OntoWeaver S: supporting the design of knowledge portals

Conference or Workshop Item

OntoWeaver-S: Supporting the Design of Knowledge Portals

Yuangui Lei, Enrico Motta, John Domingue

Knowledge Media Institute, the Open University
{y.lei, e.motta, j.b.domingue}@open.ac.uk

Abstract. This paper presents OntoWeaver-S, an ontology-based infrastructure for building knowledge portals. In particular, OntoWeaver-S is integrated with a comprehensive web service platform, IRS-II, for the publication, discovery, and execution of web services. In this way, OntoWeaver-S supports the access and provision of remote web services for knowledge portals. Moreover, it provides a set of comprehensive site ontologies to model and represent knowledge portals, and thus is able to offer high level support for the design and development process. Finally, OntoWeaver-S provides a set of powerful tools to support knowledge portals at design time as well as at run time.

1 Introduction

The semantic web [2] is the vision of next generation of the World Wide Web. It extends the current web by associating well-structured meaning with web resources. The major advantage of doing so is that it enables the information understandable and consumable not only for humans, but for computers as well. A substantial amount of efforts have been made for delivering the Semantic Web [19, 13, 15, 5, 21, 16]. In particular, a number of knowledge portals have been built, which allow users enjoying the benefits gained from the semantic web technology on information sharing and exchanging in specified communities. Examples include OntoWeb portal [19] (http://ontoweb.aifb.uni-karlsruhe.de/), Esperonto portal (http://www.esperonto.net) and KMi Semantic Portal (http://plainmoor.open.ac.uk/ksp). These knowledge portals typically employ a domain ontology as a share basis for information communication and information exchanging. The typical functionalities provided by knowledge portals include:

- *Information provision*, which allows community users to submit information and make contributions to their communities.
- *Information presentation (or visualization)*, which visualizes the underlying data content coming from different sources.
- *Information querying*, which allows users to make queries over the underlying data sources.

Building such knowledge portals is a complex task. An ad-hoc development methodology often results in few re-usable components, costly and time consuming development processes, and poor performance on maintenance. In this context, a few approaches and tools have been developed [14, 8, 4, 18], which attempt to either automate the process of generating knowledge portals [4, 18] or facilitate and guide the activities involved in the design of knowledge portals [14, 8]. However, a number of requirements of knowledge portals have not been fully addressed in these frameworks:

- The need for accessing and providing remote web services. This requires the design frameworks of knowledge portals to be integrated with web service management platforms to allow end users to share their web services with others in their communities.
- The need for high-quality user interfaces. This calls for modelling languages with the expressive capability to describe sophisticated user interfaces and presentation styles,

which provide appropriate facilities to allow end users to navigate and manipulate the back-end data sources and access remote web services.
- The ability to present personalized views of knowledge portals to individual users. This requires comprehensive customization support from the design frameworks.

OntoWeaver-S is an ontology-based application, which offers comprehensive support for the design and development of knowledge portals. In particular, it addresses the issues mentioned above by the following approaches:

- OntoWeaver-S extends OntoWeaver [9, 10, 11] by means of integrating the web service framework IRS-II [15] within the data-intensive web site design framework. In this way, OntoWeaver-S provides support for the target knowledge portals accessing and managing remote web services.
- OntoWeaver-S extends the OntoWeaver site view ontology to describe sophisticated site views of knowledge portals, including navigational structures and user interfaces, which allow the access to the back-end data sources and web services. In particular, web services can be specified within the site view components to allow the invocation of web services and the presentation of the web services results.
- OntoWeaver-S addresses the customization issues for knowledge portals by means of the inherited customization framework (from OntoWeaver).
- OntoWeaver-S offers a set of powerful tools for supporting knowledge portals at design time as well as at run time.

In this paper we focus on the support provided by OntoWeaver-S for the design and development of knowledge portals. We begin in section 2 by explaining the rationales of OntoWeaver-S. We then illustrate the OntoWeaver-S approach to modelling knowledge portals in section 3. Thereafter, in section 4 we discuss the customization support. In section 5 we describe the design-time support and the run-time support that the OntoWeaver-S tools provide for knowledge portals. In section 6 we describe the related work. Finally in section 7 we conclude our work and present the future work.

