Using Agents & Artifacts to Access the Semantic Web A Recommender System Case Study

Jéssica Pauli de C. Bonson[1], **Elder Rizzon Santos**[1]

[1]Laboratório IATE – Universidade Federal de Santa Catarina(UFSC)
Florianópolis – SC – Brazil

{jpbonson, elder}@inf.ufsc.br

Abstract. *In this paper we integrate the access of agents to the Semantic Web by means of the Agents & Artifacts model, using the Cartago framework. Our case study is a recommender system to define metadata of a Learning Object according to a metadata standard. The agents are able to provide recommendations by querying the DBPedia through artifacts. The contribution of our work is to develop a prototype integrating MAS to the Semantic Web using artifacts.*

1. INTRODUCTION

In this paper we research the use of the Agents & Artifacts (A&A) [Omicini et al. 2008a] model to ease the access of agents in a Multi-Agent Systems (MAS) to the functionalities of the Semantic Web. To test our approach we adopted the case study of a recommender system for the educational area. The A&A model emerged from the need of modelling the environment of a Multi-Agent Systems (MAS) as a first-class entity [Ricci et al. 2007, Omicini et al. 2008b, Ricci et al. 2008, Weyns et al. 2007]. Artifacts can model tools and resources to help the agents execute their tasks at runtime. In this work the artifacts model the semantic searches to access the Web of Data [Berners-Lee et al. 2001], more specifically, the DBPedia [Bizer et al. 2009]. Our agents access this repository to get recommendations for the metadata of a Learning Object (LO) through inferences on the partial knowledge provided by the user. Following the knowledge representation available on the Semantic Web, our artifacts aid the agents on the recommendations by querying linked data sources considering context-specific categories, classes or individuals.

2. The Recommender Model

In this paper we developed a recommender system for the metadata fields of a LO, based on an application profile of the metadata standard OBAA. To generate the recommendations our system uses a MAS composed of agents and artifacts specialized in accessing the DBPedia entities through SPARQL queries. The system execution starts with the user providing the partial metadata and requesting the system to provide recommendations for other metadata fields. The GUI gathers the partial knowledge and sends it to the MAS, the agents then use the artifacts to query the DBPedia to find recommendations based on the partial knowledge provided. Finally, the recommendations are sent back to the GUI that shows them to the user.

We focused the metadata Title, Description and Keywords of the OBAA profile, that can easily be mapped to some of DBPedia properties, such as dbpedia-owl:abstract, rdfs:comment, db-prop:title, db-prop:name, foaf:name and rdfs:label. To generate the recommendations, the partial data is transformed into keywords through Natural Language

Processing (NLP), using the Apache Lucene. The keywords are ranked based on their frequency, and the most frequent ones are provided to the agents at the artifacts. Each keyword is obtained by three agents, each one specialized in a type of semantic search. The agents process the keywords in parallel through semantic artifacts that query the DB-Pedia using SPARQL, returning ontology individuals that are similar to the keyword. The properties of the most returned individuals are used as a basis for the recommendations.

Figure 1 describes the recommender model in more details. The cloud represents the MAS, that contains the agents and artifacts. The InputArtifact and the OutputArtifact work as an integration between the MAS and the GUI, and are responsible for the agents coordination, where the agents get the source data to process and put the resultant outputs, respectively. The main part of the system is composed by nine agents and artifacts specialized in three types of semantic searches at DBPedia: individuals, classes and categories, where category is an informational structure derived from Wikipedia. Each of these artifacts are used by only one agent, we did it so the system can process the queries in parallel, because the A&A model doesn't allow to more than an agent using an artifact at the exact same time. The semantic artifacts provide two ways of performing a semantic search, where the difference is a trade-off between quality and quantity. The artifacts return the results as the individuals that were more similar to the keyword.

