Hotel Recommendations – Full example

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.

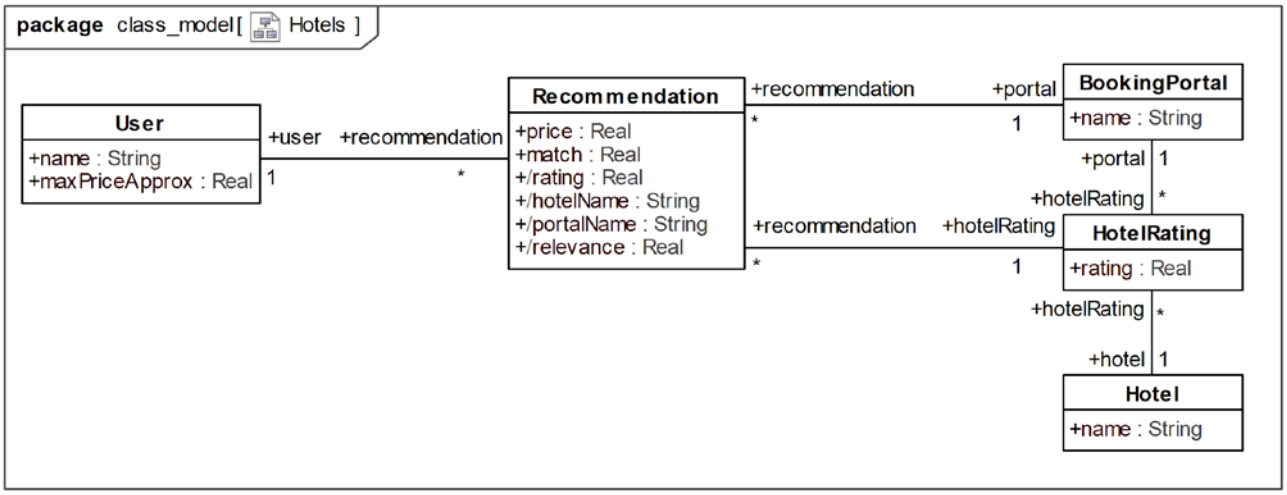


Fig. 1. Class diagram with the main entities of the HotelRecommendation system.

`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.

