On Enriching Ajax with Semantics: The Web Personalization Use Case

Kay-Uwe Schmidt[1], Ljiljana Stojanovic[2], Nenad Stojanovic[2], and Susan Thomas[1]

[1] SAP Research, CEC Karlsruhe, Vincenz-Prießnitz-Str. 1,
76131 Karlsruhe, Germany
`{Kay-Uwe.Schmidt,Susan.Marie.Thomas}@sap.com`
[2] FZI Forschungszentrum Informatik, Haid-und-Neu-Straße 10-14,
76131 Karlsruhe, Germany
`{Ljiljana.Stojanovic,Nenad.Stojanovic}@fzi.de`

Abstract. With the dawn of Ajax the capabilities of tracking user behavior multiplied. The same holds for the capabilities of adapting the user interface in a Web browser. To provide meaningful adaptation, the events, context and elements of an Ajaxified Portal must be given meaning. We show the use of ontologies as a model for user-related context and portal-related content. Content-related concepts are used to annotate Ajax widgets to associate them with meaning. As a user navigates a portal and fires events related to the widgets, a semantically rich user model is built, enabling suitable adaptation. Both the user model and the adaptation are based on ontologies and logic rules. Since user tracking and portal adaptation in the era of Ajax, now takes place on the client-side we present a resource-saving approach to executing adaptation rules in the browser. The approach is applied in an e-Government case study.

Keywords: User Adaptivity, Ajax Portal, Semantic Web, e-Government.

1 Introduction

In most e-Government projects to date, technology was in the center of the project and not the user, although the user, e.g., the citizen or business person, is the one who shall in the end use all the new and exciting online e-Government services. As long as all efforts are technology driven, e-Government will not take off, and will not reach its full potential. To confront different citizens with a one-size-fits-all Web interface is not the optimum way to deliver public sector services because every person is an individual with different knowledge, abilities, skills and preferences. The conventional brick-and-mortar office has a more human face because the clerk can respond to different people in different manners. That is why people tend to use the conventional office rather than the e-Government services. To transfer some of the humanity to e-Government portals, it is necessary to build adaptive portals for public services. Such user-adaptive portals will increase the usability, and, thus, the acceptance of e-Government, enabling administrations to achieve the, as yet, elusive efficiency gains and user satisfaction that are the primary goals of e-Government projects.

This paper describes an approach for adaptation that addresses these issues. This approach results in a system that is both user-adaptive and self-adaptive. By 'user-adaptive', we mean an interactive system that acquires a model of the individual user, and utilizes that model to adapt itself to the user. Such adaptation usually involves some form of learning, inference or decision making (paraphrased from [1]). By 'self-adaptive', we mean a system that observes the results of its actions and adapts itself to improve future performance. Such adaptation can be automatic or mediated by a human. This paper, however, concentrates on user-adaptivity.

Of the types of functions user-adaptation might fulfill – identified in [1] – the approach focuses on two: first, 'help with system use', in particular, help that enables a user to efficiently use an offered e-Government service; second, support for information acquisition, in particular, the information related to the offered e-Government services.

To achieve user-adaptivity we use a new approach that combines the power of Ajax, the underlying technology of Web 2.0, with Semantic Web technologies to create a client-side semantic framework for capturing the meaning of user behavior, recognizing the user's situation, and applying rules to adapt the portal to this situation.

Although, in this paper, the discussion centers around e-Government portals, the architecture is easily generalizable to other types of portals, since at an abstract level most portals can be said to offer some combination of services and associated information.

The rest of this paper is organized as follows. Section 2 presents some examples of e-Government services, and derives requirements on adaptive e-Government portals. Sections 3 and 4 explain the advantages of Ajax and Semantic Web technologies when it comes to meeting these requirements and achieving user-adaptation. Section 5 presents our approach, which combines Ajax and Semantic Web technologies. Section 6 compares the approach to related work. Section 7 indicates the direction of future work. Finally, Section 8 concludes the paper with acknowledgements.

2 Motivating Examples and Requirements

A typical service provided by an e-Government portal is submission of an application form related to a building project. Such a service is actually a complex process, subject to regulations that require the submission of different forms at different times, depending on the type of building project. For an inexperienced user the challenge starts here. With lack of background knowledge of the building regulations the user is confused and does not know which form to choose for her building project. Such users, unfamiliar with the portal and the specific service, need guidance to prevent them from getting stuck in the portal shallows. On the other hand, for an architect who works daily with the virtual building application within an e-Government portal, any guidance would only hinder her smooth sailing. Therefore, there is, among other things, a need to cater for different skill levels like novice, average and expert.

