An Ontology-driven Approach to Web Site Generation and Maintenance

Yuangui Lei, Enrico Motta and John Domingue

Knowledge Media Institute
The Open University
Walton Hall, MK7 6AA
{y.lei, e.motta, j.b.domingue}@open.ac.uk

Abstract. Building and maintaining a data-intensive web site is costly and time-consuming and a number of approaches have addressed this problem using a model-based methodology. This paper presents IIPS (Intelligent Information Presentation System), a system that uses an ontology-driven approach to site generation and management. IIPS provides a suite of visual tools, which make it possible to model a data-intensive web site at a conceptual level, using site, interface and domain ontologies. As a result, the site designer can focus on the conceptual structure of the target web site and associated resources, independently of its realization. IIPS also provides explicit mapping mechanisms, which make it possible to generate quickly site implementations from the conceptual model. IIPS improves over existing model-based approaches to web design, by providing knowledge-level support for all aspects of web design, including site and resource specification, presentation and domain data.

1 Introduction

As the web is becoming the major computing platform for sharing data, the need to develop sophisticated data-driven applications to exploit the Internet is increasing in domains such as knowledge portals, electronic commerce, digital libraries, and distance learning. Nevertheless, web application development and maintenance remain costly and time-consuming. To address this problem, many researchers have proposed the use of model-based methodologies to try and simplify the whole process of generation and maintenance of data-intensive web applications [1,2,3,4].

These approaches typically separate the specification of the web site from the domain data. However, most of them, e.g. [1,2], only provide methodological guidance to define site models, rather than explicit conceptual modelling support. Moreover they do not support automatic site generation. As a result, developers still need to do a lot of work to realise site specifications and to integrate these with domain data. Some approaches do provide knowledge-level support for site modelling, e.g. WebML [3] and OntoWebber [4], however they fail to model user interface issues, such as page layouts and graphic user interfaces.

In this paper we describe an Intelligent Information Presentation System (IIPS), which improves over existing approaches to web site generation and maintenance. IIPS uses an ontology-driven approach to site generation and management: it provides a suite of visual tools, which make it possible to model a data-intensive web site at a conceptual level, using site, interface and domain ontologies. The site ontology models the navigational structure and the compositional structure of a generic data-intensive web site, the interface ontology models web-based user interfaces and the domain ontology specifies the data relevant to the site. Thus, the site designer can focus on the conceptual structure of the target web site and associated resources, independently of its realisation. IIPS also provides explicit mapping mechanisms, which make it possible to generate quickly site implementations from the conceptual model. An important advantage of this approach is that by exploiting the conceptual, explicitly represented specifications of the web site and the interfaces, IIPS is able to reason about these, e.g., in order to customise presentations in an intelligent way for different types of users and devices.

The paper is organized as follows: section 2 presents an overview of the IIPS system; section 3 describes the IIPS ontologies; section 4 introduces automatic site generation through ontology mapping; section 5 illustrates some initial ideas about how IIPS can exploit the ontological specifications to generate smart, customised interfaces; section 6 discusses the IIPS solution to site maintenance; section 7 describes the initial prototype implementation of IIPS. Finally, sections 8 and 9 compare and contrast IIPS with other relevant approaches and discuss future work.

2 Overview of the IIPS System

Fig. 1 shows the framework of IIPS. As shown in the figure, IIPS accepts a domain ontology as input, and produces a data-intensive web site. The IIPS approach is based on the following methods:

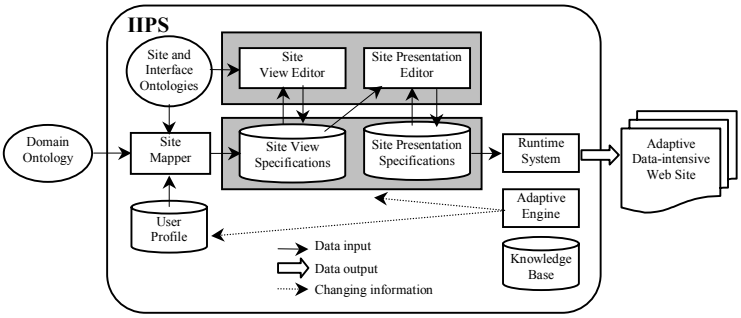

Fig. 1. IIPS Framework

- *Ontology-driven site generation and maintenance*. IIPS defines a set of ontologies to model data-intensive web sites, and uses a domain ontology to drive the target web site generation and maintenance. To create data-intensive web sites with IIPS, developers only need to provide a domain ontology to describe the domain data structure, and the system will generate default web site specifications automatically through mapping. The main advantage of this approach is that developers, especially domain experts, can focus on developing the domain ontology, checking the consistency of the domain ontology, and developing the conceptual structure of the target web site and associated resources.
- *Use of declarative site specifications* to facilitate tool construction and maintenance. The declarative nature of a site model offers many potential benefits over traditional hard-coded site specifications. First, it facilitates the construction of tools to assist developers at design-time, and end-users at run-time, for the declarative model provides a common representation which can be reasoned about [5]. Second, it supports rapid prototyping and iterative development. Developers can construct prototype systems rapidly based on the default system generated by mapping. Finally, it allows a measure of tool independence, so that a site could be reengineered using a different tool set.
- *Use of RDF as the underlying knowledge representing language* to represent ontologies, site specifications, and target web sites. Resource Description Framework (RDF) [6] is a foundation for processing metadata, which provides interoperability between applications that exchange machine-understandable information on the Web. RDF schema [7] provides a mechanism to define particular vocabularies for RDF documents. However, it is not powerful enough to describe the constraints on and relationships among ontologies. At the moment, we use RDF schema to represent the basics of the site ontology, the interface ontology and domain ontologies, and exploit OCML [8] to describe constraints and relationships. We use RDF statements to describe site specifications and annotate target web sites. Later we will consider using DAML+OIL [9] or OWL [10] as the underlying language to represent ontologies.
- *Separation of presentations from site contents.* IIPS uses a *site view specification* to describe the navigational structure, the compositional structure, and contents of a web site, and uses *a presentation specification* to describe presentation instructions, including layouts and visual appearances. This approach separates presentation from site view specification completely. As a result, one site view can be rendered according to different presentation instructions, thus creating totally different presentations.
- *Provision of a set of graphic tools* to support site generation and management. The tools suite of IIPS consists of *a site mapper* to generate default site specifications automatically; *a site editor* to allow developers to edit the site views and presentations manually, and to allow end users to customize site views; *a runtime system* to render site specifications to a web site; *an adaptive engine* to provide adaptive interfaces to end users; and *an ontology editor* to allow users to edit and extend ontologies.

3 Modelling of Web Site

3.1 Site Ontology

The site ontology is defined to model the navigational structure and the compositional structure of a data-intensive web site on the basis of pre-existing site modelling approaches. It conceptualizes a generic data-intensive web site at an implementation independent level, and makes use of the user interface ontology to model the user interfaces of a data-intensive web site. Fig. 2 provides an overview of the site ontology:

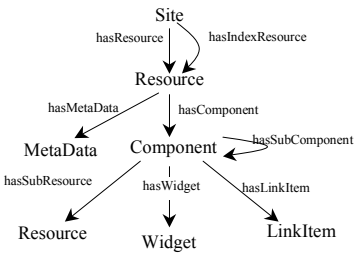

Fig. 2. Overview of the Site Ontology

- The class *Site* models a web site as a logical collection of resources. It has slots *hasResource* and *hasIndexResource*. The slot *hasResource* specifies resources a web site contains. The slot *hasIndexResource* describes the entry point of a web site that helps users to navigate through.
- The class *Resource* models web resources such as web pages and Java applets. It contains a slot *component* that describes contents of a resource, and a slot *hasMetadata*. To model typical resources which appear in data-intensive web sites, IIPS defines a series of resource primitives, which are shown in table 1.
- The class *Component* models contents of resources. It consists of slots: *hasSubResource* that specifies resources that may appear in a component, *hasSubComponent* that describes sub-components, and *hasWidget* and *hasLinkItem* that describe widgets and hyperlink items. IIPS defines five component primitives to model typical types of contents that can be used to compose resources. Details are shown in table 1.
- The class *Widget* models basic interface elements that can present any kind of information at a conceptual level. It is an abstract class to describe widgets. IIPS defines three primitives to model abstract widgets as shown in table 1.

- The class *LinkItem* models contents that have an associated hyperlink. It has three slots: *hasAssociatedResourceURI* to specify associated resource, *hasParameter* to filter the resource content, and *output* to present prompt information.
- The class *DataItem* describes domain data in the site ontology and the interface ontology. It has two subclasses: *ClassItem* and *SlotItem*. The class *ClassItem* describes domain class entities. The class *SlotItem* models domain properties or slots.
- The class *MetaData* models metadata for resources.

