Going out for a jog on a rainy day

1. Introduction

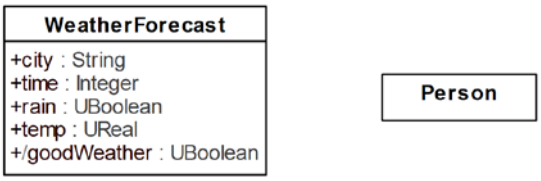
This example tries to illustrate the types of systems in which their users (let's call them agents) receive information from external data sources that are not completely reliable or may contain inaccuracies and therefore have an associated uncertainty which is normally expressed as a probability that represents the degree of confidence on the value provided by the source.

In addition to this (objective) degree of confidence, users of these systems may also associate some subjective uncertainty to that of the source. For example, a weather forecast (WF) system predicts that the probability of rain tomorrow in Malaga is 60%. However, Bob does not trust this information because he knows that the predictions are made from data taken from the airport, whose conditions are different from those in the city center. So he also adds some subjective uncertainty to the weather forecast predictions. Of course, the subjective uncertainty assigned to the same information by different agents may vary, depending on their personal history, experiences and beliefs (e.g., their individual level of trust in a data provider or in the sources).

To further complicate matters, in most systems the users do not work in isolation, but have to exchange information and interoperate with each other to achieve their goals. Think, for example, of agents driving unmanned vehicles in a city, or the avatars of human beings carrying out collaborative tasks such as games, performing robotic surgery or even software modeling? They must be able to combine their (subjective) opinions to reach a consensus on, for example, the next action to perform.

2. The initial system

Suppose that three users (Ada, Bob and Cam) want to go out for a walk, and use two independent weather services to know about the outside temperature and expected chances of rain.

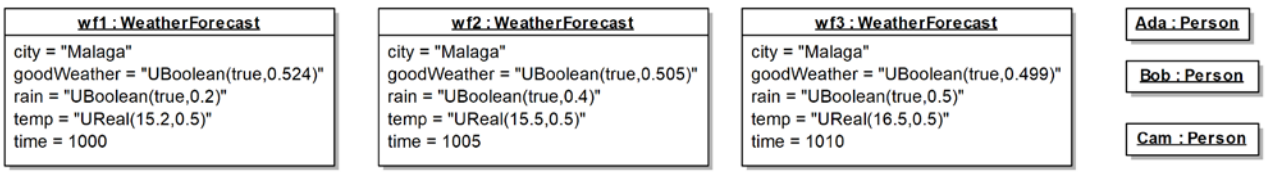


Note the use of datatypes UBoolean and UReal to represent the chances of rain and the temperature measurements. These extended datatypes represent Real and Boolean values enriched with uncertainty [Bertoa et al, 2020]. Attribute goodWeather is derived using the following OCL expression:

goodWeather : UBoolean derive = (not self.rain) and (self.temp >= 15.0)

Notice how the type system takes care of propagating the corresponding uncertainty.

With this, our system composed of the three agents and the estimated weather of Málaga at three moments in time can be represented by the following object diagram:



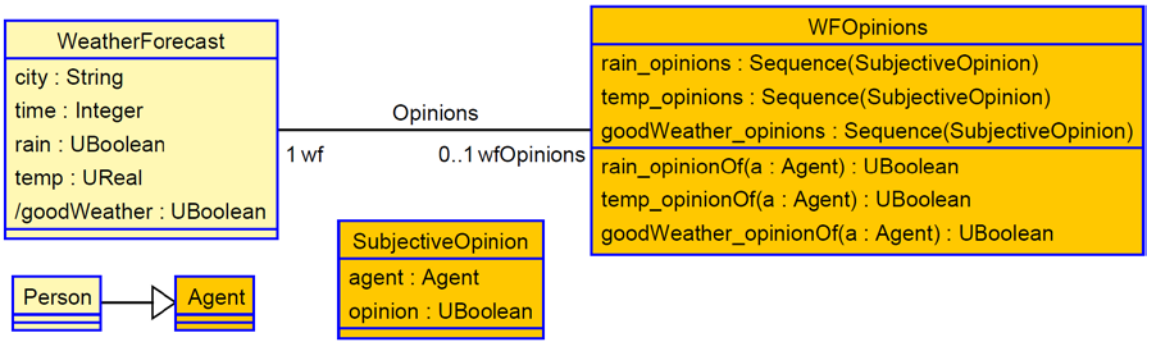
We can see how, according to that system, the best moment for going out for a jog is at time 1000 when the chances for having good weather are higher (and above 0.5). Moment 1010 can be discarded (below those chances, the agents will not go out).

