applied for each particular scenario, as described in Sect. 4.3.
(7) Then, decisions can be made depending on the resulting fused opinions, and their degree of uncertainty (see the considerations below about *the cost of being wrong*).
(8) Finally, in light of the resulting decisions, the belief agents can iteratively reconsider their individual opinions or the fusion operator to be used, and the process can start again until a consensus is reached.

A particular scenario occurs when there are no different independent models, one from each agent, but the model is the same for all, and the agents add their opinions on this common model. In such a case, only steps (2), (3), (5), (6), (7) and (8) apply.

5.2 Considerations

Let us discuss here some further considerations about this process.

Object models vs. class models. Systems are represented by means of object models, and therefore our proposal is expected to be used by belief agents to annotate elements of such models, such as instances, links or attribute values. Of course, it would be possible to add subjective opinions to class models, too. In this case, we would be treating the UML class models as an instance model conforming to the class model that represents the UML language. Then, annotations on a UML class model would represent how sure the modeler is about these classes, relations and attributes. In other words, we would be using our approach to allow agents to express their opinions about the entity types and relationship types used to represent the domain model, which is a particular type of Design Uncertainty [17, 18, 63]. The treatment of this kind of uncertainty is beyond the scope of this paper, and is left as part of future extensions of our work.

Probabilistic Logic to represent uncertainty. Users can employ probabilities to represent their degrees of belief, in case they prefer to use probabilistic logic —although this would mean neglecting their degree of uncertainty. This can be done by using values of type `UBoolean` instead of `SBoolean`. It is worth mentioning that `UBooleans` can be used wherever `SBoolean` values are expected because of the subtyping relationship defined between these two datatypes.

Uncertainty and "the costs of being wrong." In this approach, the resulting decisions, in addition to their associated degrees of *belief* and *disbelief*, will also be accompanied by their associated degree of *uncertainty*. This is relevant in situations where the risk of making decisions with high uncertainty may lead to errors involving huge losses or costs [51]. For example, a facial recognition system that mistakenly identifies a murder suspect, an automatic triage system that misclassifies an emergency patient as low risk when he/she is very seriously ill, or the credit authorization system of a bank that decides to wrongly grant loans based only on the confidence (i.e., the belief component) of a prediction by a software system but without considering the consequences of neglecting its associated uncertainty. Having a measure of the degree of uncertainty associated with a resulting decision can be useful, for example, to discard any decision whose degree of uncertainty is above a given threshold. This cannot be achieved with standard probabilistic logic but it is possible with our proposal.

6 TOOL SUPPORT

We have implemented our Belief profile in MagicDraw. Once an instance model is developed, stereotyped, and its model elements are assigned opinions using the profile, the user may want to fuse opinions. For this, we have created a MagicDraw plugin.[8]

Figure 5 shows a screenshot of MagicDraw including our profile and plugin. From the point of view of the final user, one only has to either click on the fusion button on the menu or right-click and select the option *Fuse available opinions*, then the result of the fusion operators shows in the model — see the tag values in the note in green that are placed under the «`UncertainElementFusion`» stereotype. Let us remind the reader that, since these tags have been defined in the profile, they extend the instance model but do not modify it.

Internally, our plugin traverses the object models and, for each element that is stereotyped as «`UncertainElementFusion`», it collects all available opinions, performs the required validity checks (e.g., opinions must be from different agents, base rates must be equal for some of the operators, etc.) and fuses them. For the fusion, we have developed a Java library that implements all the fusion operators described in Section 4.2. The plugin calls the Java library API every time that opinions need to be fused.

[8]Both the executable files and the source code are available for download from our github repository [8].

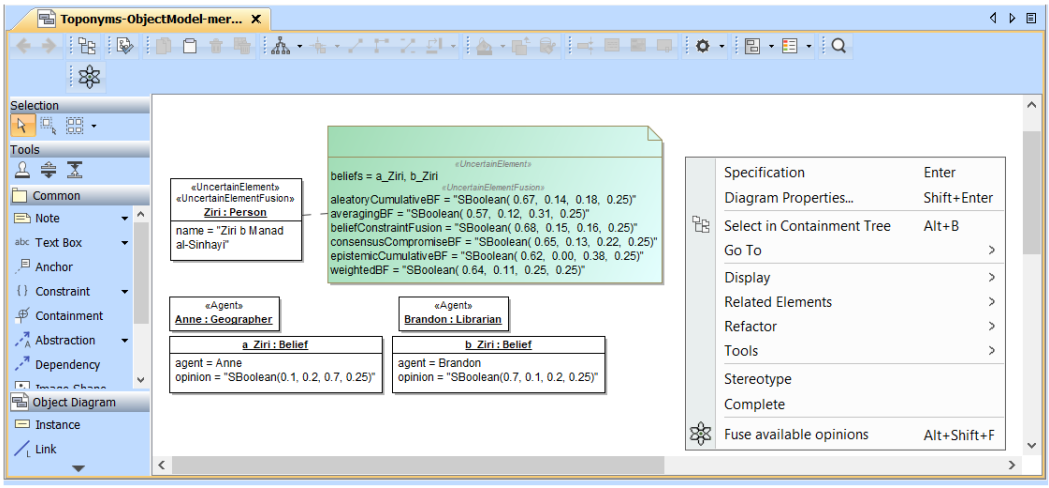


Fig. 5. Screenshot showing the use of the MagicDraw plugin on a model.

7 EVALUATION

In order to evaluate the feasibility, applicability and usability of our proposal, we have developed several application scenarios (Sect. 7.1) and we have performed an empirical study with users (Sect. 7.2).

7.1 Application scenarios

The applications cover different domains where managing vague and imprecise information is needed, and currently the solutions to represent and operate with this type of information are very limited.

(1) In the digital humanities context, we have developed a system using the DICTOMAGRED project [36] that studies the location of historical cities and placements in North Africa as they appear in ancient and modern sources, which suffer from vagueness and subjective interpretations from separate experts (historians, geographers, etc.).
(2) In the domain of Software Engineering, we present a scenario where Classifying Terms (CTs) [23, 25] have been used to automatically generate test cases. Our approach is used to add beliefs to the generated test cases (i.e., model instances) and to reason about them before they are accepted and used for testing purposes or discarded.
(3) Considering videogame digital stores, we have developed an application to track opinions in user reviews in Steam (a videogame digital distribution service). In Steam, users describe their personal opinions and experiences playing a game.
(4) We also developed an application for controlling the AC system of a room where users can express their confidence on the sources of information and reach agreements. It was briefly described in Sect. 2.1.
(5) As an example of an information provider system, we have developed an application to decide whether to perform outdoor activities based on weather forecast services, where users can express their individual trust in them.
(6) Another case shows how an AI-empowered recommender system is the source of uncertainty and different opinions. We show how separate friends that are going to travel together, agree on the hotel in which they are going to stay considering their individual trust on the travel

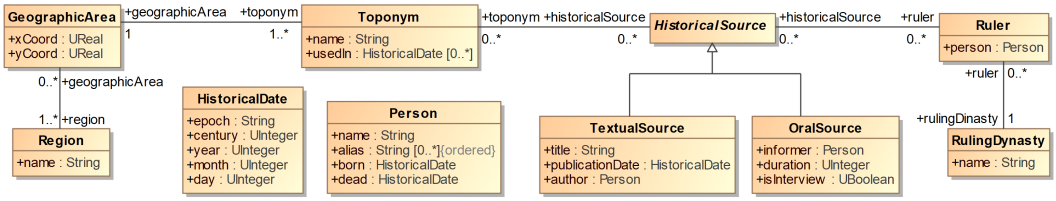


Fig. 6. Toponyms – Domain Model.

agencies and hotel booking services offering hotel rooms. These travel agencies and services use a machine learning algorithm that provides a confidence (i.e., an objective uncertainty) for each recommendation that they make.

These scenarios are described in the companion website to this paper [9]. They aim at demonstrating the expressiveness of our proposal in different application areas, thus responding to research question RQ1. Namely, the first scenario was chosen from [38] to allow us to compare our proposal with theirs, using a similar example. The second one shows a use case of our proposal in the context of software engineering. The third scenario illustrates our approach in environments that collect feedback from a multitude of users, in this case videogame reviews in a digital store. Scenarios (4) to (6) allow users of different domains (smart systems, information providers and recommender systems) to assign opinions to their instance models. In all cases, we also show how the individual opinions can be combined to reach agreements, thus addressing research question RQ2.

For illustration purposes, the following subsections (7.1.1, 7.1.2, and 7.1.3) describe the application scenarios (1), (2), and (3). The detailed description of the rest can be found in [9].

7.1.1 Vagueness in the information provided by historical sources. This case study is taken from the DICTOMAGRED project [36] in the digital humanities domain, which analyses historical sources (e.g., oral testimonies and legal documents), most of them in Arabic. Such sources contain geographical references describing routes through different areas in the Maghreb, their place names (toponyms), and other related historic events. The main goal of the project is "to provide a software tool for humanities specialists to retrieve information about the location of toponyms in North Africa as they appear in historical sources of medieval and modern times" [36]. This project has been extensively used in other proposals [37–39] that employ domain models to represent the inherent vagueness.

Figure 6 shows a class model that represents the types of entities and relationships needed to capture the toponyms of a region. *Toponyms* are located in a *geographical area* that belongs to one or more *regions*. The names of the toponyms have been mentioned in different *historical sources*, dated when certain *rulers* where governing particular *dynasties*.