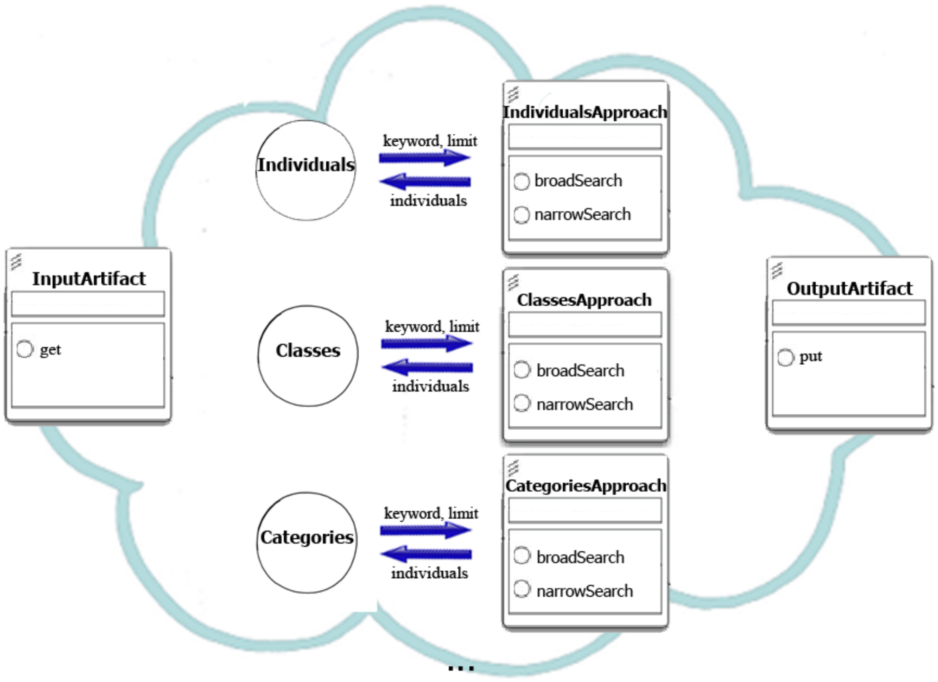

Figure 1. Overview of the recommender model.

As an example, the individual approach process the keywords in order to obtain individuals that contain the keyword in the properties db-prop:title or db-prop:name. A shorter version of the SPARQL query for the narrow search of this approach is shown below. The words preceded by the symbol @ are parameters that will be informed by the agents when calling the artifact's operation. The query states the prefixes, defines the result being returned at SELECT and the search criteria at WHERE. The filters ban the results that belong to categories which results are poor related with an educational context. The parameters that the agent informs in this query are the keyword being searched, the depth of the search, and the limit of results.

```
PREFIX db-prop: <http://dbpedia.org/property/>
PREFIX dbpedia-owl:
   <http://dbpedia.org/ontology/>
PREFIX rdf:
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:
   <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia_cat:
   <http://dbpedia.org/resource/Category:>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT distinct ?object
WHERE {
{?object db-prop:title ?title . ?title
   <bif:contains> "+@keyword+" . }
UNION
{?object  db-prop:name ?name. ?name
   <bif:contains> "+@keyword+" . }
filter (NOT EXISTS
   {?object dcterms:subject ?Category .
   ?Category skos:broader dbpedia_cat:Sports
   option(TRANSITIVE, T_DISTINCT,
   t_max(@t_max)) }) .
filter (NOT EXISTS
   {?object dcterms:subject ?Category .
   ?Category skos:broader dbpedia_cat:Companies
   option(TRANSITIVE, T_DISTINCT,
   t_max(@t_max)) }) .
}
LIMIT @limit
```

Below we show a shorted version of the code for the classes_approach agent. It starts by discovering the artifacts that it will use through the goal myTools. Those are the artifacts for input, output and the one specialized on the classes approach. Then the goal consumeItems checks if new keywords are available in the InputArtifact at 0.5 seconds intervals. When a keyword is obtained it is used in the processItem goal, which executes the semantic search. First, the agent calls the artifact's operation execute_broad. the the agent uses the results from the broad search to execute a narrow search. If after executing the narrow search there are no results, the agent will send to the OutputArtifact the results from the broad search, otherwise the results from the narrow search will be sent. This decision is made by the goal decideOutcome. Before sending the results to the OutputArtifact through the put operation, the agent calls the operation execute_getIndividuals to obtain the individuals from each class as the final result from the semantic search.

```
!observe.

+!observe : true
  <- .wait(200);
    ?myTools(A1, A2, Id);
    !consumeItems(Id).

+!consumeItems(Id): true
<- .wait(500);
get_for_ClassesApproach(Item);
!processItem(Item, Id);
!!consumeItems(Id).

(...)

+!processItem(Keyword, Id) : true
<-  execute_broad(Keyword, 30, R1)
   [artifact_id(Id)];
execute_narrow(R1, R2)[artifact_id(Id)];
isEmpty(R2, Test)[artifact_id("input")];
!decideOutcome(R1, R2, Test, Id).

+!decideOutcome(R1, R2, Test, Id) : Test
<- execute_getIndividuals(R1, 20, R3)
   [artifact_id(Id)];
put(R3).

(...)

+?myTools(A1, A2, A3): true
  <- lookupArtifact("input",A1);
    lookupArtifact("output",A2);
 .my_name(N);
 lookupArtifact(N,A3).

(...)
```