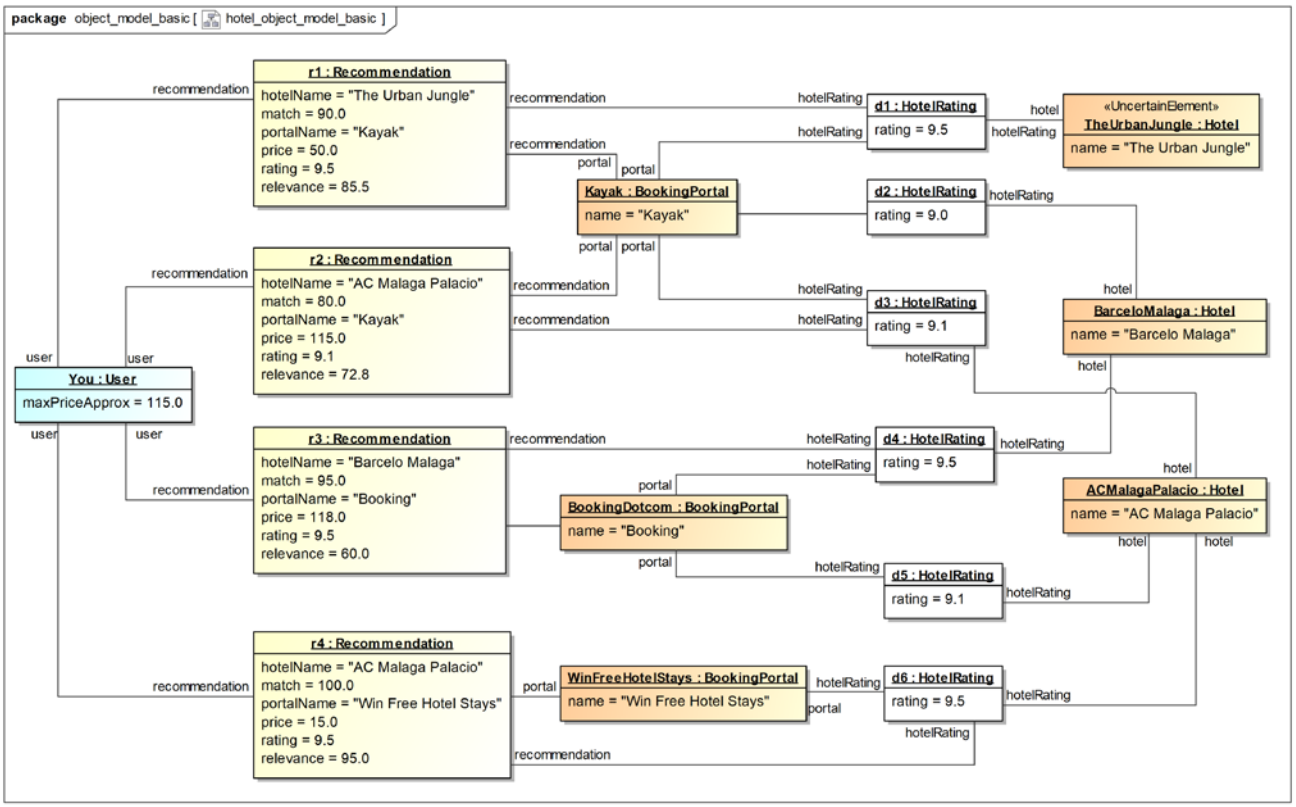


Fig. 2. An object diagram with four recommendations for one user.

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 to stay at the “AC Malaga Palacio” hotel, with a match of 100% and a price per night of 15 Euros.

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:

- TRUE = SBoolean(1.0, 0.0, 0.0, 1.0) -- 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
- FALSE = SBoolean(0.0, 0.0, 1.0, 0.0) -- projection: 0.0

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

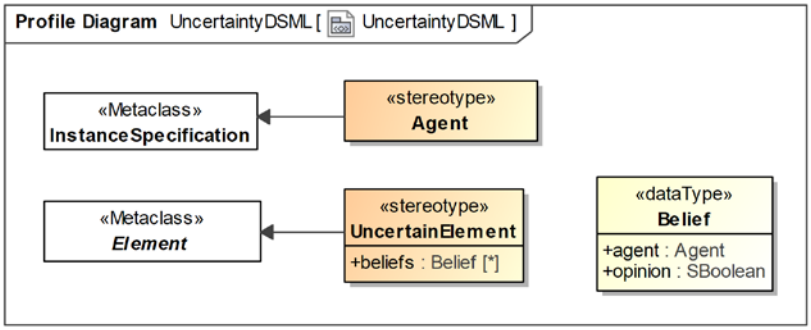


Fig. 3. The UML Profile for expressing degrees of belief on UML model elements.

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 on 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:

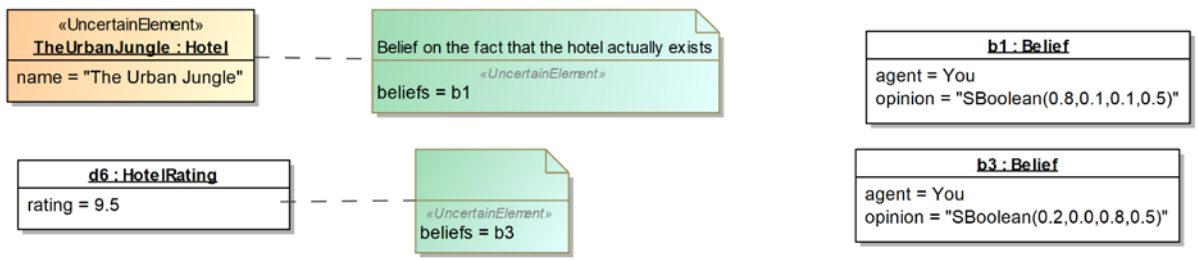
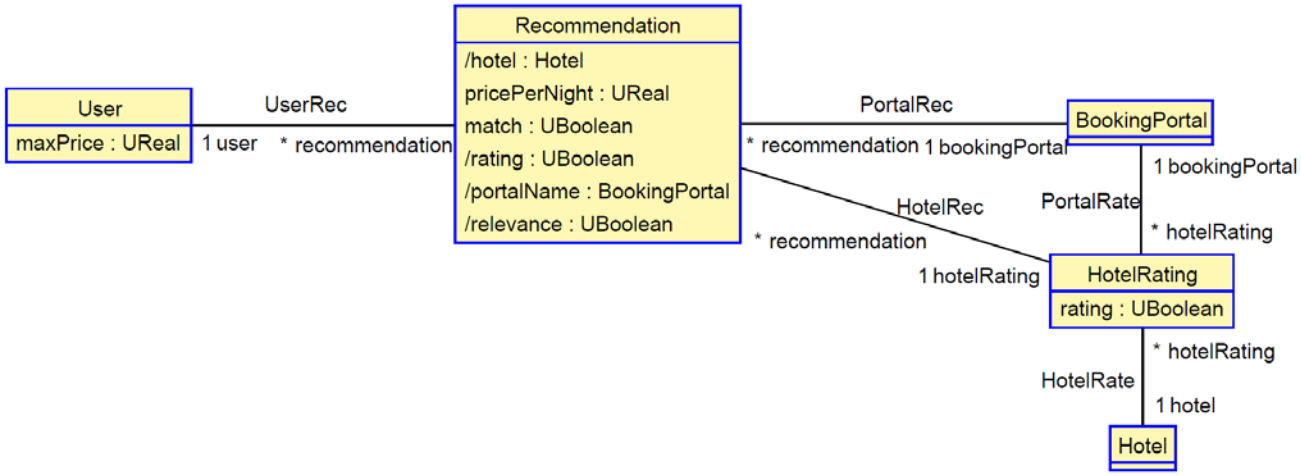
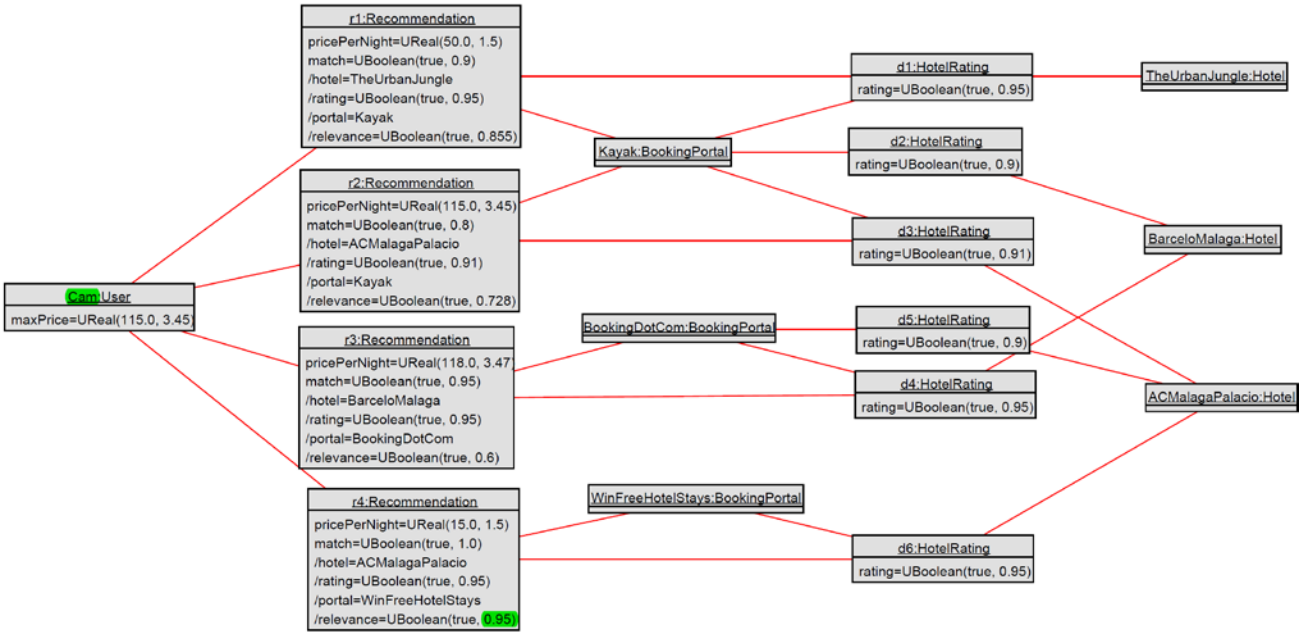