Suppose now that two researchers, Anne (a Geographer) and Brandon (a Librarian), after consulting different sources, have independently created two different object models about the `Ashir` toponym located in the region known today as the Maghreb. Note that these sources may not be reliable, provide imprecise or incomplete descriptions and details, or may have been altered over the years, with different versions of the same facts (such as it frequently happens with oral sources).

Following the methodology presented in Sect. 5, Anne and Brandon added their *opinions* to those elements in their own model that they were not completely sure about. Then, they merged their models. Being both of them foresighted, they contacted Carol, a well-known expert in the field, to solve their potential disagreements on the most controversial topics. She was not asked to create another object model but to provide her opinions on the merged model by Anne and Brandon. Note that, while merging the models and revisiting their opinions, they can slightly change their

mind based on the discoveries of their colleagues. More details about this process and the agents' individual models can be consulted in our technical report [9]. Figure 7 shows the merged object model enriched with the opinions of the three agents. Remember that model elements with no assigned belief are assumed to be dogmatic trues, i.e, `SBoolean(1,0,0,1)`. Now it is time to use the fusion operators to combine the agents' opinions to produce a fused opinion that can be assumed to better reflect the truth. As we explained before, this is a challenging process in general because different belief fusion situations can occur in practice, and may require different fusion operators depending on the purpose and nature of the fusion process [29, 66]. Let us show in the following how decisions have been made for each particular case.

Fusing opinions, situation #1: Anne and Brandon have studied the same toponym, but they have used two different textual sources. While Anne has used *Kitab al-masalik*, Brandon opted for *Mu yam al-buldan*, which has led to contradictory information. They try to agree on what is the best source to keep studying together, but in case they doubt about the usefulness of the source that the other person has chosen, they will keep using the one they originally selected. Note that they do not doubt about the existence of the textual source but about its usefulness to study the toponym, hence they have placed their subjective opinion on the association between the toponym and the source.

Fusion operator: The fusion operator that matches the situation the best is *belief constraint fusion* (BCF): no room for compromise in case of totally conflicting arguments.

Result: After revisiting the class model, Anne is stubbornly sure that she is using the only source that contains true information and that Brandon's source is completely mistaken. Therefore, she has assigned `SBoolean(0,1,0,0.5)` to the existence of the association between `Ashir`; and `Bakri_Ts` and `SBoolean(1,0,0,0.5)` between `Ashir` and `Yakit_Ts` to express that she does not believe that `Bakri_Ts` is a textual source to study the toponym `Ashir`, while `Yakit_Ts` is. In contrast, Brandon considers that Anne's source is not correct and that his is better. Therefore, he assigns `SBoolean(0.8,0.1,0.1,0.5)` and `SBoolean(0,1,0,0.5)` to the association between `Ashir` and `Yakit_Ts` and `Bakri_Ts`, respectively.

Note that, since there are two textual sources (`Yakit_Ts` and `Bakri_Ts`), each textual source is a binary domain (it can be either considered or not) and therefore the prior probability (i.e., the base rate) that Anne and Brandon have assigned to both sources is 0.5.

The result of the BCF operator on the opinions on the association between `Ashir` and `Bakri_Ts` is *undefined* (due to the contrary opinions); and the result of the BCF operator on the opinions on the association between `Ashir` and `Bakri_Ts` is `SBoolean(0, 1, 0, 0.5)`. Since they cannot agree on studying only one source, they decide that each one will keep studying their own and both instances are kept in the object model.

Fusing opinions, situation #2: Even studying different sources, both Anne and Brandon seem to have discovered that someone called *Hammad* was a ruler. Nevertheless, Brandon also thinks that there was another ruler during that period of time named *Ziri*. Anne, Brandon and Carol annotate the model with their subjective opinions about the existence of the two rulers.

Fusion operator: After inviting Carol to give her opinion, the amount of independent evidence increases (one more agent/source of knowledge) and the degree of uncertainty must be reduced (due to her expertise). Therefore, the *Aleatory Cumulative Belief Fusion* (ACBF) operator is the most appropriate.

Result: Given this information, it could have happened that: the ruler was *Ziri*, the ruler was *Hammad*, both *Ziri* and *Hammad* were rulers, or none of them was. Then, we are facing a quaternary domain for which the agents' opinions' base rate must be 0.25. After applying the operator on the opinions on both instances, the results are:

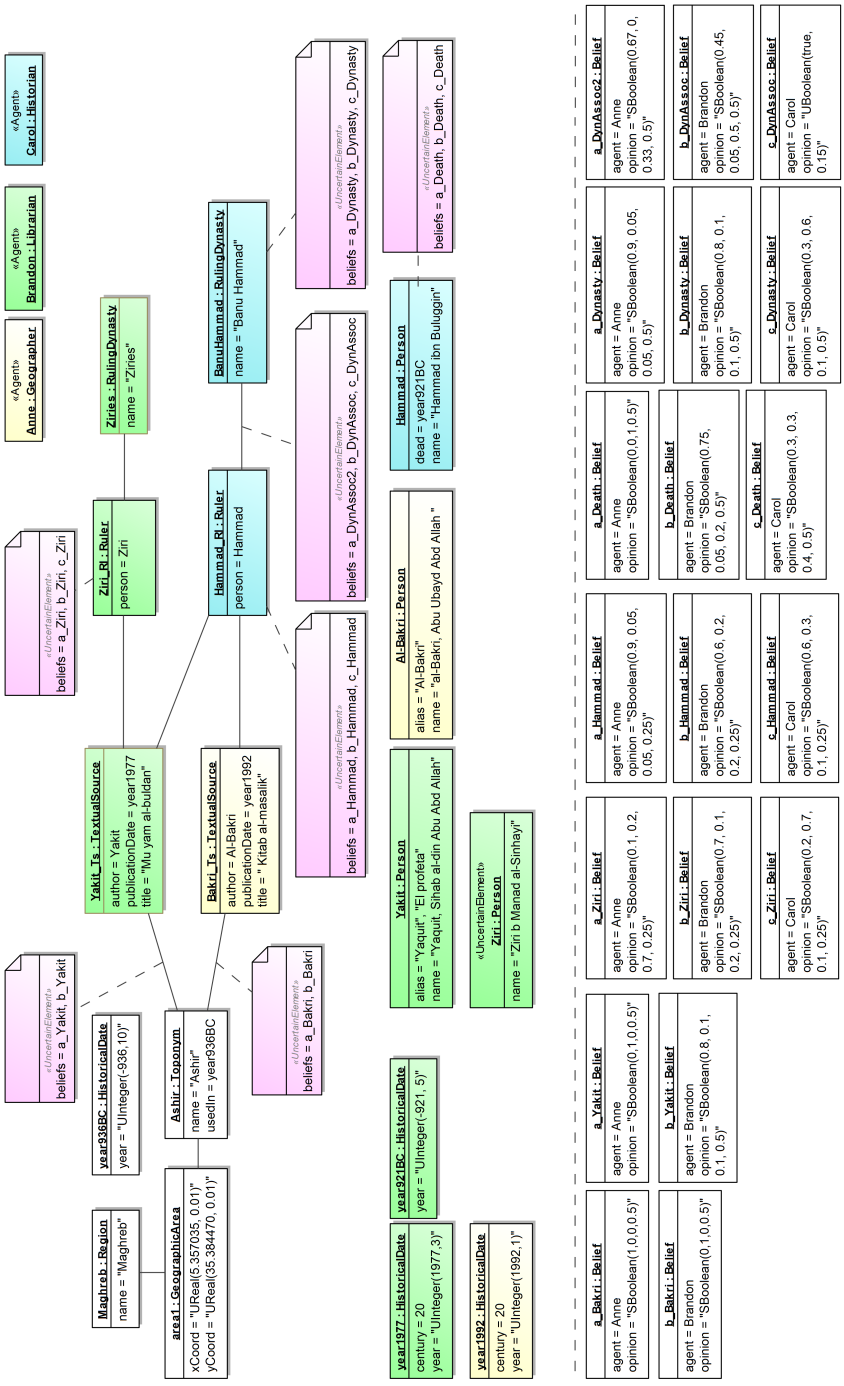


Fig. 7. Toponyms - Dictomagred Object Model (merged)

```
ACBF(b_Ziri, a_Ziri, c_Ziri)       = SBoolean(0.39, 0.54, 0.07, 0.25)
ACBF(b_Hammad, a_Hammad, c_Hammad) = SBoolean(0.82, 0.15, 0.03, 0.25)
```

The group's opinion about the existence of the ruler *Ziri* is not very much supported with a belief below 0.4, a disbelief of 0.54 and not much uncertainty; while there seem to be a consensus on the existence of the ruler *Hammad*, supported by a belief of 0.82. The three agents decide to agree that *Hammad* was a ruler while *Ziri* was not.

Fusing opinions, situation #3: The three agents discussed about *Hammad*'s death date. Brandon found some hints on the textual source that he studied that made him believe that *Hammad* may have died around 921BC and he supports this with the opinion `SBoolean(0.75,0.05,0.2,0.5)`. Carol found this date sensible (`SBoolean(0.3,0.3,0.4,0.5)`), while Anne was not convinced at all (`SBoolean(0,0,1,0.5)`).

Fusion Operator: The agents decide to explore the results of applying the *Averaging Belief Fusion* (ABF) operator, to give each opinion the same weight; and the *Weighted Belief Fusion* (WBF) operator, to give greater weight to more confident opinions.