3. Adding degrees of belief to the model elements in OCL

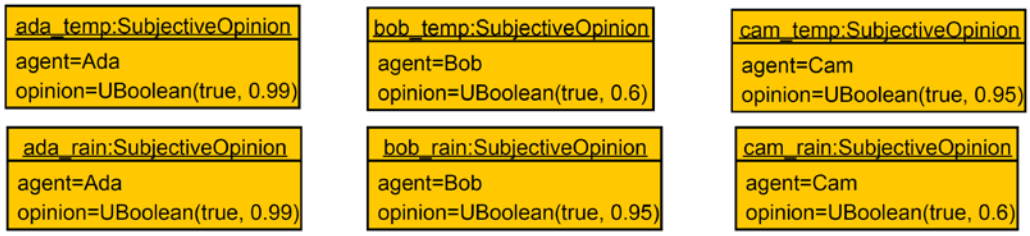
Although the previous system is already able to deal with some type of uncertainty, still the agents cannot express any degree of belief about the model elements, representing their opinions.

In a previous work [Burgueño et al, 2019] we showed how to associate Bayesian probabilities to model elements representing their beliefs. These probabilities were called credences, and with them it was possible to enrich the models so that agents' belief could be explicitly stated. Credences were expressed by means of UBoolean values representing the corresponding probability.

The following conceptual model extends the previous one and allows capturing this information:[1]



Each WeatherForecast object may then have an associated WFOpinions object that holds the opinions of the agents about its rain, temp and goodWeather slot values. For example, suppose the following degrees of belief of the three agents about the temp and rain values provided by the Weather Forecast service:



Note how Ada trusts the temperature and rain readings provided by the WeatherForecast service. Bob is 95% sure about the rain forecast but is only 60% confident on the temperature estimations, while Cam is basically the opposite. Class WFOpinions calculates the corresponding opinions, as follows:

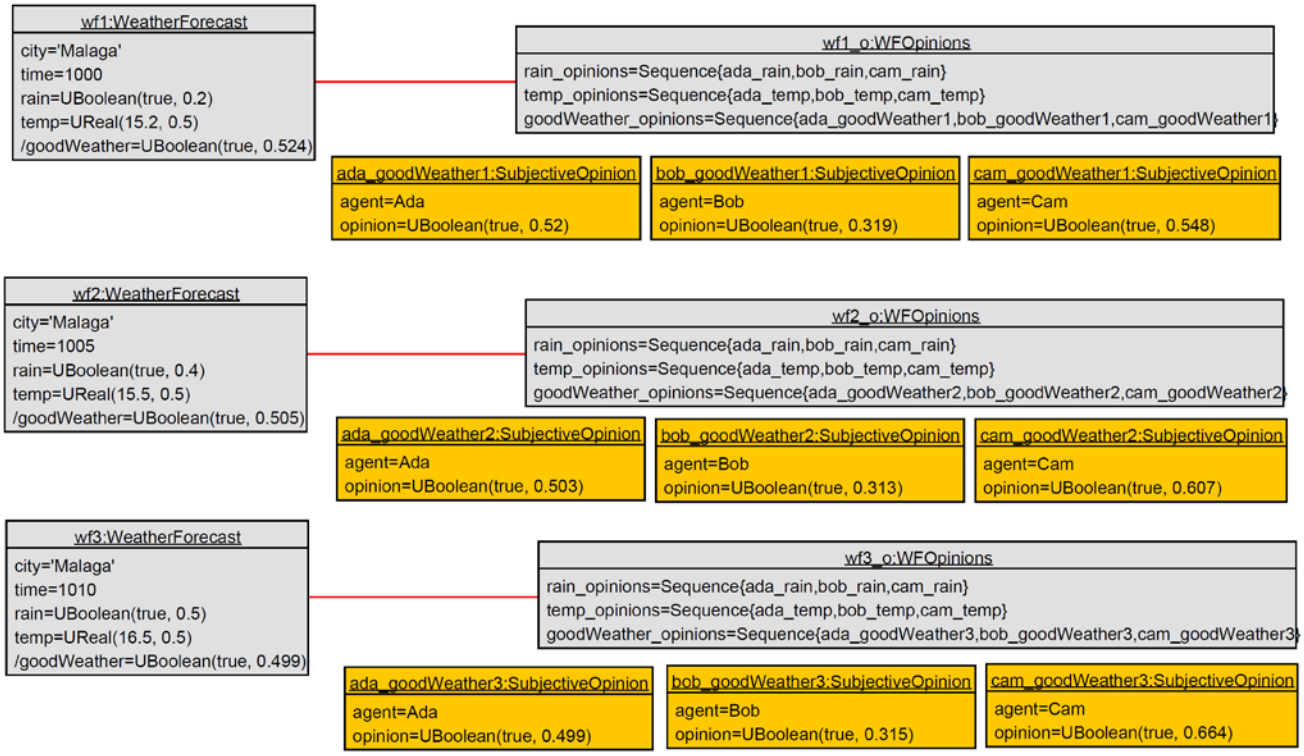
```
class WFOpinions
operations
  rain_opinionOf(a:Agent) : UBoolean =
    let agentOpinion : Sequence(SubjectiveOpinion) = self.rain_opinions->select(t|t.agent=a) in
    if agentOpinion=null or agentOpinion->isEmpty then UBoolean(true,1.0)
    else agentOpinion->collect(opinion)->last()
    endif
  temp_opinionOf(a:Agent) : UBoolean =
    let agentOpinion : Sequence(SubjectiveOpinion) = self.temp_opinions->select(t|t.agent=a) in
    if agentOpinion=null or agentOpinion->isEmpty then UBoolean(true,1.0)
    else agentOpinion->collect(opinion)->last()
    endif
```

