Design of customized Web applications with OntoWeaver

Design of Customized Web Applications with OntoWeaver

Yuangui Lei, Enrico Motta, John Domingue
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
{y.lei, e.motta, j.b.domingue}@open.ac.uk

ABSTRACT
OntoWeaver is our conceptual modelling methodology and a tool that support the specification and implementation of customized web applications. It relies on a number of different types of ontologies to declaratively describe all aspects of a web application. This paper focuses on the OntoWeaver customization framework, which exploits a user model, a customization rule model, and a declarative site model, to enable the design and development of customized web applications at a conceptual level. OntoWeaver makes use of the Jess inference engine to reason upon the site specifications and their underlying site ontologies according to the customization rules and the valuable user profiles to provide customization support in an intelligent way. The ontology-based approach enables the target web applications to be represented in an exchangeable format. Hence, the management and maintenance of web applications can be carried out at a conceptual level without having to worry about the implementation details. Likewise, the declarative nature of the site specifications and the generic customization framework allow the specification of customization requirements to be carried out at the conceptual level.

Categories and Subject Descriptors
I.2.1 Applications and Expert System

General Terms
Design, Human Factors, Management

Keywords
Web Site Modelling, Customization Modelling, Web Site Design

INTRODUCTION
So far work in the area of customized web application design has focused on the implementation of the specific customization methods, such as user model-driven adaptive approaches [2], automatic personalization methodologies [15], and manual customization approaches [13]. However, current tools for web site design and implementation provide very limited support for web site customization.

K-CAP'03, October 23–25, 2003, Sanibel, Florida, USA.

To address this problem, a few conceptual web modelling approaches consider customization issues as an important modelling dimension. Examples include WebML [3] [4], OOHDM [16] [17], Hera [5], and OntoWebber [8]. They support one-to-one web site delivery by providing a personalization model or a customization model that describe user groups, individual users, customization context, and customization rules. The customization frameworks of these proposals offer mechanisms that allow site designers to pre-define customization rules or different site views to enable delivering customized web applications for user groups or individual users. However, these approaches do not offer explicit customization specification models that facilitate specifying customization requirements for individual users at a high level. For instance, the WebML approach [3] [4] requires developers to write business rules to enable the personalization of the target web applications. However, it does not offer an explicit specification rule model to allow developers to complete the task at a high level.

The WUML approach [9] [10] provides a generic customization model that allows for the specification of customization rules at high level. However, due to the fact that it does not provide means to model the target web application, it requires developers to slice the target web application into the so-called *stable part* and *variable part*, and define adaptation hooks in the web application to allow for the specification of customization actions in customization rules. As a consequence, it limits the possibilities for customization by forcing developers to anticipate what may be customized.

In [12], we presented IIPS, an ontology-driven approach to the design and maintenance of data-intensive web sites. OntoWeaver extends IIPS, by introducing a generic customization framework and tools for building and maintaining customized web applications. The key ideas of our approach to achieve customization are the following:

- The use of different types of ontologies – the site view ontology, the presentation ontology, and the domain ontology, to drive the processes of the design, management and maintenance of data-intensive web applications. The ontology-based approach enables the target web applications to be represented in an exchangeable format. As a result, site management and maintenance can be carried out at the conceptual level. Furthermore, the declarative nature of the site

specifications allows for the intelligent presentation of information according to the requirements of different contexts (e.g. individual users and devices) as the specifications can be reasoned upon.

- The provision of a generic customization framework to enable the customization support in an intelligent way at a high level of abstraction. The OntoWeaver customization framework provides meta-level models that enable the specification for user models and customization rules at a conceptual level, and makes use of the Jess inference engine [7] for the intelligent customization support by reasoning about the site specifications and the customization rules according to the profile of user individuals. Unlike WUML [9][10], OntoWeaver is considerably flexible as everything can be customized. There is no need for a prior decision about what can be subject to customization: the entire model is available to the customization engine.
- To provide with the visual tools that could support the entire life-cycle of a customized data-intensive web application at a high level, including modelling, design, and maintenance.

This paper is organized as follows: first, we describe the overview of the OntoWeaver methodology for web site design; then, we present the OntoWeaver customization framework; in the next section, we use the conference paper review system as a case to show how the OntoWeaver approach can be used to design a customized web application; thereafter we discuss the usages of the OntoWeaver customization approach; and finally in the last two sections we describe the related work, conclusion, and future work.

