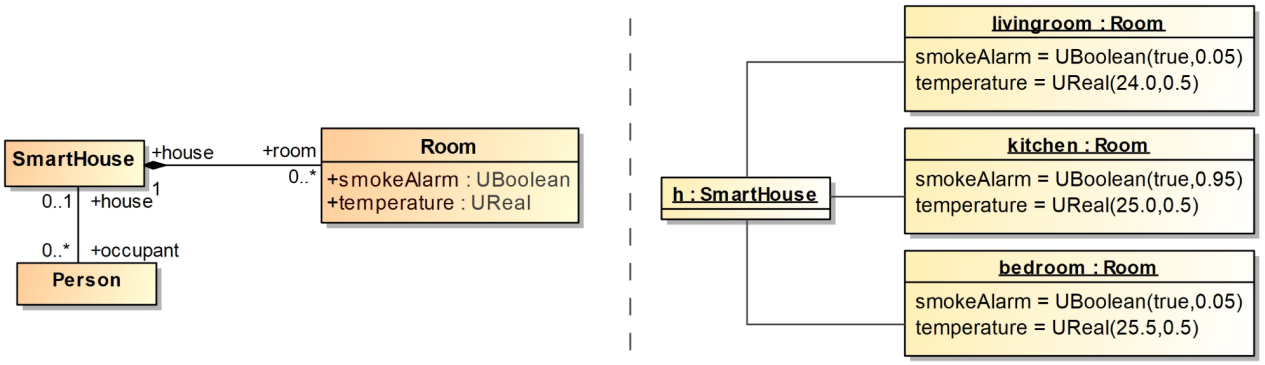
Exercise 1A: Smart rooms - used to illustrate the use of the UML Profile

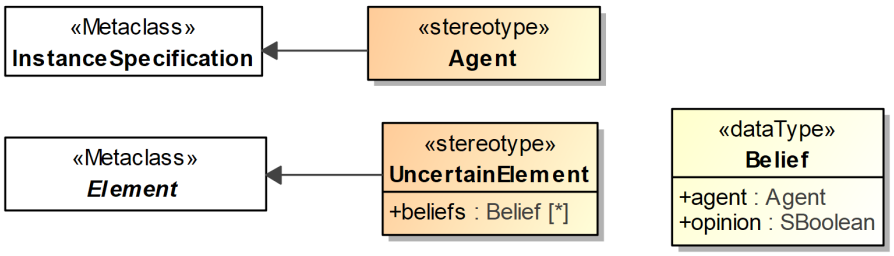
Let us suppose a smart house whose rooms are equipped with a temperature sensor and a smoke alarm. According to their manufacturers' information, the accuracy of the temperature sensor is ± 0.5 degrees and the reliability of the smoke alarm is 5%.

The diagrams below show the conceptual model of the system (left) and an object model that describes a house and three of its rooms (right). Note the use of the UML extended datatypes `UReal` and `UBoolean` that represent Real and Boolean values enriched with uncertainty [Bertoa et al, 2020].