Moreover, adaptation, for example adaptation to skill level, is needed for each service, since an expert in one service may be a novice in another. For example, the architect, expert at building applications, may be a novice hen it comes to submitting

an application for child support. In fact, many services will only rarely be used by any one user, so the majority of users will probably remain novices in their use.

Given this example we derive five basic requirements for adaptive e-Government portals. Firstly, a portal must provide guidance and information that matches the users, e.g., the different skill levels and interests of its citizens. Secondly, as citizens typically use e-Government services rarely they should not be bothered with providing and maintaining any user profiles.

The third important requirement is to observe the crucial usability principles such as responsiveness, predictability and comprehensibility, controllability and unobtrusiveness. Initial emphasis, in regard to usability, is placed on providing accurate, but unobtrusive, guidance, when and where it is needed by the user.

The fourth general requirement is that the portal should be subject to continual improvement. Explicit user feedback can be enormously helpful here, but the feedback requested should be relevant to the services that the user executed.

A fifth, and final, important requirement is related to the nature of e-Government services, and the fact that there are multiple units of e-Government at different levels, e.g., local, regional and national. Given the similarity of many of the services offered by these different units, there is an enormous potential for efficiency gains through sharing best practices. Therefore, the fifth general requirement is the ability to share successful adaptation strategies and rules.

In the rest of the paper, we describe an approach that meets these identified requirements.

3   Mashup of Ajax and the Semantic Web

As indicated by the definition of user-adaptivity given in Section 1, acquisition of a model of the user is the indispensable pre-requisite to adaptation. The second step is then to use this model to perform the adaptation. As shown by the requirements analysis of Section 2, it cannot be assumed that the system has any previous information about a user, so that in each user session the user model has to be acquired from scratch. Effectively, this means that the user model is based on the user actions during a session. Therefore, it is essential to be able to track and interpret these actions as accurately as possible. This section shows that Ajax enables the required fine-grained tracking of user behavior and that, additionally, it provides a richer set of adaptation options than standard HTML-based Web technology. The next section then shows how semantics enables interpretation of the user behavior that can be collected using Ajax.

In Adaptive Hypermedia Systems (AHS) adaptation strategies were already studied [2] and are well understood for conventional Hypermedia systems. Under conventional techniques the creation of HTML pages on a remote server and the typical request-response user paradigm of the Web are subsumed. With conventional techniques, the tracking of user clicks, the user modeling, as well as the adaptation all take place on the server. This limits the possibilities of user tracking to the user requests seen by the server [3], which is actually a subset of the user clicks. Furthermore adaptation can only take place when a user requests a new page, which then is adapted to her needs. On the fly adaptation without reloading the whole page in not obtainable.

With the dawn of Ajax in early 2005 [4] a new potential of tracking a user's browsing behavior, as well as new adaptation strategies arose. With Ajax the look and feel of Web pages can be transformed to that of desktop applications. This is the result of the seamless combination of three powerful technologies in Ajax: Asynchronous communication, JavaScript and XML. Using asynchronous communication, just the needed data can be obtained from the server without reloading the whole page. Additionally, JavaScript as a language to execute code in a browser, and XML for the ad-hoc manipulation of a Web page make it possible that a user can be supported without explicitly clicking a link on a page. Thus help and guidance can be already provided when the user behavior is recognized as searching for additional information without server communication explicitly originated by the user.

With Ajax the range of user actions that can be tracked is extended beyond just mouse clicks. For example, scrolling, mouse over and keystroke events can be tracked enabling the detailed recording of user actions on the client-side. In the world of the HTTP requests-response paradigm the Web server is not able to obtain such detailed information. A Web server can only track a subset of user clicks. It misses browser events, like the Back button and cached links. The well-known problem of assigning clicks to users is also solved on the fly, since user tracking takes place directly on the client-side. Additionally, the user's Web browsing behavior can be processed directly on the client and the browser can react immediately to recognized behavioral patterns.

The advanced user tracking possibilities are also accompanied by sophisticated adaptation techniques formerly only seen in desktop applications, like tool tips and fading help windows. However, this rich model of user actions and new adaptation options can only be leveraged if their meaning is machine readable as discussed in the next section.

4   Semantics-Based Adaptation

In this section we show how semantic technologies and in particular ontologies can be utilized for automatic adaptation of an e-Government portal to the individual requirements of the users. We firstly motivate the reasons for using ontologies and thereafter we introduce the semantic model of adaptive portals. Even though the paper is motivated by using e-Government examples, the proposed approach is general enough to be applied in any other domain.