ONTOWEAVER OVERVIEW

The OntoWeaver methodology is an ontology-based approach to the design and management of customized data-intensive web sites. Figure 1 shows the framework of the OntoWeaver methodology. The key concept behind OntoWeaver is that it employs a set of ontologies to abstract all aspects of the target web applications, including domain data structures, navigational structures, user interfaces, presentations, end users, and customization rules. In particular, OntoWeaver proposes *a site view ontology* that models navigational structures and user interfaces, and provides a flexible mechanism that allows for the composition of user interfaces, by means of a set of constructs: *SiteResource*, *ResourceComponent*, *SubResource, Output*, *DynamicOutput*, *LinkItem*, *DynamicLinkItem*, *Input*, *Command,* and *Service*. Moreover, OntoWeaver defines *a presentation ontology* to separate the specification of visual appearances and layouts from site views. The target web applications are represented by the *site view specifications* and the *site presentation specifications* in terms of the site view ontology and the presentation ontology. These specifications are then compiled into HTML-based dynamic web pages automatically. A more detailed discussion about the site ontologies can be found in [12].[1]

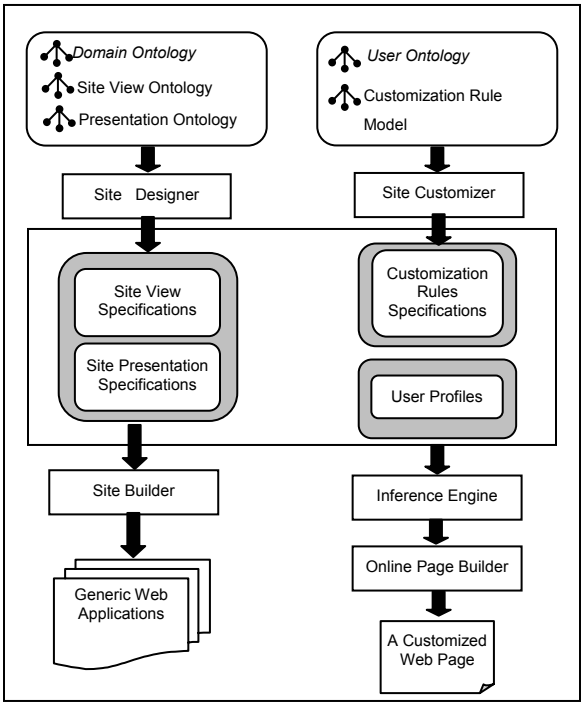

Figure 1 The OntoWeaver framework

OntoWeaver strictly separates the domain data model, site view model, and the presentation model. This architecture *per se* already guarantees design time customization support. Essentially site developers can make use of this modular approach to define: (1) at the site view level, different site views over the same domain model for different user groups or different types of devices; and (2) at the presentation level, different layouts and appearances for the same site view thus giving flexibility for the requirements of different user groups. However, the dynamic (i.e. run-time) customization requirements of individual users can not be supported simply by this modular architecture. The customization should take place dynamically according to the contextual information of each user individual. To this purpose, OntoWeaver proposes a generic customization engine to take an advantage of the declarative site specifications to enable individualization of the target web applications. It achieves this goal by enabling the derivation of customized site views from the same generic site view according to the rule specifications and user profiles at run-time.

With OntoWeaver, the design of a customized data-intensive web application involves two major activities: specifying a general web site and specifying customization requirements. The specification of a general web site

[1] Please note that [12] actually becomes a slightly older version the one used in this paper.

comprises designing the domain ontology, specifying site views, and specifying the visual appearances and layouts over the site views. The customization design involves specifying the user ontology, creating different site views and presentations for different user roles or devices, and specifying customization rules for the individualization.

In order to facilitate the specification and management of web applications, OntoWeaver provides a suite of tools to allow developers to specify and manage customized web applications at a conceptual level. The tool suite is made up of an *Ontology Editor* allowing users to browse and manipulate ontologies; a *Site Designer* supporting the web application design at design time; a *Site Customizer* allowing developers or web administrators to carry out customization design at a conceptual level; a *Site Builder*, which compiles the site specifications into web site implementations; an *Online Page Builder*, which generates customized web pages on the fly according to the inference results; and an *OntoWeaver Server*, which is responsible for providing back-end service supports for the OntoWeaver tools. The online services include reading and updating server-side site specifications, customization rules, and user profiles, and invoking the Jess inference engine to create customized site specifications.