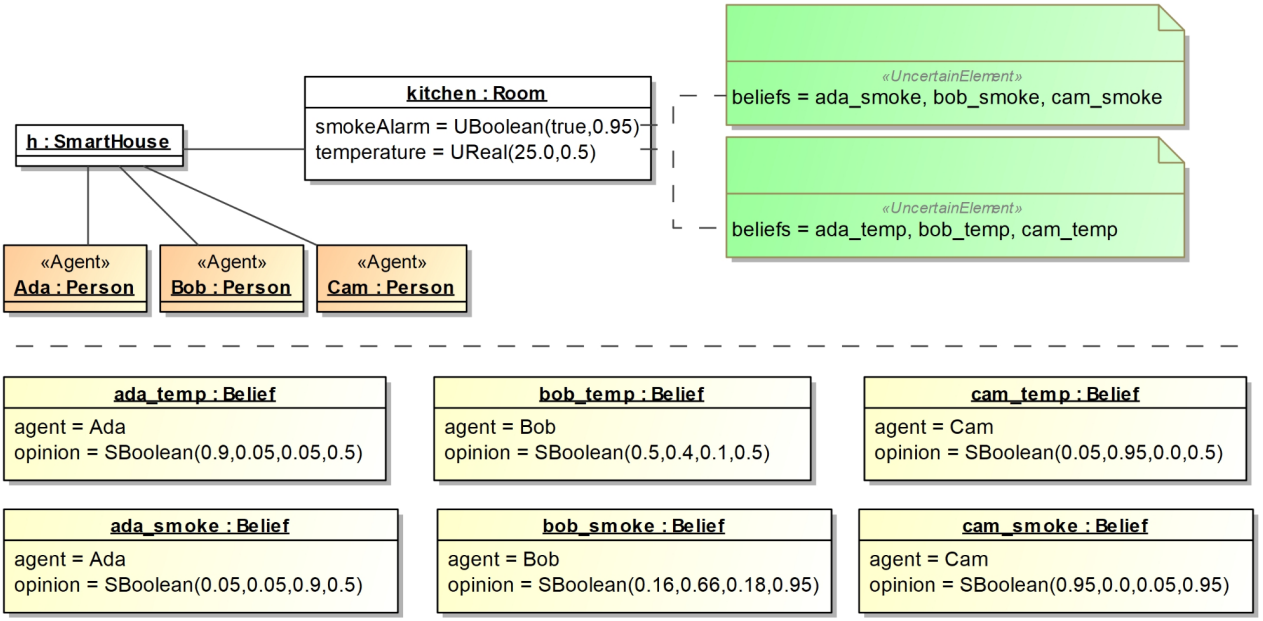


In many situations, the users of such systems also associate subjective uncertainty with that of the sources. For example, one of the house occupants, Ada, does not trust the temperature readings of the kitchen because he knows that the sensor is close to the oven and therefore the measurements can be fairly unreliable. So Ada would add some subjective uncertainty to the objective results of the sensor's measurements. Of course, this type of subjective uncertainty assigned to the same sensor by different users can vary, depending on their personal history, experiences, and beliefs (e.g., their individual level of trust in the manufacturer). Thus, other occupants may have different opinions, or trust, on the sensors' readings and consequently on their reactions to those, i.e., whether to think that there is a fire and hence call the fire brigade, or simply to ignore the alarm.

To add subjective opinions to the model elements we will use the Belief Uncertainty UML Profile, shown below. Stereotype `Agent` can be attached to users that can express opinions about the model elements. Stereotype `UncertainElement` is used to indicate that a model element is subject to the opinion of one or more agents, and it holds in its tag value `beliefs` the set of opinions by the agents, expressed as pairs (agent, opinion). Opinions are expressed by means of subjective logic binominal opinions.



Using this profile, an example of how to represent the opinions of the three house occupants (Ada, Bob and Cam) about the sensors in the **`kitchen`** room is shown in the following object diagram.



We can see how the three occupants have different subjective opinions on the values of the **`temperature`** and **`smokeAlarm`** slots of this object: Ada is quite confident in her opinions about the temperature readings but very uncertain about the smoke alarm. Bob is unsure about the temperature sensor and does not believe in the smoke alarm. Cam trusts in both devices.

Exercise 1A.

Once you have read this explanations and the example description, please respond to Questionnaire 1A available at https://forms.gle/dyDJHEUr6xbZS8sc9

Exercise 1B: Hotel Recommendations

1.1. Introduction

Let us suppose a User who wants to book a hotel in a given city to attend a conference, and decides to employ the services of booking portals such as BookingDotCom or Kayak to get the best deal that matches their preferences.

The class diagram below shows a model with the main entities involved in the application.