2 OntoWeaver-S Rationales

As shown in figure 1, OntoWeaver-S views a knowledge portal as a four-layer architecture: *a data layer*, *a site view layer*, *a presentation layer* and *a customization layer*. The data layer consists of data content that can be accessed and manipulated by means of accessing knowledge portals. It comprises *a domain ontology*, which serves as a share basis for knowledge exchanging, *structured semantic web content*, which represents instances of the domain ontology concepts, *unstructured conventional web content*, which is submitted by community users, and *remote web services*, which can be accessed through the user interfaces of knowledge portals. The site view layer defines sophisticated

site views, which support the navigation within knowledge portals and the access to the data layer of knowledge portals. The presentation layer concerns the visual appearance and layout of knowledge portals. The customization layer describes customization requirements. It applies customization rules to the presentation layer, the site view layer, and the data layer to present customized views to individual users.

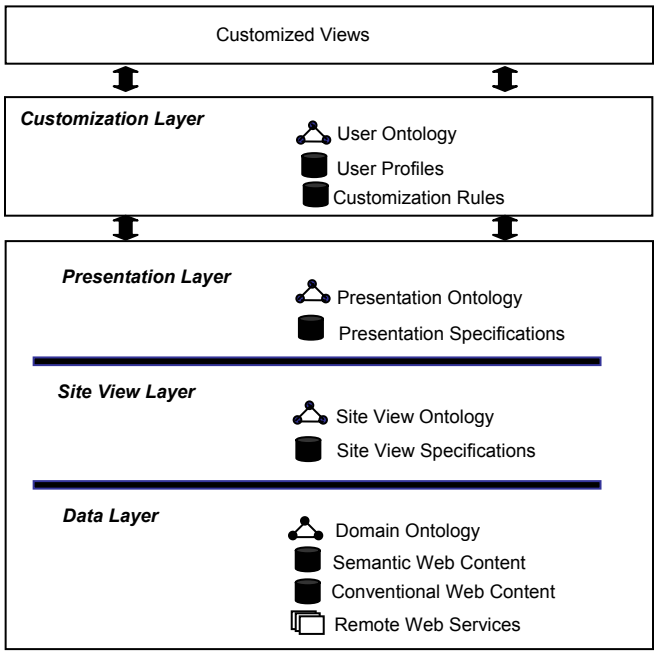

Fig.1. The architecture of knowledge portals in OntoWeaver-S

OntoWeaver-S provides *a site view ontology*, *a presentation ontology*, *a customization framework*, and *a set of tools* to support the design and development of knowledge portals. Specifically, the site view ontology offers a set of comprehensive constructs to describe the site view layer; the presentation ontology allows the design of the presentation layer to be carried out at a high level of abstraction; and the customization framework offers high level support for the specification of customization requirements. Moreover, OntoWeaver-S is integrated with a comprehensive platform IRS-II to support the provision and access of remote web services for knowledge portals.

2.1 Integrating Web Services into Knowledge Portals

IRS-II is an implemented infrastructure, which has been developed in our lab, the Knowledge Media Institute (http://kmi.open.ac.uk). It supports the publication, discovery, and execution of semantic web services. The following informal specification shows the task description of a semantic web service, which answers requests for flights in accordance with the given user requirements.

Task Ontology: *flight-service*
Task Name: *find-flights*
Input Roles: *from-place (type: city)*
to-place (type: city)
depart-time (type: time-point)
arrival-time (type: time-point)
budget (type: amount-of-money)
Output Role: *flights (type: Flight)*

To invoke this semantic web service, a user (or an application) simply asks for the task to be achieved in terms of the task name *find-flights* and the task ontology name *flight-service,* the IRS-II *broker* then selects an appropriate problem solving method (*PSM*) and then uses *grounding information* to locate and invoke the corresponding web service – see [15] for a detailed description of IRS-II. In particular, the input roles carry parameters for executing the corresponding web service; the output roles store the service results. Please note that IRS-II only supports one output role at the moment. The data type of the output role can be primitive e.g. String and Integer, or non-primitive, e.g. being a domain class. When the data type of an output role is not primitive, IRS-II uses XML [24] to represent results. For example, IRS-II uses XML to represent the results of the web service find-flights, which are instances of the class *Flight* defined in the domain ontology of the semantic web service.

OntoWeaver-S employs IRS-II as a platform to integrate web services into knowledge portals. On the one hand, OntoWeaver-S relies on IRS-II to enable the access of remote web services, as IRS-II is able to locate and invoke remote web services and pass results back. On the other hand, OntoWeaver-S uses IRS-II as a platform to allow the provision of web services for knowledge portals.

Figure 2 shows the process of accessing remote web services in an OntoWeaver-S generated knowledge portal. OntoWeaver-S provides a run-time tool, called *Service Integrator*, to integrate IRS-II with knowledge portals. Specifically, the Service Integrator collects information from a knowledge portal, then calls the IRS-II server (by means of IRS-II APIs) to invoke the specified web service and gets results from IRS-II, and finally the Service Integrator passes the service results back to the knowledge portal. This process is invoked by end users when they submit information for accessing web services. Please see [12] for more details.

