Client-Side Adaptation: An Approach Based in Reutilization Using Transversal Models

Sergio Firmenich[1], Silvia Gordillo[1,2], Gustavo Rossi[1], and Marco Winckler[3]

[1] LIFIA, Facultad de Informática,
Universidad Nacional de La Plata and Conicet Argentina
[2] CiCPBA
{sergio.firmenich,gordillo,Gustavo}@lifia.info.unlp.edu.ar
[3] IRIT, Université Paul Sabatier, France
winckler@irit.fr

Abstract. Improving navigability in Web applications is essential for applications success. Our research aims to improve the user experience by applying novel techniques such as concern-sensitive navigation. Concern-sensitive navigation allows enriching Web pages with content related to the context in which they are accessed. In previous works, we showed how this technique can be applied during the development process; at present we are working in a client-side adaptation approach to apply concern-sensitive navigation to existing applications. This approach is open to other adaptation kinds which are illustrated in this position paper. We also outline a set of tools we are developing to simplify the process of adaptation.

Keywords: Concern-sensitive navigation, User experience, Client-side adaptation.

1 Introduction and Motivation

The Web is in constant evolution. Social sites like Facebook or LinkedIn, interactive encyclopedias like Wikipedia or multimedia servers such as Flickr have changed our way to access information. As a final result of this evolution, we could say that the user experience has been improved in a remarkable way, though we still need to struggle to make popular applications more "adaptable". A constraint in Web engineering approaches to adaptation is that the user's information which is used to adapt an application is many times restricted to a single application's boundary. In this way, very often the user is not considered as accessing many applications at the same time, and this has as a result that the user's experience is not accordingly improved.

In previous papers we presented concern-sensitive navigation (CSN) [2] as a conceptual tool to improve navigation by enriching or adapting a link's target node according to the user's concern in the link's source node. In this way, when the user navigates from node A to node B, his concern in A is used to adapt B. On the other hand, when the user navigates from C to B, B is adapted using the concern in C; therefore the contents and links in B slightly change according to the navigation source. If both nodes A and B belong to the same application, we say that it is an *intra-application* adaptation, otherwise, it is an *inter-application* adaptation. We are interested in client-side (CS) adaptation of existing applications; this means that the adaptation is not only performed in the client, but that adaptation engineers might be third parties (potential users instead of developers of the adapted application).

This position paper presents a novel realization of CS adaptations. First, we focus on CSN adaptations, achieving reuse of adaptation code by using transversal models between different applications. We outlined our tool to ease the development process.

The structure of the paper is as follows. In Section 2, we explain how we build CSN at the CS, and how these adaptations can be reused. In section 3 we show new types of adaptations that can be realized at the CS. In section 4 we conclude and present some ideas for further research.

2 Concern-Sensitive Navigation: A Client-Side Approach

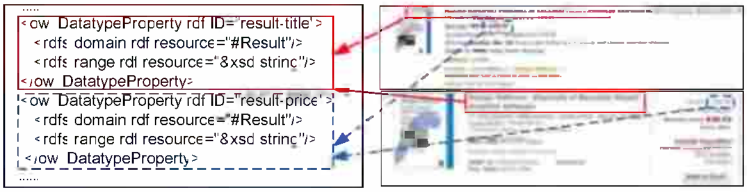
In [2] we presented a novel realization of CSN, by performing the adaptation in the CS of existing applications, using scripts (not necessary developed by the designers of the adapted application). The adaptation process is facilitated by a weaving engine; in [2] we used the Grease Monkey (GM)[1] weaving engine.

To make these scripts stable we extensively used the concept of Modding Interface (MI) [1]. A MI is a high-level application model aimed to create an abstract layer over the DOM. A Modding Interface contains concepts and concepts' properties to abstract the real DOM elements. Two files are used to define a Modding Interface: an OWL to define the model and a XSLT for mapping elements between model and the DOM. In this way, the scripts access DOM elements indirectly by manipulating concept instances. In [1] it is the application developers' responsibility to publish the MI, as well as to maintain the matching between models and underlying DOM elements. Our approach uses this MI as an abstraction mechanism to achieve reusable adaptations and easier maintenance. Note that a developer who wants to create an adaptation must define the MI for the target application. In [2] we presented our MI development tool; later we show how we generalized the idea and propose improvements for the tool.

