A Model of User Preferences for Semantic Services Discovery and Ranking

José María García, David Ruiz, and Antonio Ruiz-Cortés

University of Seville
Escuela Técnica Superior de Ingeniería Informática
Av. Reina Mercedes s/n, 41012 Sevilla, Spain
josemgarcia@us.es

Abstract. Current proposals on Semantic Web Services discovery and ranking are based on user preferences descriptions that often come with insufficient expressiveness, consequently making more difficult or even preventing the description of complex user desires. There is a lack of a general and comprehensive preference model, so discovery and ranking proposals have to provide *ad hoc* preference descriptions whose expressiveness depends on the facilities provided by the corresponding technique, resulting in user preferences that are tightly coupled with the underlying formalism being used by each concrete solution. In order to overcome these problems, in this paper an abstract and sufficiently expressive model for defining preferences is presented, so that they may be described in an intuitively and user-friendly manner. The proposed model is based on a well-known query preference model from database systems, which provides highly expressive constructors to describe and compose user preferences semantically. Furthermore, the presented proposal is independent from the concrete discovery and ranking engines selected, and may be used to extend current Semantic Web Service frameworks, such as WSMO, SAWSDL, or OWL-S. In this paper, the presented model is also validated against a complex discovery and ranking scenario, and a concrete implementation of the model in WSMO is outlined.

Keywords: User Preferences, Ontology Modeling, Semantic Web services, Service Discovery, Service Ranking.

1 Introduction

Semantic Web Services (SWS) definition frameworks provide comprehensive tools to describe services and their interactions. However, preferences cannot be described at the same detail level, *i.e.* users cannot define complex desires for a concrete service request. For instance, WSMO goals [19] only support the description of requirements about a request in the form of capabilities, but preferences to rank services fulfilling these requirements cannot be directly expressed by using a standard WSMO goal definition, which only provides means to define non-functional properties / values pairs. In other words, preferences are not considered first-class citizens in WSMO, in comparison to service capabilities, whose

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 1–14, 2010.

definitions are more expressive. Other frameworks, such as OWL-S [16] or SAWSDL [5], do not even define a specific model to describe user requests at all.

Discovery and ranking proposals try to fill this gap, extending SWS frameworks to support preferences definition, or just providing separate user preferences descriptions. There is a variety of formalisms proposed to define preferences, such as tendencies [4,22,24], relative weights [15,17,18,23], or utility functions [8,13,25]. These formalisms actually determine the level of expressiveness of each proposal, while resulting in a high coupling between user preferences definition and its corresponding discovery and ranking implementations.

In order to overcome these limitations, a highly expressive, intuitive model of user preferences is presented in the following. This proposal adapts a well-known model from database systems [12] that allows to define preferences constructively and user-friendly. In this paper, a user preference ontology is described and validated using a discovery scenario from the SWS Challenge[1]. Additionally, the proposed ontological model is applied to WSMO definitions, extending its *goal* element in order to allow the specification of preferences using our model.

The rest of the paper is structured as follows: Sec. 2 discussed the related work on preference definition. Then, in Sec. 3 our proposal is thoroughly presented and validated against a scenario from the SWS Challenge, along with an implementation in WSMO. Finally, Sec. 4 sums up our work and presents the benefits of our contribution.

2 Related Work

Concerning the representation of preferences, there are several approaches in the literature based on different formalisms. Thus, preferences modeled as utility functions have been widely used in economics [6,10] and web systems [1,8,25]. Another formalism based on partial orders were proposed in database systems field [3,12]. The main difference between these two formalisms is that the former constitutes a quantitative approach while the latter is qualitative.

Although quantitative approaches are more general because most preference relations can be defined in terms of utility functions, there are some intuitive preferences that cannot be captured by these functions [3]. On the other hand, qualitative approaches have higher expressiveness and are more intuitive and user-friendly, though they are not directly applicable to a SWS scenario because they do not take into account that properties may be expressed using different abstraction levels depending on the stakeholder. Our proposal constitutes a hybrid model, where both qualitative and quantitative preferences can be expressed.

Preference queries from database systems can be also applied to the Semantic Web scenario with ease. Thus, Siberski *et al.* [20] extend SPARQL query language with a preference clause similar to the proposed by Kießling [12], which in turn serves as the foundations for the preference model presented in Sec. 3.1. Applied to SWS discovery and ranking scenario, there are some approaches in the literature [9,14], though their preference model is essentially based on utility functions.

