

Interactive Ontology-Based User Knowledge Acquisition: A Case Study

Lora Aroyo¹, Ronald Denaux¹, Vania Dimitrova², and Michael Pye²

¹ Eindhoven University of Technology, Faculty of Mathematics and Computer Science, P.O.Box 513, 5600 MB Eindhoven, The Netherlands

² School of Computing, University of Leeds, LS2 9NA, Leeds - UK
Contact: l.m.aroyo@tue.nl, vania@comp.leeds.ac.uk

Abstract. On the Semantic Web personalization technologies are needed to deal with user diversity. Our research aims at maximising the automation of acquisition of user knowledge, thus providing an effective solution for multi-faceted user modeling. This paper presents an approach to eliciting a user's conceptualization by engaging in an ontology-driven dialog. This is implemented as an OWL-based domain-independent diagnostic agent. We show the deployment of the agent in a use case for personalized management of learning content, which has been evaluated in three studies with users. Currently, the system is being deployed in a cultural heritage domain for personalized recommendation of museum resources.

1 Introduction

The rapid expansion of semantics-enriched services on the Web results in an exponential growth of the users of these services. They differ in their capabilities, expectations, goals, requirements, preferences and usage context. The one-size-fits-all approach to developing Web applications is becoming inappropriate. To fulfill the Semantic Web vision of improving the way computers and people work together, *context-aware* and *user-adaptive* technologies that take into account the *users' perspective* are needed [?]. User-adaptive systems automatically tailor their behavior to the needs of individual users. They can recommend items or documents that relate to the user's interests, provide links to relevant resources according to the user's goal, or offer explanation when needed. With the addition of explicit semantics, user-adaptive systems become context-aware.

A critical aspect in the realization of personalization is the acquisition of knowledge about the users. Traditionally it is represented in a *user model* [?]. The depth of the represented understanding about the users can vary from simple user profiles that focus mainly on users' preferences to sophisticated models that capture users' conceptualizations [?]. In the *open world* context the latter capture individual users' viewpoints, spanning from knowledge engineers to naive users. They also define the semantics of the user knowledge representation, thus facilitating the effective integration of the users within semantics-aware systems. This enables knowledge-enhanced reasoning to align the perspectives of

users and system designers, where various mismatches in meaning can be discovered and taken into account for more effective adaptation. The acquisition and maintenance of user conceptual models requires robust methods that integrate seamlessly in semantics-enhanced Web-based systems. An example of such an approach is proposed in here and is illustrated in an e-learning case study.

Why did we chose an e-learning use case? E-learning is a key application domain where the empowering role of semantics-enhanced technologies is being acknowledged. The Web is becoming the most popular educational medium nowadays, at schools, universities, and for professional training [?]. A prominent new stream of research on Educational Semantic Web [?] is being established. Recent successes in this field include semantics-based annotation and sharing of educational resources [?,?,?], as well as supporting the construction and sharing of knowledge among communities of learners and teachers [?,?]. Although there are initial attempts to develop semantic-aware personalization technologies for Web-based educational systems [?,?], this research is still in an embryonic stage. On the other hand, adaptive learning systems are well advanced in the addressing of the user' needs [?]. As pointed by Wolpers and Nejdll, future research in the Educational Semantic Web should include, among others, capturing the perspectives of different users based on observations from a variety of sources, and representing these perspectives in interoperable learner models [?]. The approach presented in this paper is a contribution in this direction. Furthermore, the proposed approach is domain-independent, and is currently being instantiated in museum and digital library domains.

The paper describes an ontology-based dialog agent, called OWL-OLM, that elicits and maintains a model of the user's conceptualization. The architecture of OWL-OLM is presented in Section 2, and the main components are described in the follow-up sections: domain ontology and a user model (Section 3) and dialog maintenance (Section 4). We illustrate the deployment of OWL-OLM within an RDF/OWL-based system for personalized management of learning content, called OntoAIMS Resource Browser (Section 6). It uses a multi-faceted user model built by collecting, interpreting and validating user data from diverse sources, such as user preferences, diagnostic dialog and monitoring the user interaction with the system [?]. The evaluation of OntoAIMS Resource Browser is outlined in Section 7. Finally, we discuss related and future work.