Fig. 4. Examples of application of the UML profile on some model elements.

An alternative representation does not use the profile, but additional attributes are added to store the agents' opinions. This representation will be shown here using the tool USE. Let's start with the model of the system, which is the same as shown above.



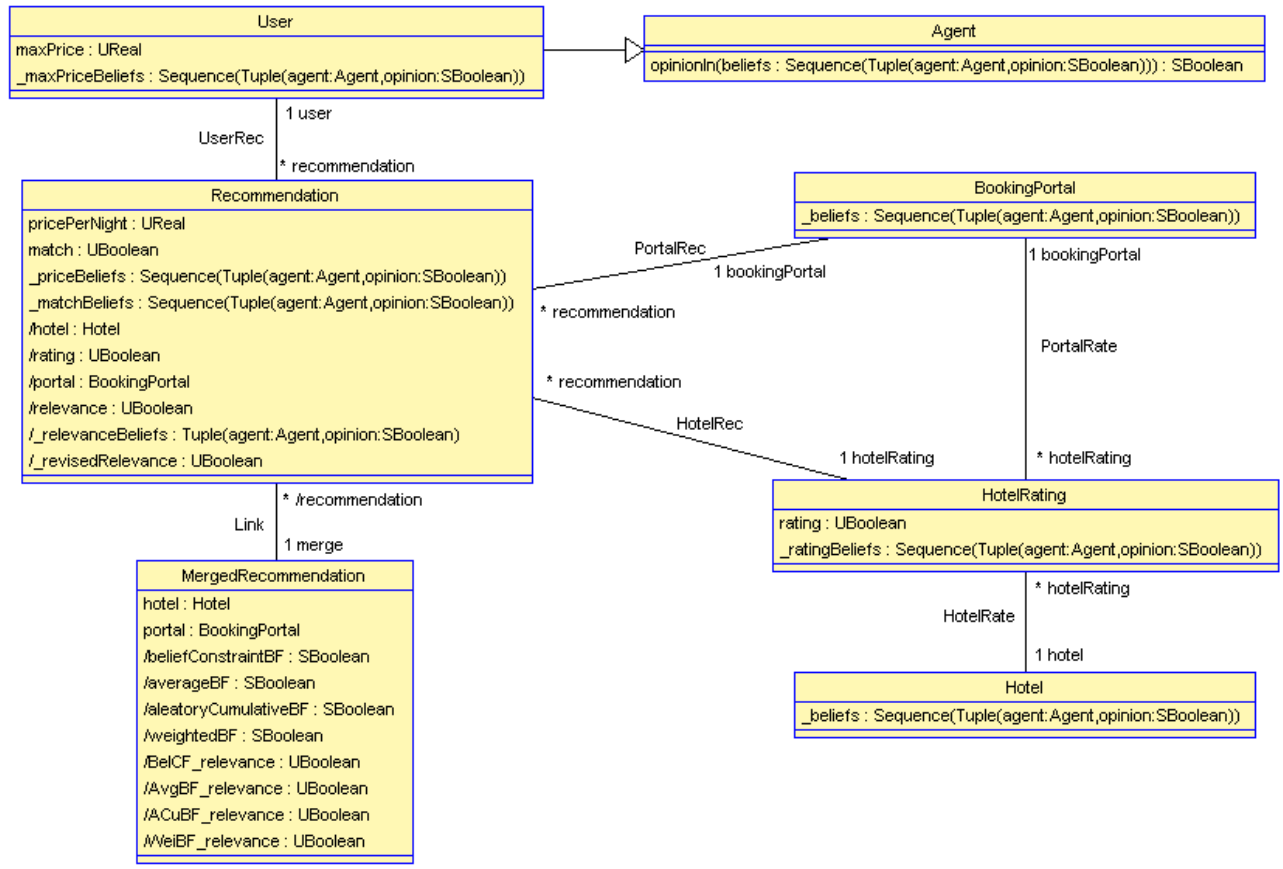
The, the recommendation for a given user, **Cam**, could be as follows:



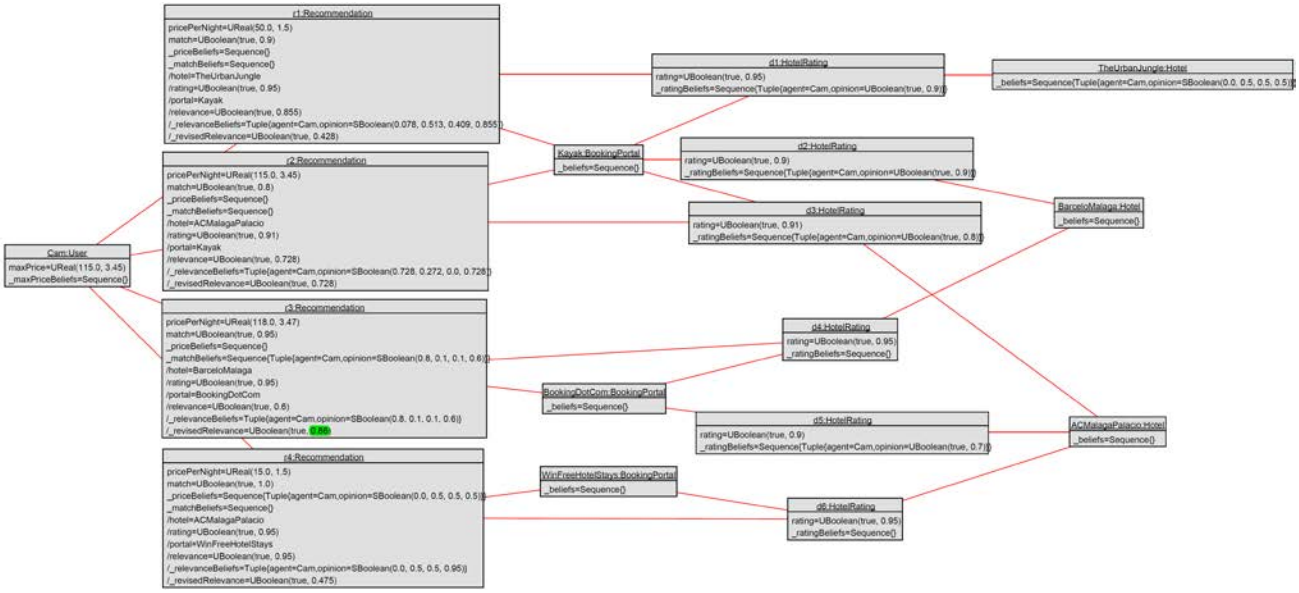
According to that model, Cam is recommended to book through the WinFreeHotelStays portal, and stay at the AC Malaga Palacio, with a price of 15 euros per night.

However, Cam keeps pondering whether this is indeed a valid recommendation, because the price seems suspiciously low, and she has never heard of that booking portal. She has never heard of the hotel called "TheUrbanJungle" either. So she decides to assign beliefs to some of these objects and the values of attributes, before committing to book any hotel.

To add information about the users' beliefs, we define who are the belief agents (in this case, all users are agents) and add an attribute for each element that is subject to vagueness. This results in the extended model of the system.



Then, using this model, she annotates the model with opinions, and the situation changes. Now, the best offer is through BookingDotCom, to stay at the BarceloMalaga. Cam feels more confident now about her booking, and also about the hotel she will stay in.