Result: Since the result of both operators support the death date more strongly than not, the agents agree to keep the date of *Hammad*'s death in the final model.

```
ABF(b_Death, a_Death, c_Death) = SBoolean(0.53, 0.12, 0.35, 0.5)
WBF(b_Death, a_Death, c_Death) = SBoolean(0.63, 0.12, 0.25, 0.5)
```

Fusing opinions, situation #4: While the group has agreed to keep studying the ruler called *Hammad*, they discuss about the dynasty he belongs to. Despite Anne's certainty about the fact that it belongs to the *Banu Hammad* dynasty (as reflected in her opinions), Brandon and Carol are not so sure that this was the case. Brandon believes that the dynasty existed but has some concerns about the fact that *Hammad* was part of it (i.e., his concerns are reflected in the association). In contrast, Carol doubts about both facts: the existence of the object `BanuHammad` and the link. Note also that Carol has not specified her subjective opinion about the association but she has only used a probability (i.e., *credence*). This probability represented with an `UBoolean` will be automatically lifted to the corresponding Subjective logic value `SBoolean(0.15,0.85,0,0.15)` when needed; for instance, to operate with other `SBoolean` values.

The group would like to make their findings public, but before going to the press with a statement like "there was a ruler named *Hammad* who belonged to the *Banu Hammad* dynasty," they need to see what their collective belief is about this.

Fusion operator: To compute the collective belief, we need to aggregate each individual belief about the different instances and then combine the resulting beliefs with an appropriate fusion operator ($\otimes$). The individual opinions are aggregated using an `and` operator. Hence, the OCL expression is:

$\otimes$ (`av_Hammad and av_DynAssoc and av_Dynasty`, `pm_Hammad and pm_DynAssoc and pm_Dynasty`, `lb_Hammad and lb_DynAssoc and lb_Dynasty`)

In this situation, the three agents have the same credibility and they are observing the same fact, and hence the most suitable fusion operator $\otimes$ is the *Averaging belief fusion* (ABF).

Result: Using the ABF operator, the result of the formula above is `SBoolean(0.26,0.37,0.37,0.05)`, whose projection is 0.28, and its degree of uncertainty is 0.37. This is not sufficiently convincing and thus the three experts decide to continue studying the toponym before making any press release.

Note how these operators can be used in standard OCL expressions, given that they form part of the extended UML and OCL type system that we have developed, and where `SBoolean` is just another primitive datatype [40].

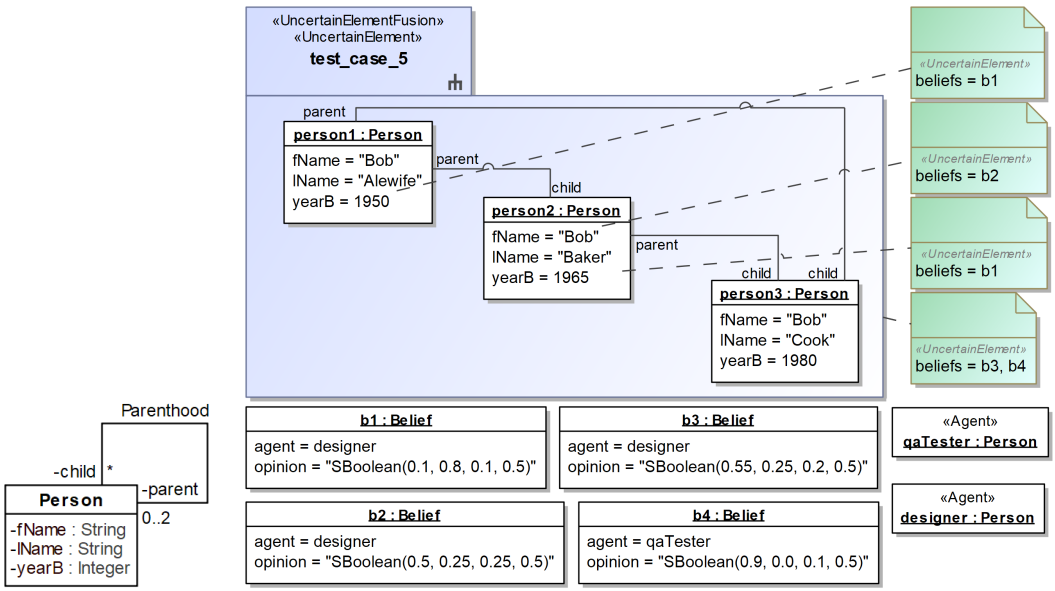
7.1.2 Test Case Generation in Software Modeling. The generation of suitable test cases is an essential task for achieving effective software testing, which is one of the key activities in Software Engineering. The use of Model-Based Testing techniques [65] has proved to be successful in this field, using the system models to automatically generate test cases [54]. In this context, as important as automated test case generation is the selection and prioritization of the most effective tests. A good selection of test cases significantly increases the effectiveness of the testing process by detecting more faults with a smaller test suite, as well as their efficiency, since reducing the number of tests also decreases the testing execution time [57].

Although automation is often desirable, there are situations where the intervention of test experts is necessary in the selection and prioritization of test cases. For example, current model validators generate many test cases that are either too *similar* (from the testing point of view), or cannot be considered *realistic* [34]. To tackle the generation of an excessive number of similar test cases, model-based approaches such as *Classifying Terms* (CTs) [23] use equivalence partitioning techniques to automatically guide the test case generation so that the number of test cases is reduced to only those that represent distinct cases from a given user's perspective, achieving good results [25]. An additional problem is that some of the automatically generated test cases either do not represent realistic situations or are not meaningful to the tester, and therefore need to be removed from the final test suite in order to avoid superfluous tests which sometimes are time-consuming and do not add any value to the testing process. Identifying such unrealistic or meaningless test cases is not an easy task, and currently it is mainly performed directly by the test experts. The problem is that such decisions are subjective in many cases, and depend on the persons making them.

To illustrate this situation with a simple example, let us assume that we are dealing with the domain model from [23] that represents Families (shown in Figure 8a). In this model, each person has a first name (`fName`), a last name (`lName`) and a birth year (`yearB`). The parenthood relation between persons is captured by means of an association. Using CTs, a set of diverse test cases can be automatically generated, distinguishing between different types of families according to their structure: with no kids, just parents and children, including grandparents too, etc. Figure 8b shows one of the test cases (represented as an UML object diagram inside the Package `test_case_5`) that was automatically generated for the class diagram in Fig. 8a. Although syntactically correct, this case does not seem to represent a meaningful situation: it shows a family where Bob Alewife has a child (Bob Cook) with one of his other children (Bob Baker). Besides, they both had their first children when they were 15.

When a designer faces a situation like this, instead of directly discarding the whole model—which could lead to discarding all the automatically generated models [34]—they could annotate the model elements they are not sure about with their own beliefs. This is precisely where our proposal can be of great help. Thus, given the doubts that our designer had about the automatically generated object model, he decided to annotate those elements he was not sure about using our profile. For instance, he thinks that the birth years of `person1` and `person2` are not appropriate as they indicate that both persons have a kid when they were 15 years old. Therefore, an opinion of `SBoolean(0.1, 0.8, 0.1, 0.5)` is assigned to them. Furthermore, he thinks that not all three people should be named Bob. Therefore, he also assigned the belief `b2` to the name of `person2`. All this information is captured in the object model presented in Fig. 8b.

In addition, the designer assigned an opinion to the test case as a whole, i.e., considering it as a single entity. This was done by assigning a belief to the package that contains the model elements. We can observe how, in Fig. 8b, the package has been stereotyped as an uncertain element and the belief `b3` has been assigned to it. In such a case, the belief that an agent holds about a concrete model element is conditional to the belief stated on the test case. Since they are independent beliefs,



(a) Domain model. (b) Generated test case annotated with opinions.

Fig. 8. Test case generation - example using the Families domain model from [23].

the designer could compute the updated belief of a model element using the "and" operator. For example, for the attribute `yearB` of `person1`, the updated value is calculated with the formula: "`b1 and b3`", which returns the value `SBoolean(0.08, 0.85, 0.07, 0.25)`. Given the resulting low degree of belief and high disbelief, the designer decides to refine the test case by manually updating the attribute `yearB`.

Of course, more than one person may express their opinion about a test case or any of its elements. In this case, they need to reach a consensus. For example, let us assume that our designer is working together with a QA tester, who is also asked to annotate the test case with their personal beliefs. The QA tester does not question any of the model elements, only is slightly uncertain about the overall case. This is why she decides to assign an opinion of `SBoolean(0.9, 0.0, 0.1, 0.5)`, as shown in belief `b4`, because she thinks that the test case might be useful for testing the system since it could cover a corner case.

To reach a consensus, the most appropriate fusion operator is Averaging Belief Fusion (ABF) because the opinions are dependent and they should have the same weight. Thus, `ABF(b3,b4) = SBoolean(0.78, 0.08, 0.13, 0.50)`. Given the result of the fusion operator (a high degree of belief with low uncertainty), both the designer and QA tester decide to keep the test case , but with the modifications suggested by the designer. This example illustrates how the use of our profile provides a support tool for making informed decisions in this context.

7.1.3 Videogame reviews on Steam. Steam[9] is a videogame digital distribution service, which offers services such as a digital store, digital rights management, video streaming, social networking and APIs for supporting in-game achievements and microtransactions. In terms of market share, the Steam platform is currently the largest digital distribution platform for PC games, with nearly 30,000 games and over 100 million users.

