A Tool Support for Web Applications Adaptation using Navigation History

Sergio Firmenich[1], Marco Winckler[2], Gustavo Rossi[1]
[1]LIFIA, Facultad de Informática, Universidad Nacional de La Plata and Conicet Argentina
{sergio.firmenich, gustavo}@lifia.info.unlp.edu.ar
[2]ICS-IRIT, Université Paul Sabatier, France
winckler@irit.fr

Abstract. Currently the Web is a platform for performing complex tasks which involve dealing with different Web applications. However users still have to face these tasks in a handcrafted way. In this paper we present a novel approach that combines concern-sensitive adaptation and navigation history to improve the user experience while performing a task. We have developed some simple though powerful tools for applying this approach to some typical tasks such as trip planning and house rental. We illustrate the paper with a simple though realistic case study and compare our work with others in the same field.

1 Introduction

One of the interesting facets of Web evolution is the end-users interaction with Web content [2]. At first, users could only browse through contents provided by the web site. Later, users could actively contribute with content by using tools (e.g. wikis) embedded into the web site. Recent technologies provide users with tools for changing the way Web content is displayed. For example, visual Mashups [7, 17], support the integration of content hosted by diverse web sites and Greasemonkey scripts [10] allow users to change third part web applications by adding content and/or controls (e.g. to highlight search results in Amazon.com which refer to Kindle).

These tools follow the concept of *Web augmentation* [4] by extending what users can do with Web content. However, they provided limited support to tasks requiring navigation of many Web sites. For example, a user planning a holiday trip to Paris might ultimately visit several web sites such as *expedia.com* for flights, *booking.com* for hotels, *wikipedia.org* for general information about the city and *parisinfo.fr* for points of interest, current events or expositions in Paris... From the users' point of view, the navigation of all these web sites is part of the same task. The existing augmentation techniques are of little help in this case. GreaseMonkey scripts can adapt the content on a specific Web site but it will require much effort to make it generic enough to integrate information provided by different applications. Mashup for *expedia.com* will not necessarily integrate information from users' preferred web sites (e.g. *airfrance.fr*, *venere.com...*). If Web sites provide public APIs, Mashups can be extended but it does not prevent users to learn how to do it beforehand. Quite often, users' tasks are associated with opportunistic navigation on different Web sites, which is difficult to predict [14]. In this context, effective Web augmentation should overcome to main barrier: i) to take into account the different web sites visited by users; and ii) to adapt target web sites accordingly to unpredictable user needs.

This paper investigates the use of a tool support for creating flexible, light-weight and effective adaptations to support users' tasks during the navigation of diverse Web applications. Our goal is to support users' tasks by keeping his actual concern (and related data) persistent through applications. For example, allow the reuse of dates provided on *expedia.com* for booking a flight to search hotels at *booking.com*. Another example, allow the inclusion of new links letting users to navigate from *parisinfo.fr* to related articles at *wikipedia.com* whenever he needs further explanation about a topic. Hereafter we present the tools we have developed to solve this kind of problem. Section 2 shows a view at glance of our approach for Client-Side Adaptation (CSA) of Web applications. Section 3 presents the tools we have developed. Section 4 presents how we have validated our approach with end-users. Section 5 discusses related work and section 6 presents conclusions and future work.

2 The underlying approach

Our approach is based on concept of concern-sensitive navigation. We say that a Web application (or specifically a Web page) is concern-sensitive (CS) when its contents, operations and outgoing navigation links can change (or adapt) to follow the actual situation (concern) in which it is accessed [9]. Concern-sensitive navigation is different from context-aware navigation, where other contextual parameters (location, time, preferences) are considered. The main difference is that concern-sensitive is driven by a specific user goal, quite often volatile and difficult to generalize.

Figure 1 shows an example of concern-sensitive navigation across two applications: Google Maps (as the source of navigation) and Wikipedia (as the target). The left-side of Figure 1 displays Wikipedia links in the map of Paris; once selected, these links trigger the exhibition at the right-side of Figure 1 the corresponding map and a set of links to those Wikipedia articles in the surroundings of the current one.