2.1 Applications Families and Reusable Adaptations

In our research we have seen that many applications may share similar MI models, therefore we introduced the concept of application family. An application family is a set of applications sharing several model's concepts. Since adaptation scripts access the model's concepts and not the DOM, we can reuse an adaptation across several applications (in the same family). In Figure 1 we show an application family example. Note that the same model is shared between Amazon.com and BarnesAndNoble.com. At left, we show a MI model containing the concept "Result" with two properties: the resulting product's name, and the product's price. At right, we show how each property is mapped to different DOM's elements in each application, consequently an adaptation accessing to "Result" instances could be reused in both applications.

[1] GraseMonkey: https://addons.mozilla.org/en-US/firefox/addon/748

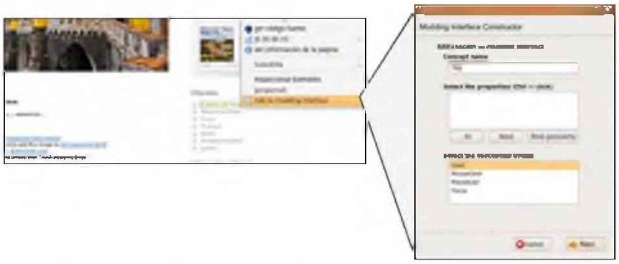


```
.....
<owl DatatypeProperty rdf ID="result-title">
  <rdfs domain rdf resource="#Result"/>
  <rdfs range rdf resource="&xsd string"/>
</owl DatatypeProperty>
<owl DatatypeProperty rdf ID="result-price">
  <rdfs domain rdf resource="#Result"/>
  <rdfs range rdf resource="&xsd string"/>
</owl DatatypeProperty>
.....
```

Fig. 1. Search results in different e-commerce applications

2.2 Developing CSN Adaptations

Developing a CSN adaptation using this approach is feasible but can be difficult even for an experienced developer. The first step to solve this problem is to raise the abstraction level in MI construction process. As part of this research we have developed a tool, which helps developers to define MI's concepts by highlighting DOM elements. In this way, developers are relieved from low-level technical issues. The tool also supports the definition of a CSN adaptation by selecting the source and target application, and choosing what information from the source node will be used.



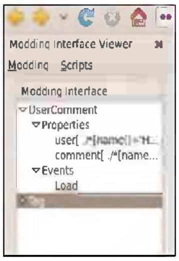


Fig. 2.1. Adding concepts to the Modding Interface **Fig. 2.2.** MI Viewer

In Figure 2 we show the first step in this process, Figure 2.1 shows how the MI's concepts are defined, while in Figure 2.2 the concepts already defined are presented.

When the concepts are defined, our tool supports the CSN adaptations construction by selecting source and target nodes. In this process, the adaptation developer must specify which concept instances will be stored to use them in the target application.

3 Generalizing Client-Side Adaptation

CSN adaptations are a useful way to improve the user experience. We have done experiments with users, and they have appreciated the improvements achieved with CSN enhancements. In broader terms, we could say that CS adaptation is gaining followers. Users are really interested in adapting existing Web applications. GM is really used around the world, some scripts have been downloaded million of times. However, not all possible CS adaptations are related with the navigational path followed by the user. Moreover, the typical GM scripts under utilizes the CS adaptations possibilities. With this in mind, we present our classification of kinds of CS adaptations:

- Static adaptations: this adaptation kind will be applied every time that the application's page loads. This is a very important adaptation because with them we can achieve, for example, an improvement in the application's accessibility. An adaptation could hide instances of concepts or reorganize UI elements to do more efficient the reader's work.
- CSN adaptations: adaptations which improves the user's experience by adapting applications considering the user's concern. Several examples of this adaptation kind can be found in [2].
- Navigational history adaptations: adaptations based in the user's navigation. This kind of adaptation was initially described in History-Based [3]. However, when this technique is used dynamically at the CS, the possibilities increase considerably since we can use the navigation history through many applications and not only into an application's scope.
- Task-aware adaptations: adaptations which take into account the user's current activity, e.g. while he works with several applications. For example, in Figure 3, we can see a common Web usage scenario enriched with task-aware adaptation. In Figure 3.1 the user is reading a Wikipedia's article. After that, the user opens a new tab and searches for "adaptivity" in Google. At the end, in Figure 3.3, when the user returns to Wikipedia, the page has been adapted by adding a link to the Wikipedia's article about "adaptivity".