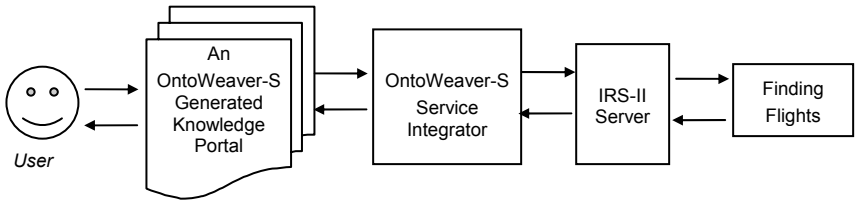

Fig.2. The process of accessing web services in OntoWeaver-S generated knowledge portals

3 Modelling Knowledge Portals

As mentioned earlier, OntoWeaver-S offers a site view ontology to model the site view layer of knowledge portals. Figure 3 shows an overview of the site view ontology. It models a web site as a collection of logical resources; the logical resources describe web pages and are abstracted as compositions of resource components; and the resource components are described as compositions of a number of site view elements, e.g. output elements, input elements, command elements, and sub resource components.

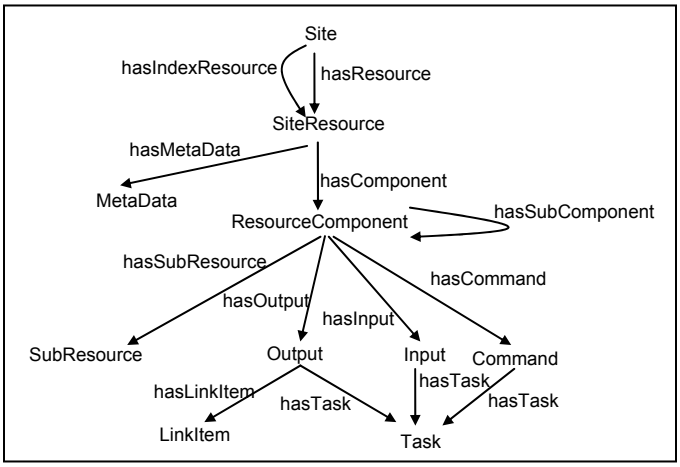

Fig.3. An overview of the OntoWeaver-S site view ontology

The site view ontology offers *a set of navigational constructs* to facilitate the specification of complex navigational structures and *a set of user interface constructs* to allow for the composition of sophisticated user interfaces. The user interface constructs

can be further classified into *atomic user interface constructs*, which describe atomic user interface elements that can not be further decomposed into other elements, and *composite user interface constructs*, which describe composite user interface elements.

On the one hand, these constructs can provide a fine grained level support for user interface composition. On the other hand, these constructs support the high level specification of access to semantic web services. Specifically, within the user interface constructs, remote web services are described in terms of *tasks*, *input roles* and *output roles*, which comply with the semantic representation approach employed in IRS-II. Please note that the concept of output role in OntoWeaver-S is slightly different from IRS-II when the data type of the output role of a web service is a class entity. As a class entity may have a number of slots, the output roles in OntoWeaver-S in this context refer to the slots of the result instances of the corresponding web service. For example, to present the results of the web service *find-flights*, the output roles specified in the user interface elements are slots of the class *Flight*.

3.1 Modelling Site Structures

By the term of site structure, we mean the coarse-grained level structure of an entire knowledge portal. A site structure comprises an index page node, which defines an entry point for the knowledge portal, a number of page nodes, which form the navigation space, and the URI of a domain ontology, which specifies the domain ontology for information sharing in the target knowledge portal. Like OntoWeaver, OntoWeaver-S relies on the construct *Site* to describe the components of web sites, *SiteResource* to define page nodes and *LinkItem* to express link relations between the page nodes. In particular, at the coarse-grained level, the detailed content of page nodes is not concerned. Each page node contains *a meta-data part* to describe its initial purpose and *a navigation component part* to hold links, which allow navigation from one page node to another. As the principle of specifying site structures in OntoWeaver-S remains the same with that in OntoWeaver, we will not detail the specification here. Please refer to [11] for details.

3.2 Modelling the Composition of Web Pages

Figure 4 shows a sample user interface, which visualizes the results of the web service *find-flights*. As illustrated in the figure, the user interface is composed of a number of static output elements and dynamic output elements. Like OntoWeaver, OntoWeaver-S abstracts the user interface of a web page as the composition of *resource components.* Each resource component is modelled as a composition of a number of sub elements. The sub elements can be composite elements, which in turn contain a number of elements. Thus, complex user interfaces can be easily composed.