Table 1. Primitives of resources, components, and widgets

Class Name	Description	Slot List
Resource	Modelling web resources.	• hasComponent • hasMetaData
IndexResource	Serving as an entry point of a web site	• hasNavigationComponent
IndexedResource	Presenting indexed information about a set of instances of domain entities	• hasIndexComponent
DatalistResource	Presenting detailed information about a set of instances of domain entities	• hasDataComponent • hasLinkItem • hasParameter
Knowledgeacquisition Resource	Allowing end users to input facts about domain entities	• hasKaComponent
SearchResource	Allowing end users to make queries	• hasSearchComponent • hasDataComponent
Component	Modelling contents of composing resources	• hasSubResource • hasSubComponent • hasWidget
InputComponent		
KaComponent	Modelling contents to allow users to input facts about domain entities	• hasClassItem • hasKaCommand
SearchComponent	Modelling contents to allow users to make queries	• hasClassItem • hasSearckKey • hasSearchCommand
OutputComponent		
NavigationComponent	Modelling contents of presenting navigation information	• hasLinkitem
IndexComponent	Modelling contents of presenting indexed instances of domain entities	• hasLinkItem • hasIndexKey
DataComponent	Modelling contents of displaying detailed information about a set of domain instances	• hasClassItem
Widget	Modelling basic interface elements	
Input	Modelling widgets allowing users to input facts	• hasSlotItem • hasDefaultValue • hasInputType • hasStyle
Output	Describing widgets presenting information	• hasOutputType • hasValue
Command	Modelling widgets allowing user to invoke a task	• hasTask

3.2 Interface Ontology

Since more and more web sites employ complex graphic user interfaces to facilitate interactions with end users, it is no longer adequate to focus only on data content and navigation structure as many approaches do. User interfaces should be modeled to conceptualize interface design knowledge and provide an explicit interface knowledge base for interface generation.

The interface ontology in IIPS defines four classes to model web-based user interfaces: *Presentation, Template, Layout,* and *Container,* and a series of *mapping rules* to provide presentation guidelines for user interface generation. The *presentation* class defines presentations for interface elements. It has a slot *dataResourceURI* to specify a resource object that a presentation will work on, a slot *layout* and a slot *template*. Class *Layout* models ways to construct a presentation. Class *Template* is defined to facilitate reusing presentations. It has a sub class *WidgetTemplate* to model templates to render conceptual widgets. The *Container* class models interface elements which hold other interface elements, such as windows, forms, dialogs, and panels. The *mapping rules* define a set of rules for mapping data types to widgets. For example, Boolean data can be mapped to a check box, a radio-button, or a text output.

3.3 An example

To illustrate the usage of IIPS site ontologies to model a data-intensive web site, we use the web site of the Knowledge Media Institute at the Open University (http://kmi.open.ac.uk) as a data-intensive web site example. As a site instance, the KMi web site contains an index page and a list of resources to present information. Fig. 3 shows a fragment of RDF statements describing the KMi web site (The namespace prefix '*so*' refers to the namespace of IIPS site ontologies: xmlns:so=”http://kmi.open.ac.uk/ylei/iips/siteontology/”).

```
<rdf:Description rdf:about="http://kmi.open.ac.uk">
  <rdf:type rdf:resource="http://kmi.open.ac.uk/ylei/iips/siteontology#Site" />
  <so:IndexResource rdf:resource="http://kmi.open.ac.uk/home-f.cfm"/>
  <so:Resource>
    <rdf:Bag>
      <rdf:li rdf:resource="http://kmi.open.ac.uk/people/members.html"/>
      <rdf:li rdf:resource="http://kmi.open.ac.uk/people/affiliate.html"/>
   </rdf:Bag>
   …
  </so:Resource>
</rdf:Description>
```

Fig. 3. Specifications describing the KMi web site

To illustrate how to describe web pages presenting information about domain entities, we use *kmi-member* as an entity example. Fig. 4(a) shows the screenshot of the kmi-member web page. This page is made up components to display instantiations of the class *kmi-member*. As shown in fig. 4(b), the site view specification of this

page contains one data component which displays instantiations of the class *kmi-member*. The data component is made up of outputs which display prompt messages

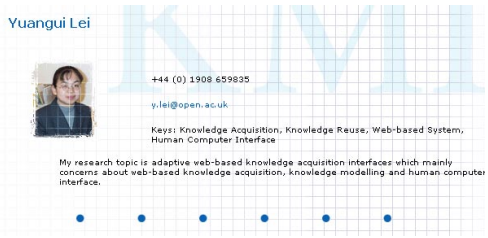

Fig. 4(a). Screenshot of the KMi-member page

```
<rdf:Description rdf:about="http://kmi.open.ac.uk/members.htm">
    <rdf:type rdf:resource="http://kmi.open.ac.uk/ylei/iips/siteontology#Resource"/>
    <so:Component rdf:resource="http://kmi.open.ac.uk/viewspec/components/kmi-member-dataComponent"/>
</rdf:Description>

<rdf:Description rdf:about="http://kmi.open.ac.uk/viewspec/components/kmi-member-dataComponent">
   <rdf:type rdf:resource="http://kmi.open.ac.uk/ylei/iips/siteontology#DataComponent"/>
   <so:ClassItem rdf:resource="http://kmi.open.ac.uk/kmi-ontology/kmi-member" />
   <so:Output>
     <rdf:Description rdf:about="http://kmi.open.ac.uk/images/bullet" >
       <so:OutputType>image</so:OutputType>
       <so:Value>http://kmi.open.ac.uk/img/text/b-bullet.gif</so:Value>
     </rdf:Description>
   </so:Output>
  <so:DynamicOutput>
     <rdf:Description rdf:about="http://kmi.open.ac.uk/viewspec/slotentities/kmi-member-name">
       <rdf:type rdf:resource="http://kmi.open.ac.uk/ylei/iips/siteOntology#DynamicOutput"/>
       <so:OutputType>text</so:OutputType>
      <so:SlotItem rdf:resource="http://kmi.open.ac.uk/kmi-ontology/ kmi-member-name" />
     </rdf:Description>
 </so:DynamicOutput>
    …
</rdf:Description>
```

