Ontology-based User Model

Ontology-based user model defines concepts representing user characteristics and identifies relationships between individual characteristics connected to domain ontology. Such a model is (after its population) used by presentation tools to provide personalized navigation and content. Model can be employed also in content organizing tools (e.g., perform sorting of items based on user's preferences).

User ontology in project MAPEKUS[1] is composed of two standalone ontologies, which separate domain-dependent and general characteristics:

- generic-user ontology – defines general user characteristics;
- publication-user ontology – defines characteristics bound to the domain of publications represented by domain ontology.

1.1 Domain independent model

1.1.1 Class User

`User` is a primary class of domain-independent user model (Fig 1.1). It has following data and object properties:

- `hasMaxAge`, `hasMinAge` – data property of type `xsd:int` represents upper and lower boundary of interval which contains user's age;
- `hasChild` – data property of type `xsd:boolean` has the value true or false depending on whether a user has at least one child or not;
- `livesInRegionOfSize` – data property of type `xsd:int` represents number of citizen in user's region of residence;
- `hasCharacteristic` – object property represents domain independent characteristics, its range are instances of type `UserCharacteristic`;
- `includes` – object property with a range instances of type `DomainSpecificUser` represents domain-specific parts of user model.

[1] MAPEKUS project, `http://mapekus.fiit.stuba.sk`

1.1.2 UserCharacteristic class

Class `UserCharacteristic` has following properties:

- `hasTimeStamp` – data property of type `xsd:string` represents a time stamp of this characteristic;
- `hasCountOfUpdates` – data property of type `xsd:int` represents number of actualization (updates) of this characteristic;
- `hasSource` – object property with a range instances of type`UMSource` represents characteristic's source;
- `contributesTo` – object property with a range instances of type `Goal` represents a goal, which relates to this characteristic;
- `hasRelevance` – object property represents relevance of this characteristic in order to achieve `Goal` linked by `contributesTo` property; its range instances are of type `c:LevelOrdering`;
- `hasConfidence` – object property represents confidence of this characteristic (deqreee of quality of user characteristic estimation); its range instances are of type `c:LevelOrdering`.

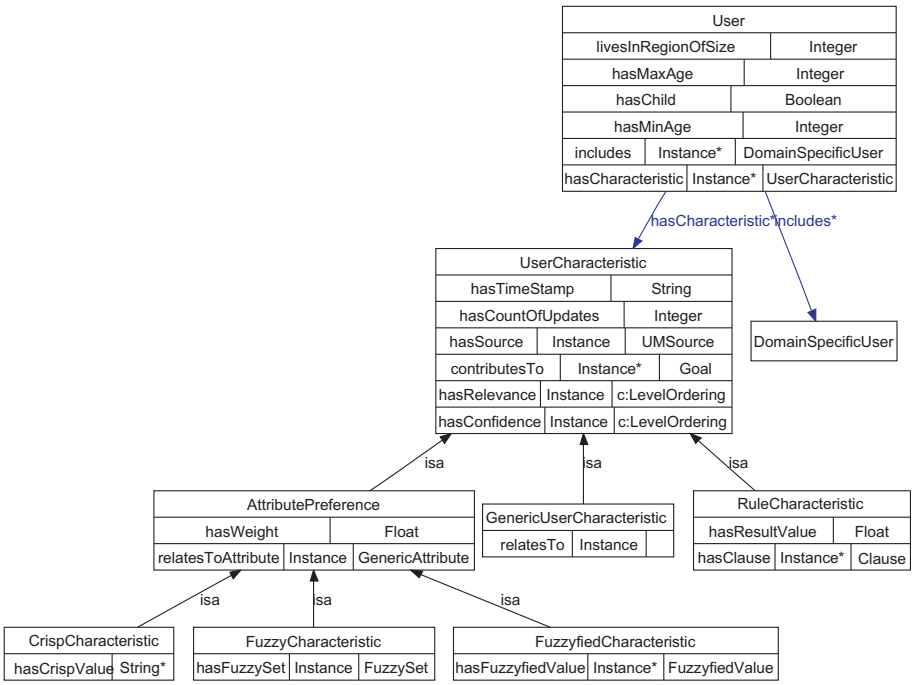

Figure 1.1: Domain-independent user model.

`UserCharacteristic` class has following subclasses:

- `AttributePreference` – represents local preferences. One subclass of this class is used for an individual characteristic. It can be `FuzzyCharacteristic`, `CrispCharacteristic` or `FuzzyfiedCharacteristic`.
- `GenericUserCharacteristic` – represents characteristic in general, which can be used to express relationship to any entity of user and domain model.
- `RuleCharacteristic` – represents global preferences in the form of rules, e.g.,: $resultValue = 0.5$ `IF` $(goodSalary >= 0.7\ AND\ goodPosition >= 0.4)$.