Suppose now that Ada and Bob, two friends of Cam, decided to attend the conference, and also got the same recommendations from the same booking portals. However, they also expressed their opinions on some of the model elements and they have a revised set of relevance ratings for the hotels according to their opinions. This is summarized in the model shown below.

Note that when their individual opinions are added to the model, Ada prefers to go to the hotel TheUrbanJungle, Bob to the AC Malaga Palacio, and Cam to the Barcelo hotel.

3. Combining opinions from different agents

Imagine that the three friends decide to stay to the same hotel, if possible. The three models decorated with the opinions of all users can be utilized to try to reach a consensus about where to stay, if possible.

To accomplish this task, Subjective Logic provides a set of operators that allow merging opinions by different agents about the same model elements. These are the so-called “fusion operators.” This is of paramount importance for permitting collaborative modeling and enabling cooperative work between the agents, currently a strong requirement for many systems.

Each fusion operator was designed for a specific purpose. Depending on the situation, the modeler has to decide which fusion operator is the most suitable one. They can be classified according to two main categories:

Willingness to compromise. The **Belief Constraint Fusion** (BCF) operator can be used when the agents have already committed their choices and will not change their minds, at the potential cost of not being able to reach a consensus. In contrast, the **Consensus and Compromise Fusion** (CCF) transforms conflicting opinions into vague beliefs, being suitable in situations where we look for consensus if it exists, and for a vague opinion otherwise.

Cumulative information. The **Aleatory** and **Epistemic Cumulative Belief Fusion** (aCBF and eCBF, resp.) operators are more suitable in situations where the amount of independent evidence increases when more sources are included (i.e., when more agents give their opinion). Contrarily, the **Averaging Belief Fusion** (ABF) operator is better when some dependency is assumed between sources and more sources will not necessarily mean more evidence, i.e., more agents providing opinions does not imply being closer to the truth because each agent has their own perception about the observed fact. Finally, the **Weighted Belief Fusion** (WBF) operator also assumes dependency between sources but the opinions are weighted depending on their confidence, i.e., the stronger the confidence (or, equivalently, the smaller the uncertainty), the higher the weight.

The following model shows the results of fusing the opinions. In this case, all operators coincide that the best option for the three friends to stay in at the AC Malaga Palacio hotel.