4.1   Advantages of Using Ontologies for Adaptation

There are several reasons to build our approach upon the intensive use of semantic technologies. Firstly, ontologies enable semantic interpretation of user behavior in a portal, which enables meaningful, effective and context-aware adaptation.

The building permission example from Section 2 is elaborated next to show how Ajax and semantics together enable such context-aware adaptation. Assume that the user, who wants to apply for building permission, goes to the appropriate e-Government Web site. And, on this site, the user finds a list of hyperlinks to forms related to building permits. But, she does not know which one is appropriate for her building project. Being based on Ajax, the Web site implements mouse-over help for

these hyperlinks. The user knows this, and places the mouse on a hyperlink for a time to make the help appear. Then the user does this for a second hyperlink, but still does not choose a form. Assuming that the hyperlinks have been associated with concepts in the ontology, the system can now make a semantic interpretation of the user's behavior. In this case, the conclusion would be that the user has a strong interest in the concepts associated with the two mouse-over hyperlinks, and that the user needs help choosing a form. In response to this context, the system can offer the user help. Not only that, this help can be tailored to the user by taking account of the concepts in which the user showed interest, concluded from her current navigation path and behavior. As explained later, adaptation such as this is based on using semantic annotation of a page and its structural elements (e.g. hyperlinks).

A second reason to use ontologies is that ontologies used in rules can make adaptation logic more explicit. This declarative representation, expressed as rules using concepts and relations from the ontology, helps the domain experts inspect, understand and even modify the rationales behind adaptive functionality. For example, the hierarchical organization of e-Government services allows the expert to model adaptation rules on a more abstract level, i.e., covering more than one concrete service (e.g. building permission service, independently of the type of building such as house, office, etc). This reduces significantly the number of rules and makes maintenance of the system much easier.

Finally, ontologies facilitate sharing knowledge between portals, especially for those offering similar services (e.g. two municipalities in one state are similar). For example, the best practices gathered in issuing building permits in one portal (e.g. inexperienced users need an additional explanation regarding the hyperlink "required documents") can be easily transferred to other portals that implement the same regulations for issuing building permits. This sharing is greatly facilitated by the fact that all of the terms used (e.g. additional explanation, hyperlinks, "required documents" etc.) are well defined. It is clear that the benefits for the users as well as for e-Government are enormous, since the public administration can improve its performance at much less expense.

4.2   Ontology-Based Model of Adaptive Portals

Since the data relevant for adaptation is rather sparse, or a great deal of interpretation must be done to turn it into actually useful information, we have developed the ontology-based model of adaptive portals. This model (the so-called Portal Adaptation Ontology) is used to decide if an adaptation should take place and how to do that. A part of the ontology is shown in Figure 1. The full version can be found in [5]. The ontology represents all aspects relevant for adaptation such as Web site structure (Web Portal Ontology), Web site content (Content Ontology), user profiles (User Ontology), and Web site usage data as well as knowledge about the adaptation process itself (Adaptation Ontology). Ajax-enabled Web pages as well as the UI elements contained by those pages will be annotated with individuals and concepts form the Portal Adaptation Ontology.
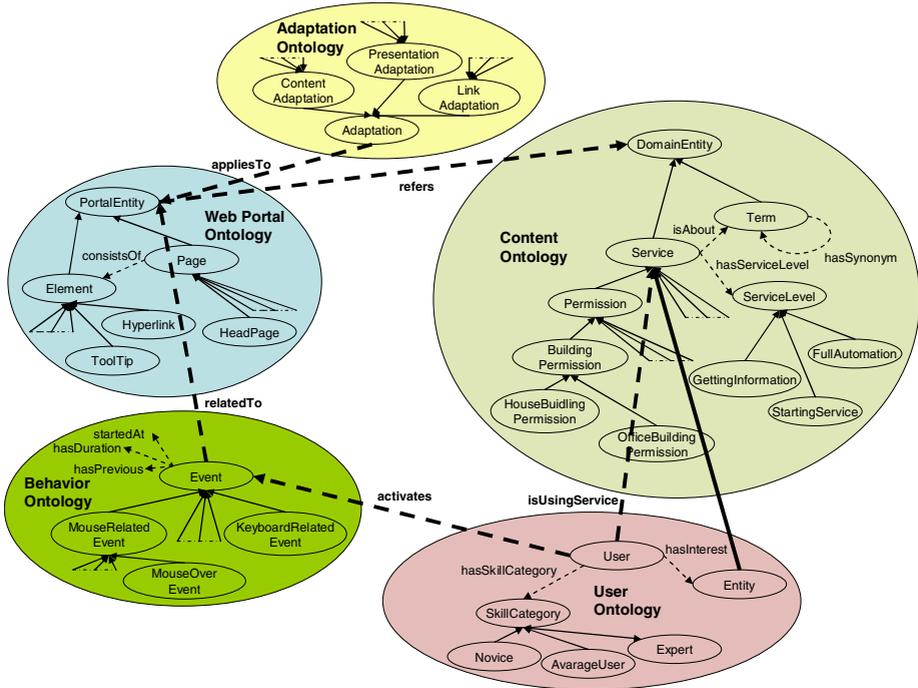
Fig. 1. A part of the Portal Adaptation Ontology showing several entities of the included ontologies as well as dependencies between them