[1] `http://sws-challenge.org`

Additionally, their model is coupled with their actual implementation of discovery and ranking, consequently limiting their expressiveness. A more comprehensive comparison and overview of these approaches can be found in [7].

Other proposals to model preferences on SWS are more focused on ranking mechanisms, so their preference model are specifically tailored towards their implementations. Toma *et al.* [22] presents a multi-criteria approach based on logic rules, modeling preferences by including simple annotations to WSMO goals, in a similar fashion as policies defined in Palmonari *et al.* [17], though they provide more facilities to express relative weights and different offered policies. García *et al.* [8] provide means to semantically define preferences as utility functions, integrating both logic rules and constraint programming to rank services with respect to those preferences. Finally, a hybrid approach to SWS ranking is proposed by Kerrigan [11], where preferences are described using instances from an ontology, as in our proposal, while distinguishing filtering and ordering preferences that are used at different stages of his solution.

3 Defining an Ontology of User Preferences

In the following, an ontology representing our proposed preference model is presented in detail. In order to validate this model, an application to the Logistics Management scenario from the SWS Challenge[2] is also discussed. Finally, we depict how to extend WSMO framework so that users can define their preferences inside WSMO goals.

3.1 User Preferences Model

As discussed before, service descriptions and user preferences should be semantically described at the same detail level. Therefore, there is a need for the definition of an ontological model that leverages preference descriptions as first-class citizens in the discovery and ranking scenario. This model has to provide intuitive and user-friendly facilities to easily define both requirements and preferences, so that service descriptions can be matched with user requests. Furthermore, these facilities have to conform a sufficiently expressive model so that a user can describe any kind of preference, without being limited by a concrete formalism or representation.

In order to specify a preference model, firstly we need to establish a clear separation between requirements that have to be met, and preferences that have to be taken into account once requirements have been fulfilled. Typically, requirements are hard constraints that are used to filter service repositories in the discovery process, while preferences are used to rank previously discovered services so that the most appropriate service can be selected after the ranking process. Therefore, preferences define a strict partial order in our model, providing a framework to compare and rank a set of services.

[2] The complete scenario description can be found at `http://sws-challenge.org/wiki/index.php/Scenario:_Logistics_Management`

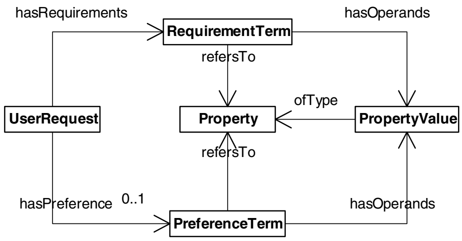

Fig. 1. Upper ontology of user preferences

Figure 1 shows the upper ontology of our model, which is represented using a UML-based notation for OWL ontologies [2] that is also used throughout the rest of the paper. The root concept in our proposed model is called `UserRequest`, which consists on the materialization of user desires with respect to a particular service request. These desires are described using `RequirementTerms` and possibly a `PreferenceTerm`. Thus, hard requirements are defined by instances of subclasses of `RequirementTerm`, and the same applies to `PreferenceTerm`. Both requirements and preferences are related with one or more properties, which are referred inside each term, and with some property values that act as operands for each term specialization. Properties and property values depend on the specific domain being used within each scenario, so instances (or specializations) of Property and PropertyValue classes may be probably described using an external domain ontology.

Requirements definition has been widely discussed in the literature, and SWS frameworks provide sufficiently expressive facilities to define them, so in the following we will focus on preference modeling. In Sec. 3.2, requirement terms are simply considered as property / property value pairs, which is sufficient to describe user requests in the validation scenario.

Concerning preferences, Fig. 2 presents the middle ontology of our model, where a hierarchy of available preference constructors is introduced. Thus, a preference term can be an `AtomicPreference`, or a composition of two preference terms by applying one of the sub-classes of `CompositePreference`. On the one hand, atomic preferences are those which refers to a single property, and can describe either a qualitative or a quantitative preference that users may have with respect to the referred property. On the other hand, composite preferences relate different preferences between them, so that a complex preference can be described. These complex constructors are defined for two preferences in our model, though they can be generalized to a greater number of preferences obviously.