Fig. 4(b). The site view specification of the *kmi-member page*

```
<rdf:Description rdf:about="http://kmi.open.ac.uk/presentationspec/presentation/kmi-member-dataComponent-presentation">
<rdf:type rdf:resource="http://kmi.open.ac.uk/ylei/iips/siteontology#Presentation"/>
<so:Container rdf:resource="http://kmi.open.ac.uk/presentationspec/container/dataContainer" />
<so:DataResourceURI>http://kmi.open.ac.uk/viewspec/components/kmi-member-dataComponent </so:DataResourceURI>
<so:Layout>
 <rdf:Description rdf:about="http://kmi.open.ac.uk/presentationspec/presentation/layout/kmi-member-data-component">
  <rdf:type rdf:resource="http://kmi.open.ac.uk/ylei/iips/siteontology#Layout"/>
   <so:Presentation>
    <rdf:Description rdf:about="http://kmi.open.ac.uk/presentationspec/presentation/kmi-member-name">
      <so:DataResourceURI>http://kmi.open.ac.uk/viewspec/slotentities/kmi-member-name</so:DataResourceURI>
      <so:Template rdf:resource="http://kmi.open.ac.uk/presentationspec/template/larger-blue-text" />
    </rdf:Description>
   </so:Presentation>
   …
 </rdf:Description>
 </so:Layout>
</rdf:Description>
```

Fig. 4(c). The presentation specification of the *data component of kmi-member.* The dataResourceURI property specifies the data component as the resource object that the presentation will work on

for each slot, and dynamic outputs which display the value of slots of each instantiation of the class *kmi-member*. The presentation of this page is constructed from that of the data component. Fig. 4 (c) shows fragments of code appearing in the site presentation specification of the *kmi-member data component*.

4 Automatic Site Generation through Ontology Mapping

The idea of using a domain ontology to drive software generation is not new. Researchers in the knowledge acquisition area have developed several knowledge acquisition meta-tools, such as DASH [11], Protégé 2000 [12], and Knote [13], which use a domain ontology to drive the knowledge acquisition tool generation. The major advantage of this methodology is that developers, especially domain experts, can specify software tools readily, and that a pre-existing ontology can be used as the basis for the specifications.

IIPS uses the same methodology in driving data-intensive web site generation. Unlike approaches mentioned above where tool specification and implementation are tightly coupled together, IIPS defines a set of comprehensive ontologies to model the target software – data-intensive web sites explicitly, and conceptualizes a target web site at a high level without being concerned about the implementation. As a result, the target web site can be rendered in different ways.

The automatic site generation mainly involves site mapping, which is responsible for generating site specifications through mapping a domain ontology to the site ontology. IIPS uses relationships among the classes defined in a domain ontology to drive the design of the structure and content of the target web site. It makes use of the following site mapping rules to generate site specifications:

- IIPS can identify the top nodes of the domain ontology, and map them to an index resource, which serves as the entry point of the target web site.
- Each class, which needs to be instantiated during the run time of the target web site, is mapped to a series of resources. These resources include a data list resource which presents instances, an indexed resource which displays index information about a set of instances, a search resource which contains search-components to allow users to make a query, and a knowledge-acquisition resource which enables users to input facts about the class.
- The process of instantiating a knowledge-acquisition component involves scanning properties (slots) of the given class, and mapping properties to widgets according to mapping rules defined in the interface ontology. Each slot is mapped to an input widget to allow users to enter data, and an output widget to present a prompt message. In addition, command widgets, which are associated with knowledge acquisition tasks, are needed in the knowledge-acquisition component.
- The search component instantiation process is similar to the knowledge acquisition component. The difference lies in its command widget, which is associated with a search task.

The default site specification generated by ontology mapping contains conceptual structures and contents. The run-time system of IIPS can create default interfaces and presentations for them.