Web Portal Ontology: The way that the Web site is physically laid out as well as the structure of each page can be useful toward understanding usage behavior and interpreting system suggestions. Additionally, the semantic information about the reasons why the structure exists in the way that it does may also be useful. Thus, the Web Portal Ontology contains entities representing the types of pages (such as Head Page, Navigation Page, FAQ, Combined Page etc.) and the structural elements of a page (e.g. Hyperlink, Figure, Table, Content, etc.). We note here that information about page structure can be used to derive or to verify the type of a page [6]. For example, a Navigation Page is a page with small content/link ratio; short time spent on page and is not a maximal forward reference.

Content (Domain) Ontology: The content[1] of Web pages themselves is essential to determining particular topical interests and understanding the relationships between pages. The Content Ontology consists of concepts and relations modeling the meaning of services/information offered by an e-Government portal. This includes already existing categorization[2] of e-Government services (such as residential affairs,

---

[1] By 'content' we assume the meaning and not the syntax of a page.

[2] It has been developed based on the existing standards for modeling life events such as the Swiss Standard eCH-001 that aims to give an overview over all relevant e-Government services in Switzerland and therefore to provide a consistent and standardized classification of the services.

residential permissions, identification, certifications, naturalization citizenship, moving, education, etc.) as well as typical e-Government terminology (e.g. building permission, building application, etc.).

User Ontology: The user is modeled through the concept User and its properties such as hasInterest, hasSkillCategory, etc. As already mentioned the values of these properties are determined on the basis of user actions during the session. For example, to determine the interest of the user the content/meaning of the pages the user visited is taken into account. Indeed, semantic annotation of pages using the entities from the Content Ontology is used to derive this information. Returning to the example from Section 2, the system would conclude that the user has a strong interest in the concepts associated with the two moused-over hyperlinks, since these concepts define the meaning of hyperlinks.

The hierarchy of user skill categories includes, at the first level, concepts such as Novice, AverageUser and Expert. For example if a user often goes back to the previously visited page, then we assume she is overwhelmed and has become unable to navigate effectively and is therefore classified as a novice. We note that the skill category of the user implicitly applies only to the service that the user is currently using, since the scope of the user categories is limited. That is that the categories are not valid on the global portal level but on page/service[3] level. For instance a user familiar with building applications might be categorized as an experienced user on the appropriate pages in the e-Government portal which deals with building applications. On the other hand, she might be a domain novice when trying to enroll her child in a public school.

Behavior Ontology: The most important data set is the recording of interactions of users with the Web site, in other words, the way that the Web site is used. Even though, this is by far the most abundant collection of data, provided by Ajax, it is, however, the least informative on its own and needs to be enriched with semantics and interpreted.

However, interpreting event data is difficult if the data is not normalized into a common, complete, and consistent model. This entails not only reformatting the data for better processing and for achieving readability, but also breaking it down into its most granular pieces. For example, the system has to be able to recognize all mouse-related events such as mouse-down, mouse-move, mouse-out, mouse-over, mouse-up, etc. Moreover, interpretation involves filtering out unwanted information to reduce analytical errors or misrepresentations. For example, the system should be able to condense the received events into a single event directly indicating a problem. Returning to the example from the beginning of this section, the adaptation should be generated only if two mouse-over events occur sequentially within a session. Finally, interpretation involves acquiring more information from outside the scope of the original event data, for example, from a page the event occurred on (e.g. replacing the meaningless information such as name and target of a hyperlink with the meaning of this hyperlink).

---

[3] A page must be annotated with the service it belongs to in order to enable the system to link the user with a service.

To cover all these requirements we have developed the Behavior Ontology that structures information about the user's interactions and relationships and/or dependencies between interactions. The main purpose of the ontology is to store all the interactions of the user which might help to identify her experience, actual context and goals.

The most important concept of this ontology is the concept Event that describes what happened, why it happened, when it happened, and what the cause was. The structure of the hierarchy of events reflects the underlying technology used for capturing events, i.e. Ajax. For example, events are decomposed at the first level into the event categories: keyboard, button, mouse, focus and general events. Each of these categories is further specialized. For example, keyboard-related events occurring when a user hits a key contain events such as key-down, key-up and key-press. The category of general events cover load, unload, submit, error handling, etc.