1.1.3 UMSource class

`UMSource` class does not have any data or object properties. It is assumed, that such instances exist in domain-specific parts of a model that represent ways how the user model gets populated with data (automatically by software tools and manually from human intervention).

1.1.4 Goal class

`Goal` class does not have any data or object properties. It is assumed that that such instances exist in domain-specific parts of a model that represents user goals in the particular domain.

1.1.5 AttributePreference class

`AttributePreference` has following properties:

- `hasWeight` – data property of type `xsd:float`. It is uded to compute weighted average if no rules are available.
- `relatesToAttribute` – object property represents an attribute which is bound to the characteristic. Its range instances are of type `GenericAttribute`

1.1.6 GenericUserCharacteristic class

`GenericUserCharacteristic` has following properties:

- `relatesTo` – object property with a range instances of any class from user and domain model.

1.1.7 RuleCharacteristic class

`RuleCharacteristic` class (Fig 1.2) has following properties:

- `hasResultValue` – data property of type `xsd:float` represents the resulting value of the rule.
- `hasClause` – object property with a range instances of type `Clause` represents individual clauses of the rule.

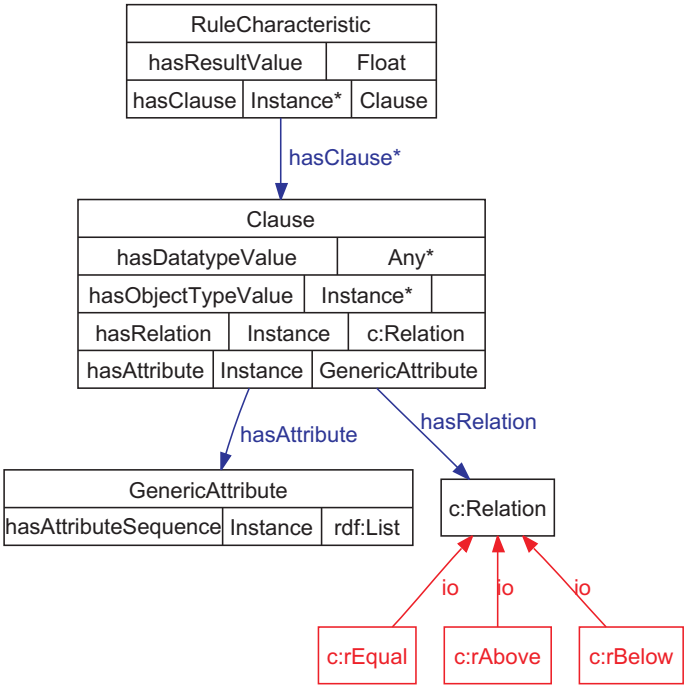

Figure 1.2: Part of a model representing a Rule Characteristic

1.1.8 Clause class

`Clause` class represents a clause part of the rule – a condition applied on a particular attribute. It has following properties:

- `hasAttribute` – object property represents an attribute on to which the property is related. It has range instances of type `GenericAttribute`
- `hasDataTypeValue`/`hasObjectTypeValue` – data/object property, which represents a value of respective attribute;
- `hasRelation` – object property with a range instances of type `c:Relation` represents a type of relation ($<$, $>$, $=$).

1.1.9 GenericAttribute class

`GenericAttribute` class represents properties of any object. It is required to create a subclass along with instances in a domain specific part of the model. The class has following properties:

- `hasAttributeSequence` – object property with a `rdf:List` as a type of range instaces, which represents a list of properties from a base class of domain specific model (i.e., a `Publication` in MAPEKUS project).

1.1.10 CrispCharacteristic class

`CrispCharacteristic` class holds evidence of tolerable values of properties, if these can not be ordered naturally (e.g., places or company names). It has following properties:

- `hasCrispValue` – data property of type `xsd:string` represents a list of values, which are of user's interest.

1.1.11 FuzzyCharacteristic class

`FuzzyCharacteristic` class (Fig. 1.3) represents fuzzy characteristic for properties, whose values can be naturally ordered (e.g., a salary, received degree). It has following properties:

- `hasFuzzySet` – object property with a range instances of type `FuzzySet` represents fuzzy set, which assigns a number $R, R \in < 0, 1 >$ to each value. Higher the number is, better the value is for the user.

1.1.12 FuzzySet class

Fuzzy set is determined by its member function. This function assign a number from $< 0, 1 >$ to each item. If an item is assigned 0, it does not belong to the set. If an item is assigned 1, it belongs to the set. Values from open interval (0,1) are interpreted as a partial membership in the set. The shape of the set is acquired by linking all its points.