[1] Note that this representation in OCL corresponds to the previous version of our OCL approach, which was closer to the representation with the UML Profile and did use Tuples. However, it complicated the computation of derived attributes and relied on operations that should be explicitly invoked, instead of automatically computing the derived values. We need to change this example to conform to the new representation. Anyway, both the results and line of discourse are completely valid.

```
  goodWeather_opinionOf(a:Agent) : UBoolean =
      not (self.rain_opinionOf(a) and self.wf.rain)
      and (self.temp_opinionOf(a) and (self.wf.temp >= 15.0))
end
```

The first two operations return the credence assigned to the corresponding attribute by an agent (or a full confidence if the agent has not expressed any opinion about it). The last one propagates the confidence through the derivation expression.

With this, the resulting decisions about going out are shown in the following object model:



We can see how Ada is happier to go out at time 1000, while Cam prefers time 1010, even though it was initially discarded because the value of goodWeather was less than 0.5. Bob does not like to go out at any time.

Apart from our previous work cited above, currently there are other proposals that also enable the representation of confidence in models, using different notations such as Bayesian probabilities or Fuzzy logic. However, they all suffer from two main limitations. Firstly, the agents are able to express their degrees of belief using just probabilities, so they cannot express ignorance. For example, when the agent 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, because this would mean that x and not(x) are equally likely, which does not represent ignorance since it is already quite informative [Josang, 2016]. In general, forcing an agent to set probabilities with little or no confidence could lead to unreliable conclusions [Muñoz et al, 2020]. In our example, how certain or uncertain were Bob and Ada about the credence they assigned to the data sources?

The second limitation is that existing approaches only enable the representation of individual opinions in software models, but without allowing to reason about the combined opinions of the agents. In particular, we need to count on mechanisms and operations that permit merging individual agents’ opinions in order to reach consensus whenever they are possible. For example, given Ada, Bob and Cam’s opinions, what should they decide about going for a run together today?