2 Architecture of OWL-OLM

OWL-OLM uses the STyLE-OLM framework for interactive ontology-based user modeling [?] and extends it to work with an OWL-based domain ontology and a user model. The architecture of OWL-OLM is presented in Figure 1.

The *Dialog Agent* is the main OWL-OLM component which maintains the user-knowledge acquisition dialog. The user-agent interaction is geared towards achieving the goal of the user, according to which the agent defines its *dialog goals*. For example, in a learning situation the user may want to be recommended what to read next on a certain topic. The goal of the dialog agent in this case

would be to assess the current state of the user's knowledge according to the requirements for the particular course task. In a museum case, the user may want to be recommended which painting to view next in the context of the user's current preferences. To do so, dialog subgoals are defined (e.g. to probe the user's knowledge of concepts related to the task or the preferences set). In addition, the user may want to clarify some part of the domain, so he asks questions. Dialog subgoals will be then to answer the user's questions and help him clarify the particular domain aspects.

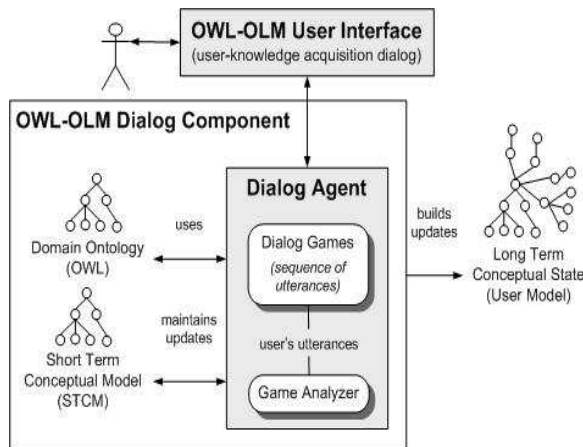


Fig. 1. The architecture of OWL-OLM.

The agent uses also a *Game Analyzer* that analyzes each user's utterance to decide the agent's response and to update the user's short-term conceptual state, see Section 4.2. When a dialog episode finishes, the short-term conceptual state is used to update the user's *Long-Term Conceptual State*, referred to as the *User Model*. The belief revision algorithm used in [?] is employed.

The user interacts with the system by using a graphical *user interface*, illustrated in Figure 2. The interface uses JGraph³ to present, create, and modify graphical utterances. The main components of OWL-OLM are described next.

3 Domain Ontology and a User's Conceptual Model

OWL-OLM is built as a user modeling component to be integrated in Semantic Web applications. OWL-OLM follows a general dialog framework that is domain independent and produces OWL-based user model. The only restriction imposed is that a URI of a domain ontology has to be provided, and that this ontology has to be defined in OWL. For the current instantiation, example from which

³ <http://www.jgraph.com/>

A *Domain Ontology* built in OWL is used to maintain the dialog and to update the user's *Short-Term Conceptual State*. The latter is also represented in OWL and provides a model of the user's conceptualization gathered throughout the dialog. Section 3 gives details.

The dialog agent maintains a *dialog episode goal*, which is divided into *sub-goals* that trigger *Dialog Games* - sequences of utterances to achieve a specific sub-goal, see Section 4.1.

The agent uses also a *Game Analyzer* that analyzes each user's utterance

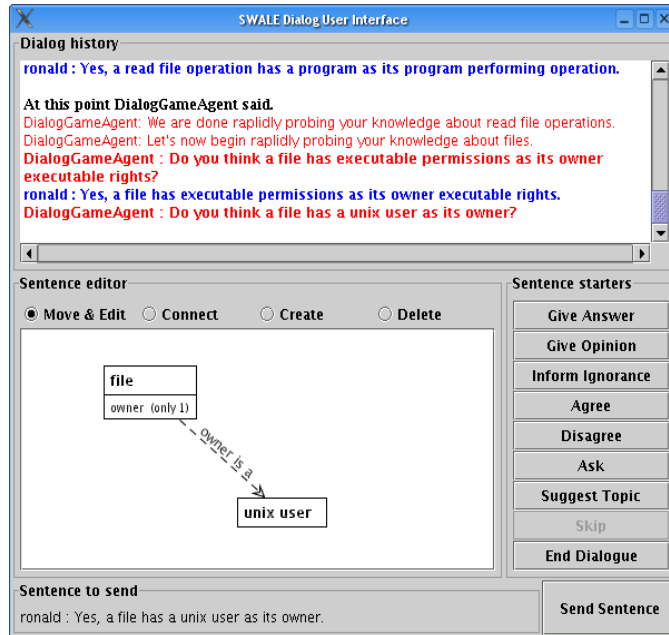


Fig. 2. Example of OWL-OLM interface - utterance (2) from the dialog in Section 5.

was given in Section 5, we use basic Linux ontology⁴ which includes concepts from an introductory Linux course taught at the University of Leeds, UK. The ontology was built by a domain expert from Eindhoven University of Technology by using Protégé [?], and is used in a Linux course, see Section 6.