`FuzzySet` class has following properties:

- `hasPoint` – object property with a range instances of type `FuzzySetPoint` represents individual points of a fuzzy set;
- `hasType` – object property with a range instances of type `FuzzySetType` represents one of four set types.

1.1.13 FuzzySetPoint class

`FuzzySetPoint` class has following properties:

- `hasY` – data property of type `xsd:float` represents rating from 0 to 1;
- `hasX` – data property of type `xsd:float` represents numerical value of an object;
- `hasXString` – data property of type `xsd:string` represents a label of value.

FuzzySet point has coordinates x, y and a label. If we want to plot a fuzzy set, we plot `hasXString` values on x-axis. If the items of a set are number, than the labels are their textual representations (x=1, xstring="1"). If the items of a set are values such as Bc., Mgr. Phd., the value `hasX` would contain symbolic numerical values (xstring = "Bc.", x = 1; xstring = "Mgr.", x = 2).

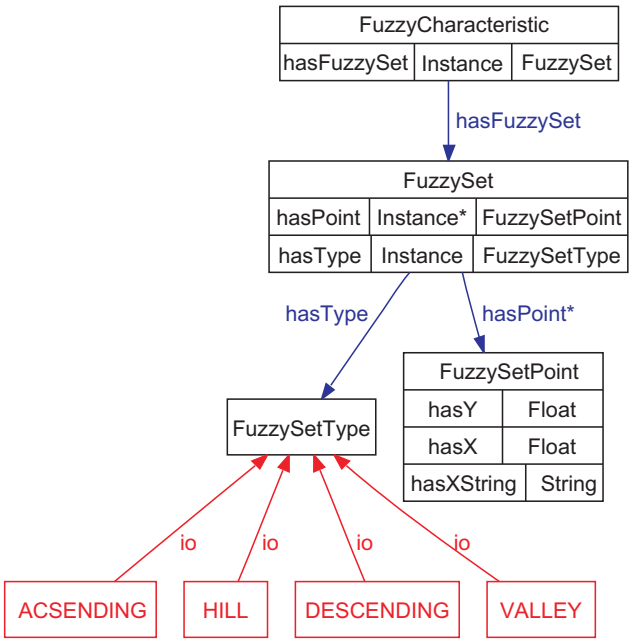

Figure 1.3: Representation of fuzzy characteristics.

1.1.14 FuzzySetType class

`FuzzySetType` class represents types of fuzzy sets, which expresses the shape of member function. The class has following instances:

- `ASCENDING` – the member function is ascending;
- `DESCENDING` – the member function is descending;
- `HILL` – the member function has a triangle- or trapezoid-like shape;
- `VALLEY` - inverse of `HILL`, it reaches its maximum at the edges of function's domain.

1.1.15 FuzzyfiedCharacteristic class

Despite of the fact, that values of a particular properties can not be naturally ordered (places, company names, etc.), we can assign a 0-to-1 preference to some values. It is not a fuzzy set, as we do not have an x-axis, so we do not know the order. However, we can still use it when selecting the best objects.

`FuzzyfiedCharacteristic` class has following properties:

- `hasFuzzyfiedValue` – object property represents values of properties along with their numerical rating. Its range instances are of type `Fuzzyfied-Value`

Figure 1.4: Domain specific user for publications domain.

1.1.16 FuzzyfiedValue class

`FuzzyfiedValue` class has following properties:

- `hasString` – data property of type `xsd:string` represents values of properties, which can not be ordered naturally;
- `hasEval` – data property of type `xsd:float` taking the value from 0 to 1, which express to what extent the user prefers this value.

1.2 Domain-specific model

In general, there could be several domain-specific models, which are connected with domain-independent model. In the MAPEKUS project, we use a domain of publications.

1.2.1 PublicationSpecificUser class

Connection of domain-specific model to the domain-independent model is realized through `PublicationSpecificUser` class (Fig. 1.4) as a subclass of `DomainSpecifiUser` class of a domain-independent model.

`PublicationSpecificUser` has following properties:

- `hasCharacteristic` – object property with a range instances from a union of classes `RuleCharacteristic` and `AttributePreference`.
- `hasVisitedPublication` – object property represents records about visited publications. Its range instances are of type `VisitedPublication`.

1.2.2 VisitedPublication class

`VisitedPublication` defines a record about interaction of user with domain content.It has following properties:

- `hasDateOfVisit` – data property of type `xsd:string` represents a date, when the interaction occurred.
- `hasPublication` – object property with a range instances of type `p:Publication` represents a domain content (a publication);
- `hasRating` – data property of type `xsd:int` represents user's rating of the publication.

1.2.3 PublicationAttribute class

Class `PublicationAttribute` is a subclass of a `GenericAttribute` class and replaces it in the domain of publications.