THE ONTOWEAVER CUSTOMIZATION FRAMEWORK

The OntoWeaver customization framework is based on four components: *a user ontology*, which provides means to describe user model in the domain specific context; *a customization rule model*, which provides mechanisms to enable the specification of customization rules; *a set of site ontologyies* (i.e. the site view ontology and the presentation ontology), which offer meta models to enable the specification of declarative site models; and *an inference engine*, which reasons about site specifications with customization rules according to the facts of the user profiles.

User Model

The user model describes information about end users, e.g. preferences, knowledge, potential goals, and so on. In most cases user models are domain dependent. Only site designers or web administrators can define the user model in a way that exactly reflects end users in a specific domain. Hence, OntoWeaver only provides a basic user ontology that facilitates the definition of the domain-specified user model on the basis of it.

Since customized web sites should be responsive to the needs of individual users, the user model is one of the most important components for customization. First, it enables customization conditions to be defined at a high level. Second, its instantiations, user profiles, are used to evaluate conditions of customization rules, and decide whether the corresponding customization rules are to be fired or not. Each customization action takes place only when the context of end users meets a certain condition. For example, in a customized conference paper review system, the device-dependent customization takes place when the user device meets certain conditions.

Customization Rule Model

In order to provide dynamic customization support for the target web applications, we propose the usage of a series of rules, called customization rules, specifying conditions that certain customizations should happen, and actions that actually realize customizations.

The OntoWeaver rule model is made up of a condition part and an action part. The condition part describes a condition that has to be satisfied for the customization to take place. The action part describes the adaptation actions, e.g. adding/hiding/modifying components, or setting presentation or layout properties for components.

Customization Condition Model

A customization condition can be atomic, which is made up of a single condition; or composite, which is composed by a list of conditions by means of the logical operators such as *AND*, *OR*, and *NOT*. Each condition is represented by means of a *class entity URI*, which specifies the relevant class entity abstracted in the user model; a *parameter name* which describes the slot entity of the specified class entity, the value of which will be evaluated at run time; *a relation operator* to express the condition that the value of the specified slot entity should match with the specified value; a *specific value* being used to test the corresponding condition; and *a logical operator*, which connects the current condition with the following one to compose a complex condition. The logical operators offer OntoWeaver the capability of composing complex conditions. Figure 2 shows an example that describes a customization condition, where the user role is "reviewer" or "PC". The customization condition is made up of two sub-conditions, which are connected together by means of the logical operator 'OR'.

```
<rdf:Description rdf:about="customization_rule0/condition" >
   <so:condition>
   <rdf:Description rdf:about="customization_rule0/condition0" >
      <so:classEntityURI>User</so:classEntityURI>
      <so:paramName>role</so:paramName>
      <so:dataType>String</so:dataType>
      <so:paramRelation>=</so:paramRelation>
      <so:paramValue>reviewer</so:paramValue>
      <so:logicalOperator>OR</so:logicalOperator>
   </rdf:Description>
   </so:condition>
   <so:condition>
   <rdf:Description rdf:about="customization_rule0/condition1" >
      <so:classEntityURI>User</so:classEntityURI>
      <so:paramName>role</so:paramName>
      <so:dataType>String</so:dataType>
      <so:paramRelation>=</so:paramRelation>
      <so:paramValue>PC</so:paramValue>
   </rdf:Description>
   </so:condition>
</rdf:Description>
```

Figure 2 An example of a customization condition

Customization Action Model

A customization action is made up of three components: a property *siteEntityURI* which specifies the URI [1] of the

element in the site view model that the customization will work on; a property *objectType* which specifies the customization type (e.g., site view, presentation, and layout); and a *modification* part which expresses the customization details i.e. how to change the value of the specified slot of the intended customization object (e.g. content, presentation, and layout) about the specified element.

OntoWeaver supports three types of customization actions: *site view customization*, which customizes site structures, user interfaces and data content; *presentation customization*, which tailors visual appearances; and *layout customization*, which individualizes the constructive organizations for site view components, e.g., web pages and their components.

Regarding the site view customization, OntoWeaver distinguishes five typical customization actions: *hiding* /removing *components*, *modifying the content of components*, *adding pre-existing components*, and *creating new components*. It is made possible by using the URI mechanism [1] to represent components of the site view model in OntoWeaver. Each component can be retrieved and manipulated by referencing its URI. Hence, OntoWeaver supports the customization actions over the site structures, user interfaces, and presentations. In addition, OntoWeaver allows the customization of data content by means of specifying parameters for user interface elements that deal with dynamic data content.