OWL-OLM uses Jena 2.1 [?] extensively for input and output of OWL ontologies and models, creating and changing OWL resources and resolving domain ontology queries (in the Dialog Agent). The OWL generic reasoner from Jena is employed to make inferences from the domain ontology.

As shown in the OWL-OLM architecture (Figure 1), the dialog agent extracts a user's *Short Term Conceptual State*, used to tune the user's *Long Term Conceptual State*. The main idea of a conceptual state is that it gives a partial model of a user's conceptualization which is linked to one or more existing ontologies. A user's conceptual state is defined in OWL-OLM as a triple of URIs pointing to a *Conceptual model*, a *Domain ontology* and a *User*. The *conceptual model* is specified in OWL resembling an ontology specification, i.e. it defines classes, individuals, and properties, and uses OWL properties to define relationships. It makes links to resources defined in the domain ontology, when possible (a resource can represent a concept or a relationship between concepts [?]).

To indicate how the conceptual model is derived, a set of properties⁵ is used:

⁴ SWALE project, <http://wwwis.win.tue.nl:8080/~swale/blo>

⁵ An RDF specification of those properties, <http://wwwis.win.tue.nl/~swale/aimsUM>

- **times_used_correctly**: the number of times that the user has used a resource correctly, i.e. the way in which the resource was used was supported by the domain ontology either because the resource was found in the domain ontology or because it could be inferred from it.
- **times_used_wrongly**: the number of times that the user has used a resource in a way that contradicts the domain ontology.
- **times_affirmed**: the number of times that the user has stated he knows about this resource.

The above properties are related to classes, individuals, and properties in the conceptual model. The following excerpt shows the state of the class `Filesystem_node` (called by its label `file` in the example in Section 5).

```
<rdf:Description rdf:about="blo:Filesystem_node">
  <rdfs:comment rdf:datatype="xml:string">
    Any set of data that has a pathname on the
    filesystem.
  </rdfs:comment>
  <rdfs:label>file</rdfs:label>
  <rdf:type rdf:resource="owl:Class"/>
  <aimsUM:times_used rdf:datatype="xml:long">
    12</aimsUM:times_used>
  <aimsUM:times_used_correctly
    rdf:datatype="xml:long">
    10</aimsUM:times_used_correctly>
  <aimsUM:times_used_wrongly
    rdf:datatype="xml:long">
    2</aimsUM:times_used_wronlgy>
  <aimsUM:times_affirmed rdf:datatype="xml:long">
    3</aimsUM:times_affirmed>
  <aimsUM:times_denied rdf:datatype="xml:long">
    1</aimsUM:times_denied>
</rdf:Description>
```

This example shows that the user has used the class `Filesystem_node` a total of 12 times; 10 times supported by the domain ontology and twice not supported. He has stated 3 times that he knows the concept `Filesystem_node` and once that he does not. Classes, individuals, object properties, and datatype properties are annotated in the same way.

We also need to capture the relations between concepts that the user builds. In order to associate the properties described above with these relations we use *reified statements* [?]. For instance, in utterance (15) in the example given in Section 5 the user states that class `Move_file_operation` is a subclass of `Command`. We can create a reified statement referring to this relationship and add the `aimsUM:times_used_wrongly` property to this statement, as shown below:

```
<rdf:Description rdf:nodeID="A273">
  <rdf:type
    rdf:resource="rdf:Statement"/>
  <rdf:subject
    rdf:resource="blo:Move_file_operation"/>
  <rdf:predicate
    rdf:resource="rdfs:subClassOf"/>
  <rdf:object rdf:resource="blo:Command"/>
  <aimsUM:times_used
    rdf:datatype="xml:long">1
  </aimsUM:times_used>
```

```

<aimsUM:times_used_wrongly
  rdf:datatype="xsls:long">1
</aimsUM:times_used_wrongly>
</rdf:Description>

```

The above denotes that the user has used the `rdfs:subClassOf` relationship between `Move_file_operation` and `Command`. This is more detailed than only associating properties with the individual classes, as in the first excerpt.