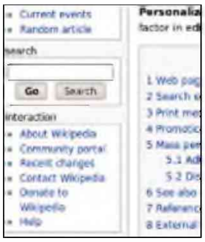


Fig. 3.1. User is reading a Wikipedia's article

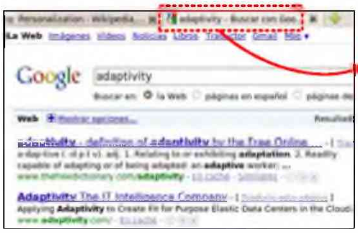


Fig. 3.2. User opens another tab to search for "*Adaptivity*" in Google

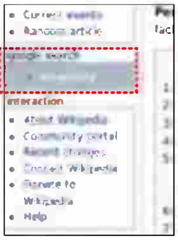


Fig. 3.3. Wikipedia is adapted by a context-aware adaptation

Note that this kind of adaptations can't be achieved easily using well-known Web development methodologies. In comparison with CS adaptations, their adaptation approaches have two disadvantages: 1) the adaptations are restricted by application developers, 2) users are not taken into account as users of other applications.

Security issues are important here; with our CS approach, we can know all about the user's activity. Obviously, this must be done in a secure way and we must be sure that the information shared by applications is only used in the adaptation code. A way to do it is using credentials when an adaption does a request which require privilegies. Performance is important too. In our tests we didn't perceive a lost in this, after all, and for now, the adaptations are only JavaScript code which is very usual nowadays.

3.1 Improving Support for Client-Side Adaptations

The adaptation kinds presented above imply new features not contemplated in our previous work and which GM can't offer. Because GM scripts don't have privileges, they can't know when another page is loaded, and they are not able to share data among different applications. On other hand, when GM is used, it is the developer who must coordinate different tasks: create and export models, create and export adaptations scripts, install scripts in GM and store the MI in an accessible site. Therefore, we choose to develop a tool which will work both with GM and a GM script which will be the connection between the adaptations and our tool. We aim to improve this process by making the tool responsible of: (1) constructing and storing transversal models, (2) responding to adaptation requests about context and data.

The following list enumerates new features of the extended tool we are building:

- Features to make the information accessible in different context: saving and restoring contextual data is very important when the adaptation is developed in an inter-application way.
- Support for task-aware adaptations: these adaptations will strongly depend on context changes; for example, the user is accessing a specific application, or even with more granularity, the user is accessing a specific functionality in an application. The tool must have an API to register changes in context elements and create corresponding events for these changes. In this way, the adaptations that are pending of these changes can be performed when the event occurs.

4 Concluding Remarks and Further Works

Providing Web users with appropriate information and functionalities according with his actual concern or preferences is an interesting research topic. The adaptation approaches of mature Web methodologies have usually two main restrictions: (1) adaptations are defined by application developers; (2) generally, adaptations are restricted to a single application, so users are not considered as users of other applications. Our approach solves these restrictions by enabling users to adapt applications in a reusable and straightforward way. As a proof of concept, we have outlined our client-side realization of CSN presented in [2]. We are now working in a broader classification of client-side adaptation kinds and we are extending our tool to support them.

References

1. Diaz, O., Arellano, C., Iturrioz, J.: Layman tuning of websites: facing change resilience. In: Proceeding of WWW 2008 Conference, Beijing, pp. 127–128. ACM, New York (2008)
2. Firmenich, S., Rossi, G., Urbieta, M., Gordillo, S., Challiol, C., Nanard, J., Nanard, M., Araujo, J.: Engineering Concern-Sensitive Navigation Structures. In: Concepts, tools and examples, JWE (accepted 2010)
3. Brusilovsky, P.: Adaptive Navigation Support. In: The Adaptive Web: Methods and Strategies of Web Personalization, pp. 263–290. Springer, Heidelberg (2007)