[9]https://store.steampowered.com/

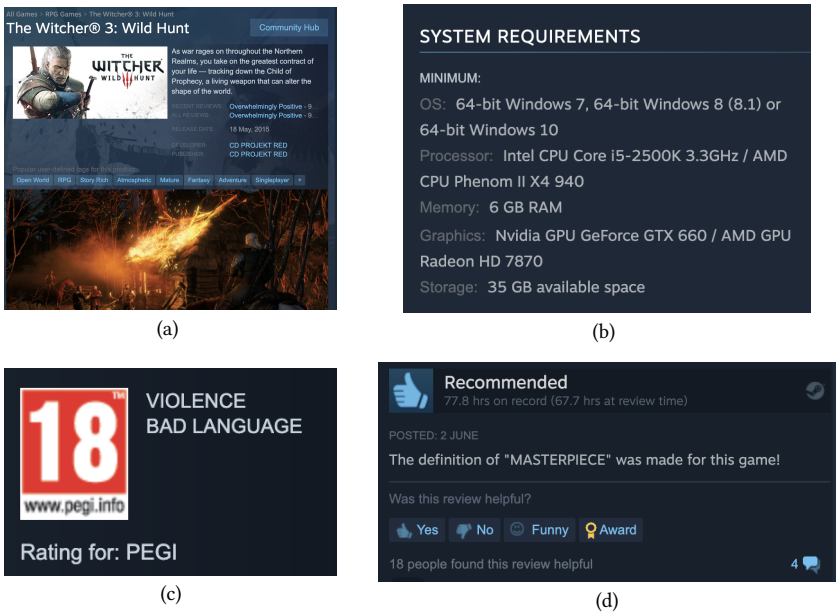


Fig. 9. (a) Steam store product page; (b) System requirements; (c) PEGI rating; (d) Sample user review.

Steam's digital store includes a dedicated page for each game in their catalog (see Fig. 9). The information included in this page mixes both factual information provided by the game developer and opinions and user reviews contributed by Steam users. For instance, the following information is provided about a particular game:

- The Pan European Game Information (PEGI)[10] classification, providing a recommended age label and a list of content descriptors.
- The list of language translations for in-game information, dialog and/or subtitles.
- The list of downloadable content (DLC) that provide extras to the game, such as additional maps, missions, characters, items or skins.
- The list of supported platforms as well as the minimum and recommended system requirements (CPU, RAM memory, graphics card, available storage) for each one.

Regarding opinions, users can assign tags to games (such as "single-player" o "role-playing game") as well as write textual reviews. Furthermore, users can show their agreement with a particular review by endorsing it ("Was this review helpful?").

Steam reviews offer an interesting dataset from the point of view of *opinion mining* and *sentiment analysis*, which has been analyzed in the software engineering literature (*e.g.*, [31]). User reviews may show conflicting opinions regarding a particular game. Some examples of controversial topics are software bugs, the minimum system requirements (with users explaining their first-hand experiences with incompatibilities, crashes or bad performance) and the amount of hours of content provided by the game. Some of these controversial topics are also analysed by specialized social gaming services, such as HowLongToBeat.[11]

In the following, we discuss how our uncertainty modeling approach can be used to study the opinions of several users about a particular videogame. First, Fig. 10 shows the domain model for a

[10]https://pegi.info
[11]https://howlongtobeat.com

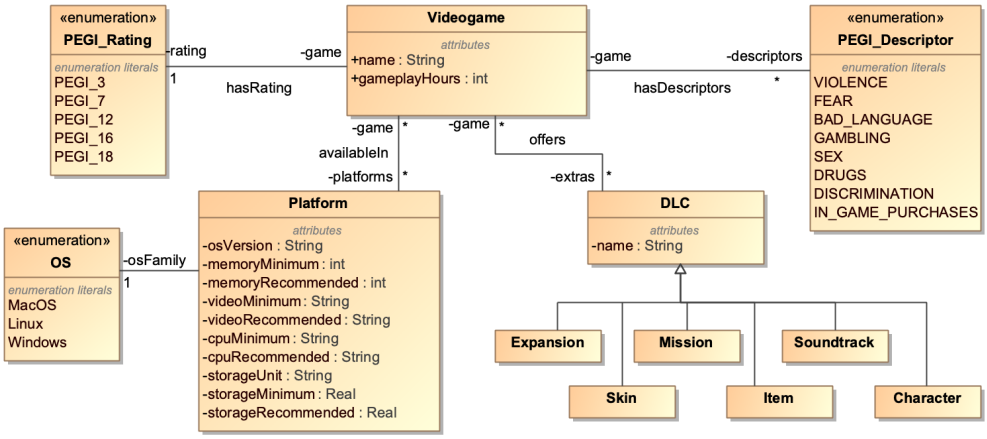


Fig. 10. Videogame domain model.

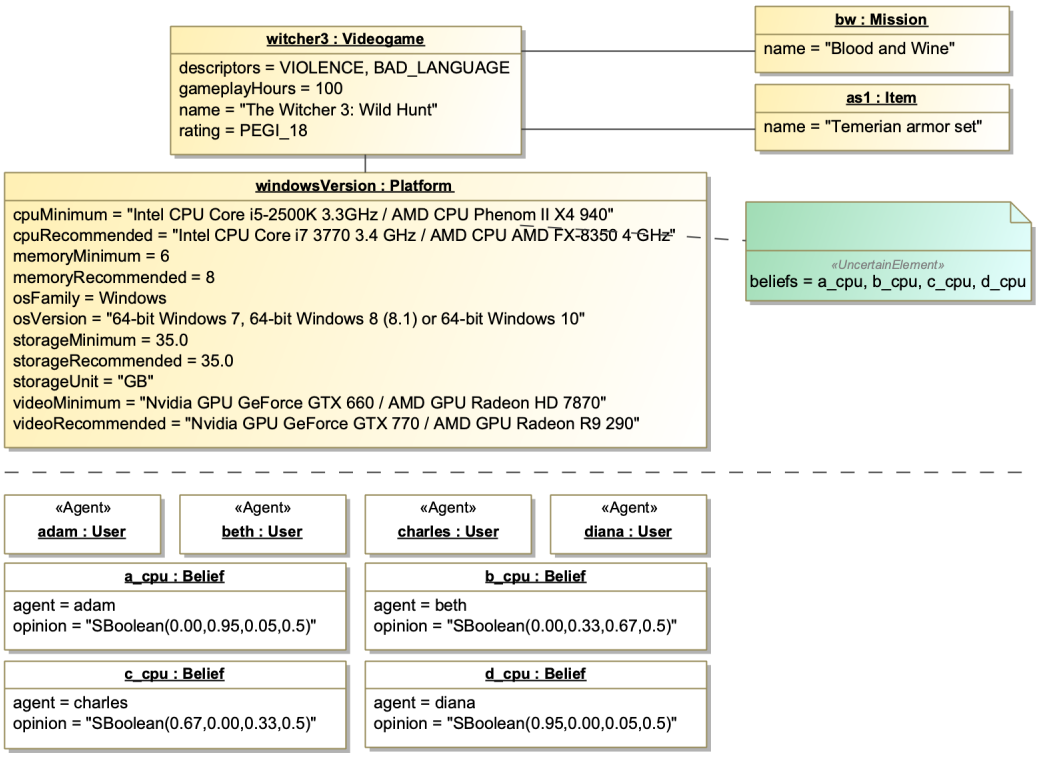


Fig. 11. Object model of the videogame "The Witcher 3: Wild Hunt".

videogame in the Steam store. Then, it is possible to create object models with the information of particular games. For instance, we have selected a popular role-playing game ("The Witcher 3:

Review	Votes	Subjective opinion	Projection
Adam: *"My computer has twice the requirements and still it does not run at 100%, there are always some glitches or bugs"*	18	`SBoolean(0.00, 0.95, 0.05, 0.5)`	0.03
Beth: *"It needs better hardware compatibility. Users with AMD hardware (I'm not included) are having a rough time, from what I hear"*	114	`SBoolean(0.00, 0.33, 0.67, 0.5)`	0.34
Charles: *"I have a modest PC and I can run it at 1080p/high with decent and stable FPS, so if you meet the requirements you should not have any trouble running it"*	34	`SBoolean(0.67, 0.00, 0.33, 0.5)`	0.84
Diana: *"Using a full AMD PC, the game runs perfectly in ULTRA quality with no glitches even though the game uses proprietary technology from NVIDIA"*	33	`SBoolean(0.95, 0.00, 0.05, 0.5)`	0.98

Table 3. Sample user reviews discussing the minimum system requirements.

Wild Hunt"[12]) and built the object model shown in Fig. 11.[13] As a popular game, it has attracted a very large number of reviews (more than 400k), some of them showing conflicting opinions about system requirements. Some users consider them to be sufficient while others disagree.

Let us illustrate our proposal in the case of one particular requirement, namely the CPU required to run the game (i.e., the value of attribute `cpuMinimum` in class `Platform`). The value of this attribute constitutes the "fact" about which users express their subjective opinions.

To study these subjective opinions, we consider the reviews of four Steam users, which we call Adam, Beth, Charles and Diana for convenience. Table 3 describes sample fragments from their reviews discussing system requirements as well as the number of votes given by other community members that considered each review helpful. The number of endorsements can be used as a measure of the popularity of that opinion among the player base, but can also be interpreted as the endorsers' own opinions, and therefore consider endorsers as agents who have expressed their opinions, too.