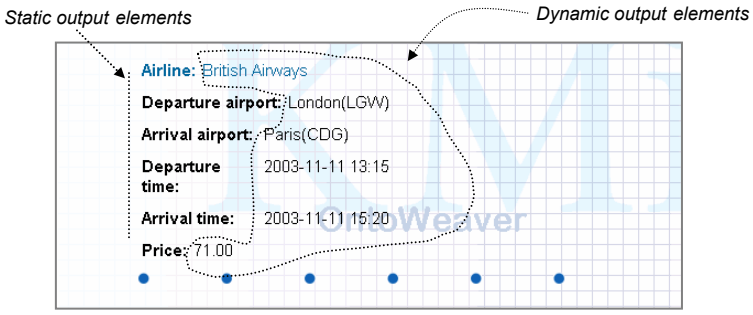

Fig.4. A user interface sample for visualizing the results of the web service *find-flights*

3.3 Modelling Dynamic Features of Knowledge Portals

As mentioned earlier, the typical dynamic features of knowledge portals include i) *information visualization*, which publishes the dynamic data content coming from the underlying data sources or from remote web services; ii) *information provision*, which allows the provision of information for updating the underlying knowledge bases, iii) *web service access*, which allows end users to access web services, and iv) *information querying*, which allows end users to make queries over the back-end knowledge bases. OntoWeaver-S provides a set of user interface constructs to describe the user interfaces, which realize these dynamic features. Moreover, these user interface constructs can be easily adjusted to meet particular requirements of knowledge portals, as OntoWeaver-S provides fine-grained constructs to allow the declarative representation of user interface elements.

3.3.1 Information Visualization

Information visualization in knowledge portals presents dynamic information, which comes from the underlying knowledge bases, web resources or remote web services. OntoWeaver-S relies on the following constructs to enable the composition of the user interfaces for information visualization:

- *DynamicOutput*, which models basic user interface elements that present the dynamic value of the specified field of a given class entity or the dynamic value of the specified output role of a given web service. The construct *DynamicOutput* has a number of attributes: the attributes *hasTask* and *hasOutputRole* are used in the case of presenting dynamic values of web services; the attributes *hasClassEntity* and *hasSlotEntity* are used to specify values from the specified slot of a class entity.
- *OutputComponent*, which describes the composite user interface element for publishing the value of the specified slot of the given class entity or the value of the

specified output role of the given web service. An output component typically comprises an output element, which presents explanations about the dynamic content, and a dynamic element, which displays the dynamic content.

- *DataComponent*, which abstracts user interface elements that visualize instances of a specified class entity or results of a specified web service. The user interface shown in figure 4 presents the result of the web service *find-flights*. As illustrated in the figure, the user interface comprises a number of static output elements explaining the meaning of the dynamic values and a number of dynamic output elements presenting dynamic values. The following RDF code illustrates the specification of the user interface (The prefix '*svo*' refers to the namespace of the site view ontology: xmlns:svo=*"http://kmi.open.ac.uk/people/yuangui/siteviewontology#"*).

```
<rdf:Description rdf:about="flights-result-page/datacomponent" >
  <rdf:type rdf:resource="&svo;DataComponent" />
  <svo:task rdf:resource="find-flights"/>
 <svo:outputComponent>
   …
 </svo:outputCompoenent>
</rdf:Description>
…
<rdf:Description about="flights-result-page/datacomponent/airline/dynamicoutput" >
        <so:outputType>text</so:outputType>
        <so:task rdf:resource="find-flights" />
        <so:outputRole rdf:resource="find-flights/airline" />
</rdf:DescripItion>
```

3.3.2 Information Provision

The information provision is realized through knowledge acquisitions forms, which allow users to submit information to knowledge portals. The submitted information can be instances of the underlying domain ontology classes, web resources or information for invoking the specified service. Please note that in this context, the service can be a built-in service provided by OntoWeaver-S or a remote web service, which has been made available through IRS-II. In particular, a knowledge acquisition component can be used for information provision, information query and web services access. OntoWeaver-S proposes a set of constructs to model the composition of knowledge acquisition forms:

- *Input*, which abstracts the input fields for allowing end users specifying information for particular slots of the specified class entity or for particular input roles of the specified web service.
- *Command*, which describes the interface elements for submitting information. In particular, the specification of a command element indicates the associated service and the result page node, which intends to present results of the associated service.
- *InputComponent*, which describes the composite user interface elements for allowing the information provision for the specified slot of the given domain class entity or for the specified input role of the associated service. An input component typically contains an input element for presenting an input field and an output element for presenting an explanation about the input field.