Adaptation Ontology: This ontology was derived from the taxonomy of adaptive hypermedia systems [2]. We distinguish between content, presentation and link adaptation. Each of these types can be further categorized. For example, adaptation of navigation which realizes adaptation by changing the links of the system (i.e. LinkAdaptation) can be realized by several techniques such as DirectGuidance, LinkSorting, LinkHiding, LinkAnnotation, LinkGeneration or MapAdaptation. Each technique might also be realized in several ways. For example, LinkHiding concerns links that are not considered relevant for a user (at the current time), and can be realized by hiding, disabling or removing links.

As shown in Figure 1, all the previously mentioned ontologies are combined in the Portal Adaptation Ontology that models adaptive functionality formally and explicitly. Moreover, it is enriched with rules[4], as discussed below, to enable automation of the adaptation process. In this way, we provide a logical characterization of self-adaptive e-Government systems. We note here that we use the OWL-DL ontology language to represent ontologies. Rules are encoded in the SWRL[5] language, and the KAON2[6] inference engine is used to perform ontology and rule-based reasoning.

We classify the rules into two types based upon their roles in the adaptation process:

Categorization Rules: These rules assign a current user to the predefined user categories. For example[7], a user is an expert for a service, if she uses a bookmark to load a page representing this service.

```
FORALL hasSkillCategory(U,"Expert") ←
  User(U) AND Service(S) AND isUsingService(U,S) AND
  Page(P) AND refers(P,S) AND bookmarkUsage(E) AND
  relatedTo(E,P) AND activates(U,E).
```

---

[4] Concepts and relations defined in the Portal Adaptation ontology directly or indirectly through included ontologies are used in rules.

[5] Semantic Web Rule Language: http://www.w3.org/Submission/SWRL

[6] KAON2: http://kaon2.semanticweb.org/

[7] Note that all terms used in these examples belong to the Portal Adaptation Ontology. Additionally, due to complexity of SWRL format, rules are represented using FLOGIC syntax.

Adaptation Rules: These rules automate corrective actions, i.e. adapt the content, structure or layout of a page to the current user based on the category to which she belongs. For example, if a user is not an expert, and if she spent more than 100ms reading a tool tip of some element on a page, then context-sensitive and content-sensitive help explaining the meaning of this element should be shown to this user.

```
FORALL D ShowAdditionalInformation(T,D) ←
  User(U) AND hasSkillCategory(U,"Expert") AND
  MouseOver(E) AND activates(U,E) AND hasDuration(E,t)
  AND greater(t,TimeConstant⁸) AND ToolTip(T) AND
  relatedTo(E,T) AND DomainEntity(D) AND refers(T,D).
```

5   Bringing Together Semantics and Ajax

In this section we relate the ontologies, introduced in the previous section to Ajax technology, and describe how their combination enables adaptivity. There are four key elements to achieving adaptivity: annotation, event interpretation and correlation, the user model and adaptation to the user. These four are explained in the initial part of this section. Then a more detailed description of the user model is given. After that we discuss the main challenges of integrating semantics and Ajax.

The fundamentally new in the idea to marry Ajax with the semantic Web is that JavaScript events are associated with concepts, thereby, becoming meaningful 'words' in the interaction with the user. By means of appropriate annotations, the context of JavaScript events can be recognized and the portal can react accordingly. The annotations come from our Portal Adaptation Ontology (see Section 4) and are stored in a knowledge base.

Semantics are indirectly associated to events by annotating the UI elements which fire events as the user interacts with the portal. The UI elements, also called Widgets, of an Ajax page can be annotated with concepts from the Content Ontology (see Section 4) e.g. the concept of a building application in an e-Government context. They can also be annotated with concepts related to the purpose of the widget e.g. with the concept of 'navigation' for a widget meant to navigate the user to a sought-after service.

Events on their own, even when coupled with semantics, are not enough. First, sequences of events have to be correlated into more meaningful units. Thus, simple JavaScript events like mouse over events are combined into compound events, which can, in turn, be a starting point for subsequent correlations. A compound event can consist in such a way of multiple simple events. Based on the list of the compound events, a model of the user can be derived, i.e. a proper instantiation of the Behavior Ontology (see Section 4) will be generated.