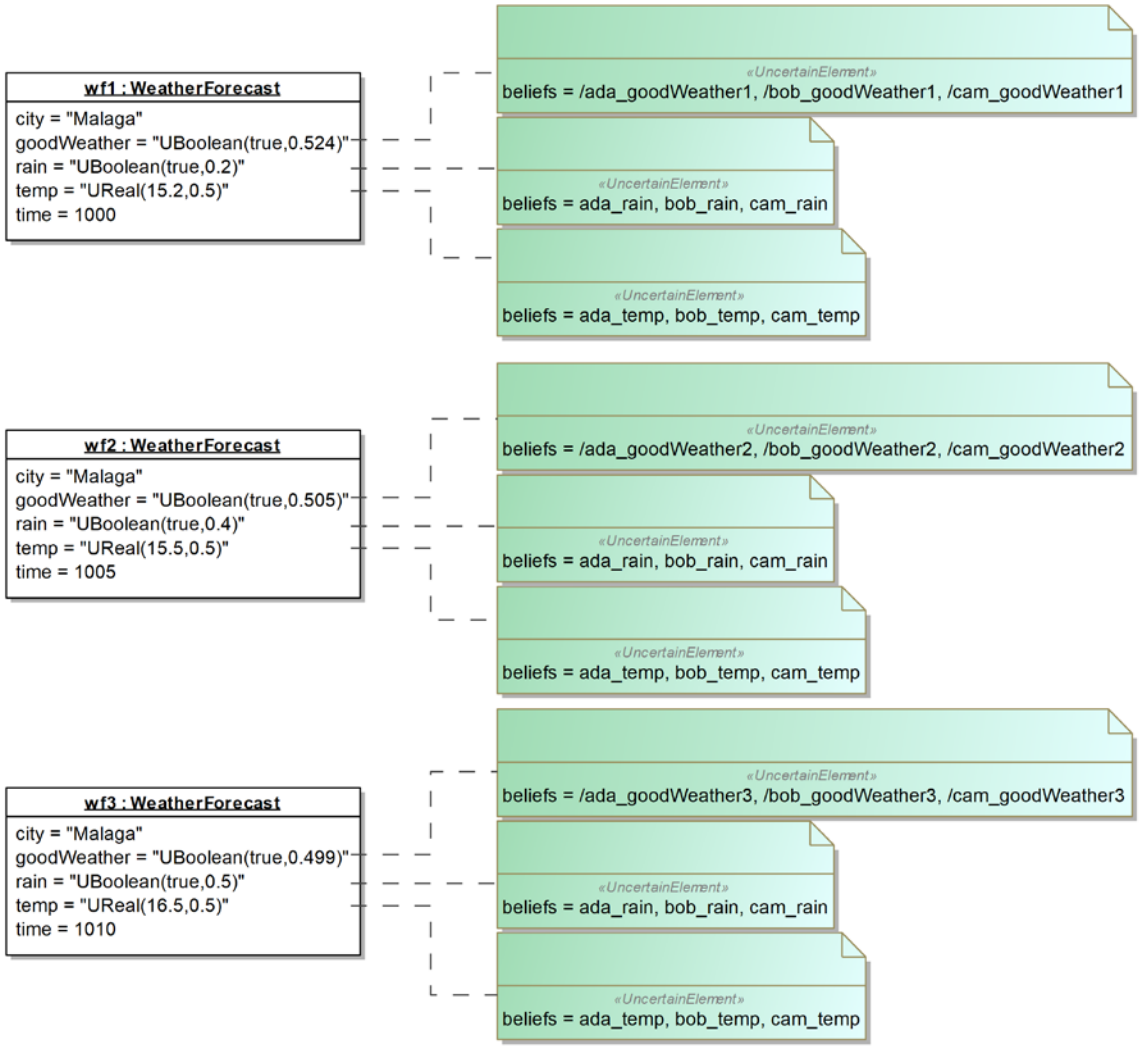
4. Adding Subjective opinions to the model elements

To address these issues, our proposal makes use of Subjective logic, replacing probabilities (i.e., credences) by subjective opinions. Thus, instead of providing a probability, agents express their opinions using a 4-tuple (b,d,u,a) of numbers between 0 and 1, where "b" is the degree of belief that the statement is true, "d" is the degree of disbelief, "u" is the degree of uncertainty and "a" is the prior probability. We always have that b+d+u = 1.