4 Dialog Maintenance

4.1 Dialog Games

The dialog in OWL-OLM is organized as a series of dialog games that represent the dialog episodes. A dialog game \mathcal{DG} in OWL-OLM is defined as:

$$\mathcal{DG} = (\mathcal{C}, \mathcal{P}, \mathcal{R}, \mathcal{U}, \mathcal{S})$$

where: \mathcal{C} is a set of domain concepts and relations targeted in the game used to maintain a *global focus* [?]; \mathcal{P} is a set of *preconditions* which define conditions of the state of the system (e.g. a history of previous utterances or a user's current conceptual state) which are needed to trigger the game; \mathcal{R} is a set of *postconditions* that define conditions of the state of the system (e.g. changes in the user's conceptual state) which become valid when the game ends; \mathcal{U} is a set of *rules* to generate dialog utterances; \mathcal{S} defines the *scope* that is a set of concepts used to maintain a dialog focus space [?]. The definition follows the formalization in [?] which is derived from a linguistic dialog games model [?].

The building blocks of every dialog are *utterances*: sentences interchanged between the dialog agent and the user. Each utterance consists of three parts: an originator, intention, and OWL statement. *Originator* is the producer of the utterance, which can be the dialog agent or the user.

OWL statement is the domain-related proposition of an utterance. An OWL statement is a small OWL model that defines a set of concepts and relations. The OWL model is restricted to only *one semantic relation*, as it represents the domain-related semantics of a single utterance. OWL statements are used to (a) extract a section of an OWL ontology and only focus on this section; (b) generate text templates to render the semantics enclosed by the OWL statement, for the interface with the user; and (c) exchange very focused information about the exact relationship between a small number of classes, individuals and ontology properties, which is critical for diagnosing users and identifying mismatches between the user's conceptual model and the domain ontology, as shown later. Table 1 illustrates some OWL statements.

Intention states the dialog purpose of the utterance, i.e. the intention of the originator of the utterance. Sample intention operators used in OWL-OLM are ⁶ (1) **Give Answer**, where the originator is answering a previous utterance

⁶ See screen shots for more examples.

which was a question with the semantics enclosed in the OWL statement; (2) **Agree**, where the originator states that he agrees with the semantics enclosed in the OWL statement; (3) **Ask**, where the originator asks whether the semantics described in the OWL statement is true.

Table 1. Some types of OWL statements used in OWL-OLM

Name	Basic RDF triple(s)	Comment
Empty	none	Used in combination with intentions which do not require an OWL statement
SingleClass	c <code>rdf:type owl:Class</code>	Defines a single class c .
SingleObjectProperty	p <code>rdf:type owl:ObjectProperty</code>	Defines a single object property p without a domain or range
InstanceOf	i <code>rdf:type c</code>	Declares that i is an individual of class c
SubClass	c_1 <code>rdfs:subClassOf c_2</code>	States that class c_1 is a subclass of class c_2
ObjectProperty	p <code>rdf:domain c</code> p <code>rdf:range r</code>	Defines that class c has an object property p
QueryInstance	i <code>rdf:type c</code> i <code>rdf:type swaleQ:QuestionResource</code>	Declares that i is an individual of class c and that i is a question
QueryClassOfInd	i <code>rdf:type c</code> c <code>rdf:type swaleQ:QuestionResource</code>	i is an individual of c and c is a question
QueryDtPropDomain	p <code>rdf:domain c</code> c <code>rdf:type swaleQ:QuestionResource</code>	Class c has a datatype property p and c is a question

To achieve effective dialogs, which feel natural and efficient to the user, several problems have been tackled. Firstly, a *dialog focus* is maintained, so that the interaction is structured and organized. In terms of utterances, this means that consecutive utterances should have a concept in common whenever possible. For this, a scope of relevant concepts is maintained. Secondly, the *dialog continuity* is ensured to provide a logical exchange of utterances. This means that each utterance has to be a logical response to the previous one. Cue phrases are used when a game is opened, closed, or re-initiated, to show the logical flow and to communicate the dialog goal, as in the example in Section 5. Thirdly, to *avoid repetition of utterances* a history of the dialog is maintained to exclude already said utterances. Finally, *mixed initiative* is maintained to allow both participants in the dialog to introduce a new topic and ask clarification questions.