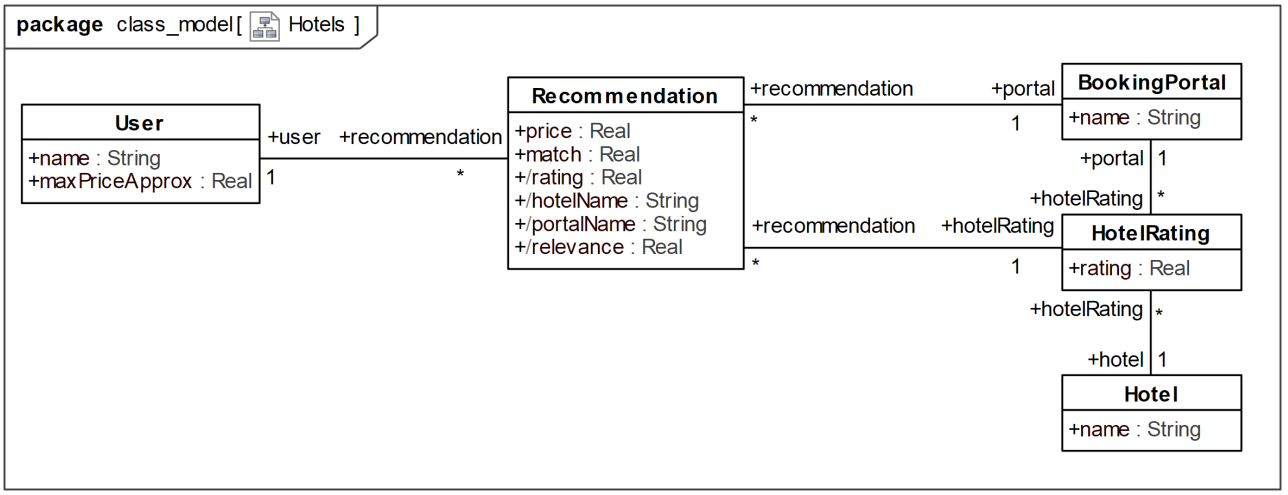


Fig. 1. Class diagram with the main entities of the HotelRecommendation system.

`Users` indicate their names and the maximum price per night they are happy to pay for a single room. Although prices are represented as `Real` numbers, in all calculations we allow for ±5% margins to express the typical tolerance we normally permit. BookingPortals have a selection of `Hotels`, for which they maintain a rating. Such a relationship between the `BookingPortals` and the `Hotels` is modeled by class `HotelRating`. Ratings are expressed as numbers in the range 0-10, with 10 indicating complete satisfaction and maximum level of quality. For a given user, a portal can provide several recommendations. Each `Recommendation` contains several attributes:

- **`hotelName`**: a derived attribute with the name of the recommended hotel. The derivation formula is expressed in OCL as: `self.hotelRating.hotel.name`
- **portalName**: a derived attribute with the name of the recommending portal. The derivation formula is: `self.portal.name`
- **`rating`**: a derived attribute with the rating of the hotel according to the portal (which in turn is normally based on the scores given by previous users of that portal about that hotel). The derivation formula in this case is: `self.hotelRating.rating`
- **`price`**: the recommended price per night for the room in that hotel (±5%).
- **`match`**: the level of correspondence between what is required by the user and what is offered by the hotel, according to the booking portal's calculations. It is represented as a Real number between 0 and 100, with 100 representing 100% correspondence. This attribute is common in service providers like Netflix, for instance.
- **`relevance`**: an aggregated indicator that computes the global expected level of suitability and satisfaction of the recommendation, based on the hotel rating, price, and match. It is a derived attribute whose derivation expression is the following:
  `100*(match/100)*(rating/10)*(price <= self.user.maxPriceApprox)`
  Note that in this formula, operator "<=" does not return true or false, but the probability that the left operand is less than the right one. This is because prices are not expressed as crisp values, but contain some margins, i.e., ±5%.

Thus, users can base their decisions on the value of attributes **`relevance`**, which can be used to sort the set of recommendations produced by the portals.

To show an example, the following object diagram shows a user ("You") who obtains four different recommendations by three booking portals, about three different hotels.