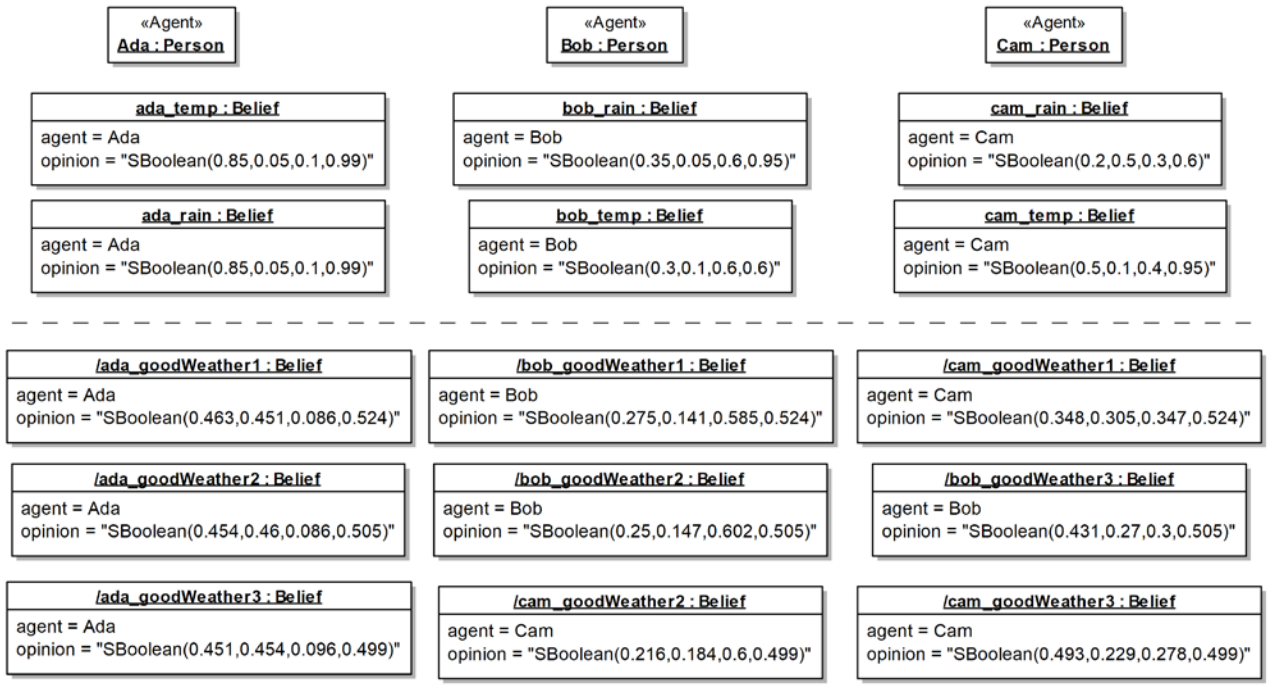
In [Muñoz et al, 2020] we extended the UML and OCL datatypes with Subjective logic opinions, creating a primitive datatype SBoolean whose values are 4-tuples (b,d,u,a). We showed how SBoolean is a supertype of UBoolean, so the following type hierarchy holds in OCL: **Boolean <: UBoolean <: SBoolean**. Among other things, this means that we can naturally embed Boolean and Probabilistic logics into Subjective logic, and the embedding respects subtyping. Projections from the supertypes to the subtypes are carried out by the corresponding operations: method "projection()" on an SBoolean value returns the projected probability [Josan, 2016] that can be used to create the proper UBoolean, and operations "toBoolean()" and "toBoolean(threshold:Real)" on a UBoolean value return true or false depending on the threshold given (which is 0.5 for the method with no parameters).

This way, agents can express not only their degree of belief but also the uncertainty with which they hold it. Subjective logic also implements several operators for fusing opinions that can be used to reach agreements when the agents' opinions diverge about the truth of a statement, as we shall later see.

The following object model shows the three weather forecast instances annotated with the opinions by the three agents.



Note the derived values of the opinions on the goodWeather attribute values. Although the three agents hold the same beliefs about the temperature and humidity values of all weather forecast measurements, their opinions about the goodWeather values are different because they get propagated through different values of the derivation expression.

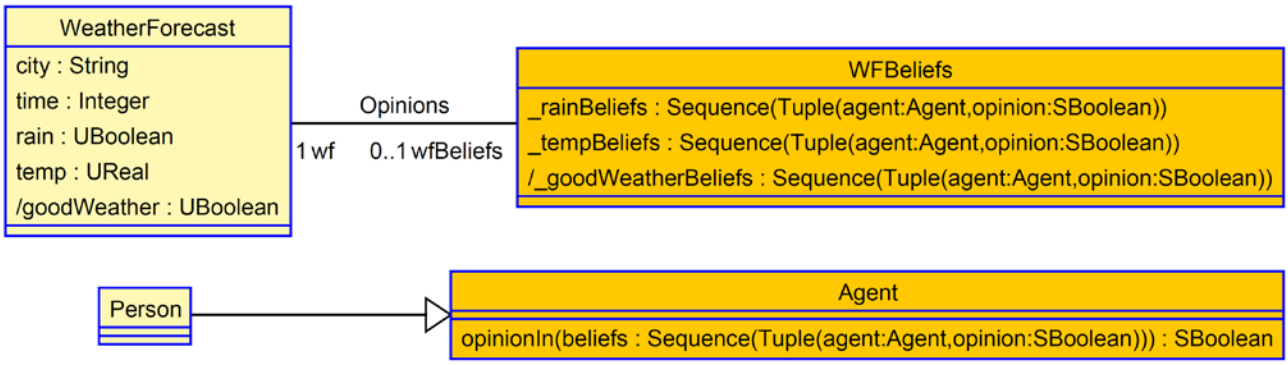


We can compute the projections of these subjective opinions, obtaining the following results:

	Time	Ada	Bob	Cam
Wf1	1000	**0.508**	**0.581**	0.530
Wf2	1005	0.497	0.554	0.582
Wf3	1010	0.498	0.515	**0.632**

We can see how Ada worsens her decisions, Bob significantly improves them (although with a high uncertainty) while Cam maintains them. We can see how Ada only wants to jog at time 1000, while Bob and Cam can go at any of the three moments in time. Bob prefers time 1000 whilst Cam prefers time 1010. It is very important to have into account the degree of uncertainty of the opinion of each agent. We can see how Ada is rather certain about her opinions, while Bob is not (his uncertainty is 0.6 in both cases). Cam is also uncertain about his opinions although not as much as Bob.

Let us show now the solution using the operationalized version of the profile, using Subjective logic. First, we need to extend the model with the agents' opinions. The extended model is shown below.

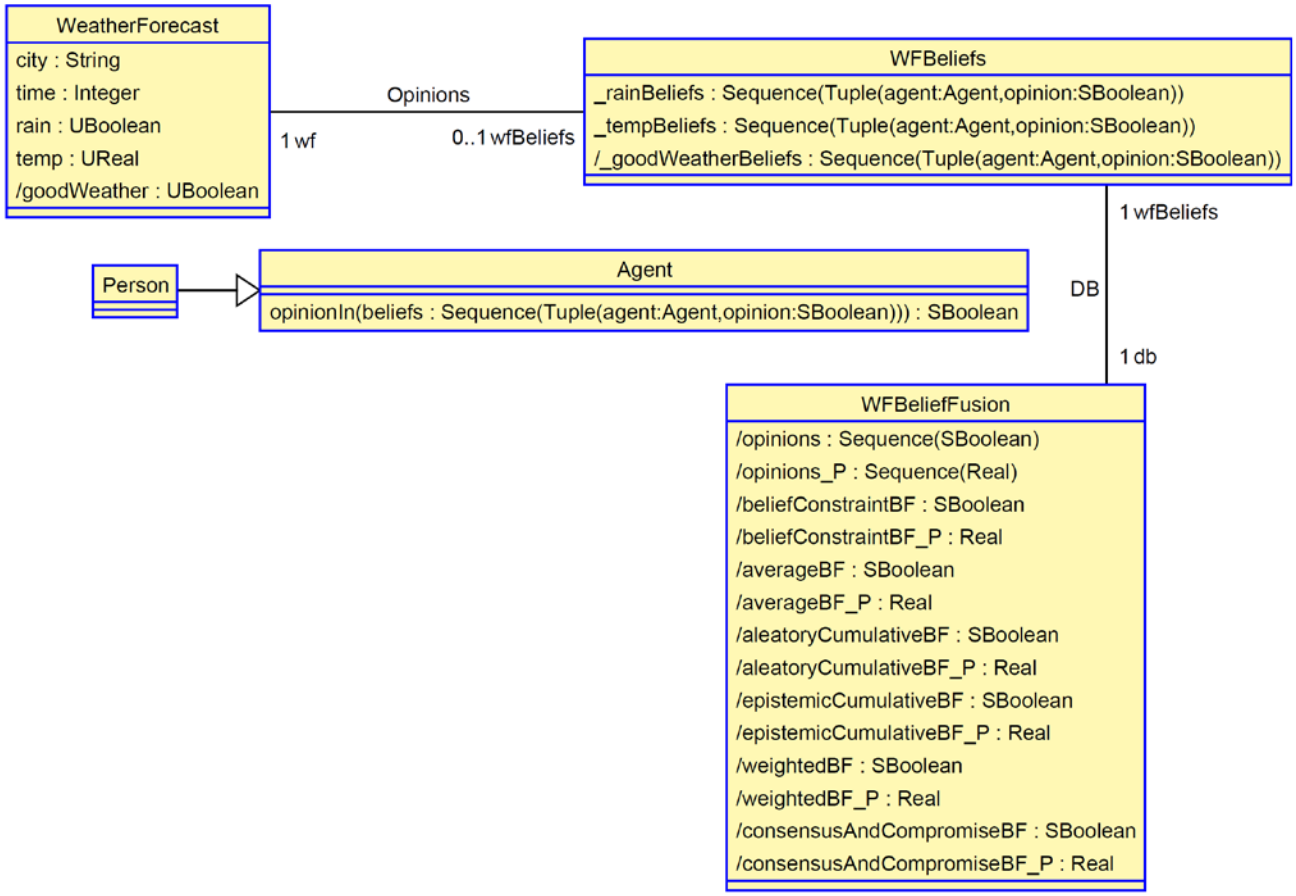


Class WFBeliefs stores the opinions held by different agents about a weather forecast, as sequences of pairs (Agent,Opinion). The agent component of the Tuple refers to the agent holding the opinion, and the opinion component refers to a subjective opinion in Subjective logic [Josang, 2016].
Given that goodWeather is a derived attribute, the corresponding values are computed by propagating the agents' opinion through the derivation formula. Notice that in this case we have preferred to use Tuples instead of objects of type "SubjectiveOpinion" because working with datatypes instead of object instances simplifies the OCL expressions and computations (OCL permits creating datatype values, but not object instances).