The user model is the basis for adaptation. One attribute of the user model is the user category. Categories are pre-defined, either on the basis of a priori knowledge, or on the basis of offline categories discovered by data mining. When possible a user is classified into one of these pre-defined categories. This is done on the basis of the

---

⁸ `TimeConstant` is a numerical value that is dynamically changed based on the log information.

context information extracted from the semantic concepts related to the simple and compound events. The list of events and the context of the user derived from it, as well as the user category are the essential attributes of the user model that enable adaptation.

Adaptation rules evaluate the current user model and generate abstract actions, which can be interpreted by the portal to adapt to the user. Abstract actions must then be converted by the portal into concrete changes to the user interface. For example, from an adaptation rule, it might follow that the user is lost in the portal shallows, and needs an assistance window in order to reach her goal. The adaptation rule only specifies that an assistance window is needed. The content of the assistance window is derived from the semantics of the events and from UI-elements linked with those events. In this way, the exact conversion of the adaptation directives conforms to the style sheets used by the portal.

In these last few paragraphs of this section, we take a closer look at the user model and the advantages of SWRL rules. All significant events generated by user interactions are collected and stored in logical event queues. The chronological sequence of the events is guaranteed by the assignment of a time stamp to each event. The rules for the correlation of events are expressed in SWRL. One advantage of using SWRL is that the Portal Adaptation Ontology can be accessed by the rules directly. A further advantage is that SWRL can be serialized as an OWL ontology. Thus, reasoning support is available.

As discussed previously, SWRL rules are used to classify the user into a category. They are also used to derive abstract actions to adapt the portal to the user. Like the events, the actions are stored chronologically in a queue ordered by a time stamp.

Both logical queues are modeled in the Behavior Ontology during design time and serve as client-side data structures driving the portal adaptation. The event queue stores the JavaScript a.k.a. Ajax events and the action queue stores the resulting adaptation steps.

5.1 Challenges Integrating Semantics and Ajax

The combination of Semantics and Ajax brings many advantages for dynamic portal adaptation. But this does not come for free. While starting the implementation of our solution we were faced with a lot of tricky challenges, all resulting from moving user tracking from the server to the client-side.

The decision to react to user behavior at the level of JavaScript events leads to a rich and verbose user model. With every new JavaScript event, the user model, and thus the user context, may change. Every such change requires execution of rules. For this reason, rule execution on the server-side, by means of the asynchronous communication facility of Ajax is infeasible; it would overload the server. So, we needed to move rule execution to the client-side. Therefore the major challenge is to implement rule execution in the client, which has limited resources and limited programming libraries.

That is, we have to deal with adaptation on the client-side, adaptation based on JavaScript event streams, and JavaScript only programming capabilities. Taking these constraints into account the following concrete challenges arise.

- Resource saving ontology-based model of adaptive portals
- Annotating Ajax pages with semantic concepts from the ontologies
- Extracting semantic annotations on the client-side
- Rule-based portal adaptation and Execution of rules on the client-side
- Portal specifics, dynamic Web pages and self-adaptivity

5.1.1   Resource Saving Ontology-Based Model of Adaptive Portals

We have already solved this challenge by carefully designing and implementing the ontologies for portal adaptation with respect to the limited resources at the client-side and to the rich user model conditioned by the verbose Ajax events (see Section 4).

Because of the resource restrictions in typical browser environments we developed new and rather small user model and behavior ontologies and did not reuse already existing but large user model ontologies like GUMO [7]. In GUMO a rich user model is proposed with many concepts and properties not applicable to the domain of adaptive portals.

5.1.2   Annotating Ajax Pages with Semantic Concepts from the Ontologies

The challenge here is to establish a link between the UI elements of an Ajax page and the concepts of the Portal Adaptation Ontology that describe them. The annotation of HTML pages with RDF triples was already a topic of several investigations and there exist a couple of solutions [8]. However, because we wanted to avoid deep changes to the portal we decided to follow a different approach.

Our idea is to store the semantic descriptions in a knowledge base. The knowledge base is an extra ontology that is not directly integrated into the Web pages, but is left on the server together with the other ontologies. But the open question is how to link all relevant UI elements of an Ajax page to the concepts which provide the semantic context information. This can be done using the optional `id` attribute provided by nearly every HTML element. As all UI elements of an Ajax page are in fact HTML elements, we can add the optional `id` attribute to every UI element we want to annotate. In order to establish a link between the annotations stored in the knowledge base and the UI elements described by them, a special property was introduced in the ontology. This property carries the value of the `id` attribute of an UI element. Thus, whenever information is needed for a certain UI element, the `id` serves as a link to its semantic annotations.

However, this raises further challenges: How to guarantee the unambiguity of the identifiers, and how to access the ontology containing the annotations from the browser?