Each preference construct from Fig. 2 is defined intuitively in the following, including a motivating example described in natural language from the SWS Challenge scenario used to validate our proposed model in Sec. 3.2, where some of these constructs are applied to describe that scenario goals. A formal definition of the described preference terms can be found at [12], where the foundations of our model are thoroughly discussed.

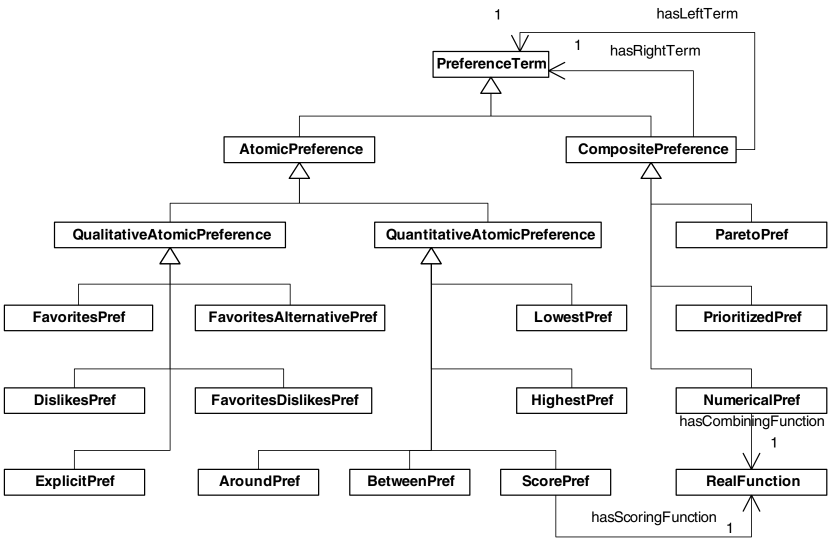

Fig. 2. Preference terms hierarchy

Qualitative atomic preferences. The first group of preferences presented in the following corresponds to the qualitative and atomic constructs, which means that every preference that belongs to this kind refers to a single, non-numerical property.

a) Favorites preference - `FavoritesPref`
A *favorites preference* defines a finite set of property values that constitute the desired values of the referred service property. Thus, services whose value for that property is a member of the *favorite set* are preferred to services that provide any other values from the property domain. An instance of this preference constructor has many operands as the cardinality of the favorite values set.
Example: I prefer WSs that provide *carriageForward* as a possible `Payment-Method`[3].

b) Dislikes preference - `DislikesPref`
As opposite to `FavoritesPref`, a *dislikes preference* define a set of property values that the service should not provide for the referred property in order to be preferred to another service whose property values coincide with any of the values in the associated *dislikes set.*
Example: I prefer WSs that do not offer *refundForDamage* as an available `Insurance` option.

[3] In each example, *italics* text correspond to property values used as operands, while `typewriter` text are used to denote properties.

c) Favorites or alternative preference - `FavoritesAlternativePref`
A *favorites or alternative preference* is an extension of `FavoritesPref`, where there are two favorite sets. The second set is called *alternative set*. In this case, services whose property values are in the favorite set are the most preferred. Otherwise their values should be on the alternative set. If this is not the case either, then the corresponding services will be undesirable, because their property values are not member of any of the two sets. In order to differentiate between favorite and alternative set, `hasOperands` object property has to be specialized to two different object properties, namely `hasFavorite` and `hasAlternative`.
Example: I prefer WSs whose `PaymentMethod` is *carriagePaid*, but if that is infeasible, then it should be *carriageForward*.

d) Favorites or dislikes preference - `FavoritesDislikesPref`
It is also possible to combine `FavoritesPref` with `DislikesPref` in the following form: service properties should have a value on the defined favorite set. Otherwise, values should not be from the dislikes set. If none of these two conditions hold, then the service will be less preferred than others fulfilling the first or the second condition. Again, `hasOperands` is specialized for this preference constructor to `hasFavorite` and `hasDislikes` object properties.
Example: I prefer WSs that provide *refundForLoss* as a possible `Insurance`, but if that is infeasible, then it should not be *refundForDamage*.

e) Explicit preference - `ExplicitPref`
An *explicit preference* can be used to explicitly represent the strict partial order between a pair of property values. Thus, a better-than graph can be defined using several *explicit preferences*. In this case, `hasOperands` is specialized to `hasLeftOperand` and `hasRightOperand`, meaning that the left operand value is better than the right operand value, with respect to the referred property.
Example: WSs that provide *carriageForward* as a possible `PaymentMethod` are more preferred than those that provide *carriagePaid*.