- *KAComponent*, which models components that present forms for allowing end users to submit information to knowledge portals. The user interface of a knowledge acquisition component is composed by a number of output elements, which present explanations for input elements, a number of input elements, which present input fields, and a command element, which allows users to submit information. The purpose of the submitted information is indicated by the specified service. In the case of semantic web content provision, the associated service is the built-in knowledge acquisition service.

3.3.3 Web Service Access

As discussed above, the functionality of accessing remote web service can be realized through knowledge acquisition forms. In particular, for those web services, which have already been made available at the design time of knowledge portals, user interfaces for accessing them can be specified at design time. For those, which are made available later, the user interfaces can be generated automatically by mapping the semantic descriptions of web services to the provided user interface constructs.

Figure 5 shows an example user interface for accessing the remote web service *find-flights*, which answers requests for flights in accordance with the given user requirements. The user interface is made up of a number of input components, which present input fields and explanations about these input fields, and a command element, which allows end users to submit information and invoke the web service. The following code illustrates the composition of this user interface. In particular, the input elements define the connections between the input fields and the input roles of the specified web service. The command element specifies the web page, which is designed to visualize the results of the web service.

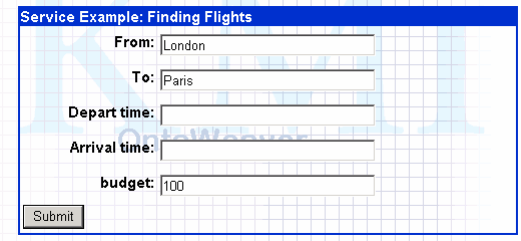

Fig.5. An user interface example for accessing the web service *find-flights*

```
<rdf:Description about="find-flights-page/kacomponent" >
  <rdf:type rdf:resource="&svo;KAComponent"/>
  <svo:task rdf:resource="find-flights"/>
  <svo:inputComponent>
      <rdf:Bag>
        <rdf:li resource="find-flights-page/kacomponent/from-place"/>
        <rdf:li resource=" find-flights-page/kacomponent/to-place"/>
        …
```

```
        </rdf:Bag>
   </svo:inputComponent>
  <svo:command rdf:resource="find-flights-page/kacomponent/command" />
</rdf:Description>
…
<!--  the specification of the command element -->
<rdf:Description about=”find-flights-page/kacomponent/command” >
    <rdf:type rdf:resource="&svo;Command”/>
     <svo:commandText>Submit</svo:commandText>
     <svo:task rdf:resource=“find-flights”/>
     <svo:resultPage rdf:resource=”flights-result-page” />
</rdf:Description>

<!--  the specification of an input element -->
<rdf:Description about=” find-flights-page/kacomponent/from-place/input” >
        <svo:task rdf:resource=“find-flights”/>
        <svo:inputRole rdf:resource="find-flights/param/from-place"/>
 </rdf:Descripltion>
```

Regarding the web service provision, OntoWeaver-S relies on IRS-II to provide user interfaces for allowing the provision of web services.

3.3.4 Information Query

As discussed in section 3.3.2, the functionality of information query can also be achieved by knowledge acquisition forms, where the associated service is the built-in query service. At the moment, OntoWeaver-S only supports simple queries over the instances of the specified class entity. Later we plan to integrate powerful semantic query tools e.g. AquaLog [20] into OntoWeaver-S.

4 Customization Support

In the context of knowledge portals, customization support is one of the major requirements for design approaches, as it delivers personalized portals for individuals, which allow them to access data sources and web services in a way that reflects their personal requirements. In particular, at the user group specified level, different views are required for different user groups. For example, advanced users can add knowledge facts to portals for particular domain classes, while others can only access such information. This kind of customization is supported by the OntoWeaver-S modular architecture, which strictly separates the domain data model, the site view model and the presentation model. In particular, different site views can be constructed over the same domain model and different layouts and appearances can be specified for the same site view model.

The customization discussed above is static, i.e. being specified at the design time. Further support for dynamic (i.e. run time) customization, which personalizes web portals according to the contextual information of end users, should be provided as well. For example, the instances of domain classes can be ordered according to users’ preferences. To support this kind of customization, OntoWeaver-S inherits the OntoWeaver

customization framework, which allows rules to be specified at a high level of abstraction. In particular, the customization framework takes advantage of the declarative specifications of all aspects of knowledge portals, offers a customization rule model and a basic user model for specifying customization rules, and employs a customization engine to produce customized web pages for individual users. As the entire site model is available to customization, the OntoWeaver-S support for customization is not restricted. More information about the customization framework can be found in [10].