5.1.3   Extracting Semantic Annotations on the Client-Side

As discussed above we are in favor of using a separate ontology for storing the semantic annotations of the portal. That saves us from dealing with the awkward extraction of Metadata embedded directly in the HTML page. It also saves us from cumbersome XML and ontology processing. The challenge is how to access the Portal Adaptation ontologies stored on a Web server from the browser on the client-side.

Based on the work done in [9] we developed a prototypical Java library that translates ontologies to JavaScript objects. These objects can be directly accessed and evaluated within an Ajax page. A first promising candidate for the encoding of the

ontologies in JavaScript is JSON [10]. The JSON string encoding the ontologies can be accessed by the Ajax page using its asynchronous communication facility.

Another issue for further investigation is to perform reasoning over the ontologies in the browser with JavaScript. Although, there exist at least one inference engine supporting JavaScript and backward-chaining reasoning [11] we decided not to use a reasoner on the client-side. There are mainly two reasons which caused this decision: Firstly, we already can perform the externalization[9] of the ontology at the server. This is only done once in advance on the server and thus has no negative implication on the portal adaptation at runtime. Secondly there is no explicit need for doing reasoning on the client-side because all HTML elements are also annotated in advance and thus well know before runtime.

Not covered by externalization are the JavaScript events and the adaptation actions because they are dynamically created at runtime. But events and actions are annotated via their targets, that is, the UI elements they are connected with. So we don't need reasoning at runtime. At this stage, having the Portal Adaptation ontologies and the link from the HTML elements to the ontologies, the open questions are: what is the most appropriate representation of the Portal Adaptation ontologies at the client, and what is the best way to synchronize the user model on the client with the user model on the server-side, in case there are rules to be executed at the server-side.

5.1.4 Rule-Based Portal Adaptation and Execution of Rules on the Client-Side

The challenge here is to develop easy to maintain rules taking into account the semantic knowledge of annotated JavaScript events. Four rule types have to be designed: Extraction, correlation, categorization and adaptation rules. Extraction rules add the semantic annotations from the knowledge base to the core JavaScript events. Since the events are only indirectly annotated by their target UI elements, some logic is necessary to combine the events with the semantics. Correlation rules combine simple JavaScript events and their semantics to an interpretable user behavior. Categorization rules evaluate the semantics of JavaScript events in order to categorize the current user properly. Based on the user categories, adaptation rules will propose appropriate adaptation strategies.

A promising rule language for OWL ontologies is SWRL. However, first prototypical implementations already show that SWRL might not be sufficient, as we also need production rules in order to fire adaptation actions. Thus, a further investigation of SWRL and other rule languages is necessary.

In order to tackle this challenge of dealing with the extraction, correlation, categorization and adaptation rules, we developed as an initial solution a server-side component that translates SWRL rules into JavaScript control statements, and into JavaScript objects or arrays, respectively. As JavaScript can be executed easily by any Web browser, the extraction, correlation, categorization, as well as the adaptation rules can now be executed at the client-side to guarantee instant portal adaptation.

To reduce the complexity of the client-side JavaScript rules, a two-stage rule-handling approach will be introduced. Simple rules are transformed into JavaScript by a server-side component, and are executed directly on the client-side. Complex rules,

---

[9] With externalization we mean the transformation of implicit knowledge of an ontology into explicit knowledge by reasoning.

with no time critical consequences for UI adaptation, remain on the server-side, in order to utilize the powerful features of ontology processing and reasoning, as well as rule execution frameworks at the server-side. Server-side rules are triggered and their results are evaluated by JavaScript callback functions encoded into the Ajax page, leveraging the XMLHttpRequest object for asynchronous communication.

5.1.5 Portal Specifics, Dynamic Web Pages and Self-adaptivity

So far, only static Web pages using the Ajax technology were examined. The open question is what effort must be made in order to support complex portals with dynamic Web pages. Another challenging area is portal self-adaptivity to achieve continual improvement. Some relevant open research questions are: Which collected data about the user and the user behavior should be recorded for the purpose of long-term adaptation? How can this data be used to check the effectiveness of adaptation rules and discover new adaptation needs?

6   Related Work

Related work to our approach includes standard models of adaptive hypermedia like [12], recent semantic-based personalization systems [13], [14] and Ajax-based personalized systems [15].

 Comparing our work with standard models for adaptive hypermedia systems like e.g. AHAM [12], we observe that they use several models like conceptual, navigational, adaptational, teacher and learner models. Compared to our approach, these models correspond to ontologies presented in Section 4, but miss their formal representation. Moreover, we express adaptation functionalities as encapsulated and reusable OWL-DL rules, while the adaptation model in AHA uses a rule based language encoded into XML.