Quantitative atomic preferences. When the referred property of an atomic preference is numerical, the following quantitative constructs may be used to express user preferences on that single property.

a) Around preference - `AroundPref`
An *around preference* determines which property value is better by determining the distance of each values to a concrete value provided as an operand of this preference term. Thus, services which provide exactly that value are preferred to the rest of them. If this is infeasible, services with closer values to the operand are preferred.
Example: I prefer WSs that provide a `BasePrice` closer to `180 Euros`.

b) Between preference - `BetweenPref`
In this case, a service should have values for the referred property between a range of values that are defined as operands in the preference (`hasOperands` is specialized to `lowBound` and `upBound`). If this is not the case, *between*

preferences prefer services closer to the interval boundaries, computing the distance as in around preferences.
Example: I prefer WSs that provide a `PaymentDeadline` within the interval of $[45, 60]$ days.

c) Lowest preference - `LowestPref`
A *lowest preference* does not have any operand, but prefer services whose property values are as low as possible for the referred service property.
Example: I prefer WSs that provide a `BasePrice` as low as possible.

d) Highest preference - `HighestPref`
In opposition to the last constructor, a *highest preference* is used when property values should be as high as possible.
Example: I prefer WSs that provide a `PaymentDeadline` as long as possible.

e) Score preference - `ScorePref`
A *score preference* basically defines a scoring function (*i.e.* a utility function like in [8]) that takes a property value as its argument and returns a real value that can be interpreted in the following form: the higher the value returned by the function is, the more preferred the property value entered as the argument. Note that this kind of preference is not as intuitive as the rest, but it is still useful when a user wants to express complex grades of preference, using for instance a piecewise function depending on the property values.
Example: I prefer WSs with the highest score with respect to `Price/Kg`, where the scoring function is defined as: $f(pricePerKg) = \frac{-1}{50} pricePerKg + 1$.

Composite preferences. The last group of preference constructs are used to compose two different preference terms by stating the preference relationship between each component term, which can be also a composite preference. Composite preferences `refersTo` property associate the preference with the union of the properties referred by component preferences.

a) Pareto preference - `ParetoPref`
A *pareto preference* P combines two preference terms P1 and P2 using the *Pareto-optimality principle*, which considers that P1 and P2 are equally important preferences. Thus, a service SWS_1 is better than another service SWS_2 with respect to P, if SWS_1 is better than SWS_2 with respect to P1 and SWS_1 is not worse than SWS_2 with respect to P2, and *vice versa*. Note that P1 is linked with P using the `hasLeftTerm` object property, and P2 using the `hasRightTerm` object property. Intutively, this preference balance the fulfillment of each preference component, so that the composite preference is the average degree of preference taking both components into account.
Example: Cf. Fig. 6 for a use case of a pareto preference.

b) Prioritized preference - `PrioritizedPref`
In the case of a *prioritized preference* P that compose two preference terms P1 and P2, P1 is considered more important than P2. Thus, P2 is evaluated only if P1 does not mind (*i.e.* service property values compared using P1 do not return enough information to rank those services).
Example: Cf. Fig. 4 for a use case of a prioritized preference.

c) Numerical preference - `NumericalPref`

Finally, a *numerical preference* is the combination of two score preferences using a function that takes the values returned by the score preferences as its arguments and returns another real number that gives information about the global preference, considering all the properties referred by concrete score preferences. Notice that component preferences must be score preferences in order to properly compose them using a combining function.

Example: Provided that $f(basePrice)$ and $g(pricePerKg)$ are already defined and they range within the interval $[0, 1]$, I prefer WSs that have a higher combined score, where the combining function is defined as:
$combF(basePrice, pricePerKg) = 0.8 * f(basePrice) + 0.4 * g(pricePerKg)$.

3.2 Validating the Model

The proposed preference model has been validated using one of the discovery scenarios from SWS Challenge. Concretely, the chosen one has been the Logistics Management scenario, because of its complexity and the inclusion of preference descriptions. It consists on seven logistics service offers, described in natural language in terms of different properties, such as price, covered geographical areas, operating hours and truck fleets, among others. Additionally, several service requests (*i.e.* user goals) applicable to this scenario are defined, which contain both hard requirements and user preferences (they are referred as *soft constraints* in the scenario) that can be used to choose the most appropriate service (*i.e.* the best one in terms of preferences) among those which fulfill hard requirements.