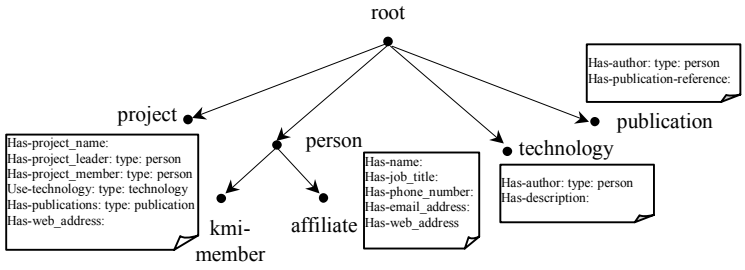

Fig. 5. Sample Ontology

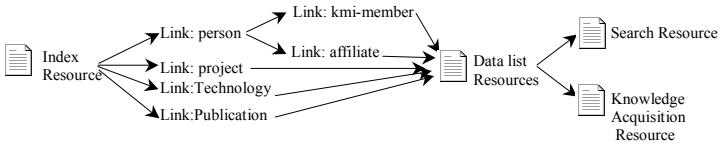

Fig. 6. The site structure generated for the sample ontology by site mapping

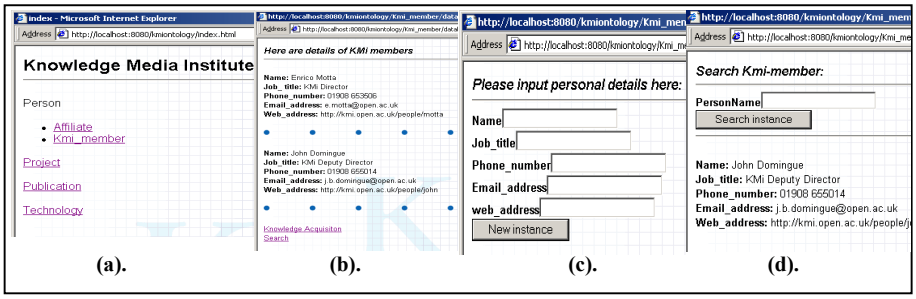

Fig. 7. Screenshots of web pages generated by the IIPS prototype system from the sample ontology. Figure (a) shows the index page. Figure (b), (c) and (d) show the data list page, the knowledge acquisition page, and search page of class *kmi-member*

Fig. 5 shows a sample ontology to illustrate the automatic site generation. There are six domain class entities. The class *person* has sub-classes *kmi-member* and *affiliate*. Fig. 6 shows the default site structure generated by the IIPS site mapper through ontology mapping.

In this example, the index resource contains five hyperlink items. The general class *person* is mapped to a foldable hyperlink item that contains further hyperlinks. Each

of the other classes is mapped to a series of resources, including a data list resource, a search resource, and a knowledge acquisition resource. Users can access these resources through accessing data list resources of each class entity. Fig. 7 shows screenshots of web pages generated by the IIPS prototype system from the sample ontology.

5 Intelligent Support for User Interface Generation

The declarative nature of the site specifications gives IIPS a capability to reason about user interfaces. The intelligent support for user interface generation happens in following cases:

- Creating different site views for different user groups. General users can only browse and customize resources restricted to their user groups. The knowledge acquisition resources are hidden from the site view. Advanced users can browse and customize pages, and input facts to the knowledge base. The developers and webmasters have the highest access to the target web site. They can browse and edit every site view, and create new user groups.
- Customizing structures, contents, and presentations of a web site according to users' needs. Due to the fact that the domain knowledge base and the site specifications are declarative, it is easy for the run-time system to exploit intelligent inference to customize site views. For example, if an end user wants more information about one kmi-member in the KMi web site, the run-time system can create a new web page through reasoning about the site view specification and the domain knowledge base. The new web page contains hyperlinks to all of the web resources about this person, including his or her home page, kmi-member page, project pages and publication pages he or she is involved in.
- Adapting user interfaces according to user profiles, which record end users' stereotypes and preferences. IIPS provides an adaptive engine to achieve this goal.

6 Site Maintenance as Ontology Manipulation

Site maintenance and management is a big issue in the life cycle of a web site. IIPS addresses this issue in three ways. First, the IIPS approach emphasizes automatic site generation from a domain ontology because it can relieve developers of developing a web site from scratch, and help them to focus on the work of developing domain ontologies. Second, the IIPS approach provides visual facilities to support developers to edit the site content and presentation manually. Finally, the IIPS approach supports automatic web site re-engineering after the domain ontology has been changed without loss of the customization information made by developers during the site editing process. Because the site specifications are declarative, the automatic site re-engineering only changes information about domain entities.

In IIPS, site maintenance can be achieved not only at the content level but also at the site specification level. At the content level, IIPS provides knowledge acquisition

forms to allow end users to make contributions to knowledge bases. At the site specification level, IIPS provides a site view editor to edit structure and content of a web site, and a site presentation editor to edit interfaces and presentations. The purpose of these editors is two-fold. First, they provide facilities for developers to edit the target web sites. Developers can utilize them to extend the site ontology through inheriting concepts in the site ontology, e.g. developers can define new types of data component to specify how the data component looks exactly. Second, they also support end users in customizing the site views restricted to their group. End users cannot change the web site, but they can choose information they are interested in and filter the irrelevant information.