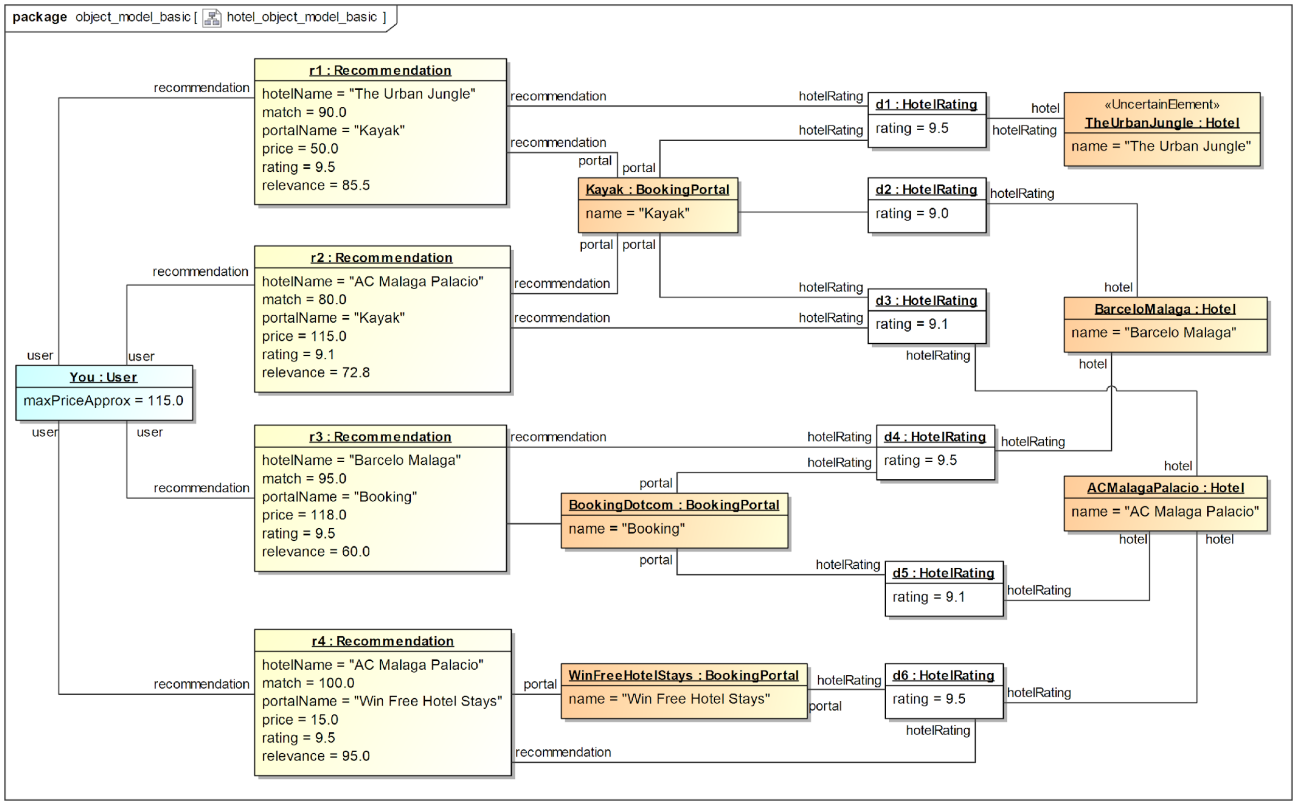


Fig. 2. An object diagram with four recommendations for one user.

The value of attribute `relevance` is automatically derived in all cases and the best option seems to be the last one, by which the portal "Win Free Hotel Stays" recommends staying at the "AC Malaga Palacio" hotel, with a match of 100% and a price per night of 15 Euros.

1.2. Assigning degrees of beliefs to model elements

By looking at this object model, we soon realize that we are not fully sure about some of the elements of the model. For example, the price offered by the last portal for the AC Malaga Hotel seems to be very low. This does not mean that it is false, but let's say we are not fully confident about it. In general, systems that represent information coming from different sources have to deal with data that sometimes is missing, inaccurate, vague, or even inconsistent. This can be due to unreliable sources, lack of knowledge, imprecise information, faulty data, or even malicious or fraudulent services.

In this work, we are interested in representing, characterizing, and processing vague and imprecise information in UML software models, considering the stakeholders' opinions and beliefs.

We aim at associating "beliefs" to model elements, and how to propagate and operate with their associated uncertainty so that domain experts can reason about software models enriched with individual opinions.

To represent degrees of belief we use Subjective Logic [Jos16] which is an extension of both Boolean and Probabilistic logics to account for uncertainty. Values in this logic are 4-tuples called "opinions." More precisely, given a Boolean logic predicate x, an opinion o(x) is the 4-tuple (b,d,u,a) where:

- b (belief) is the degree of belief that x is true.
- d (disbelief) is the degree of belief that x is false
- u (uncertainty) is the degree of uncertainty about x, i.e., the amount of uncommitted belief.
- a (base rate) is the prior probability of x without any previous evidence.

All four components of the tuple are in the range [0,1] and satisfy that b + d + u = 1.

Boolean logic is embedded in Subject logic by making the value "true" correspond to opinion (1,0,0,1), and false to (0,1,0,0). Probabilistic logic (i.e., a logic whose values are probabilities) is embedded in Subject logic by making a value p (with 0 <= p <= 1) correspond to opinion (p, 1 – p, 0, p). Given an opinion (b,d,u,a), its "*projection*" enables defining a projection from Subjective logic to Probabilistic logic. The projection of an opinion is defined by the formula $P = b + a*u$.