5 OntoWeaver-S Architecture

As shown in figure 6, the OntoWeaver-S architecture comprises the following components:

- *A knowledge warehouse* hosts ontologies, knowledge bases and specifications of knowledge portals, which are represented in RDFS [23] and RDF [22].
- *A number of server-side components* provide services for the design-time tools and the run-time tools for accessing and manipulating the sever-side knowledge warehouse, e.g. reading and updating the data repository and performing inferences over the data repository.
- *An OntoWeaver server* provides connections between the OntoWeaver-S tools and the server-side services.
- *A set of design-time tools* support the design of all aspects of knowledge portals, including domain ontologies, site view specifications, site presentation specifications, user ontologies, and customization rules.
- *A set of run-time tools* provide support for knowledge portals at run-time to achieve their functionalities.

5.1 Design-time Support

Designing knowledge portals in OntoWeaver-S can be achieved by means of *the Site Designer*, *the Site Builder,* and *the Site Customizer*. In particular, the Site Designer offers visual facilities to allow the design of generic knowledge portals. As shown in figure 7, the Site Designer offers a site structure design pane (i.e. the left pane) to facilitate the construction of site structures, a page content design pane to allow the definition of user interface elements, a presentation style design pane to support the specification of presentation styles, and a layout design pane to facilitate the layout design.

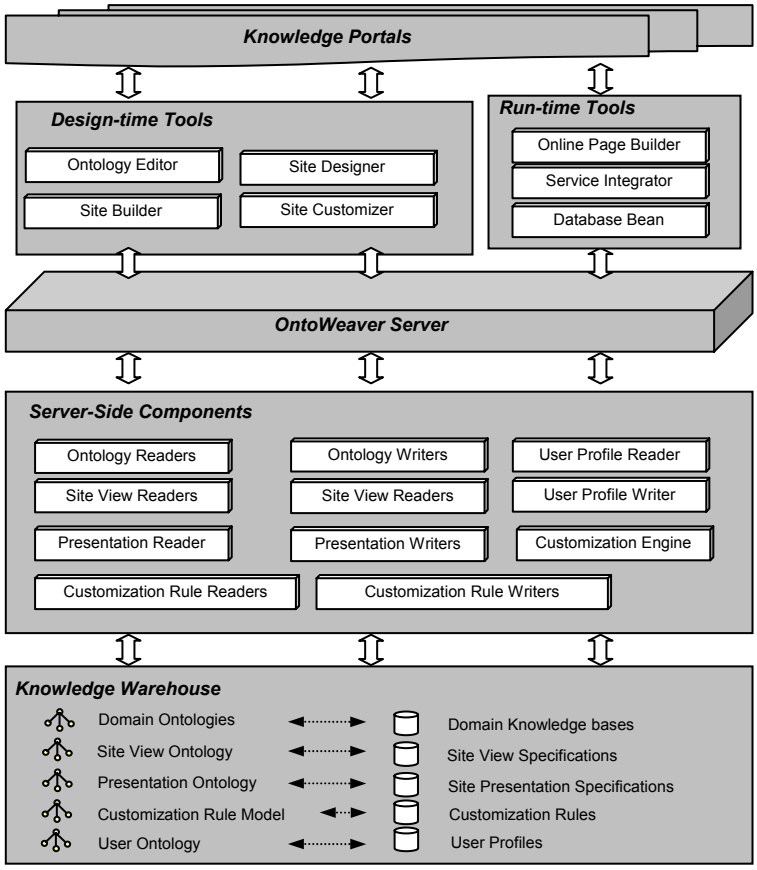

Fig.6. The OntoWeaver-S Architecture

The Site Builder compiles the specifications of knowledge portals into implementations. In particular, it augments the site view specification with the specified presentation model to produce a particular rendering of the target knowledge portal. The Site Customizer offers facilities for user group specified customization design, which allows creating different site views for different user groups, and rule specified customization design, which supports developers to specify customization rules. In addition, OntoWeaver-S provides an Ontology Editor to support the design of ontologies.