4.2 Game Analyzer

We will now explain how OWL-OLM analyzes the utterances received from the user and how the results of this analysis are used to decide the next move of the Dialog Agent and to update the user's conceptual state. The following features are analyzed:

- The *scope* and intention of two consecutive utterances are compared. If the scopes do not match, the agent has to decide whether to follow the user and change the dialog focus or to stay in the current scope;
- The *incoming utterance* is analyzed to determine whether it asks a question to the system, which triggers a question answering game;

- The *OWL statement* of the incoming utterance is compared to the domain ontology to determine whether the semantic relation described by the OWL statement is supported by the domain ontology;
- If an OWL statement is not supported by the domain ontology, the agent checks for a *recognizable mismatch* between the statement and the ontology.

During the analysis, the default OWL reasoner of Jena [?] is used to infer relationships that follow from the domain ontology. The reasoner is used to determine whether the OWL statement of an utterance is supported by the domain ontology. This is the case if and only if every RDF triple in the statement can be inferred from the domain ontology.

4.3 Mismatches

When the user submits an utterance that cannot be verified by the domain ontology, this is considered as a mismatch. We define a *mismatch* as any RDF triple in the utterance’s OWL statement which cannot be found in the existing domain ontology. In this way we can determine the type of mismatch by using the basic building blocks in OWL. The semantics of a mismatch is that a resource or relationship in the OWL statement is not supported by the domain ontology. Table 2 shows some of the types of mismatches which can be detected.

Table 2. Mismatches detected in OWL-OLM

Type	Comment
Unknown	None of the other types apply
OntClass	A class cannot be found
Individual	An individual cannot be found
ObProp	An object property cannot be found
DtProp	A datatype property cannot be found
SubClass	A subclass relationship cannot be found
InstanceOf	The link between an individual and its class cannot be found
ObProp-Rel	The domain ontology (DO) doesn’t suggest that two resources are related by this object property
DtProp-Rel	The DO doesn’t suggest that two resources are related by this datatype property
Domain	The DO doesn’t suggest that a resource is the domain of a property
Range	The DO doesn’t support that a resource is the range of a property
ObProp-Val	The DO doesn’t support that a resource has this value for this object property

By classifying the types of mismatches OntoAIMS Resource Browser is able to define dialog games to clarify each of the recognized mismatches. For instance, an example of clarification dialog is given in Section 5: utterance (19) prompts a mismatch of a type *SubClass*, which is recognized by OWL-OLM and a clarification dialog game (see utterances (14) - (19)) is initiated.

4.4 Answering Questions

Two different types of questions are catered for: those which can be answered with an yes or no (*yes/no questions*) and those where the answer contains an

element missing in the question (*open questions*). The latter are based on OWL statements that have special resource type - a *question resource*. A **question resource** is any resource in an OWL model which is marked as being the same as the resource `swale_:Question_Resource`⁷. This means we can mark any resource R in an OWL statement as being a question resource by adding the following RDF triple to the OWL model: (R `rdfs:sameAs` `swale_:Question_Resource`)

Questions are answered by transforming them to queries and feeding these to any OWL query answering mechanism. OWL-OLM uses Jena [?] to find triples in the domain ontology which match the query. A user's question may interrupt the current dialog game and initiate a new game where the Dialog Agent tries to answer the question as best as it can and to probe for relevant knowledge.

5 An Illustrative Example

We illustrate here the application of OWL-OLM for the acquisition of user knowledge with an example from the OntoAIMS Resources Browser (see Section 6). OntoAIMS employs OWL-OLM to extract a conceptual model of the learner and uses this model to recommend an appropriate to his knowledge state learning task or learning resources. The User in this example is a student who follows an introductory Web-based course on "*Linux Operating System*". The OWL-OLM Dialog Agent utilizes an existing domain ontology [?] to explore aspects of the User's conceptualization of the topic "*Files and operations on files*". The interaction is conducted in a graphical way (see fig. 2) but due to space limit is presented here only in the text form generated in the dialog history of OWL-OLM.