Fig. 1. Inter-application CSN between Google Maps and Wikipedia

We assume that concern-sensitive navigation simplifies the user's tasks by providing him sensitive information or options according to his current needs. For that purpose, the adaptation of a page *P* requires that: (a) the actual user's navigation concern (i.e. pages previously navigated, e.g. Google maps), (b) the set of relevant information from previously visited pages that are needed for adaptation (e.g. the current map), and (c) the capacity for enriching *P* with contents or links related with (a) and (b) by intervening in *P*'s DOM.

3 Tool support

The tool presented hereafter was implemented as a Firefox plug-in that provides components called *augmenters*. In our approach, users need to run adaptations (using *augmenters*) on different Web sites they are visiting to perform their tasks. This implies that users must to collect information during the Web sites navigation. A set of tools called *DataCollectors* work as a memory for user data. The other *augmenters* will then use information stored by *DataCollectors* to perform the adaptations.

3.1 Data collector and other augmenters

Two types of *DataCollectors* have been implemented: *Untyped Pocket* which implements a *copy* & *paste* behaviour for simple text (e.g. "*Paris*"); and *Typed Pocket* which allows users to label data (e.g. "*Paris*" is "*City*"). In both cases, selected information is placed into a temporary memory called *Pocket*. Data collection is supported via the contextual menu options "*put it into pocket"* (i.e. *Typed Pocket*) and *"put it into volatile pocket"* (i.e. *Untyped Pocket*). Figure 2.a illustrates the collection of a piece of information (i.e. *"Place de la Concorde"*) using the option *"put it into pocket"* which has been labelled "*PointOfInterest"*. Collected information become available into the Pocket as shown by Figure 2.b (yellow box at the upper-left side).

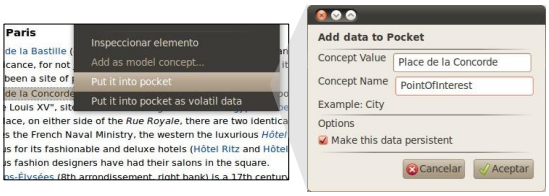

Fig 2.a. Information collection from Wikipedia using a *DataCollector*.

Fig 2.b. Resulting adaptation for the *Pocket* memory.

The *Pocket* is more than just a simple post-it as the information it stores can be used by *augmenters* to adapt Web sites. Currently available *augmenters* include:

- *Highlight*: it colors the occurrences of the data received by parameter.
- *CopyIntoInput*: it pastes the value received as parameter into an input form field. Once executed, *CoopyIntoInput* adds a listener to the click event which is removed after the first time that the target is an input.
- *WikiLinkConvertion*: it creates links to *wikipedia.com* pages using as input any occurrences values received as parameter. For example if the parameters is "*Paris*" then the link would be to the Wikipedia article about Paris.

An example of how the *augmenters* work is provided by Figure 2. At Figure 2.a the user has used a *DataCollector* augmenter to capture information at the web site *wikipedia.org*. When the user opens the *Google Maps* web site (Figure 2.b) the information collected in previous Web site (i.e. wikipedia.org) is available at the *Pocket* (the yellow box at left in Figure 2.b). Now, the contextual menu at Figure 2.b offer new augmenters based on the previous collected information, *CopyIntoInput*, *Highlight* and *WikilinkConvertion*. However which augmenters are available depend on the current site because augmenters can be generic enough to be applied to any page (e.g. *highlight*). These augmenters are illustrated by the scenarios below.

3.2 Scenario for performing adaptations

The scenario presented in this section aims at fulfilling users' needs described at section 1. While booking flights to Paris, the user collects data (cf. Figure 3.a) which will help him in the next steps to find a hotel. Relevant information ~~in~~ is labelled by the user as *departDate*, *arriveDate* and *destination*. Figure 3.b shows how the form field *destination* is filled in with the information previously collected. This scenario is executed once the user reaches the page *booking.com* (either by following a link or entering a new URL). Notice that the scenario can be instantiated because the information needed is available into de *Pocket*. So far, automatic form filling can only be done for a particular Web application (in our case for *booking.com*) but this feature can be extended using tools like Carbon [1]. This use of concern-sensitive information improves the user experience by allowing him to "transport" critical data among Web applications and use these data to adapt them.