u1Kayak:MergedRecommendation
hotel=TheUrbanJungle
portal=Kayak
/beliefConstraintBF=SBoolean(0.46, 0.3, 0.239, 0.836)
/averageBF=SBoolean(0.24, 0.224, 0.536, 0.823)
/aleatoryCumulativeBF=SBoolean(0.373, 0.349, 0.278, 0.76)
/weightedBF=SBoolean(0.211, 0.286, 0.504, 0.834)
/BelCF_relevance=UBoolean(true, 0.86)
/AvgBF_relevance=UBoolean(true, 0.681)
/ACuBF_relevance=UBoolean(true, 0.584)
/WeiBF_relevance=UBoolean(true, 0.631)
r1Ada:Recommendation
pricePerNight=UReal(50.0, 1.5)
match=UBoolean(true, 0.8)
_priceBeliefs=Sequence{Tuple(agent=Ada,opinion=SBoolean(0.0, 0.2, 0.8, 0.5))}
_matchBeliefs=Sequence{}
/hotel=TheUrbanJungle
/rating=UBoolean(true, 0.95)
/portal=Kayak
/relevance=UBoolean(true, 0.76)
/_relevanceBeliefs=Tuple(agent=Ada,opinion=SBoolean(0.291, 0.0, 0.709, 0.76))
/_revisedRelevance=UBoolean(true, 0.83)
r1Bob:Recommendation
pricePerNight=UReal(50.0, 1.5)
match=UBoolean(true, 0.6)
_priceBeliefs=Sequence{Tuple(agent=Bob,opinion=SBoolean(0.2, 0.0, 0.8, 0.5))}
_matchBeliefs=Sequence{}
/hotel=TheUrbanJungle
/rating=UBoolean(true, 0.95)
/portal=Kayak
/relevance=UBoolean(true, 0.855)
/_relevanceBeliefs=Tuple(agent=Bob,opinion=SBoolean(0.425, 0.0, 0.575, 0.855))
/_revisedRelevance=UBoolean(true, 0.917)
r1Cam:Recommendation
pricePerNight=UReal(50.0, 1.5)
match=UBoolean(true, 0.9)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{}
/hotel=TheUrbanJungle
/rating=UBoolean(true, 0.95)
/portal=Kayak
/relevance=UBoolean(true, 0.855)
/_relevanceBeliefs=Tuple(agent=Cam,opinion=SBoolean(0.078, 0.513, 0.409, 0.855))
/_revisedRelevance=UBoolean(true, 0.428)
Ada:User
maxPrice=UReal(110.0, 5.0)
_maxPriceBeliefs=Sequence{}
AChayak:MergedRecommendation
hotel=ACMalagaPalacio
portal=Kayak
/beliefConstraintBF=SBoolean(0.988, 0.012, 0.0, 0.637)
/averageBF=SBoolean(0.728, 0.272, 0.0, 0.728)
/aleatoryCumulativeBF=SBoolean(0.728, 0.272, 0.0, 0.637)
/weightedBF=SBoolean(0.728, 0.272, 0.0, 0.637)
/BelCF_relevance=UBoolean(true, 0.988)
/AvgBF_relevance=UBoolean(true, 0.728)
/ACuBF_relevance=UBoolean(true, 0.728)
/WeiBF_relevance=UBoolean(true, 0.728)
r2Ada:Recommendation
pricePerNight=UReal(115.0, 3.45)
match=UBoolean(true, 0.9)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{}
/hotel=ACMalagaPalacio
/portal=Kayak
/rating=UBoolean(true, 0.91)
/relevance=UBoolean(true, 0.439)
/_relevanceBeliefs=Tuple(agent=Ada,opinion=SBoolean(0.703, 0.097, 0.2, 0.439))
/_revisedRelevance=UBoolean(true, 0.791)
r2Bob:Recommendation
pricePerNight=UReal(115.0, 3.45)
match=UBoolean(true, 0.7)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{}
/hotel=ACMalagaPalacio
/rating=UBoolean(true, 0.91)
/portal=Kayak
/relevance=UBoolean(true, 0.637)
/_relevanceBeliefs=Tuple(agent=Bob,opinion=SBoolean(0.9, 0.0, 0.1, 0.637))
/_revisedRelevance=UBoolean(true, 0.964)
r2Cam:Recommendation
pricePerNight=UReal(115.0, 3.45)
match=UBoolean(true, 0.8)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{}
/hotel=ACMalagaPalacio
/rating=UBoolean(true, 0.91)
/portal=Kayak
/relevance=UBoolean(true, 0.728)
/_relevanceBeliefs=Tuple(agent=Cam,opinion=SBoolean(0.728, 0.272, 0.0, 0.728))
/_revisedRelevance=UBoolean(true, 0.728)
Bob:User
maxPrice=UReal(120.0, 4.0)
_maxPriceBeliefs=Sequence{}
ACBooking:MergedRecommendation
hotel=BarceloMalaga
portal=BookingDotCom