In the chosen scenario, goals B1, C1, D1 and E1 define a variety of preferences against different service properties, in addition to describe how preferences should be combined within each goal. In order to validate our preference model using this scenario, we provide in the following equivalent descriptions for each of these goals using the proposed model. Thus, textual descriptions of goals directly extracted from the scenario description are shown alongside their equivalent representation using the preference ontology presented in Sec. 3.1. For the sake of simplicity, service properties are included as instances inside the same default namespace as the goal, though a domain ontology should be externally defined, covering all the existing properties in the Logistics Management domain.

Figure 3 presents the instantiation of the goal B1 from the scenario. The goal as a whole is modeled with an instance of `UserRequest`, while each term is instantiated depending on its nature. Thus, requirements about pickup, delivery and transported goods are represented at the top of the figure. This representation is shown simplified, because requirements in every goal from the Logistics Management scenario are pairs between domain properties and their values. Consequently, in the following instantiated goals, requirements are omitted from the representation, though they can be easily described using property-value pairs.

Concerning the preference modeling, goal B1 states that the user prefers two properties, namely `PaymentMethod` and `Insurance`, to contain certain values,

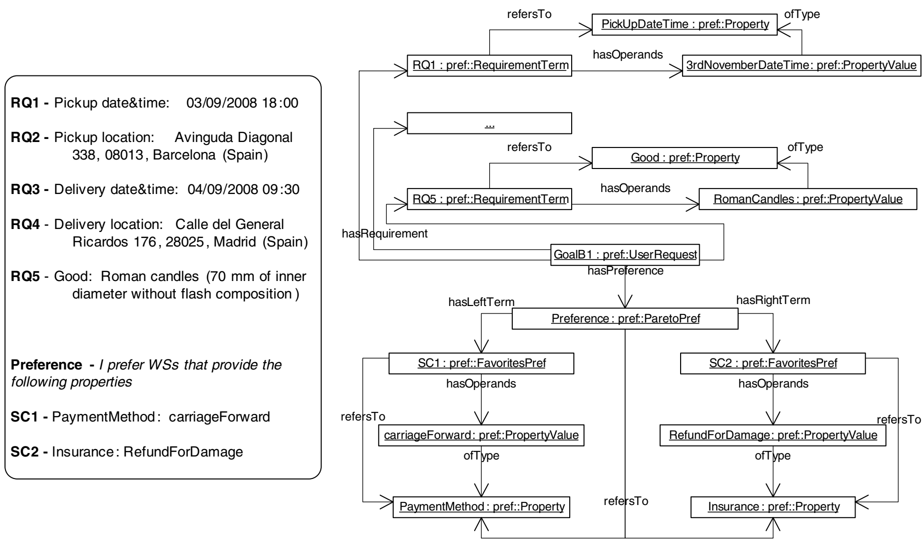

Fig. 3. Goal B1 description excerpt and its representation using our model

`carriageForward` and `refundForDamage`, respectively. Both of these soft constraints are considered qualitative preferences that define the favorite values for each property. However, the preference description do not explicitly express how to compose those two atomic preferences, so it can be inferred that a *pareto preference* can be applied to relate each one, because both atomic preferences can be considered equally important for the user.

The next goal used to validate our model is shown in Fig. 4. In this case, the atomic preferences defined in C1 are instantiated as a *favorites preference* and an *around preference*. Moreover, the preference description gives more importance to the favorites preference on the `PaymentMethod` property, taking