OntoWeaver relies on a class called *Modification* to abstract the customization action information according to the feature of the component, and its relevant definitions, by means of the *slotName-value* pair. The *slotName* expresses the slot name of the intended customization object, and the *value* describes the customized value for the specified slot.

Declarative Site Model

It is very hard if not impossible to achieve the goal of customization if the specifications of web applications are hidden from the customization framework. The OntoWeaver approach overcomes this problem by considering the customization modelling together with the web applications modelling. As a result, everything can be customized. There is no need for a prior decision about what can be subject to customization: the entire model is available to the customization engine.

The declarative site model serves as the foundation to facilitate the customization. First, the declarative site specifications and their underlying ontologies make it possible to specify customization actions precisely at a conceptual level. OntoWeaver employs RDF models [19] to represent specifications. As a result, each item possesses a URI [1] to identify itself. Moreover, its underlying ontology definition allows for the precise specification of the modifications that the action will perform. For example, if we want to customize a particular *Output* element to a particular user context, we need to be able to specify this output item using its URI, and specify the modification information about its content, e.g. changing the output style, and associating/removing hyper links with/from it, according to the definition of the class entity *Output* in the site view ontology. Second, the declarative site model provides facts to be reasoned upon to generate customized site models.

Inference Engine

Jess [7] is a Java expert system shell, which provides a scripting language to define facts and rules, and offers an inference engine to perform inferences. To enable deployment of the inference engine, we have developed a RDF-to-Jess compiler, which compiles OntoWeaver site ontologies, site specifications, user models, user profiles, and customization rules into Jess templates, facts, and rules. In particular, the condition part of the customization rule model is compiled into the left-hand-side part of the Jess rules, and the customization action part is converted into the right-hand-side. A rule in Jess looks like: LHS => RHS, where LHS is a conjunction of conditions and RHS is a conjunction of actions.

A CASE STUDY

In this section, we use the conference paper review system, which has been used as a customization example in [17] and [10], as a case to demonstrate the capability of the OntoWeaver approach to the design and development of customized web applications.

The paper review system involves three kinds of user roles: *authors*, *reviewers*, and *program committees*. Each user is concerned with different information. Authors pay visit to the web site for browsing the conference information and submitting papers; reviewers get assigned papers from and submit review articles to the conference; Program Committees (PCs) should be able to upload information about calls for papers, workshops and tutorials, browse detail information and review recommendations about each paper. In the following section, we detail the steps involved in the design of customized web applications with OntoWeaver.

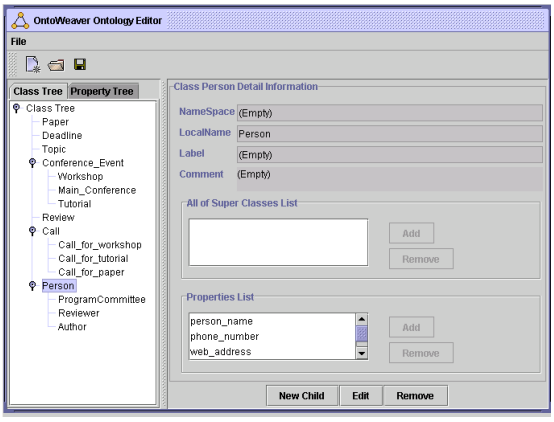

Figure 3 The domain ontology of the conference paper review system in the OntoWeaver Ontology Editor

Specifying the Domain Ontology

Figure 3 shows the domain ontology of the conference paper review system displayed in the OntoWeaver Ontology Editor. The domain ontology abstracts the underlying domain data model, by means of a set of classes and properties. The class *Conference_Event* abstracts events such as conferences, workshops, and tutorials. The class *Call* models the calls for papers, workshops, and so on. The class *Topic* describes the topics of the conference. The class *Paper* abstracts the submitted papers. The class *Review* expresses the review information about papers. The class *Person* models people involved in the paper review system. Finally, the class *Deadline* describes the important dates. The instantiation of this domain ontology forms the information about a specific conference.

Specifying the User Ontology

As mentioned before, the paper review system involves three kinds of user roles: *author*, *reviewers* and *program committees*. A user model can be very complex to abstract all kinds of user-specified information including preferences, interests, knowledge background, and so on. To enable the readability of the illustration, we extend the basic user ontology provided by the OntoWeaver customization framework, by adding a property called *interestedTopic* to abstract the interested topics of user individuals, a property called *device* to describe the devices end users use to access the system, and a set of properties to express the preference colour schemes.