Opening a probing dialog game

- (1) **DialogAgent:** I'll try to suggest a suitable task for you. I will first ask you some questions related to "*Files and operations on files*".
- (2) **DialogAgent:** Do you think a `file` has a `unix user` as its owner?
- (3) **User:** Yes, a `file` has a `unix user` as its owner.
- (4) **DialogAgent:** Do you think a `file` has a `path` as its `full name`?
- (5) **User:** I don't know whether a `file` has a `path` as its `full name`.

The agent opens the *dialog episode* and informs what the *goal* is. It then initiates a *probing dialog game* to elicit the part of the user's conceptualization related to the current task. It begins with the concept `file`, which is the main concept in this topic, and checks whether the user knows concepts linked to `file` – utterances (2) - (5). Figure 2 shows the OWL-OLM interface for utterance (2). The OWL statement of a dialog utterance is rendered in a graphical form, while the intention is rendered as a sentence opener. To construct his reply, utterance (3), the user edits the diagram with the OWL statement by using basic graphical commands. Also, when he double clicks on an object, he can edit its class/individual name, add/delete properties, or create links with other objects.

⁷ http://wwwis.win.tue.nl/~swale/swale_question

To state intention, the user uses a sentence opener, see the buttons group to the right in Figure 2. The textual form of the composed dialog utterance is shown in the bottom. To send his utterance, the user clicks on the **Send sentence** button. The utterance is added to the the dialog history, shown at the top.

User question - a new game is initiated

- (6) **User:** Can you tell me more about what can be the **group readable right** of a **file**?
- (7) **DialogAgent:** As far as I know, a **file** has a **readable permission** as its **group readable right**.
- (8) **User :** [*Skip*]
- (9) **DialogAgent:** Let us return to discussing **files**.
- (10) **DialogAgent:** We will continue with discussing about. **moving files**.
- (11) **DialogAgent:** Do you know what **moving file** is?
- (12) **User:** I think that **moving file** is a **command**.

The user interrupts the dialog by asking a question (6). The agent suspends the current dialog game and initiates a new game to *answer the user's question*. It sends a query to the domain ontology and then composes the answer. The agent then waits to see whether the user has more questions. Since the user's questions have been answered, the agent returns to the dialog game that probes what the user knows about **file**. When finished the agent closes the game that probes for **files**. It then opens a dialog game that probes for another main concept – **moving files** – and starts with probing for the parent. The user answer in utterance (12) indicates a *mismatch* (see Section 4.3). According to the domain ontology used by the agent, **moving a file** is a **file operation** not a **command**. The agent again suspends the probing game and starts a *clarification game*.

Mismatch is recognized - a clarification game is initiated

- (13) **DialogAgent:** Why do you think **moving file** is a **command**.
- (14) **DialogAgent:** Do you think **moving a file** is also a **file operation**?
- (15) **User:** Yes, I think that **moving a file** is a **file operation**.
- (16) **DialogAgent:** Do you think a **command** has a **version**?
- (17) **User:** Yes, a **command** has a **version**.
- (18) **DialogAgent:** You think that **moving file** is a **file operation**, as well as a **command**.
- (19) **User:** I don't think that **moving file** is a **command**.

The agent first checks for an alignment between the user's conceptualization and the domain ontology (utterance (14)), and then checks for inherited properties from the **subClass** link between **moving file** and **file operation**. If the answer is positive, the agent checks for an inherited property from the **subClass** link the user makes between **moving file** and **command**. Because the user does not confirm the property, the agent checks the property for **command** to highlight the inheritance link the user makes between **moving files** and **command**. At the end, the agent checks again the mismatch, which is now corrected by the user.

6 Use of OWL-OLM for Adaptive Content Management

In this section we demonstrate how OWL-OLM was utilized to enable personalization in a semantic-aware e-learning system, called OntoAIMS Resource Browser. OntoAIMS Resource Browser⁸ is an ontology-based version of the AIMS Resource Browser *adaptive information management system* [?] providing an information searching and browsing environment that enables users to identify the most appropriate course task to work on and to find and learn the relevant terminology and resources necessary for performing this particular task.