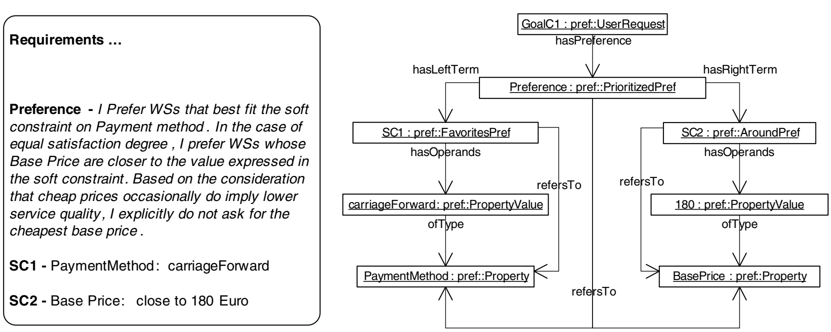

Fig. 4. Goal C1 description excerpt and its representation using our model

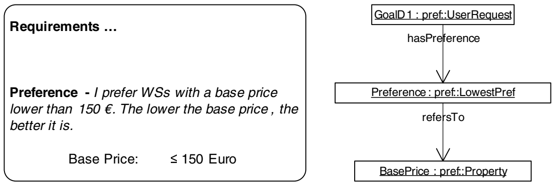

Fig. 5. Goal D1 description excerpt and its representation using our model

into consideration the preference about the `BasePrice` only if services have an equal satisfaction degree when evaluating the first preference. Thus, the most appropriate composite preference is a *prioritized preference*, because its semantics are exactly what the user is looking for in this goal.

Goal D1, which is represented in Fig. 5, is the most simple goal of the scenario. There is no composition of atomic preferences, because it only states that the `BasePrice` should be as low as possible. The limit for the price that is included in the scenario is not necessary in our solution, because the semantics of the *lowest preference* is sufficient in this case to properly rank services with respect to the stated user preferences. Nevertheless, it is possible to take that price limit into account by modeling the user preference as a *prioritized preference*, where P1 is a *between preference* on `BasePrice` with the interval $[0, 150]$, and P2 is the *lowest preference* shown in Fig. 5.

Finally, the most complex goal of the scenario is shown in Fig. 6, where some of the `refersTo` relations are omitted for the sake of clarity[4]. The different atomic preferences are composed using *pareto preferences*, because the goal E1 description explicitly states that the user wants an average satisfaction degree among the atomic preferences. Notice that SC3 is decomposed into two *favorites preferences*, because it was interpreted that `Insurance` property should have both values. If SC3 were modeled using only one *favorites preference* with the two values in the favorite set, then services that supports only one type of insurance would be considered equally preferred than those supporting both insurance values.

In conclusion, the presented validation using a relatively complex discovery and ranking scenario from the SWS Challenge proves that our proposed model is sufficiently expressive and intuitive, allowing to describe any kind of user preferences directly, user-friendly, and independently of the discovery and ranking technique to apply at a later stage. Additionally, the actual evaluation of the described preferences lead to the expected ranking results that are described in the scenario. This evaluation can be performed applying formal definitions of the equivalent preference constructs from [12]. Further validation may be performed using other scenarios and test cases, such as the shipment discovery scenario used in [8].

[4] Actually, this relation can be inferred from the type of the operands involved in each preference.