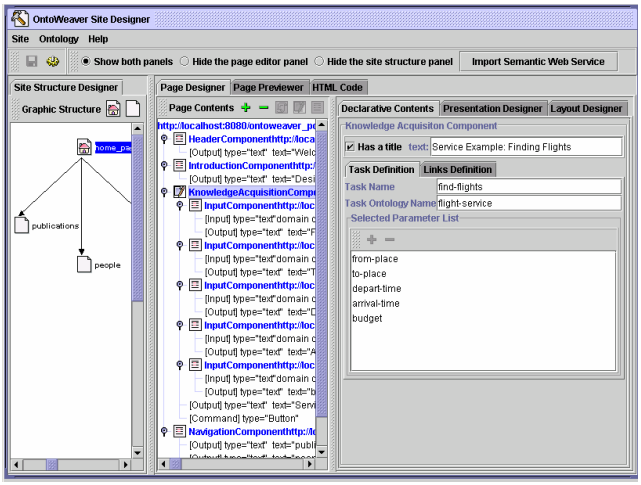

Fig.7. A screenshot of the Site Designer

5.2 Run-time Support

The run-time support that OntoWeaver-S provides includes i) retrieving and managing data stored in the back-end knowledge bases, ii) generating customized web pages on the fly according to the result of the customization engine, and iii) calling the IRS-II server to achieve the specified task with the input information and delivering the results back to knowledge portals.

6 Related Work

6.1 Related Work on Building Knowledge Portals

OntoWebber [8] is an ontology-based tool for building and managing knowledge portals. It proposes a site ontology to model knowledge portals, and thus offers support for the creation and the management of knowledge portals. However, OntoWebber does not address the support for accessing remote web services. Moreover, the requirement of creating sophisticated user interfaces for knowledge portals has not been addressed

appropriately. Specifically, the OntoWebber site ontology models site views at a coarse-grained level. For example, OntoWebber views that *a card* is a basic element to compose web pages. However, a card should consist of a number of components, e.g. texts, hyperlinks and images. Therefore, more fine-grained constructs should be proposed to allow cards to be composed and manipulated according to complex requirements.

SEAL [14] is another ontology-based approach, which addresses the design and development of knowledge portals. It employs an ontology for semantic integration of existing data sources as well as for knowledge portal presentation and management. The main functionalities the SEAL approach provides for knowledge portals include information access, information provision and information query. ODESeW [4] focuses on the automatic generation of knowledge portals from domain ontologies. It provides a number of functionalities for knowledge portals, including domain modelling, content provision, content visualization, content querying and web site administration. MetaLib [18] is a library portal solution, which relies on its Knowledge-base repository to enable the functionalities of library portals, including information querying and information visualization.

The SEAL approach, the ODESeW approach and the MetaLib approach are very different from OntoWeaver-S, as they do not provide meta-models to model knowledge portals. As a consequence, they do not offer high level support for specifying and maintaining knowledge portals. Moreover, the requirements on accessing remote web services and customization support have not been addressed in these approaches.

6.2 Related Work on Conceptual Web Modelling

Recently, a number of tools and approaches have been developed to address the design and development of data-intensive web sites. Examples include RMM [7], OOHDM [17], ARANEUS [1], WebML [3], and HERA [6]. These approaches have addressed the ability to model the underlying domain data structures and handle the dynamic data content. However, they only provide limited support for the modelling of site views (i.e. navigational structures and user interfaces), as they either do not address the composition of user interfaces, e.g. RMM [7], or only provide coarse-grained constructs to address the composition of user interfaces, e.g. ARANEUS [1], WebML [3] and HERA [6]. Moreover, they typically rely on external approaches (e.g. style sheets) to specify presentation styles and layouts for web pages.

7 Conclusions and Future Work

This paper has presented OntoWeaver-S, an ontology-based infrastructure for building knowledge portals. OntoWeaver-S distinguishes itself from other approaches in several ways. First, it offers a set of comprehensive functionalities for the target knowledge

portals, including *information visualization*, which presents dynamic data content coming from the underlying knowledge bases or remote web services, *information provision*, which allows end users to submit knowledge facts to knowledge portals, *web service access*, which allows end users to access the remote web services, and *information querying*, which allows end users to make queries over the underlying knowledge bases. Second, it offers a set of ontologies to enable the high level support for the specification of all aspects of knowledge portals. Third, it addresses customization issues of knowledge portals by means of a customization framework. Finally, it offers a set of powerful tools for supporting knowledge portals at design time as well as at run time.

An OntoWeaver-S prototype system, including all the tools mentioned in this paper, has been implemented. In the future, we will focus on i) defining constraints validating the complex site specifications and provide tools helping developers to find and correct the specifications that are either with errors or being inconsistent in the entire site model, ii) enhancing the information querying functionality by integrating with an ontology-based questing-answering engine (e.g. AquaLog [20]), and iii) bringing the semantic layer of knowledge portals to end users for enabling semantic browsing, e.g. allowing users to browsing the semantic neighbourhood of particular entities.