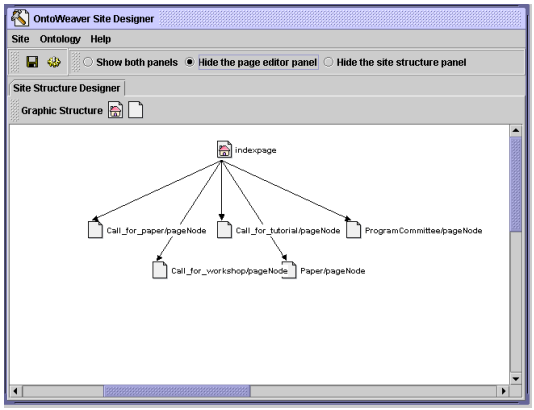

Figure 4 The simplified navigational structure of the general paper review system

Specifying the Site Views

A site view describes the site structures and user interfaces for web applications by means of the site view ontology. Here, we specify a general paper review system for generic users, who can browse the conference information and submit papers to the conference.

The general paper review system comprises six page nodes: *an index page*, *a call-for-paper page*, *a call-for-workshop page*, *a call-for-tutorial page*, *a paper-submission page*, and *a program-committee page*. Each web page is connected with other pages through the same navigation pattern. Hence, we create one navigational component describing the navigation pattern, which comprises a list of hyperlink items. The navigational component is added to each web page to enable the navigation between web pages. Figure 4 shows the simplified navigational structure of the general paper review system. Due to the fact that each web page shares the same navigation structures, the site structure is simplified to show the navigation path from the index page to other pages.

Now let us investigate the specification of the user interface for web pages. Here, we use the index page as an example to illustrate the composition of user interfaces for web pages. The index page comprises two components: the navigation component stated in the last paragraph and the data component that details the particular information about a specific conference. Each component contains a list of interface elements. Figure 5 shows the screenshot of the compositional structure of the index page visualized in the OntoWeaver Site Designer. The left pane shows the compositional structure of the index page. The right pane presents the declarative content of the selected user interface elements. It should be noted that we do not concern about the visual appearances and organizations of the web pages at this stage.

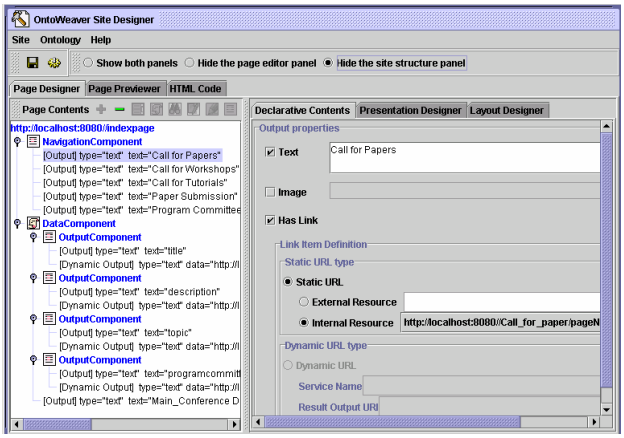

Figure 5 The screenshot of the compositional structure of the index page. The left pane shows the compositional structure of the index page. The right pane shows the content of the selected user interface element.

Presentation Specification

At this stage, site developers specify visual appearances and organizations for user interface elements of web pages in terms of *the presentation ontology*. Due to the limited space, we will not detail the specification process here. Nevertheless, OntoWeaver provides a set of constructs and visual tools to facilitate the specification process.

Customization Specification

In this section, we focus on the customization rule specification, which facilitates tailoring web pages for user individuals. To facilitate the specification and management of customization rules, OntoWeaver provides a visual tool

called *Site Customizer* (as shown in figure 6) to allow the customization rules to be defined and managed by means of user friendly interfaces.

The customization condition has been illustrated in the section of customization condition. Here we use one example to illustrate the specification of customization actions: Creating a hyperlink in the navigation component, connecting to the suggested papers for a PC when he or she shows special interests in some topics. The condition part of the customization rule should state the role of the user is 'PC'; the action part comprises three adaptation actions: (1) creating an *Output* element to present the hyperlink item for paper browsing; (2) creating a *LinkItem* element to specify the actual link details that are associated with the *Output* element; and (3) creating a *parameter* ensuring the contextual information flow through the link to refine the data content.