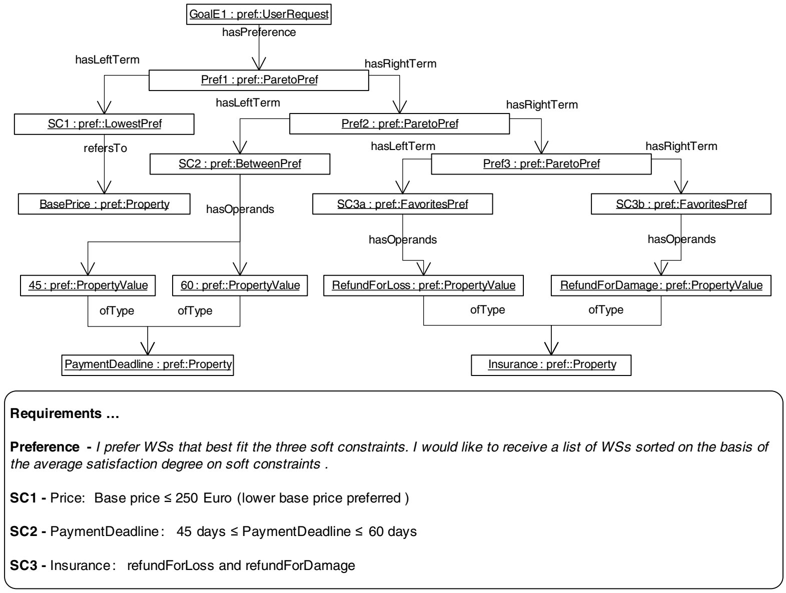

Fig. 6. Goal E1 description excerpt and its representation using our model

3.3 Implementing the Model in WSMO

Due to the fact that the proposed preference model is general and independent from the formalism, it can be applied as an extension to current SWS frameworks, such as WSMO, OWL-S, or even SAWSDL, so that these frameworks can support user preference modeling. Concerning WSMO, our proposed model can be implemented as an extension of its meta-model. Thus, user requests from our model corresponds to WSMO *goals*. Moreover, requirement terms are already supported by WSMO capabilities and interfaces, so that requirement terms described in our model can be easily translated into capabilities. However, preference terms have to be added to the specification of goals. Therefore, in order to apply our preference model to WSMO, we define a new meta-model class in Listing 1, `preferenceGoal`, which is a subclass of `goal` that adds a `hasPreference` property where preference terms can be linked with a user goal.

Listing 1. WSMO goal extended with preferences

```
Class preferenceGoal sub−Class goal
    hasNonFunctionalProperties type nonFunctionalProperties
    importsOntology type ontology
    usesMediator type {ooMediator, ggMediator}
    hasPreference type PreferenceTerm
        multiplicity = single−valued
    requestsCapability type capability
        multiplicity = single−valued
    requestsInterface type interface
```

This implementation allows a seamless integration of preference information in WSMO, without actually modifying the goal meta-model, because it is only refined. Thus, current WSMO discovery and ranking proposals could be still applied to extended goals transparently. A different approach can be found in [8], where preferences are included within `nonFunctionalProperties` section by using logic programming rules, although it is only applied to preferences defined as utility functions.

Listing 2 shows an example of how to describe a WSMO goal using our proposed implementation to include our preference model. Thus, goal D1 from Sec. 3.2 can be described in WSMO easily. The domain ontology for the Logistics Management scenario is supposed to be properly defined in `Logistics.wsml`.

Listing 2. Extended goal description with preferences from D1

```
namespace {_"GoalD1.wsml#", lm _"Logistics.wsml#",
  wsml _"http://www.wsmo.org/wsml/wsml-syntax/",
  pref _"http://www.isa.us.es/ontologies/PreferenceModel.wsml#"}

preferenceGoal GoalD1

 capability D1requestedCapability
 preference D1preference

ontology preferenceOntology

instance D1preference memberOf pref#LowestPreference
  pref#refersTo hasValue lm#BasePrice
```

From the above example, one concludes that transforming user requests modeled using our proposed ontology for preferences to a WSMO goal is a straightforward process, provided that the ontological model is expressed in WSML [21]. Also notice that the WSML variant used in Listing 2 includes new keywords to link specialized goals to preference terms which are described in a separate ontology.

4 Conclusions

In this work, a highly expressive preference model for SWS discovery and ranking is presented. This model, specified as an ontology, represents a novel approach that leverages preference descriptions so that they become a first-class citizen in SWS frameworks. Furthermore, the model has been validated using a complex discovery scenario from the SWS Challenge in order to prove the applicability of our solution to an actual discovery and ranking scenario. The main benefits of our proposed model can be summarized as follows:

– **Expressiveness.** The model is sufficiently expressive to describe complex user desires about requested services, providing a comprehensive hierarchy of preference terms.
– **Intuitive semantics.** Based on a strict partial order interpretation of preferences, the model is both user-friendly and machine-readable, so preferences may be automatically processed and inferred.

- **Qualitative and Quantitative.** Available constructs allow to express both qualitative and quantitative preferences, and even combine them in a general preference term.
- **Independence.** Our proposal is not coupled with a concrete SWS solution, neither with a discovery nor ranking mechanism, so it is not limited by the formalisms used to implement these mechanisms.
- **Extensibility.** Because the model is presented as an ontology, it can be further extended with new preference constructs with ease.
- **Applicability.** Our model can be implemented within any SWS framework, extending current proposals to leverage preference descriptions.

An implementation of our model that extends WSMO goals is also discussed. This actual application consists in a seamless extension of WSMO constructs to allow the definition of complex preferences, that can be used within any discovery and ranking solution, provided that it supports or adapts the proposed preference ontological model.

Acknowledgments. This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT projects Web-Factories (TIN2006-00472) and SETI (TIN2009-07366), and by the Andalusian Government under project ISABEL (TIC-2533).