7 Prototyping

Fig. 8 shows the major components of IIPS system. It is made up of three major components: *a knowledge warehouse, a suite of support tools*, and *a runtime system* to render the site specifications to target web sites.

The *knowledge warehouse* hosts ontologies, domain knowledge bases, site specifications, and user profiles. It serves as data repository for data represented in RDF schema and RDF statements. The *support tools* provide design-time support as well as run-time support. The *run-time system* is responsible for reading the site specifications, generating target web sites, and invoking the *adaptive engine* to provide on-line adaptive interfaces.

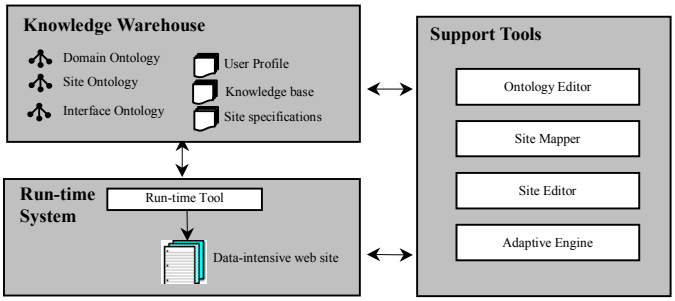

Fig. 8. Major components of the IIPS prototype system

7.1 Site Mapper

The Site Mapper is responsible for creating default site specifications through mapping the domain ontology to the site ontology. It provides facilities for developers to customize the domain ontology, preview default site views, and build default site specifications.

Customizing a domain ontology means specifying and refining domain data structures on the basis of a domain ontology. It involves selecting classes from the domain hierarchy structure, and selecting slots for each selected class. Customizing a domain ontology doesn't mean changing the domain ontology. It will not result in losing slots for classes. For example, if we only choose the class *kmi-member* and the class *affiliate*, and don't choose the class *person,* the class *kmi-member* and *affiliate* will keep all their inherited slots with them, and will not lose any information.

To illustrate the site mapping result clearly, we use the sample ontology shown in Fig. 5 as a domain ontology to drive the site generation. Fig. 9 shows the screenshot of the site preview interface. The left frame shows the site view structure, which is displayed in tree style. Each node represents a resource, the relationship between a child and a parent is hyperlink. The right frame displays detail information about the selected resource node, including its declarative contents and preview.

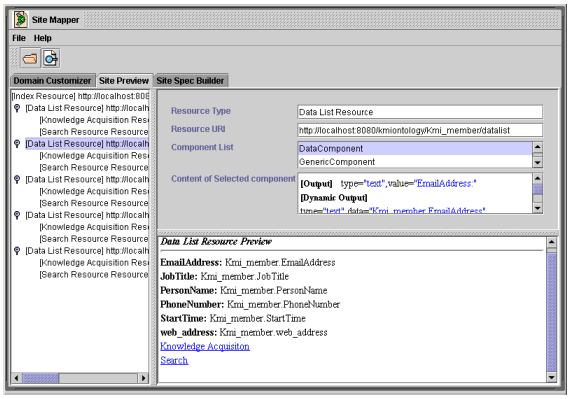

Fig. 9. Screenshot of the site preview interface in the site mapper

7.2 Ontology Editor

The Ontology Editor provides visual support for developers to edit domain ontologies. Developers can utilize it to browse and edit pre-existing ontologies, as well as create new domain ontologies. Fig. 10 shows a screenshot of the ontology editor. It mainly supports RDF schema [7] format, and also provides mechanisms to translate ontology representation between RDF schema and OCML [8]. Ontology files can be saved as both RDF Schema and OCML format. Although the definition of a class is separated from property definitions, the IIPS ontology editor provides a very straightforward way to allow users to edit classes and properties together. It provides a *class tree tab* to facilitate class editing, and *a property tree tab* to help users to concentrate on property definitions. Class editing and property editing are not separate in the ontology editor. Users can select properties for classes during the process of editing classes. At the same time, they also can achieve this by selecting domain classes for a property during the process of editing properties.

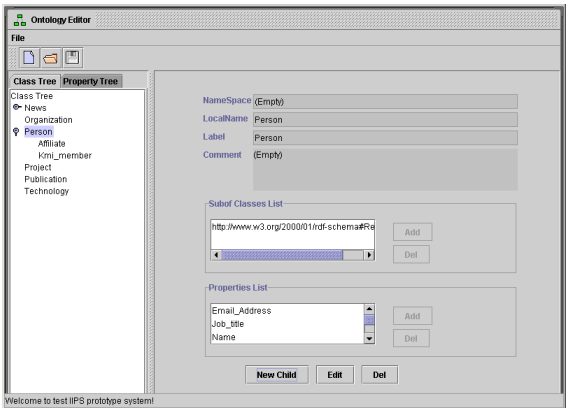

Fig. 10. A screenshot of the ontology editor

Besides the site mapper and the ontology editor, we have worked out an initial run-time system, which is responsible for reading site specifications and creating dynamic web pages. Fig. 7 shows screenshots of web pages generated by the initial run-time system.