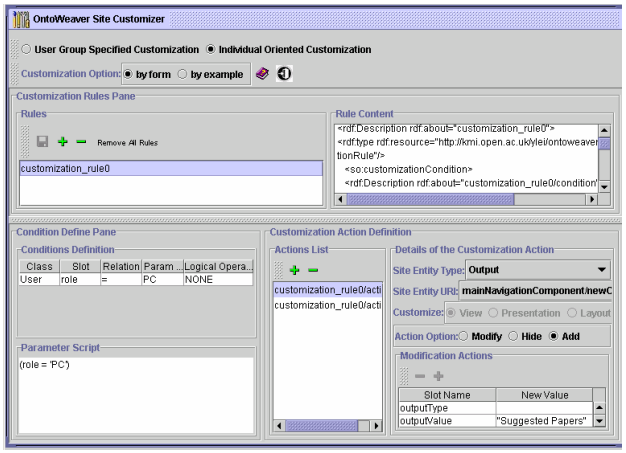

Figure 6 A screenshot of the Site Customizer. The top left pane lists customization rules; the top right pane presents the RDF statements about the selected customization rule; the bottom left pane presents the customization condition; and the right panel lists the customization actions in the current customization rule.

```
<rdf:Description rdf:about="customization_rule0/action0" >
    <so:objectType>View</so:objectType>
    <so:parentSiteEntityURI> mainNavigationComponent/ </so:parentSiteEntityURI>
    <so:siteEntityURI>mainNavigationComponent/newOutput</so:siteEntityURI>
    <so:actionType>new</so:actionType>
    <so:siteEntityType>Output</so:siteEntityType>
    <so:modification>
      <rdf:Description rdf:about="customization_rule0/action0/modification0" >
          <so:slotName>outputValue </so:slotName>
          <so:newValue>Suggested Papers </so:newValue>
      </rdf:Description>
    </so:modification>
    <so:modification>
      <rdf:Description rdf:about="customization_rule0/action0/modification1" >
          <so:slotName>linkitem </so:slotName>
          <so:newValue>mainNavigationComponent/newOutput/linkItem
          </so:newValue>
      </rdf:Description>
    </so:modification>
  </rdf:Description>
```

Figure 7(a) The specification of the customization action that creating a new output element in the navigation component

Figure 7(a) shows the specification of the first customization action. It involves creating a new Output element, and assigning the output value and the URI of the associated link item. The *objectType* specifies that the type of this customization action belongs to the site view customization. The *actionType* and the *siteEntityType* further indicate that the type of the site view customization is to create an *Output* element. The *parentSiteEntityURI* and *siteEntityURI* describe the URIs of the parent interface element and the new element. The *modification* part describes the content for the new element.

```
<rdf:Description rdf:about="customization_rule0/action1" >
    <so:objectType>View</so:objectType>
    <so:parentSiteEntityURI>mainNavigationComponent/newOutput
    </so:parentSiteEntityURI>
    <so:siteEntityURI>mainNavigationComponent/newOutput/link
    </so:siteEntityURI>
    <so:actionType>new</so:actionType>
    <so:siteEntityType>LinkItem</so:siteEntityType>
    <so:modification>
      <rdf:Description rdf:about="customization_rule0/action1/modification0" >
          <so:slotName>isExternalResource </so:slotName>
          <so:newValue>false</so:newValue>
      </rdf:Description>
    </so:modification>
    <so:modification>
      <rdf:Description rdf:about="customization_rule0/action1/modification1" >
          <so:slotName>associatedResourceURI </so:slotName>
          <so:newValue> http://localhost:8080/paper_review/paper_browsingpage
          </so:newValue>
      </rdf:Description>
    </so:modification>
    <so:modification>
      <rdf:Description rdf:about="customization_rule0/action1/modification2" >
          <so:slotName>parameter</so:slotName>
          <so:newValue> newParameterURI </so:newValue>
      </rdf:Description>
    </so:modification>
  </rdf:Description>
```

Figure 7 (b) the specification of the customization action that creating a new link item for the specified *output* element.

Figure 7(b) details the specification about the second adaptation action, which creates a new *link item*. The new link item specifies the URI and the type of the linked resource; and the URI of the associated parameter. The parameter works to refine the instances of the domain class *Paper* in the paper browsing web page by using the value of the slot *interestedTopic* as a filter. As a result, it enables the dynamic personalization of the paper browsing web page according to the different interest of user individuals. However, because of the limited space we will not detail the specification of the creation of the parameter. Nevertheless, the specification principles are the same.

Regarding the presentation customization and the device-dependent customization, they can be achieved by specifying customization conditions and actions according to requirement of different devices and individual users.