Notice that the textual opinions sometimes exhibit a degree of uncertainty, *e.g.*, second-hand opinions ("from what I hear") or statements of belief ("you should not have trouble"). Using our profile, we can define the subjective opinion that each user assigns to the attribute `cpuMinimum`, which is stereotyped as «`UncertainElementFusion`» in the object diagram.

To compute the aggregated opinion that merges the opinions of all users we employ the *Weighted Belief Fusion* (WBF) operator, because the opinions are dependent but we want to assign less weight to those opinions with higher uncertainty.

The result of merging all 199 opinions (18 × Adam, 114 × Beth, 34 × Charles, and 33 × Diana) using the WBF operator is `SBoolean(0.587, 0.314, 0.099, 0.5)`. Therefore, we can be 58.7% certain that the requirement is true, 31.4% that it is false, with an uncertainty of almost 1%. The projection of the result is 0.64, so we can initially trust (with some reservations) the vendor's claim.

7.1.4 Lessons learned. The application scenarios (sections 7.1.1, 7.1.2, 7.1.3, and [9]) illustrate several situations where belief uncertainty is relevant. These scenarios have been selected from different application domains, showing that the application of belief uncertainty is not restricted to particular domains. Moreover, we have considered different sources of belief: provided directly by different individuals or experts (e.g., Section 7.1.1 and 7.1.2) or mined from online reviews (e.g.,

[12]https://store.steampowered.com/app/292030/The_Witcher_3_Wild_Hunt/

[13]For the sake of brevity, we have not included the complete information about this game: at the time of writing, 22 different DLCs were available in the Steam store.

Section 7.1.3). In all these application scenarios, our profile was capable of capturing the beliefs of different stakeholders and the fusion operators provided a way to merge these opinions in order to make informed decisions, demonstrating the feasibility of our contribution.

7.2 Empirical evaluation

This section presents the empirical experiment that we designed and carried out to validate the usability of our UML profile and the proposed methodology. We followed the basic methodology for conducting usability studies [50]—which is derived from the classical approach for conducting controlled experiments—as well as the Empirical Standards for Software Engineering Research [49]. As recommended in [50], instead of formulating an hypothesis, this experiment aims at answering our research questions.

7.2.1 Experiment design and setup. The experiment consisted of an off-site and asynchronous screening test (Session 0) to evaluate the subjects prior knowledge and adequacy to participate in the experiment, and an on-site exercise with three parts (Sessions 1-3) and a duration of 3 hours. The complete protocol, materials and exercises provided to the subjects, the questionnaires used and the anonymized data collected can be found in our Github repository [8].

- The screening test sought to ensure that participants had a minimum level of knowledge of UML class and object diagrams, and to provide a basic introduction to uncertainty so that all subjects had the same background information.
- In Session 1 (50 min) we introduced basic notions on belief uncertainty, subjective logic, the `SBoolean` datatype, and UML profiles. Participants were asked to fill out a questionnaire (Q1) to check whether they were able to understand how opinions are represented in UML models, and their semantics.
- In Session 2 (60 min), the participants were given a small model (6 instances, 8 attributes and 4 links) with the location of the Ark of the Covenant, and they had to represent their subjective opinions about some of the model elements. During the development of this exercise, we recorded the participants' screens. Participants were asked to fill out a questionnaire (Q2) on the usability and expressiveness of the Belief Uncertainty Profile for representing opinions.
- In session 3 (70 minutes), participants were randomly organized in groups of three. During this session, we recorded the participants' screens as well as the audio of their discussions. This session had two parts. First, they had to decorate a given model with their individual opinions on one of its elements and work collaboratively to come to an agreement—warning them that any decision they made could have consequences. The participants had to individually fill out a questionnaire (Q3A) where they are asked about their experience, i.e., whether they were able to make a decision, their satisfaction with the decision, the difficulties they faced, and the method they followed to reach an agreement (if this was the case). In the second part of Session 3, we introduced the fusion operators, and asked them to select one and use it for reaching a conclusion. Note that we explained our iterative methodology to participants and asked them to put it in practice. Finally, we asked participants to respond individually to a questionnaire with two parts. The first (Q3B) was intended to report on their experience using the fusion operators. The second (Q4) contained questions about the whole experiment, such as the perceived usefulness and ease of use of `SBoolean`, the Belief Uncertainty profile and the fusion operators, their feedback on our tool and the process, and suggestions for future improvements.

Seventeen people accepted our invitation to participate in the experiment and qualified for it. Although Nielsen and other authors maintain that five users are enough for usability testing [41, 64], other authors suggest the rule of 16 ± 4 participants [2]. We run a pilot with 3 participants and the

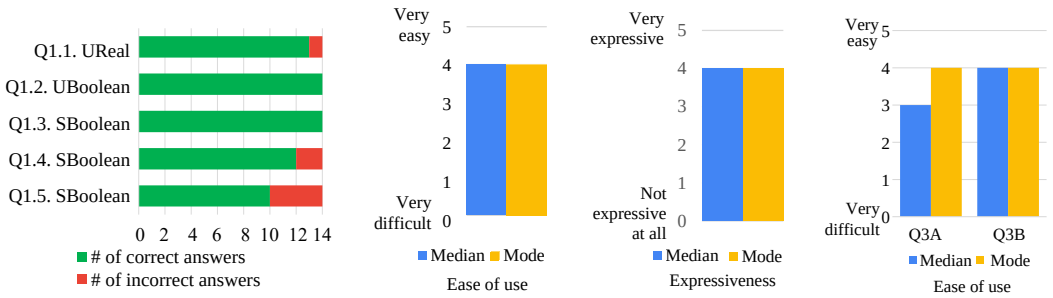
Number of participants with degree			
PhD	MSc	BSc	None
4 (29%)	5 (36%)	4 (29%)	1 (7%)

(a) Participants' education.

	Prior knowledge about				
	UML diagrams	UML profiles	MagicDraw	Uncertainty	Uncertainty on models
Median	4	2.5	3.5	2	1
Mode	4	4	4	3	0

(b) Participants' prior knowledge.

Fig. 12. Participants' profiles.



(a) Q1.1-Q1.5 (b) Q2.1 (c) Q2.2 (d) Q3A-B

Fig. 13. Results of questionnaires.

experiment with 14 participants, with which we tried to cover both situations. The similarity of the results obtained in all three groups, and their correlation with the results of the three groups combined, seem to support Nielsen's theory.

We carried out the pilot experiment with 3 subjects on November 9, 2021 and used it to refine our protocol, materials, instructions, exercises and questionnaires. Then, we discarded its results. To accommodate to the participants' agendas, we carried out the experiment on three different dates: November 11 in Malaga with 6 participants, November 18 in Barcelona with 5 participants, and November 26 in Malaga with 3 participants. The first author of the paper was always present and ensured that all the sessions were equally executed. The language used was either Spanish or English according to the participants' preferences.

7.2.2 Results. Tables 12a and 12b in Fig. 12 present the degrees that our participants hold and their prior knowledge, respectively. We used a Likert scale where 0 is *Not at all* and 5 is *High*. Then, Fig.13a shows the results of questionnaire Q1, where we checked whether they understood subjective logic opinions and uncertain datatypes. Despite the limited previous knowledge of our participants about uncertainty on domain models, we can conclude that, a brief presentation was enough for most participants to understand our proposal (syntax and semantics).

Figures 13b and 13c show the responses to Q2, where we asked participants about the ease of use and expressiveness of the profile. Results are very positive. The model that they uploaded as well as their screen recordings show that all participants were able to successfully complete the exercise.

Figure 13d shows the answers to the qualitative questions of questionnaires Q3A and Q3B. In the first one, a decision had to be made without knowing the fusion operators. All groups managed to reach an agreement, although using different strategies. For example, "*each member made a binary decision and we voted*"; "*I was unsure so I let the others decide*"; "*we discussed and each one tried to convinced the others*". Interestingly, one group reported that they calculated the average of each component of their opinions represented as `SBoolean` values and obtained `SBoolean(0.6,0,0.4,0.5)`,

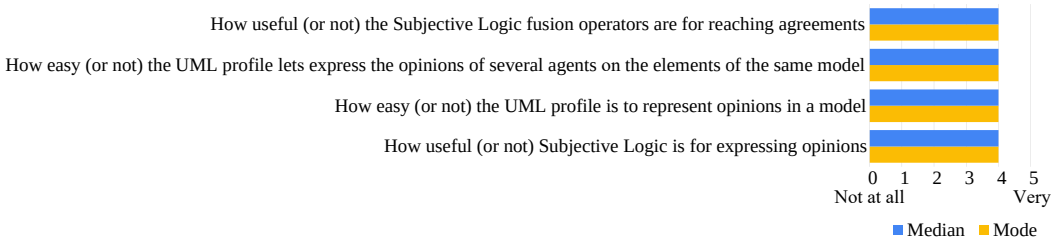


Fig. 14. Final questions and their responses.

which they use to reach a decision. We realized that, somehow, these participants were trying to find an operation to merge their opinions.

The results of Questionnaire Q3B (i.e., after using the fusion operators to make a decision) show that it was easier with them. The only reported difficulty was how to choose the most appropriate fusion operator, which we already know is not an easy problem [28]. However, they all managed to find it. Moreover, our participants found straightforward the use of our plugin to fuse opinions (one even said "*we only need to click a button*").