 The Personal Reader [13] provides a framework for designing, implementing and maintaining Web content readers, which provide personalized enrichment of Web content for each individual user. The adaptive local context of a learning resource is generated by applying methods from adaptive educational hypermedia in a semantic Web setting. Similarly [14] focuses on content adaptation, or, more precisely, on personalizing the presentation of hypermedia content to the user. However, both approaches do not focus on the on-line discovery of the profile of the current user that is one of the main features of our approach. Another difference would be the self-adaptivity.

 Recently some work has been done regarding the usage of Ajax for personalization, like [15]. However, our approach resolves the problem of the syntactical processing of the user's click stream by combines Ajax with semantic technologies. Indeed, our approach enables semantic interpretation of the user's behavior in a portal.

7   Conclusion and Outlook

This paper presented an approach to achieving user-adaptivity that combines Ajax with Semantic Web technologies. Three major advantages to this approach were

discussed. First, with this approach, user-adaptation can be more accurate and more appropriate. Better accuracy is possible because Ajax enables finer-grained tracking of user behavior, and semantic annotation enables meaningful interpretation of this more accurate record of behavior. More appropriate adaptation is enabled by the richer set of options for adaptation offered by Ajax, which makes a Web application more like a desktop application.

A second advantage of the approach is that domain experts can inspect, understand and modify the adaptation logic, since it is expressed in the form of explicit rules. Moreover, hierarchical organization of rules makes them easier to maintain.

A third major advantage is that adaptation rules can be shared by groups, like public administrations, that agree to use the same ontologies, since the rules are formulated using concepts from ontologies.

Currently we are working on solving the open research challenges. We are implementing our ideas prototypically in an Ajax-enabled JBoss Portal[10]. After finishing this, we will first evaluate our prototype, and afterwards implement parts of it in real e-Government portals.

Acknowledgements

References

1. Jameson, A.: Adaptive Interfaces and Agents. Chapter in Jacko, J. A., Sears, A. (eds.): Human-Computer Interaction Handbook (2nd ed.). Erlbaum, Mahwah, New Jersey (2006)
2. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. In User Modelling and User-Adapted Interaction, 6(2-3):87–129, July 1996
3. Mobasher, B., Cooley, Srivastava, J.: Automatic Personalization Based on Web Usage Mining. Communication of ACM, 43(8):142–151, August 2000
4. Garrett, J. J.: Ajax: A New Approach to Web Applications. http://adaptivepath. com/ publications/essays/archives/000385.php, February 2005, retrieved on 2006-11-13
5. Stojanovic, L., et al., D2: Framework for self-adaptive e-Government. Available as Deliverable D2, EU/IST Project FIT, http://www.fit-project.org/index.htm, 2006
6. Thomas, S.M., et al., D4: Identification of typical problems in e-Government portals. Available as Deliverable D4, EU/IST Project FIT, http://www.fit-project.org/index.htm, 2006
7. Heckmann, D., Schwartz, T., Brandherm. B., Schmitz, M., von Wilamowitz-Moellendorf, M.: GUMO - the General User Model Ontology. In Proceedings of the 10th International Conference on User Modeling, Edinburgh, Scotland, Jun 2005
8. Palmer, S.: RDF in HTML: approaches. http://infomesh.net/2002/rdfinhtml/, 2002, retrieved on 2006-12-28.

---

[10] JBoss Portal: http://www.jboss.org/products/jbossportal

9. Kalyanpur, A. et al.: Automatic Mapping of OWL Ontologies into Java. In Proceedings of the 16th International Conference of Software Engineering and Knowledge Engineering, pages 98–103, 2004

10. Internet Engineering Task Force: The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627, 2006, http://www.ietf.org/rfc/rfc4627.txt, retrieved on 2006-12-28.

11. Euler Proof Mechanism: http://eulersharp.sourceforge.net/, retrieved on 2006-12-28.

12. Bra, P. D., Aerts, A., Smits, D., Stash, N.: AHA! version 2.0: More adaptation flexibility for authors. In Proceedings of the AACE ELearn'2002 conference, pages 240-246, Oct. 2002

13. Dolog, P., Henze, N., Nejdl, W., Sintek, M.: The Personal Reader: Personalizing and Enriching Learning Resources using Semantic Web Technologies. AH 2004: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, 2004

14. Frasincar, F., Houben, G.: Hypermedia presentation adaptation on the semantic Web. In Proccedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002), Malaga, Spain, 2002

15. Köberl, K.: Erfassen von Benutzerkontextinformationen mit Ajax. MSc thesis, Technische Universitaet Graz, 2006