DISCUSSION

With the OntoWeaver methodology, customization can be achieved at different levels. First, different site views can be created over the same domain model for different user groups or different environments. Second, different presentations can be specified to present the same site views in different ways. Finally, the individualization of

web applications can be achieved by using the OntoWeaver customization framework.

Now we discuss how to use the OntoWeaver customization framework to enable the various types of customization approaches identified in the literature:

- *User model-driven adaptive systems* [2]. The OntoWeaver customization framework can be easily used to provide appropriate support for such systems, where user models are pre-defined, user profiles are collected from various sources, and the systems aim to provide adaptive content or presentation to individual users. They typically focus on constructing user profiles using various techniques and embed annotations to web pages to realize various adaptations. Using OntoWeaver, we can achieve the goal of these systems by designing the target web application, pre-defining a user model, and pre-defining a series of adaptive rules. The rules work with the site models according to user profiles to produce adaptive contents and presentations to user individuals.
- *Automatic personalization systems* [15]. These systems primarily use data mining algorithms to automatically discover and extract patterns from web usage data and predict user behaviour while users interact with the web. As long as the user model (not user profile) can be defined, the OntoWeaver customization approach can provide appropriate support for this kind of personalization, because OntoWeaver is only concerned with the definitions of customization rules. It does not matter whether the user profile are defined during design time or just captured during run-time, because customization happens at run time.
- *The manual customization approach*es [13]. OntoWeaver provides run-time support for user-driven customizations. In this case, users only need specify customization actions through a direct manipulation of user interface. These actions are then captured by OntoWeaver and compiled into appropriate customization rules.

RELATED WORK

Recently, a great variety of technologies and systems have been developed to achieve the goal of customization and personalization for web applications [2] [11] [14]. These approaches primarily use a user model to record user preferences and interests, and exploit various techniques to collect data for the user model, including implicitly observing user interaction, automatically discovering and extracting patterns from web usage data, and explicitly requesting direct input from the user. However, while these systems can precisely predict user's behaviour, there is not much they can do if the web application models are hidden from the personalization framework.

The My Yahoo approach [13] is a typical example of a user specified customization methodology, which allows users to select preferred modules from hundreds of available ones. It can provide a limited extent of customization support because the site descriptions are available through the format of different module descriptions. However, it does not offer a rule model to allow end users to define rules to reason about its content to provide intelligent presentation support.

WUML [9][10] proposes a generic customization model allowing the adaptation of web applications towards the context implied by ubiquity. It provides a context model that facilitates the specification of detailed information about the environment of a web application and the web application itself, and a customization rule model to enable the specification of the actual customizations. The WUML approach offers complex models to allow developers to express detailed information about the customization environment. However, it does not support modelling of web applications. As a consequence, it requires the web applications to be sliced into a stable part, comprising context-independent structure, and a variable context-dependent part, which is the subject of the adaptations. Furthermore, to enable customization, it requires web applications to provide adaptation hooks. Thus, it greatly limits the possibility for customization by forcing site developers to anticipate what may be customized.

WebML [3] [4] is another interesting approach which considers personalization issues within the design phase of web applications. It exploits user profiles, delivery specifications and a series of business rules to enable the so-called one-to-one web delivery. The delivery specifications, which express the user-specific data extractions needed to generate personalized pages, are added to the web application model as annotations. The business rules are specified to compute and store user-specific information. However, the WebML customization approach does not offer an explicit rule model that would allow rules to be specified at a high level.

OOHDM [16] [17] allows customization to be specified: (1) in the conceptual model by explicitly representing users, roles and groups and by defining algorithms that implement different rules for different users; (2) in the navigational model by defining completely different applications for each profile; and (3) in the interface model by defining different layouts according to user preference or selected devices. However, the resulting customization is static and pre-defined. The Hera approach [5] considers adaptation as an import modelling issue during the process of designing web information system. OntoWebber [8] supports the so-called coarse-grained personalization by assigning different site views for user groups, and the fine-grained personalization by modelling user individuals. None of these approaches provides an explicit customization specification model that would allow the specification of customization requirements to be carried out at a high level.

CONCLUSION AND FUTURE WORK

This paper extends IIPS [12] to OntoWeaver by proposing and introducing a customization framework into the ontology-based web modelling approach to the design and

maintenance of web applications. The OntoWeaver approach to customization is based on its conceptual modelling approach, which provides means to describe site structures, presentations and layouts, and their underlying meta-data structures declaratively in an exchangeable format. It provides a customization rule model to enable the specification of customization conditions and actions at a high level of abstraction. It offers a basic user model to allow user models to be specified in a domain specific way. It makes use of the specified user model and declarative site models to enable the precise specification of all aspects of customization rules at a conceptual level.