8 Related Work

The work on IIPS brings efforts from four areas together: data-intensive web site modelling, user interface modelling, software tool generation from ontologies, and the application of RDF to web engineering.

Related work on data-intensive web site modelling

Recently, research towards modelling of data-intensive web applications has been intensified due to the fact that the processes of web application development from scratch and web application maintenance are inefficient, time-consuming, and costly. Many modelling approaches have been proposed to tackle this problem [1,2,3,4]. Closest to our approach is the work on WebML [3]. It provides explicit site models, and supports automatic site generation. It make uses of a s*tructural model* to express domain data structure, a c*omposition* model to specify contents to composite a hypertext, a n*avigation model, a presentation model* to describe the layout and presentation, and a p*ersonalization model* to specify the features and personalization requirements of users and user groups. However, our approach models a web site much more thoroughly than WebML [3] because it models user interfaces explicitly. WebML [3] does provide a presentation model to express the layout and graphic appearance of pages. However, it only concerns the look and feel of web pages. We argue that the user interface is more than presentation which emphasizes presenting information rather than user-interface interactions. Furthermore, we emphasize the

importance of the semantics of the target web site during the site modelling process, which has not been addressed in WebML [3].

OntoWebber [4] is another site modelling approach similar to IIPS. It is an ontology-based approach to site management, and uses the RDF-based language DAML+OIL [9] as the underlying knowledge representation language. However, it fails to provide explicit mapping mechanisms to map the domain model with the site model to automate site generation.

Related work on interface modelling

A substantial effort has been made in user interface modelling [5,14,15] to try to reduce the amount of code that programmers need to produce when creating a user interface. However, most approaches have failed to become widespread due to the fact that these approaches tightly couple user interface definition with the user interface implementation, and thus lack the flexibility to be rendered in different ways.

UIML [14] and XIML [15] are recent approaches proposed to address the problem of authoring user interfaces for multiple platforms. These two languages are both declarative, appliance-independent, and generic. However they do not separate the application model from the user interface model completely.

XSL [16] addresses this problem very well. It is a language for expressing stylesheets that describe how to present an XML document. The XSL approach is domain independent. However, it focuses on the presentation of the source data. That is to say, it emphasizes presenting information rather than user-interface interactions. IIPS concerns not only the presentation of the source data, but also user interfaces, such as interface mapping rules between a data type and a user control object.

Related work on software generation from ontology

The feasibility of ontology-driven software generation has been demonstrated in the knowledge acquisition area, where various ontology-driven knowledge acquisition metatools have been developed [11,12,13]. IIPS distinguishes itself from these tools in that the target system is completely different. IIPS aims to generate a data-intensive web site. Unlike the approaches mentioned above, IIPS provides an explicit ontology to describe the target system and support ontology mapping, and conceptualizes a target web site at a high level without being concerned with the implementation. Thus, the IIPS approach is much more generic.

Related work on applying RDF to web engineering

XWMF [17] aims to create a machine-understandable web sites through exploiting RDF to model web application and its content. It provides a generic web engineering schemata and RDF as the basic vocabulary to model a web application. However, it does not provide a set of explicit models to describe web sites, therefore it is very

different from the IIPS approach, although they address similar goals of creating machine-understandable web applications.

SEAL [18] and SEAL-II [19] aim to build and manage semantic web portals on the basis of ontologies. However, they mainly focus on semantic browsing, semantic-based ranking, semantic querying, and information contribution from end users, rather than on web site modelling and automatic site generation. They do not provide explicit site models, or the mapping approach to automatic site generation, and are thus quite different from IIPS.

RSS [20], which stands for RDF (or Rich) Site Summary, is a lightweight metadata description and syndication format. It provides a vocabulary to describe a "channel" consisting of URL-retrievable items. Each item consists of a title, link, and brief description. It models a web site in a very simple way.

9 Conclusions

In this paper, we have presented IIPS, an intelligent information presentation system that uses an ontology-driven approach to drive the generation and maintenance processes of data-intensive web sites. IIPS distinguishes itself from pre-existing data-intensive web site modelling approaches in several ways. First, it provides comprehensive ontologies to model data-intensive web sites, with an emphasis on user interface modelling that has been missing in other approaches. Second, it supports automatic site generation, as well as providing a suite of visual tools to support manual management and maintenance. Finally, it provides intelligent support for user interface generation.

An initial prototype system of IIPS has been completed, including a site mapper, an ontology editor, and an initial run-time system. Future work will focus on the site editor to allow developers to edit site views and presentations and allow end users to customize a web site, and the adaptive engine to provide adaptive user interfaces for target web sites.