OntoAIMS Resource Browser uses ontologies to represent the aspects of the application semantics, to allow a strict separation of domain-dependent data, application-related data and resources, and to further enable reusability and sharing of data on the Semantic Web. A *Domain Ontology* represents the domain terms and their relationships. The content is annotated and stored in a *Resource model*, linked to the terms of the domain ontology. The course aspects are modeled as a hierarchy of course tasks in a *Course Task Model*. Each task specifies domain terms and learning objects as part of the task prerequisites, input and output. Adaptive mechanisms are employed for sequencing the course tasks and thus providing the most efficient way for the users to navigate through the structure, terminology and learning material of the course. In OntoAIMS Resource Browser, the learners use a graphical environment for browsing through a graphical representation of the domain conceptual space, and to browse through a collection of semantically annotated resources ranked by the system according to their relevance to the task and the user query, see Figure 3.

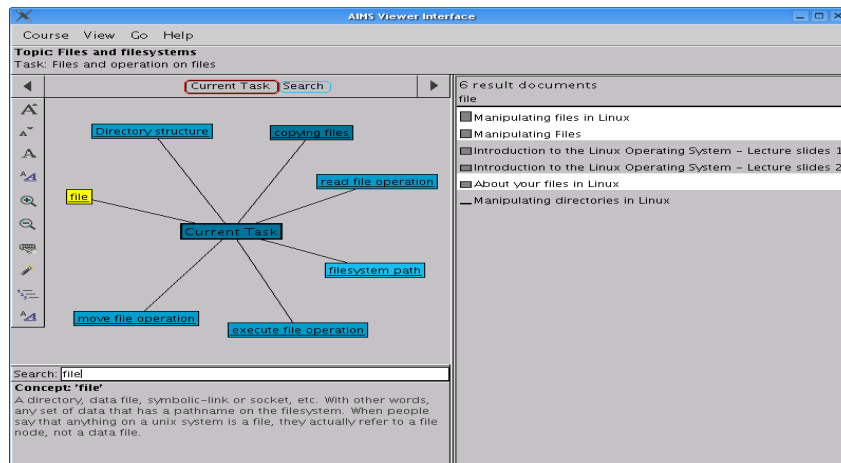


Fig. 3. OntoAIMS resource browser

⁸ Accessible at: <http://swale.comp.leeds.ac.uk:8080/staims/viewer.html>, username *visitor*, password *visitor*.

A key role in the ranking and recommending of tasks and resources in OntoAIMS is played by the *User Model*. OntoAIMS aims at covering an extensive range of user's aspects, e.g. goals, conceptual state, preferences, personal characteristics, etc. thus allowing an unobtrusive way to collect various user data. Dialogs similar to the example shown in Section 5 are conducted by the OWL-OLM agent within OntoAIMS to (a) validate the analysis of the user data, (b) elicit a user's conceptual state, and (c) build and maintain a dynamic user model as a basis for a personalized information management process.

7 Evaluation Studies with Users

Three user studies were conducted with the current instantiation of OntoAIMS Resource Browser in a domain of Linux. Initially, *six users*, postgraduate students and staff from the universities of Leeds and Eindhoven, took part in video recorded and monitored think aloud walk through sessions, as well as in detailed usability questionnaires. An improved version of the system was used in a second study with *ten first year Computing students* at Leeds. It followed a two-week introductory course on Linux. The users were asked to study resources on Linux recommended by the system. Detailed description of the OntoAIMS Resource Browser evaluation is given in [?]. A final version was presented as an Interactive Event at the AIED'05 conference and was tested with *ten international experts in Learning Management Systems*.

In general, the users appreciated the help and guidance provided by OntoAIMS resource browser and regarded the system as useful. The dialogs to discover the level of user's knowledge lasted about 5-10 minutes, which saved significant time for these users (who otherwise would have been offered reading on topics they were familiar with). Less knowledgeable users became aware of aspects of the domain they did not know (which were addressed in the OWL-OLM dialog), and liked that the system recommended basic Linux reading. OWL-OLM was regarded as a key tool seamlessly integrated in the whole environment, and was seen both as *complying with the overall goal* and *unobtrusive*. The dialog was seen as coherent by the users. The evaluation revealed also pitfalls, to be addressed in the next OWL-OLM versions, with respect to mismatches dialogs, smooth switch between the dialog and OntoAIMS Resource Browser, awareness how the conceptual model is used for adaptation, and several interface issues. The purpose of these first three studies were to test the usability and effectiveness of the dialog as a user model elicitation mechanism. With the current extension of the system and its deployment in other application domains, we plan to perform studies to test the advantages of the user modeling approach with respect to the recall and precision of resource and task recommendations.