The OntoWeaver customization approach offers the capability to provide meta-level customization support for the target web applications, as it supports web applications exploiting various techniques to collect user specified information, as long as user profiles are recorded as instances of the specified user ontology. As a result, OntoWeaver can make use of the user profiles to get contextual information of the end user, and exploit the Jess inference engine to reason about site specifications according to the obtained contextual information to present individualized web pages.

To support the design and development of customized web applications, OntoWeaver provides a suite of visual tools enabling the tasks to be carried out at a conceptual level. At the moment, a prototyped OntoWeaver infrastructure has been implemented based on RDFS [18] and RDF [19].

In the future we plan to use the new emerging semantic web standards like DAML+OIL [6] and OWL [20] as the underlying language to represent ontologies and specifications.

AKNOWLEDGEMENTS

We would like to thank Victoria Uren, Maria Vargas-vera, and Dnyanesh Rajpathak for their valuable comments on the earlier drafts of this paper.

REFERENCES

[1] T. Berners-Lee, R. Fielding, L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax.* http://www.ietf.org/rfc/rfc2396.txt.

[2] P. Brusilovsky, *Methods and techniques of adaptive hypermedia.* User Modelling and User Adapted Interaction, 1996, v6, n2-3, pp 87-129.

[3] S. Ceri, P. Fraterali, S. Paraboschi, *Data-Driven One-To-One Web Site Generation for Data-Intensive Applications.* Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, pp. 615-626.

[4] S. Ceri, P. Fratenali, A. Bongio, *Web Modelling Language (WebML): a modelling language for designing Web sites.* WWW9 Conference, May 2000.

[5] F. Frasincar, G. J. Houben, *Hypermedia Presentation Adaptation on the Semantic Web.* Second International Conference, AH2002, Malaga, Spain, pp. 133-142.

[6] Horrocks et al., *DAML+OIL*, *http://www.daml.org/2001/03/daml+oil-index*, 2001.

[7] E. Friedman-Hill, Jess, http://herzberg.ca.sandia.gov/jess/.

[8] Y. Jin, S. Decker, G. Wiederhold, *OntoWebber: Model-Driven Ontology-Based Web Site Management.* The 1st International Semantic Web Working Symposium (SWWS'01), Stanford University, Stanford, CA, July 29-Aug 1, 2001.

[9] G. Kappel, B. Pröll, W. Retschitzegger, W. Schwinger, *Modeling Ubiquitous Web Applications - The WUML Approach*, International Workshop on Data Semantics in Web Information Systems (DASWIS-2001), Yokohama, Japan, November 27-30, 2001.

[10] G. Kappel, W. Retschitzegger, E. Kimmerstorfer, B. Proll, W. Schwinger, and TH. Hofer, *Towards a Generic Customization Model for Ubiquitous Web Applications*. Proceedings of the 2nd International Workshop on Web Oriented Software Technology, 2002, pp. 79-104.

[11] A. Kobsa, J. Koenemann and W. Pohl, *Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships.* The Knowledge Engineering Review 16(2), 2001, pp.111-155.

[12] 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, 2002, pp. 219-234.

[13] U. Manber, A. Patel, and J. Robison, *Experience with Personalization on Yahoo!.* in Communications of the ACM, August 2000, Pages: 35-39.

[14] J. Mostafa, *Guest Editor's Introduction: Information Customization.* IEEE Intelligent Systems, 17(6), 2002, pp.8-11.

[15] M. Perkowitz, O. Etzioni, *Towards adaptive Web sites: Conceptual framework and case study.* Artificial Intelligence 118 (2000), pp. 245-275.

[16] G. Rossi, D. Schwabe, R. Guimarães, *Designing personalized web applications.* WWW 2001: 275-284.

[17] D. Schwabe, R. M. Guimaraes, G. Rossi, *Cohesive design of personalized Web applications,*. IEEE Internet Computing, Volume: 6 Issue: 2, March-April 2002, pp. 34 -43.

[18] *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation.* http://www.w3.org/TR/rdf-schema/.

[19] *Resource Description Framework (RDF) Model and Syntax, W3C Proposed Recommendation.* http://www.w3.org/TR/PR-rdf-syntax/.

[20] OWL Web Ontology Language, W3C Working Draft, March 2003, http://www.w3.org/TR/2003/WD-owl-features-20030331/.