To show some examples, the following list shows some of the possible values that we could assign to opinions in Subjective logic:

- CERTAIN = SBoolean(1.0, 0.0, 0.0, 0.5) -- projection: 1.0
- PROBABLE = SBoolean(0.667, 0.0, 0.333, 0.5) -- projection: 0.8335
- POSSIBLE = SBoolean(0.333, 0.0, 0.667, 0.5) -- projection: 0.665
- UNCERTAIN = SBoolean(0.0, 0.0, 1.0, 0.5) -- projection: 0.5
- IMPROBABLE = SBoolean(0.0, 0.333, 0.667, 0.5) -- projection: 0.335
- UNLIKELY = SBoolean(0.0, 0.667, 0.333, 0.5) -- projection: 0.1665
- IMPOSSIBLE = SBoolean(0.0, 0.0, 1.0, 0.5) -- projection: 0.0

To assign degrees of beliefs to UML model elements we have defined a UML profile, which is shown below:

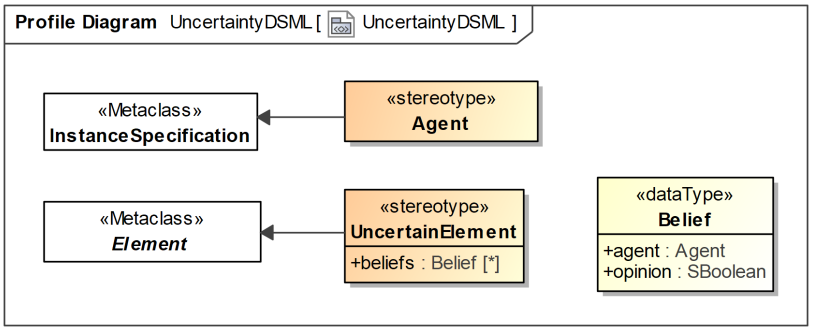


Fig. 3. The UML Profile for expressing degrees of belief on UML model elements.

Stereotype `Agent` is used to indicate the entities that hold the opinions, which can be domain experts, users, modelers, or any other model stakeholder. Stereotype `UncertainElement` is used to mark a model element as uncertain and assign a degree of belief to it. For this, datatype `Belief` represents a pair `(agent:Agent, opinion:SBoolean)` that describes the agent holding an opinion, and the opinion held by that agent. We can see how an uncertain element can be assigned a set of beliefs, so using this stereotype different agents can express their opinion about the same model element.

This profile is normally used to assign degrees of belief either to entity instances or to attributes' values. In the former case, the degree of belief expresses how sure we are about the actual occurrence (or existence) of the instance. When applied to attributes' values, it means how sure or unsure we are about these values. Note that assigning no belief to an element is equivalent to assigning a TRUE belief (i.e. complete certainty) to that model element. The following model shows an example of the application of the profile on two of the `HotelRecommendation` model elements:

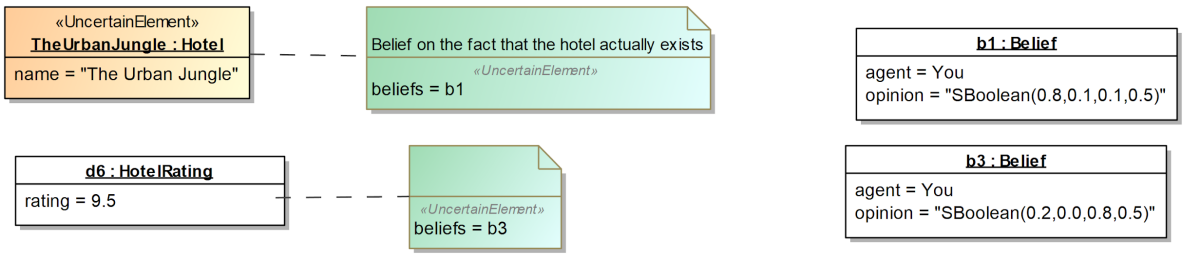


Fig. 4. Examples of application of the UML profile on some model elements.

Exercise 1B.

1. Given the object model shown in Figure 2, please associate degrees of belief to those model elements that you are not completely certain about according to your own judgment (you could even assign a degree of belief to your own specified maximum price if you wish). Use the UML profile defined in Figure 3 to accomplish this task.
2. Then, please respond to Questionnaire 1B available at https://forms.gle/NbT5evRRcbihtK7h7

The MagicDraw file is available at https://www.lcc.uma.es/~av/BUProfile/models/HotelRecommendations.mdzip.
You also need to download the BeliefUncertainty UML profile (which available at https://www.lcc.uma.es/~av/BUProfile/models/BeliefUncertaintyProfile.mdzip), and place it in the same folder.