References

1. Agrawal, R., Wimmers, E.L.: A framework for expressing and combining preferences. ACM SIGMOD 29(2), 297–306 (2000)
2. Brockmans, S., Volz, R., Eberhart, A., Loffler, P.: Visual modeling of OWL DL ontologies using UML. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 198–213. Springer, Heidelberg (2004)
3. Chomicki, J.: Preference formulas in relational queries. ACM Transactions on Database Systems (TODS) 28(4), 466 (2003)
4. Dobson, G., Lock, R., Sommerville, I.: QoSOnt: a QoS ontology for service-centric systems. In: EUROMICRO-SEAA, pp. 80–87. IEEE Comp. Soc., Los Alamitos (2005)
5. Farrell, J., Lausen, H.: Semantic annotations for WSDL and XML Schema. Technical report, World Wide Web Consortium (August 2007)
6. Fishburn, P.C.: Utility theory for decision making. Wiley, Chichester (1970)
7. García, J.M., Rivero, C., Ruiz, D., Ruiz-Cortés, A.: On using semantic web query languages for semantic web services provisioning. In: The 2009 International Conference on Semantic Web and Web Services (SWWS). CSREA Press (2009)
8. García, J.M., Toma, I., Ruiz, D., Ruiz-Cortés, A.: A service ranker based on logic rules evaluation and constraint programming. In: 2nd ECOWS Non-Functional Properties and Service Level Agreements in Service Oriented Computing Workshop, CEUR Workshop Proceedings, Dublin, Ireland, November 2008, vol. 411 (2008)
9. Iqbal, K., Sbodio, M.L., Peristeras, V., Giuliani, G.: Semantic service discovery using SAWSDL and SPARQL. In: Fourth International Conference on Semantics, Knowledge and Grid, SKG 2008, pp. 205–212 (2008)

10. Keeney, R.L., Raiffa, H.: Decisions with multiple objectives: Preferences and value tradeoffs. Cambridge Univ. Press, Cambridge (1993)
11. Kerrigan, M.: Web service selection mechanisms in the web service execution environment (WSMX). In: SAC, pp. 1664–1668 (2006)
12. Kießling, W.: Foundations of preferences in database systems. In: VLDB, pp. 311–322. Morgan Kaufmann, San Francisco (2002)
13. Kritikos, K., Plexousakis, D.: Semantic QoS metric matching. In: ECOWS 2006, pp. 265–274. IEEE Computer Society Press, Los Alamitos (2006)
14. Lamparter, S., Ankolekar, A., Studer, R., Grimm, S.: Preference-based selection of highly configurable web services. In: WWW 2007, pp. 1013–1022. ACM, New York (2007)
15. Liu, Y., Ngu, A.H.H., Zeng, L.: Qos computation and policing in dynamic web service selection. In: WWW (Alt Track Papers & Posters), pp. 66–73 (2004)
16. Martin, D., Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., et al.: OWL-S: Semantic markup for web services. Technical Report 1.1, DAML (2004)
17. Palmonari, M., Comerio, M., De Paoli, F.: Effective and Flexible NFP-Based Ranking of Web Services. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 546–560. Springer, Heidelberg (2009)
18. Pathak, J., Koul, N., Caragea, D., Honavar, V.G.: A framework for semantic web services discovery. In: WIDM 2005: Proceedings of the 7th annual ACM international workshop on Web information and data management, pp. 45–50. ACM Press, New York (2005)
19. Roman, D., Lausen, H., Keller, U.: Web service modeling ontology (WSMO). Technical Report D2 v1.3 Final Draft, WSMO (2006)
20. Siberski, W., Pan, J.Z., Thaden, U.: Querying the Semantic Web with Preferences. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 612–624. Springer, Heidelberg (2006)
21. Steinmetz, N., Toma, I. (eds.): The Web Service Modeling Language WSML. Technical report, WSML, WSML Working Draft D16.1v0.3 (2008)
22. Toma, I., Roman, D., Fensel, D., Sapkota, B., Gomez, J.M.: A multi-criteria service ranking approach based on non-functional properties rules evaluation. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 435–441. Springer, Heidelberg (2007)
23. Vu, L.H., Hauswirth, M., Porto, F., Aberer, K.: A search engine for QoS-enabled discovery of semantic web services. International Journal of Business Process Integration and Management 1(4), 244–255 (2006)
24. Wang, X., Vitvar, T., Kerrigan, M., Toma, I.: A QoS-aware selection model for semantic web services. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 390–401. Springer, Heidelberg (2006)
25. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)