Figure 14 shows the results of the qualitative questions asked to the participants in questionnaire Q4. They all seem to be satisfied with our proposal. It is worth mentioning that the analysis of the participants' screen and voice recordings confirm that they successfully applied our methodology and our tool supported all the process (e.g., they defined their opinions, iteratively refined them, and used the fusion operators to reach a consensus). Finally, in the open-ended questions, all the participants except one described our approach with positive adjectives such as interesting, novel, easy-to-use, intuitive, etc. Only one participant complained about the fusion operators saying that he "*considers the decision wiser when fully* [...] *debated orally and humanly weighted.*"

7.2.3 Lessons learned. Overall, most participants found our proposal useful, expressive and usable. We observed how the learning curve is steeper at the beginning, which is reflected in some incorrect answers to the questions in questionnaire Q1. However, as the experiment progressed and participants became familiar with all the concepts of our proposal, they were able to use it faster and correctly. The participants also expressed that the selection of the most suitable fusion operator was the most difficult part. However, the results show that all of them were able to succeed in this task. We believe that, with some training, good documentation and/or examples, this aspect does not hinder the application of our proposal. Nevertheless, we plan to investigate how we can better assist users in this step in the future.

The fact that participants explicitly saw the opinions of other members before and during their consensus-building discussions made them realize that their opinions could be refined. This shows that our iterative methodology was faithfully put in practice.

7.3 Threats to Validity

Threats to validity are inherent to every empirical study. In our case, we will discuss those related to the validation conducted using application scenarios (Sect. 7.1), and those that might apply to the controlled experiment with users (Sect. 7.2). We also discuss the mitigation actions that we employed for each threat, where appropriate. Threats are classified into four categories: internal, construct, external, and conclusion validity [69].

7.3.1 Internal validity. Threats related to the factors that could affect the results of our evaluation.

- *Communication.* The questionnaires and all the materials were in English, which is not the native language of all the participants. *Mitigation:* We allowed the participants to perform the discussions and provide textual answers in their native language.
- *Background.* Some participants could not have the necessary knowledge about UML and its profiles to be considered for our experiment. *Mitigation:* We performed a screening test before the experiment to select only subjects with the required level of knowledge.
- *Separate days.* We performed the experiment in three different days, trying to recreate the same conditions for each of them. However, there might be some differences in the explanations and the questions asked by the participants that could make the sessions not identical. *Mitigation:* We followed the same protocol for each of the sessions in order to reduce significant differences between them. Furthermore, one of the authors was present in all experiments, ensuring they all followed the same protocol.

7.3.2 Construct validity. These threats are related with those issues that might arise during research design, which are concerned with the relationship between theory and what is observed.

- *Protocol.* As for the methodology followed to conduct the experiment, it can never be guaranteed to provide sufficient detail for the success of the study. *Mitigation:* We followed the basic methodology for conducting usability studies [50] as well as the Empirical Standards for Software Engineering Research [49].

7.3.3 External validity. These threats are related to the extent to which it is possible to generalize the findings and conclusions of this study beyond the experiment context.

- *Selection.* The conclusions may depend on the choice of a particular application domain. *Mitigation:* We have developed six case studies covering a diversity of application domains. These scenarios consider both merging opinions provided by individuals (e.g., DICTOMAGRED) as well as opinions from groups of individuals that could be mined from online forums (e.g., the videogame store). Moreover, all our cases try to recreate realistic situations.
- *Modeling notation.* In this work, we have studied beliefs in instances of domain models described using the UML notation. The usability of our approach has not been assessed in the context of other modeling formalisms.
- *Population.* The experiments have been conducted with participants with an academic background (students, lecturers, researchers, ...) and in an academic setting. Although this is referred as convenience sampling [68] and is common practice in controlled experiments, this imposes a threat. *Mitigation:* Participants with different education levels (PhD, MSc, BSc) have been selected for our experiment.

7.3.4 Conclusion validity. Threats to the conclusion validity are concerned with the issues that affect the ability to draw correct conclusions and whether the experiments can be repeated.

- *Sample size.* The number of participants in the experiments may be insufficient to draw correct conclusions. *Mitigation:* The experiments have included 14 participants. As previously, mentioned considering 16 ± 4 participants [2] is considered sufficient in this type of studies.

8 RELATED WORK

In previous works [4, 6], we showed how to use probabilities to represent objective uncertainties in UML and OCL as primitive datatypes [3]. More recently, we have incorporated subjective logic [28] into the UML and OCL type system, being able to represent opinions with some degree of uncertainty [40]. This paper build on our previous contribution [40] to extend our work in [6] so that experts can enrich software models with their individual opinions, merge them with the opinions of other experts, and reach agreements.

The explicit identification and treatment of vague information has been considered in several domains where the decision-making process is important:

- In the digital humanities context, historians and geographers need to deal with information that comes from historical sources (including manuscripts and oral testimonies). Some of these sources are unreliable, incomplete or even inconsistent. Besides, different experts may have different interpretations of the same facts [1]. Several initiatives exist to deal with such vague information. For example, the ConML language [38] allows the annotation of the elements of the instance models with information representing the modeler's *confidence* in the truthfulness of that element, such as the exact location of a settlement or the actual author of a manuscript. Although in ConML, several experts can express their individual opinions, too, these opinions cannot be reconciled. Furthermore, its treatment of uncertainty is rather basic, with a Likert scale of vagueness. The approach proposed in [1] permits historians to express their intuitions using probabilistic logic, and then reason about the models enriched with this information. Using subjective logic would allow them to explicitly express their degrees of uncertainty when needed, too.
- Other authors have also used subjective logic to express trust, using the information provided by external users or even crowds in different application domains, such as museum collections [10], maritime navigation [11] or cultural heritage [42]. In these works, users not not express their subjective opinions, they merely provide information (e.g., tags to classify museum items) that is used as evidence to compute the degrees of trust, sometimes combined with other information, such as provenance (i.e., the origin of the information) or reputation, to improve trust assessment.
- Imprecision and vagueness are also inherent in AI applications, whose results always have a certain degree of reliability and are often accompanied by their estimated accuracy and precision [5, 7, 26]. This provides some kind of objective confidence in the results of AI applications such as image recognition, product recommendation, traffic prediction, or information extraction from natural language specifications [59]. On top of that, users of these applications may also hold some kind of subjective confidence in their results. For example, the weather forecast for tomorrow may indicate a 55% probability of rain, which can be considered an *objective* confidence. But then, two different users may have different opinions in that prediction and hence in the confidence associated to it, maybe because they do not trust the application equally. For this reason, one user may assign a confidence of 30% to rain (probabilistic confidence), while another user, who trusts the application, will give a 70%. Current proposals for representing degrees of confidence in domain models only allow the assignment of objective confidence to the instance model elements, but are unable to express the subjectivity held by distinct agents about them and to reason about combined opinions.
- In the Software Engineering domain, checking whether a software product meets its specifications is essential, particularly when the requirements involve safety or critical issues. However, product requirements are often expressed in natural language and thus contain vague and ambiguous statements that are difficult to verify [59]. Some proposals use different algorithms to annotate the instance model elements with the accuracy and completeness with which they fulfill the system requirements. For example, in [60] the authors assign a *confidence score* to each element of a Feature Model indicating the probability that textual requirements documents identify that element as relevant to the product line. These confidence scores provide an objective measure of that degree of confidence, although on top of them users may have some subjective judgments (opinions) about the actual relevance of the requirement, or the fitness of the feature to meet the requisite. How to express such

subjective opinions and to fuse them to reach a consensus are still unresolved issues, which our proposal addresses.

With regard to the explicit representation of uncertainty in software models, the survey [63] provides a comprehensive summary of current solutions. Among them, several proposals deal with belief uncertainty, using different theories. For example, probability theory [14, 19] is used in many proposals [1, 4, 6]. Other authors have proposed other approaches including possibility theory (based on fuzzy logic [52, 70]), plausibility (a measure in the Dempster-Shafer theory of evidence [56]) or uncertainty theory [33].

For example, in the context of performance modeling [47] and self-adaptive systems [48], a taxonomy has been proposed to identify and classify different types of uncertainties that may occur in a model. Nevertheless, once identified, the uncertainty must be managed manually by the designer. Moreover, this approach it is unable to capture opinions from different agents or reason about them.

In the field of software architecture, [30] allows designers to evaluate alternative architectures in the presence of incomplete information. This analysis requires cost-benefit estimates for different alternatives as well as information about their dependencies. Using this information, an optimal architecture is computed for a given set of parameters and goals. Meanwhile, [16, 35] allow designers to specify architecture requirements using fuzzy logic. Then, fuzzy inference engines are used to infer the best-fitting architecture from this knowledge. [61] applies similar fuzzy logic techniques to the modeling of quantitative requirements in software development. Again, these methods do not allow the description of different opinions by different agents or reaching conclusions based on combining these opinions.

A deeper comparison among these theories falls out of the scope of this paper, although a detailed discussion can be found in [33]. Our decision was based on the need to explicitly express the degree of uncertainty, which is neglected by these proposals and it is critical when making decisions with some associated risk.