Data Collection in Expedia.com

Fig 3.a. Information extraction from *expedia.com*

Form filling in Booking.comwith data collected previously

Fig 3.b. Form filling in *booking.com* with information collected in previous web sites

Figure 4 exemplifies the use of the augmenter *WikiLinkConvertion*. In this scenario, we assume the user has previously visited the web site *Wikipedia.com* and selected his *PointOfInterest*. Then the user opens the web site *Parisinfo.com*. It is worthy noting that the *Pocket* features all information previously collected at the web site *Wikipedia.com*. Now, the user right clicks over *PointOfInterest*, a contextual menu offer the option "*Convert to Wiki Link*" that, once selected by the user, will run the augmenter ~~*WikiLingConvertion*~~*WikiLinkConvertion*, thus creating new links on the current page of the web site *ParisInfo.com* allowing the navigation to the web site *Wikipedia.com* to the corresponding page associate to the information *PointOfInterest*. The augmenter *Highlight* works in a similar way but it changes the colour of the text instead of created links. Due to space reasons, the augmenter *Highlight* is not illustrated here.

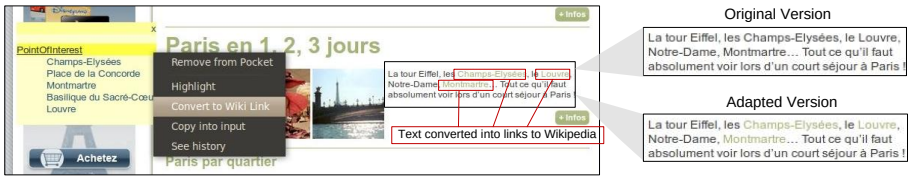

Fig 4. Text plain converted into link to add personal navigation.

4 Evaluation of the approach

To validate our approach and actual usage of the tools, we have conducted a usability study with end-users. The goal of this evaluation was to investigate if CSA is usable for solving common tasks whilst navigating web. The adaptations investigated in this study explored the following augmenters: *Highlight* for changing color of important information, *WikiLinkConvertion* for creating new links to Wikipedia, *DataCollector* for recording information for later usage, and *CopyIntoInput* for automating filling in forms.

The study was run with 11 participants (6 males and 5 females, aged from 23 to 46 years old). All participants were experienced Web users (i.e. > 5 years using the web) that browse the web as part of their daily activities (in average 4,1 hours of navigation on the web per day, SD=2,4 h). We have focused on experienced users because we assume that they are more likely to formulate special needs for adapting Web pages than novices with the Web. Participants were asked to fill out a pre-questionnaire, following they were introduced to the system (i.e. 2-5 minutes training) and asked to conduct five tasks at their workplace, followed by a final interview and a System Usability Scale questionnaire (i.e. SUS, [5]). The SUS has been as a complement to user observation as it is widely used in comparative usability assessments in the industry.

The five user's tasks concern a trip planning to Paris for visiting the art exhibition "*De Stijl et Mondrian*". The initial setup was a Web page advertising that art exhibition. The tasks were: 1) to collect required data for planning the trip including dates, keywords and locations; 2) to book a hotel in Paris near the exposition for the week-end of February 18th 2011; 3) to select a hotel in the neighborhood of "*Les Marais*"; 4) to record information about the hotel; 5) to create a relationship between the actual Web of the exhibition and the Web site *wikipedia.com*.

Usability was measured in terms of time to accomplish tasks, number of tasks performed successfully, and user satisfaction (via a questionnaire). Users were also asked to rate every task from 1 to 5 (from very easy to very difficult).