3. Conclusion and Future Work

In this paper we developed a recommender system where agents access the Semantic Web by means of artifacts that model the environment. The system is able to obtain recommendations for LO metadata using the partial knowledge provided by the user to process semantic queries on DBPedia through SPARQL. The contributions and results of this work can be observed in two points of view:

a) Metadata recommenders systems: To the best of our knowledge, in the context of recommenders specific for educational LO metadata there are no works available to compare our results with. The system developed in this paper is an initial prototype for this context. We noticed that our systems' recommendations were useful to obtain more information about a concept, but most of the time they were unrelated with the educational context. We believe that it happened because DBPedia isn't a semantic repository specific for educational purposes, then lacking the pedagogical information necessary for the recommendations.

b) Architectures that enable agents to access the Semantic Web: The main objective of our paper was to developed a model of system where agents can have access to the Semantic Web by means of artifacts, so this is context where we are able to make most of our comparisons. The main difference between our work and the related work ([Klapiscak and Bordini 2009, da Silva and Vieira 2007, Kagal et al. 2003, Zou et al. 2003, Chen et al. 2004, Katasonov and Terziyan 2008]) regarding the access to the Semantic Web by agents is that in this work the integration is accomplished by means of artifacts.

Due to the adoption of artifacts, our model provides reusability: By using artifacts agents are able to access the Semantic Web without the need to implement code or an architecture specific for them. Also, conceptually any BDI agent is able to use artifacts are developed. Other relevant points are that artifacts decreases the computational burden on the agent side, since the agents just activate the desired operation at the artifact, and they can perform other tasks while the artifact process the operation.

The semantic artifacts use SPARQL queries with a predefined general structure. In the current version the customizable parts by the agents are the keywords being searched, the prefixes, the filters and some parameters of the search. In a future version we could use the functionality of internal properties of the artifacts, which can be modified at runtime by agents to change the services being offered by the artifact. So the agents would be able to customize the queries to DBPedia at runtime. Other possible future works include: real time recommendations; utilize the contents of the LO or the user's personal data as context; and an empirical evaluation of the system.

References

Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5):28–37.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). Dbpedia-a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165.

Chen, H., Finin, T., Joshi, A., Kagal, L., Perich, F., and Chakraborty, D. (2004). Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 8(6).

da Silva, D. M. and Vieira, R. (2007). Argonaut: Integrating jason and jena for context aware computing based on owl ontologies. *Agent, Web Services, and Ontologies Integrated Methodologies*, page 19.

Kagal, L., Perich, F., Chen, H., Tolia, S., Zou, Y., Finin, T., Joshi, A., Peng, Y., Cost, R. S., and Nicholas, C. (2003). Agents making sense of the semantic web. In *Innovative Concepts for Agent-Based Systems*, pages 417–433. Springer.

Katasonov, A. and Terziyan, V. (2008). Semantic agent programming language (s-apl): A middleware platform for the semantic web. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing*, ICSC '08, pages 504–511. IEEE Computer Society.

Klapiscak, T. and Bordini, R. H. (2009). Jasdl: A practical programming approach combining agent and semantic web technologies. In *Declarative Agent Languages and Technologies VI*, pages 91–110. Springer.

Omicini, A., Ricci, A., and Viroli, M. (2008a). Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456.

Omicini, A., Ricci, A., and Viroli, M. (2008b). Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456.

Ricci, A., Viroli, M., and Omicini, A. (2007). Give agents their artifacts: the a&a approach for engineering working environments in mas. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 150. ACM.

Ricci, A., Viroli, M., and Omicini, A. (2008). The a&a programming model and technology for developing agent environments in mas. In *Programming multi-agent systems*, pages 89–106. Springer.

Weyns, D., Omicini, A., and Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous agents and multi-agent systems*, 14(1):5–30.

Zou, Y., Finin, T., Ding, L., Chen, H., and Pan, R. (2003). Using semantic web technology in multi-agent systems: a case study in the taga trading agent environment. In *Proceedings of the 5th international conference on Electronic commerce*, pages 95–101. ACM.