class WFBeliefs
attributes
_rainBeliefs : Sequence(Tuple(agent:Agent,opinion:SBoolean)) init: Sequence{}
_tempBeliefs : Sequence(Tuple(agent:Agent,opinion:SBoolean)) init: Sequence{}
_goodWeatherBeliefs : Sequence(Tuple(agent:Agent ,opinion:SBoolean)) derive =
Agent.allInstances->iterate(a ; acc : Sequence(Tuple(agent:Agent,opinion:SBoolean))=Sequence {} |
acc ->append(Tuple{agent = a,
opinion = a.opinionIn(_rainBeliefs).applyOn(not self.wf.rain) and
a.opinionIn(_tempBeliefs).applyOn(self.wf.temp >= 15.0)}))
end

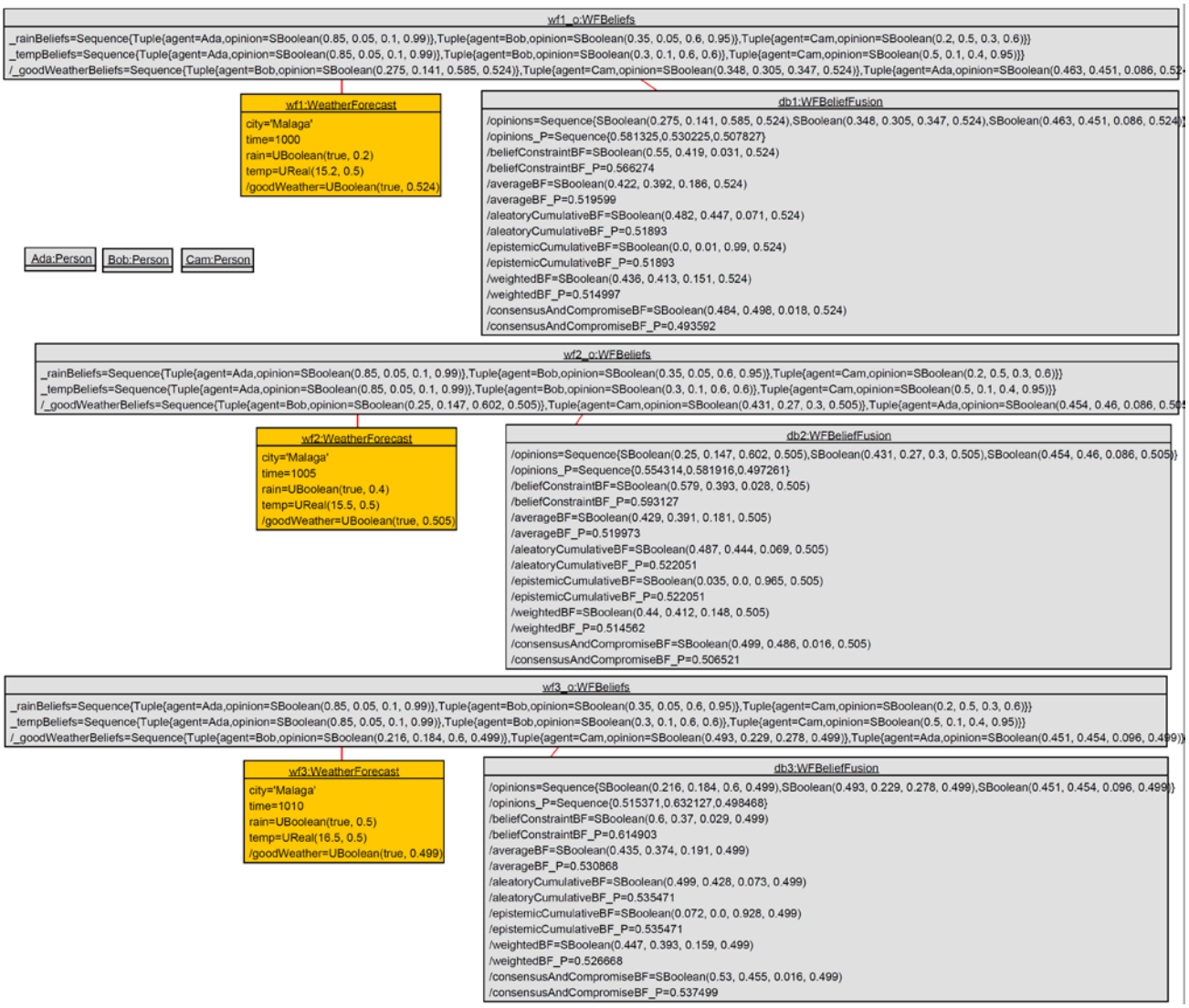
Method "applyOn(u:UBoolean):SBoolean" of datatype SBoolean propagates the opinions, this time using datatype SBoolean operation "applyOn(c : UBoolean)" that returns a SBoolean whose prior probability corresponds to the confidence of the UBoolean, and adjusts the rest of the tuple values accordingly.