Acknowledgements

We would like to thank Murray Altheim and Arthur Stutt for their insightful comments on earlier drafts of this paper.

References

1. Franca Garzotto, Paolo Paolini and Daniel Schwabe, HDM--a model-based approach to hypertext application design, ACM Trans. Inf. Syst. 11, 1 (Jan. 1993), Pages 1 – 26.
2. T. Isakowitz, E.A. Stohr and P. Balasubramaninan, RMM: A Methodology for Structured Hypermedia Design, Communications of the ACM, August 1995.
3. Stefano Ceri, Piero Fratenali, Aldo Bongio, Web Modelling Language (WebML): a modelling language for designing Web sites, www9 Conference, Amsterdam, May 2000.

4. Yuhui Jin, Stefan Decker, Gio Wiederhold, OntoWebber: Model-Driven Ontology-Based Web site Management, Semantic Web Workshop, Stanford, California, July 2001.
5. P.Szekely, P.Sukaviriya, P.Castells, J.Muthukumarasamy,and E.Salcher, Declarative interface models for user interface construction tools: the MASTERMIND approach, In Proc. EHCI'95, 1995.
6. Resource Description Framework (RDF) Model and Syntax, W3C Proposed Recommendation, http://www.w3.org/TR/PR-rdf-syntax/.
7. Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation, http://www.w3.org/TR/rdf-schema/.
8. Motta E., Reusable Components of Knowledge Modelling: Case Studies in Parametric Design Problem Solving, IOS Press, Amsterdam, 1999.
9. Ian Horrocks, Frank van Harmelen, Peter Patel-Schneider, Tim Berners-Lee, Dan Brickley, Dan Connolly, Mike Dean, Stefan Decker, Dieter Fensel, Pat Hayes, Jeff Heflin, Jim Hendler, Ora Lassila, Deb McGuinness, Lynn Andrea Stein, DAML+OIL, http://www.daml.org/2001/03/daml+oil-index, 2001.
10. Jeff Heflin, Raphael Volz, and Jonathan Dale, Requirements for a Web Ontology Language, W3C Working Draft, 7 March 2002, http://www.w3.org/TR/2002/WD-webont-req-20020307/.
11. Henrik Eriksson, Angel R. Puerta, and Mark A. Musen, Generation of Knowledge-Acquisition Tools from Domain Ontologies, Int. J. Human-Computer Studies (1994) 41, 425-453.
12. William E.Grosso, Henrick Eriksson. Ray W. Fergerson, Johh H. Gennari, Samson W. Tu, and Mark A. Musen, Knowledge Modelling at the Millennium, In Proc. the 12th International Workshop on Knowledge Acquisition, Modelling and Management (KAW'99) Banff, Canada, October 1999, http://smiweb.stanford.edu/pubs/SMI_Abstracts/SMI-1999-0801.html.
13. Enrico Motta, Simon Buckingham Shum, and John Domingue, Ontology-driven document enrichment: principles, tools and applications, Int. J. Human-Computer Studies (2000) 52, 1071-1109.
14. Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, Jonathan E. Shuster, UIML: An Appliance-Independent XML User Interface Language, WWW8 Conference Paper, Toronto Convention Centre, Toronto, Canada, May 11-14, 1999.
15. Angel Puerta and Jacob Eisenstein, XIML: A Common Representation for Interaction Data, in Proceedings of the 7th international conference on Intelligent user interfaces, pp. 214-215, 2002.
16. Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Tony Graham, Paul Grosso, Eduardo Gutentag, Alex Milowski, Scott Parnell, Jeremy Richman, Steve Zilles, Extensible Stylesheet Language (XSL) Verson 1.0, 2001, http://www.w3.org/TR/xsl/.
17. Reinhold Klapsing, Gustaf Neumann, Wolfram Conen: Semantics in Web Engineering: Applying the Resource Description Framework, IEEE MultiMedia Journal, Vol. 8, No. 2, April-June, 2001.
18. Nenad Stojanovic, Alexander Maedche, Setffen Staab, Rudi Studer, SEAL-A Framework for Developing SEmantic PortALs, K-Cap 2001 - First International Conference on Knowledge Capture, Oct. 21-23, 2001, Victoria, B.C., Canada.
19. Hotho, A., Maedche, A., Staab, S., & Studer, R., SEAL-II — the soft spot between richly structured and unstructured knowledge. Journal of Universal Computer Science, vol. 7, no. 7 (2001), 566-590.
20. Gabe Beged-Dov, Dan Brickley, Rael Dornfest, Ian Davis, Leigh Dodds, Jonhathan Eisenzopt, David Galbraith, R.V. Guha, Ken MacLeod, Eric Miller, Aaron Swartz, and Eric van der Vlist, RDF Site Summary (RSS) 1.0, http://groups.yahoo.com/group/rss-dev/files/specification.html, 2000.