References

1. P. Atzeni, G. Mecca, P. Merialdo, *Design and Maintenance of Data-Intensive Web Sites,* proceeding of the 6th int. Conference On Extending Database Technology (EDBT), Valencia, Spain, March 1998.
2. T. Berners-Lee, J. Hendler, and O. Lassila, *The Semantic Web*, Scientific American, 2001.
3. S. Ceri, P. Fratenali, A. Bongio. *Web Modelling Language (WebML): a modelling language for designing Web sites*. WWW9 Conference, Amsterdam, May 2000.
4. O. Corcho, A. Gomez-Perez, A. Lopez-Cima, V. Lopez-Garcia, and M. C. Suarez-Figueroa, *ODESeW. Automatic Generation of Knowledge Portals for Intranets and Extranets*, In Proceedings of the 2nd International Semantic Web Conference 2003 (ISWC 2003), 20-23 October 2003, Florida, USA.
5. J. B. Domingue, M. Dzbor, *Magpie: Browsing and Navigating on the Semantic Web*, In Proc. of the Conference on Intelligent User Interfaces, January 2004, Portugal.
6. F. Frasincar, G. J. Houben, *Hypermedia Presentation Adaptation on the Semantic Web.* Second International Conference, AH2002, Malaga, Spain, pp. 133-142.
7. T. Isakowitz, E.A. Stohr and P. Balasubramaninan, *RMM: A Methodology for Structured Hypermedia Design*, Communications of the ACM, August 1995.
8. Y. Jin, S. Decker, G. Wiederhold, *OntoWebber: Model-Driven Ontology-Based Web site Management*, Semantic Web Workshop, Stanford, California, July 2001.
9. Y. Lei, E. Motta, and J. Domingue, *An Ontology-Driven Approach to Web Site Generation and Maintenance,* In proceedings of 13th International Conference on Knowledge Engineering and Management, Sigüenza, Spain 1-4 October 2002, pp. 219-234.

10. Y. Lei, E. Motta and J. Domingue, *Design of Customized Web Applications with OntoWeaver*, in proceedings of the International Conference on Knowledge Capture, October, Florida, USA, 2003, pp 54-61.
11. Y. Lei, E. Motta and J. Domingue, Modelling Data-Intensive Web Sites With OntoWeaver, in proceedings of the international workshop on Web-based Information System Modelling (WISM 2004), Riga, Latvia, 2004.
12. Y. Lei, E. Motta and J. Domingue, OntoWeaver-S: Integrating Web Services into Data-Intensive Web Sites, in Proceedings of the WWW2004 workshop on Application Design, Development and Implementation Issues in the Semantic Web, New York, 2004.
13. F. Lima and D. Schwabe, *Application Modelling for the Semantic Web*, in Proceedings of the First Latin American Web Congress (LA-WEB 2003).
14. A. Maedche, S., R. Studer, Y. Sure, and R. Volz, *Seal --- Tying up information integration and Web site management by ontologies*. IEEE Data Engineering Bulletin, March 2002.
15. E. Motta, J. Domingue, L. Cabral, and M. Gaspari, *IRS-II: A Framework and Infrastructure for Semantic Web Services*, in Proceedings of the 2nd International Semantic Web Conference 2003 (ISWC 2003), 20-23 October 2003, Florida, USA.
16. D. Quan, D. Huynh and D. R. Karger, *Haystack: A Platform for Authoring End User Semantic Web Applications*, in Proceedings of the 2nd International Semantic Web Conference 2003 (ISWC 2003), 20-23 October 2003, Florida, USA.
17. D. Schwabe and G. Rossi, *an Object Oriented Approach to Web-Based Application Design*, Theory and Practice of Object Systems 4(4), 1998, Wiley and Sons, New York, ISSN 1074-3224).
18. T. Sadeh and F. Walker, *Library Portals: toward the Semantic Web*, New Library World, volume 104 (1184/1185) pp 11-19.
19. P. Spyns et al., *OntoWeb - a Semantic Web Community Portal*. In Proc. Fourth International Conference on Practical Aspects of Knowledge Management (PAKM), December 2002, Vienna, Austria, 2002.
20. V. Lopez and E. Motta, *Ontology-driven Question Answering in AquaLog,* In Proceedings of 9th international conference on applications of natural language to information systems, Manchester, 2004.
21. R. Volz, D. Oberle, S. Staab, B. Motik, *KAON SERVER - A Semantic Web Management System*, in Alternate Track Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003, ACM, 2003.
22. W3C, *Resource Description Framework (RDF) Model and Syntax*, available online at http://www.w3.org/TR/PR-rdf-syntax/.
23. W3C, *Resource Description Framework (RDF) Schema Specification 1.0*, available online at http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
24. W3C, *Extensible Markup Language (XML) 1.0, Second Edition*, available on line at http://www.w3.org/TR/2000/REC-xml-20001006.