All participants used the tools presented during the training period to perform the tasks. Users completed the tasks in approximately 37 minutes (SD=9 minutes). The results show that, generally, participants appreciate the concept of CSA and the tool support. In the pre-questionnaire, when asked if they would like to modify the web pages they visit, 2 of 11 participants said no because "it could be very time consuming". Notwithstanding, all participants said that our tools for client-side adaptation are useful and that they are willing to use it in the future. Adaption across different web site was described as "natural" by 7 participants and a "real need" by 5 of them. The tool *DataCollector* was the most successful applied by all participants; it was considered the very useful and a "good substitute for post-its". However, success rate varied according the augmenter employed: *CopyIntoInput* was considered very easy to use by participants and employed successfully by 10 of them (90,9%). The augmenter ~~*highlight*~~ *Highlight* (72% of success rate, 8 participants) was considered easy to use but 5 users blamed it because they only can be applied to the exact word previously selected and users cannot choose the color and/or the policy used to highlight different pieces of information. Participants were very impressed by the augmenter allowing links to Wikipedia from concepts, i.e. *WikiLinkConvertion*;

despite the fact it was considered extremely useful, the success rate with this augmenter was the lowest in the study, i.e. 18%, due to two main issues: the fact that links can only be created from typed information; lack of visual feedback (i.e. an icon) indicating where that action was possible. Nine participants (81,8%) mentioned that using the augmenters improve their performance with tasks, one user said it could be faster without the augmenters and the other one didn't see any difference. This user perception has been confirmed by the time recorded during task execution using augmenters *WikiLinkConvertion* and *CopyIntoInput*.

This study also revealed some usability problems that motivate further development in the tool. For example, users requested to have a visual indicator allowing them to distinguish where augmenters have been applied (ex. links on the web site x links created with the augmenter *WikiLinkConvertion*). Users intuitively tried to activate some of the augmenters using *Drag & Drop* which is an indicator for further research of more natural interaction with *augmenters*. The most frequent suggestions for new augmenters include "automatic filling forms", "create links to other web sites than Wikipedia", and "automatic highlight at the web page of information previously collected". This positive analysis is confirmed by a SUS score of 84,9 points (SD = 5,5), which is a good indicator of general usability of the system.

5 Related work

The field of Web applications adaptation is broad; therefore, for the sake of conciseness we will concentrate on those research works which are close to our intent. The interested reader can find more material on the general subject in [6]. As stated in the introduction we can identify two coarse-grained approaches for end-user development in Web applications: i) mashing up contents or services in a new application and ii) adapting the augmented application, generally by running adaptation scripts in the client side.

Mashups are an interesting alternative for final users to combine existing resources and services in a new specialized application. Visual and intuitive tools such as [16] simplify the development of these applications. Since most Web applications do not provide Web services to access their functionality or information, [11] proposes a novel approach to integrate contents of third party applications by describing and extracting these contents at the client side and to use these contents later by generating virtual Web services that allow accessing them.

The second alternative to build support for their tasks is Web augmentation [4], where applications are adapted instead of "integrated" in a new one. This approach, as indicated in [2] is very popular since it is an excellent vehicle for crowdsourcing. Many popular Web applications such as Gmail have incorporated some of these user-programmed adaptations into their applications like the mail delete button (See http://userscripts.org/scripts/show/1345). GreaseMonkey [10] is the most popular tool for Web augmentation, and its scripts are written in JavaScript. The problem with these scripts is their dependence on the Document Object Model (DOM) used to organize the Web page; if the DOM changes the script can stop working. In [8] the authors propose a way to make GreaseMonkey scripts more robust, by using a conceptual layer (provided by the Web application developer) over the DOM.

While we share the philosophy behind these works, we believe that it is necessary to go a step further in the kind of supported adaptations. In [9] we showed how to use the actual user concern (expressed in his navigational history) as an additional parameter to adapt the target application. By using the scripting interface we managed to make the process more modular and by defining adaptations for application families (e.g. social networks) we improved the reuse of adaptation scripts. In the following sections we show how to broaden the approach allowing end users to select which concrete information can be used to perform the adaptation, therefore improving the support for his task and providing support for building more complex adaptations. Some tasks are repeated several times and then users make the same process each time. This problem has been tackled in [3,13] with the CoScripter tool. CoScripter is a Firefox Plug-in which allows users to record his interactions with a Web Site, and then, they can repeat the process automatically later. The approach is a bit flexible, for example, the whole process can be repeated with other information in form inputs that those used in the original recorded execution, but always using the same fixed Web sites. In this way, CoScripter is not useful when users need to change slightly the process, for example by changing which is the target Web application. However, both CoScripter's goals and our approach's goals are different, because with CoScripter Web applications are not adapted (not further that fill forms with values) and volatile requirements are not contemplated.