There are other modeling works that deal with belief uncertainty in models, but they usually focus on aspects different from the ones we have addressed here. For instance, some works tackle the uncertainty on the models themselves and on the decision of the right type of models to use depending on the system properties that we want to capture [32]. Other works deal with the uncertainty of the design decisions, of the modeling process, or of the domain being modeled [15, 18, 21, 53]. Our work does not deal with the possible modeling choices; we just treat a model as a set of statements [55] and permit assigning degrees of belief to them. Nevertheless, we believe there is some overlap between belief and design uncertainty that we will explore as part of our further work.

9 CONCLUSIONS AND FUTURE WORK

Belief uncertainty is a type of epistemic uncertainty in which a belief agent is uncertain about the truth of a statement. In this work, we have presented our approach and methodology to capture, represent and operate with belief uncertainty in domain models. In order to achieve our goal, we use Subjective Logic to capture the subjective opinions of the agents involved in the modeling process of a domain, including their degree of belief, disbelief and uncertainty. We have created a UML profile to represent the agents' beliefs in instance models in such a way that the domain model is extended but not altered. In addition, we have addressed the challenge of combining the opinions from different belief agents on the same model elements and have provided a mechanism to operate with these opinions. This mechanism supports decision-making by enabling agents to arrive at global, informed decisions by using optimal strategies to merge individual opinions.

This is of utmost relevance for decision makers, since decisions based on probabilities with low confidence could lead to exorbitant mistakes [51].

The validation performed was aimed at answering the two research questions we initially posed: how can users capture and explicitly represent their opinions about the instances of domain models (RQ1), and then reason about these opinions to reach informed decisions (RQ2). First, we showed how opinions could be expressed in several exemplar applications from different domains, illustrating different applications of our proposal. Next, the experiment conducted has shown how different sets of users could effectively use our UML profile to express their opinions about instance models, and then use the fusion operators to help them reach conclusions successfully, i.e., users were able to successfully apply our methodology and our tool supported the entire process.

As part of our future work, we plan to empirically evaluate both the usability and usefulness of our approach with users in real contexts.

Moreover, we are aware of the UML/OCL benefits to represent and operate with beliefs, but we also understand their drawbacks. Depending on the final user profile, their background and competences, UML may not be the most appropriate base language for our approach . In this sense, we would like to explore different notations such as tailored DSLs, easier to adapt to the vocabulary and technical knowledge of specific user communities.

Finally, there are different reasons for an agent to be uncertain about a particular statement (conflicting sources, second-hand opinions, incomplete information, ...). We are interested in defining a taxonomy for these reasons, in order to improve the annotation of a model. We believe this information can be useful when choosing a particular belief fusion operator. This could also be very useful when extending our approach to other types of models, such as sequence diagrams and activity diagrama that focus more on the dynamics of the system and not its static/data perspective.

ACKNOWLEDGMENTS

We would like to thank the reviewers of the paper for their insightful comments and very valuable suggestions, which have significantly helped us to improve it. This work is partially supported by the Spanish Government unders project LOCOSS (PID2020-114615RB-I00), CoSCA (PGC2018-094905-B-I00) and MBTI4A (P20-00067-FR); and TRANSACT, which has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007260. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Netherlands, Finland, Germany, Poland, Austria, Spain, Belgium, Denmark, and Norway.

REFERENCES

[1] Jacky Akoka, Isabelle Comyn-Wattiau, Stéphane Lamassé, and Cédric du Mouza. 2020. Contribution of Conceptual Modeling to Enhancing Historians' Intuition–Application to Prosopography. In *Conceptual Modeling*. Springer, 164–173.

[2] R. Alroobaea and P. J. Mayhew. 2014. How many participants are really enough for usability studies?. In *Proc. of the 2014 Science and Information Conference (SAI)*. IEEE, 48–56. https://doi.org/10.1109/SAI.2014.6918171

[3] Manuel F. Bertoa, Loli Burgueño, Nathalie Moreno, and Antonio Vallecillo. 2020. Incorporating measurement uncertainty into OCL/UML primitive datatypes. *Softw. Syst. Model.* 19, 5 (2020), 1163–1189. https://doi.org/10.1007/s10270-019-00741-0

[4] Loli Burgueño, Manuel F. Bertoa, Nathalie Moreno, and Antonio Vallecillo. 2018. Expressing Confidence in Models and in Model Transformation Elements. In *MODELS'18*. ACM, 57–66. https://doi.org/10.1145/3239372

[5] Loli Burgueño, Jordi Cabot, and Sébastien Gérard. 2019. An LSTM-Based Neural Network Architecture for Model Transformations. In *Proc. of MODELS'19*. IEEE, 294–299. https://doi.org/10.1109/MODELS.2019.00013

[6] Loli Burgueño, Robert Clarisó, Jordi Cabot, Sébastien Gérard, and Antonio Vallecillo. 2019. Belief uncertainty in software models. In *Proc. of MiSE@ICSE'19*. IEEE, 19–26. https://doi.org/10.1109/MiSE.2019.00011

[7] Loli Burgueño, Robert Clarisó, Sébastien Gérard, Shuai Li, and Jordi Cabot. 2021. An NLP-Based Architecture for the Autocompletion of Partial Domain Models. In *Proc. of CAiSE*, Vol. 12751. Springer, 91–106.

[8] Loli Burgueño, Paula Muñoz, Robert Clariso, Jordi Cabot, Sebastien Gerard, and Antonio Vallecillo. 2021. Belief Fusion Plugin - Git Repository. https://github.com/atenearesearchgroup/belief-fusion-plugin.

[9] Loli Burgueño, Paula Muñoz, Robert Clarisó, Jordi Cabot, and Antonio Vallecillo. 2021. Dealing with beliefs in domain models — companion website. http://atenea.lcc.uma.es/projects/BeliefsInModels

[10] Davide Ceolin, Archana Nottamkandath, and Wan Fokkink. 2013. Semi-automated assessment of annotation trustworthiness. In *11th Annual Conference on Privacy, Security and Trust*. IEEE, 325–332.

[11] Davide Ceolin, Willem Robert Van Hage, Guus Schreiber, and Wan Fokkink. 2013. Assessing trust for determining the reliability of information. In *Situation awareness with systems of systems*. Springer, 209–228.

[12] Marco Console, Paolo Guagliardo, and Leonid Libkin. 2018. Propositional and Predicate Logics of Incomplete Information. In *Proc. of KR'18*. AAAI Press, 592–601.

[13] Andrew Critch. 2016. Credence — using subjective probabilities to express belief strengths. http://acritch.com/credence/

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

[15] Naeem Esfahani and Sam Malek. 2013. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II (LNCS, 7475)*. Springer, 214–238.

[16] Naeem Esfahani, Sam Malek, and Kaveh Razavi. 2013. GuideArch: guiding the exploration of architectural solution space under uncertainty. In *35th International Conference on Software Engineering (ICSE)*. IEEE, 43–52.

[17] Michalis Famelis and Marsha Chechik. 2019. Managing design-time uncertainty. *Softw. Syst. Model.* 18, 2 (2019), 1249–1284. https://doi.org/10.1007/s10270-017-0594-9

[18] Michalis Famelis, Rick Salay, and Marsha Chechik. 2012. Partial Models: Towards Modeling and Reasoning with Uncertainty. In *ICSE'12*. IEEE Press, 573–583.

[19] W. Feller. 2008. *An Introduction to Probability Theory and Its Applications*. Wiley.

[20] Martin Fowler. 2011. *Domain-Specific Languages*. Addison-Wesley.

[21] David Garlan. 2010. Software Engineering in an Uncertain World. In *Proc. of the FoSER Workshop at FSE/SDP 2010*. ACM, 125–128.

[22] Martin Gogolla, Fabian Büttner, and Mark Richters. 2007. USE: A UML-Based Specification Environment for Validating UML and OCL. *Science of Computer Programming* 69 (2007), 27–34.

[23] Martin Gogolla, Antonio Vallecillo, Loli Burgueño, and Frank Hilken. 2015. Employing classifying terms for testing model transformations. In *Proc. of MODELS'15*. IEEE Computer Society, 312–321. https://doi.org/10.1109/MODELS.2015.7338262

[24] Deshuai Han, Qiliang Yang, and Jianchun Xing. 2014. Extending UML for the modeling of fuzzy self-adaptive software systems. In *Proc. of CCDC'14*. IEEE, 2400–2406. https://doi.org/10.1109/CCDC.2014.6852575

[25] Frank Hilken, Martin Gogolla, Loli Burgueño, and Antonio Vallecillo. 2018. Testing models and model transformations using classifying terms. *Software and Systems Modeling* 17, 3 (2018), 885–912. https://doi.org/10.1007/s10270-016-0568-3

[26] Eyke Hüllermeier and Willem Waegeman. 2019. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. arXiv:1910.09457 [cs.LG]

[27] Audun Jøsang. 2001. A Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 9, 3 (2001), 279–212. https://doi.org/10.1142/S0218488501000831

[28] Audun Jøsang. 2016. *Subjective Logic - A Formalism for Reasoning Under Uncertainty*. Springer. https://doi.org/10.1007/978-3-319-42337-1

[29] Audun Jøsang, Dongxia Wang, and Jie Zhang. 2017. Multi-source fusion in subjective logic. In *Proc. of FUSION'17*. IEEE, 1–8. https://doi.org/10.23919/ICIF.2017.8009820

[30] Emmanuel Letier, David Stefan, and Earl T Barr. 2014. Uncertainty, risk, and information value in software requirements and architecture. In *36th International Conference on Software Engineering (ICSE'2014)*. 883–894.