/beliefConstraintBF=SBoolean(0.799, 0.201, 0.0, 0.5)
/averageBF=SBoolean(0.57, 0.43, 0.0, 0.57)
/aleatoryCumulativeBF=SBoolean(0.57, 0.43, 0.0, 0.256)
/weightedBF=SBoolean(0.57, 0.43, 0.0, 0.256)
/BelCF_relevance=UBoolean(true, 0.799)
/AvgBF_relevance=UBoolean(true, 0.57)
/ACuBF_relevance=UBoolean(true, 0.57)
/WeiBF_relevance=UBoolean(true, 0.57)
r3Ada:Recommendation
pricePerNight=UReal(118.0, 3.47)
match=UBoolean(true, 0.8)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{Tuple(agent=Ada,opinion=SBoolean(0.8, 0.1, 0.1, 0.8))}
/hotel=BarceloMalaga
/rating=UBoolean(true, 0.95)
/portal=BookingDotCom
/relevance=UBoolean(true, 0.256)
/_relevanceBeliefs=Tuple(agent=Ada,opinion=SBoolean(0.341, 0.559, 0.1, 0.256))
/_revisedRelevance=UBoolean(true, 0.366)
r3Bob:Recommendation
pricePerNight=UReal(118.0, 3.47)
match=UBoolean(true, 0.6)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{}
/hotel=BarceloMalaga
/rating=UBoolean(true, 0.95)
/portal=BookingDotCom
/relevance=UBoolean(true, 0.57)
/_relevanceBeliefs=Tuple(agent=Bob,opinion=SBoolean(0.57, 0.43, 0.0, 0.57))
/_revisedRelevance=UBoolean(true, 0.57)
r3Cam:Recommendation
pricePerNight=UReal(118.0, 3.47)
match=UBoolean(true, 0.95)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{Tuple(agent=Cam,opinion=SBoolean(0.8, 0.1, 0.1, 0.8))}
/hotel=BarceloMalaga
/rating=UBoolean(true, 0.95)
/portal=BookingDotCom
/relevance=UBoolean(true, 0.6)
/_relevanceBeliefs=Tuple(agent=Cam,opinion=SBoolean(0.8, 0.1, 0.1, 0.8))
/_revisedRelevance=UBoolean(true, 0.96)
Cam:User
maxPrice=UReal(115.0, 3.45)
_maxPriceBeliefs=Sequence{}
ACWin:MergedRecommendation
hotel=ACMalagaPalacio
portal=WinFreeHotelStays
/beliefConstraintBF=SBoolean(0.0, 0.875, 0.125, 0.77)
/averageBF=SBoolean(0.0, 0.5, 0.5, 0.76)
/aleatoryCumulativeBF=SBoolean(0.0, 0.75, 0.25, 0.95)
/weightedBF=SBoolean(0.0, 0.5, 0.5, 0.76)
/BelCF_relevance=UBoolean(true, 0.096)
/AvgBF_relevance=UBoolean(true, 0.36)
/ACuBF_relevance=UBoolean(true, 0.238)
/WeiBF_relevance=UBoolean(true, 0.36)
r4Ada:Recommendation
pricePerNight=UReal(15.0, 1.5)
match=UBoolean(true, 0.9)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{Tuple(agent=Ada,opinion=SBoolean(0.0, 0.5, 0.5, 0.5))}
/hotel=ACMalagaPalacio
/rating=UBoolean(true, 0.95)
/portal=WinFreeHotelStays
/relevance=UBoolean(true, 0.855)
/_relevanceBeliefs=Tuple(agent=Ada,opinion=SBoolean(0.0, 0.5, 0.5, 0.855))
/_revisedRelevance=UBoolean(true, 0.428)
r4Bob:Recommendation
pricePerNight=UReal(90.0, 0.1)
match=UBoolean(true, 0.5)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{Tuple(agent=Bob,opinion=SBoolean(0.0, 0.5, 0.5, 0.5))}
/hotel=ACMalagaPalacio
/rating=UBoolean(true, 0.95)
/portal=WinFreeHotelStays
/relevance=UBoolean(true, 0.475)
/_relevanceBeliefs=Tuple(agent=Bob,opinion=SBoolean(0.0, 0.5, 0.5, 0.475))
/_revisedRelevance=UBoolean(true, 0.238)
r4Cam:Recommendation
pricePerNight=UReal(15.0, 1.5)
match=UBoolean(true, 1.0)
_priceBeliefs=Sequence{}
_matchBeliefs=Sequence{Tuple(agent=Cam,opinion=SBoolean(0.0, 0.5, 0.5, 0.5))}
/hotel=ACMalagaPalacio
/rating=UBoolean(true, 0.95)
/portal=WinFreeHotelStays
/relevance=UBoolean(true, 0.95)
/_relevanceBeliefs=Tuple(agent=Cam,opinion=SBoolean(0.0, 0.5, 0.5, 0.95))
/_revisedRelevance=UBoolean(true, 0.475)