Another class "WFBeliefFusion" will also be added to the model, in order to easily capture the results of the fusion operations on the sets of opinions by the individual agents. Therefore, the complete conceptual model is the following:



With this, agents can provide more expressive opinions about their judgements using Subjective logic, as shown below. Note that the prior probabilities of all opinions are exactly the same as before, i.e., they correspond to their credences (see previous Section). However, they have now been qualified by adding the degree of uncertainty each agent has about the statement they make about their confidence on the source.

Using this information, the resulting model is the following:



Subjective logic provides several fusion operators to combine opinions. Let us examine now the results of the belief fusion operators associated to each option (wf1, wf2 and wf3) after merging the opinions of Ada, Bob and Cam. This is captured by the corresponding instances of class Dashboard (db1 associated to wf1, db2 to wf2 and db3 to wf3), which are shown in the object diagram above. The results are summarized in the Table below. Only one value is below 0.5 (C&C on Wf1). Cells in orange indicate the projection of subjective values with uncertainties above 0.5. Cells in blue indicate the higher values of each column. Besides, cells in italics correspond to operators that cannot be applied in our case because in our case the evidences of each agent are dependent (they all express their opinion about the same fact at the same moment).

	BCF	**CCF**	**ABF**	**WBF**	***ACBF***	***ECBF***
Wf1	0.566	***0.494***	0.520	0.515	*0.519*	*0.519*
Wf2	0.593	0.507	0.520	0.515	*0.522*	*0.522*
Wf3	**0.615**	**0.537**	**0.530**	**0.527**	***0.536***	***0.536***

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. In our case, our reasoning is as follows.

- The Belief Constraint Fusion (**BCF**) is used when the agents decide that there is no room for compromise and do not want to give in on their opinions.
- The Consensus & Compromise fusion (**CCF**) is suitable when agents are willing to give in on their opinions to reach agreements, which also fits our situation.
- The Averaging Belief Fusion (**ABF**) operator assumes that the agents' opinions are dependent, which is our case because they observe the same fact at the same moment, and each one is based on the same evidences.
- The Weighted Belief Fusion (**WBF**) operator also works as ABF but gives more weight to more certain opinions, unlike ABF where all opinions have the same weight even when some of them are very uncertain. In our case, it makes sense that uncertain opinions (i.e., people without strong preferences), should have more weight. Therefore, this operation fits even better than ABF.
- The two cumulative fusion operators assume that the opinions are independent, which is not the case here.

We can see how in all cases the combined opinions with higher values correspond to time 1010 (Wf3) and they are all above 0.5, so the agents decide to go jogging at that moment in time.

References

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

[Bertoa et al, 2020] Manuel F. Bertoa, Loli Burgueño, Nathalie Moreno, and Antonio Vallecillo. "Incorporating measurement uncertainty into OCL/UML primitive datatypes." Software & System Modeling 19, 5 (2020), 1163–1189. https://doi.org/10.1007/s10270-019-00741-0

[Burgueño et al, 2019] Loli Burgueño, Robert Clarisó, Jordi Cabot, Sébastien Gérard, and Antonio Vallecillo. "Belief uncertainty in software models." In Proc. of MiSE@ICSE 2019, pp. 19-26, ACM, 2019. https://doi.org/10.1109/MiSE.2019.00011

[Muñoz et al, 2020] Paula Muñoz, Loli Burgueño, Victor Ortiz, and Antonio Vallecillo. "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 .