8 Related Work & Discussion

Different perspectives of enabling personalization on the Semantic Web are being addressed recently. Our work on OntoAIMS Resource Browser is situated

within the field of User-adaptive Web-based Information Systems [?] that capitalize on adaptation techniques to provide individual users with the *right* service at the *right* time and in the *right* way. The OWL-OLM approach is similar to the interactive sessions used in HUMOS [?]. While HUMOS uses a very simple probing dialog to elicit initial, fairly basic profile of the user, OWL-OLM considers more in-depth interactions that extract enhanced user models. By using Semantic Web technologies, enriched semantics for the user data can be achieved to allow for more efficient reasoning and, thus, more accurate user modeling results. Stojanovic [?] in his comprehensive query refinement approach proves also the need and the benefit of an interactive 'step-by-step' query refinement process, which considers the user's perspective on the domain conceptualization in the retrieval process. The approach capitalizes on the improvement of the user and context modeling versus the improvement of the information retrieval process. In addition to this, our work shows how such interactive approach can be realized and visualized with an OWL domain ontology. An example of an NLP-based interactive knowledge acquisition interface is given by Chklovski and Gil [?]. As in OWL-OLM it shows the need for a clarification dialog with the user to correctly specify domain knowledge from multiple perspectives. Moreover, the authors prove the efficiency of using semantic rich structures to "collect semantically interpretable knowledge while interacting in natural language", as exemplified in OWL-OLM as well.

A strong argument is being formed recently to stress the importance of sharable and reusable user models, as well as personalization methods on the Semantic Web[?]. Both OWL-OLM and OntoAIMS Resource Browser contribute to an on-going work in this area [?,?]. The work we present on eliciting a user's conceptualization based on an existing domain ontology relates to the research on aligning and reconciling ontologies, reviewed in [?,?]. However, there is a crucial difference between the *user-expert* alignment considered in OWL-OLM and the alignment of two (or more) expert ontologies considered in existing, widely used tools, such as PROMPT [?]. Results from their empirical studies on aligning expert user's ontologies as well as the existing, robust ontology aligning algorithms (e.g. using word concordances and synonyms) can be very useful to extend the OWL-OLM mismatch patterns. Most of the methodologies applied for building shared ontologies have a dialog part at some stage to enable experts to clarify aspects of their conceptualizations. This confirms that approaches like OWL-OLM are viable for capturing a user's conceptualization. Finally, the user modeling dialog in OWL-OLM is similar to negotiation between agents who share knowledge and clarify meaning, e.g. [?].

9 Conclusion and Future Work

We presented OWL-OLM - a novel framework for eliciting a user's conceptualization based on an ontology-driven dialog. We focused on the formal specification and the architectural design of the diagnostic dialog by illustrating the following aspects: (a) maintaining dialog coherence, (b) answering different questions

and (c) identifying mismatches. OWL-OLM makes extensive use of Jena for OWL-based reasoning to maintain the dialogue and update the user model. We demonstrated the utilization of OWL-OLM in OntoAIMS Resource Browser - an RDF/OWL-based software architecture for adaptive learning content management. User studies showed that the user model extracted by OWL-OLM could be used to improve personalization in OntoAIMS Resource Browser. Other applications explored are *digital libraries* and *museums*, where effective help can only be provided if a user's view on the subject domain is considered. Possible other applications include *online banking*, where a probing dialog can be used to quickly identify what conceptual models the users have of key terms, or *online catalogues*, where the search is driven by some taxonomy which may often differ from the user's perception of the domain. In overall, the novel aspects demonstrated in this paper are: (a) ontological approach for integration of methods for eliciting and utilizing user models; (b) improved adaptation functionality resulted from that integration, validated in studies with real users; (c) support of interoperability and reusability on the educational Semantic Web.

Future work will focus on the development of a good classification of user's mismatches and patterns for clarification dialog based on systematic studies of empirical and computational approaches for ontology aligning and reconciliation. In-depth studies are needed to design effective knowledge elicitation tools suited not for ontology engineers, but for users with a wide range of experiences. Finally, it appears useful to provide also a text form for communication and to allow the users to choose a preferred interaction medium.

Acknowledgments. The research was partly supported by the UK-Netherlands Partnership in Science program and the EU Network of Excellence PROLEARN.