[31] Dayi Lin, Cor-Paul Bezemer, Ying Zou, and Ahmed E Hassan. 2019. An empirical study of game reviews on the Steam platform. *Empirical Software Engineering* 24, 1 (2019), 170–207.

[32] Bev Littlewood, Martin Neil, and Gary Ostrolenk. 1995. The role of models in managing the uncertainty of software-intensive systems. *Reliability Engineering & System Safety* 50, 1 (1995), 87 – 95.

[33] Baoding Liu. 2018. *Uncertainty Theory* (5 ed.). Springer.

[34] José Antonio Hernández López and Jesús Sánchez Cuadrado. 2021. Towards the Characterization of Realistic Model Generators using Graph Neural Networks. In *Proc. of MODELS'21*. IEEE, 58–69. https://doi.org/10.1109/MODELS50736.2021.00015

[35] Ioanna Lytra and Uwe Zdun. 2013. Supporting architectural decision making for systems-of-systems design under uncertainty. In *International Workshop on Software Engineering for Systems-of-Systems*. 43–46.

[36] Miguel Angel Manzano, Helena de Felipe-Rodríguez, and Laura Gago-Gómez. 2018. DICTOMAGRED: Diccionario de Toponimia Magrebí. https://dictomagred.usal.es/

[37] Patricia Martín-Rodilla and Cesar Gonzalez-Perez. 2018. Representing Imprecise and Uncertain Knowledge in Digital Humanities: A Theoretical Framework and ConML Implementation with a Real Case Study. In *Proc. of TEEM'18*. ACM,

863–871. https://doi.org/10.1145/3284179.3284318
[38] Patricia Martín-Rodilla and Cesar Gonzalez-Perez. 2019. Conceptualization and Non-Relational Implementation of Ontological and Epistemic Vagueness of Information in Digital Humanities. *Informatics* 6, 2 (2019), 20. https://doi.org/10.3390/informatics6020020
[39] Patricia Martín-Rodilla, Martin Pereira-Fariña, and Cesar González-Perez. 2019. Qualifying and Quantifying Uncertainty in Digital Humanities: A Fuzzy-Logic Approach. In *Proc. of TEEM'19*. ACM, 788–794. https://doi.org/10.1145/3362789.3362833
[40] Paula Muñoz, Loli Burgueño, Victor Ortiz, and Antonio Vallecillo. 2020. Extending OCL with Subjective Logic. *Journal of Object Technology* 19, 3 (Oct. 2020), 3:1–15. https://doi.org/10.5381/jot.2020.19.3.a1
[41] Jakob Nielsen. 2020. How Many Test Users in a Usability Study? https://www.nngroup.com/articles/how-many-test-users/
[42] Archana Nottamkandath, Jasper Oosterman, Davide Ceolin, Wan J Fokkink, et al. 2014. Automated Evaluation of Crowdsourced Annotations in the Cultural Heritage Domain.. In *URSW*. 25–36.
[43] Object Management Group. 2014. *Object Constraint Language (OCL) Specification. Version 2.4.* OMG Document formal/2014-02-03.
[44] Object Management Group. 2015. *Unified Modeling Language (UML) Specification. Version 2.5.* OMG document formal/2015-03-01.
[45] Object Management Group. 2017. *Precise Semantics for Uncertainty Modeling (PSUM) RFP.* https://www.omg.org/cgi-bin/doc.cgi?ad/2017-12-1 OMG Document ad/2017-12-1.
[46] Victor Ortiz, Loli Burgueño, Antonio Vallecillo, and Martin Gogolla. 2019. Native Support for UML and OCL Primitive Datatypes Enriched with Uncertainty in USE. In *Proc. of OCL@MODELS'19 (CEUR Workshop Proceedings, Vol. 2513)*. CEUR-WS.org, 59–66. http://ceur-ws.org/Vol-2513/paper5.pdf
[47] Diego Perez-Palacin and Raffaela Mirandola. 2014. Dealing with uncertainties in the performance modelling of software systems. In *10th International ACM Sigsoft Conference on Quality of Software Architectures*. 33–42. https://doi.org/10.1145/2602576.2602582
[48] Diego Perez-Palacin and Raffaela Mirandola. 2014. Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation. In *5th ACM/SPEC International Conference on Performance Engineering*. 3–14.
[49] Paul Ralph, Sebastian Baltes, Domenico Bianculli, Yvonne Dittrich, Michael Felderer, Robert Feldt, Antonio Filieri, Carlo Alberto Furia, Daniel Graziotin, Pinjia He, Rashina Hoda, Natalia Juristo, Barbara A. Kitchenham, Romain Robbes, Daniel Méndez, Jefferson Molleri, Diomidis Spinellis, Miroslaw Staron, Klaas-Jan Stol, Damian A. Tamburri, Marco Torchiano, Christoph Treude, Burak Turhan, and Sira Vegas. 2020. ACM SIGSOFT Empirical Standards. *CoRR* abs/2010.03525 (2020).
[50] Jeff Rubin and Dana Chisnell. 2008. *Handbook of Usability Testing. Second Edition: How to Plan, Design, and Conduct Effective Tests*. Wiley Publishing, Inc.
[51] Stuart Russell and Frank Chen. 2020. Controlling AI. podcast. https://a16z.com/2020/01/16/controlling-ai-human-compatible/
[52] Stuart J. Russell and Peter Norvig. 2010. *Artificial Intelligence. A Modern Approach* (3 ed.). Prentice Hall.
[53] Rick Salay, Marsha Chechik, Jennifer Horkoff, and AlessioDi Sandro. 2013. Managing requirements uncertainty with partial models. *Requirements Eng.* 18, 2 (2013), 107–128.
[54] Markus Scheidgen. 2015. Generation of Large Random Models for Benchmarking. In *Proc. of BigMDE @ STAF'15 (CEUR Workshop Proceedings, Vol. 1406)*. CEUR-WS.org, 1–10.
[55] Ed Seidewitz. 2003. What Models Mean. *IEEE Software* 20, 5 (Sept. 2003), 26–32. https://doi.org/10.1109/MS.2003.1231147
[56] Glenn Shafer. 1976. *A Mathematical Theory of Evidence*. Princeton University Press.
[57] Muhammad Luqman Mohd Shafie, Wan Mohd Nasir Wan-Kadir, Horst Lichter, Muhammad Khatibsyarbini, and Mohd Adham Isa. 2022. Model-based test case generation and prioritization: a systematic literature review. *Softw. Syst. Model.* 21, 2 (2022), 717–753. https://doi.org/10.1007/s10270-021-00924-8
[58] Jie Sheng, Li Yan, and Zongmin Ma. 2019. 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)*. Springer, 589–600. https://doi.org/10.1007/978-3-030-21920-8_52
[59] Anjali Sree-Kumar, Elena Planas, and Robert Clarisó. 2018. Extracting software product line feature models from natural language specifications. In *International Systems and Software Product Line Conference (SPLC'2018)*. ACM, 43–53. https://doi.org/10.1145/3233027.3233029
[60] Anjali Sree-Kumar, Elena Planas, and Robert Clarisó. 2021. Validating Feature Models With Respect to Textual Product Line Specifications. In *Proc. of VaMoS'21*. ACM, 1–8. https://doi.org/10.1145/3442391.3442407
[61] Dan E. Tamir, Carl J. Mueller, and Abraham Kandel. 2016. Complex Fuzzy Logic Reasoning-Based Methodologies for Quantitative Software Requirements Specifications. In *Computational Intelligence and Quantitative Software Engineering*. Springer, 153–172.

[62] Juha-Pekka Tolvanen and Steven Kelly. 2018. Effort Used to Create Domain-Specific Modeling Languages. In *Proc. of MODELS'18*. ACM, 235–244. https://doi.org/10.1145/3239372.3239410

[63] Javier Troya, Nathalie Moreno, Manuel F. Bertoa, and Antonio Vallecillo. 2021. Uncertainty representation in software models: A survey. *Software and Systems Modeling* (2021). https://doi.org/10.1007/s10270-020-00832-3

[64] Carl W. Turner, James R. Lewis, and Jakob Nielsen. 2006. *Determining Usability Test Sample Size* (2 ed.). Vol. 3. CRC Press, 3084–3088.

[65] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A taxonomy of model-based testing approaches. *Softw. Test. Verification Reliab.* 22, 5 (2012), 297–312. https://doi.org/10.1002/stvr.456

[66] Rens Wouter van der Heijden, Henning Kopp, and Frank Kargl. 2018. Multi-Source Fusion Operations in Subjective Logic. In *Proc. of FUSION'18*. IEEE, 1990–1997. https://doi.org/10.23919/ICIF.2018.8455615

[67] Markus Voelter, Sebastian Benz, Christian Dietrich, Birgit Engelmann, Mats Helander, Lennart C. L. Kats, Eelco Visser, and Guido Wachsmuth. 2013. *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org. http://www.dslbook.org

[68] Claes Wohlin, Martin Höst, and Kennet Henningsson. 2003. Empirical Research Methods in Software Engineering. In *Empirical Methods and Studies in Software Engineering, Experiences from ESERNET*, Reidar Conradi and Alf Inge Wang (Eds.). Vol. 2765. 7–23.

[69] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer.

[70] Hans-Juergen Zimmermann. 2001. *Fuzzy Set Theory – and Its Applications* (4 ed.). Springer Science+Business Media.