Other near work is [15], which is other Firefox plug-in addressed to improve user experience by empowering his browser with commands with different goals. With MozillaUbiquity users execute commands (developed by themselves) for specific operation, for example to publish some text from the current Web page in a social network. Anyway, these commands are executed under user demand, and adaptations are not made automatically. Although MozillaUbiquity makes short the distance between two distinct Web Applications, to move information from one of them to another is not fully exploited.

6 Conclusions and future work

In this work we have presented a novel approach of CSA driven by the integration of information through the navigation of several web applications. The underlying idea is to support concern-sensitive adaptations on Client-Side in order to improve users experience while they are doing tasks in many web sites. The tools presented in this paper are based on a framework (not presented here) that allows the development of augmenters beyond those presented. The present study allows us to investigate new strategies of *Web augmentation* with end users. A user testing experiment was performed to demonstrate the feasibility of the strategy of CSA. Despite some usability problems found with the actual tools, the preliminary results show that approach is very promising and can indeed help users to solve complex tasks that require information exchange between different web sites. We observe an increasing interest in the development of tools that can make users more active with respect the way they access content provided by Web applications. Notwithstanding, there is a few empirical studies that investigate the user experience of user driven CSA, in particular when adapting third-party web sites. As far as the adaptation across different web sites is a concern, we haven't found in the literature any other tool

allowing users to freely adapt web pages accordingly to previously navigation of web pages. The results presented here remains preliminary but it provides many insights for discussions, including: users' needs for performing complex tasks among different web sites, development of new strategies of end-user programming of the web, impact of user-driven adaptation of web site. Our next steps include the investigation of CSA beyond a single user session, for example, when navigation of different web sites occurs in a long period of time.

References

1. Araújo, S., Gao, Q., Leonardi, E., Houben, G. Carbon: Domain-Independent Automatic Web Form Filling. In: Proc. of ICWE2010, (Vienna, 2010), Springer.
2. Arellano, C., Díaz, O., Iturrioz, J. Script Programmers as Value Co-creators. In Proceeding of ICWE Workshops, (Vienna, 2010), Springer, 417-420.
3. Bogart, C., Burnett, M., Cypher, A., Scaffidi, C. End-user programming in the wild: A field study of CoScripter scripts. In.: Proceeding of EEE Symposium on Visual Languages and Human-Centric Computing, (Germany, 2008), 39-46.
4. Bouvin, N. O.. Unifying Strategies for Web Augmentation. In: Proc. of the 10th ACM Conference on Hypertext and Hypermedia, 1999.
5. Brooke, J. (1996) "SUS: a 'quick and dirty' usability scale". In: Usability Evaluation in Industry. London: Taylor and Francis.
6. Brusilovsky, P. Adaptive Navigation Support. in The Adaptive Web: Methods and Strategies of Web Personalization, Springer, 2007, 263-290.
7. Daniel, F., Casati, F., Soi, S., Fox, J., Zancarli, D., Shan, M. Hosted Universal Integration on the Web: The mashArt Platform. In Proceeding of ICSOC/ServiceWave (Stockholm, 2009), 647-648.
8. Diaz, O., Arellano, C., Iturrioz, J. Layman tuning of websites: facing change resilience. In.: Proc. of WWW2008 Conference, (Beijing, 2008), 127-1128.
9. Firmenich, S., Rossi, G., Urbieta, M., Gordillo, S., Challiol, C., Nanard, J., Nanard, M., Araujo, J. Engineering Concern-Sensitive Navigation Structures. Concepts, tools and examples. JWE 2010, 157-185.
10. Greasemonkey, At: http://www.greasespot.net/ (last visit on Jun. 7, 2011).
11. Han, H., Tokuda, T. A Method for Integration of Web Applications Based on Information Extraction. in Proceeding of ICWE (New York, 2008), Springer.
12. Ikeda, S., Nagamine, T., and Kamada, T. Application framework with demand-driven mashup for selective browsing. In: Proc. of the 10th Int. Conf. on Information Integration and Web-based Applications & Services (iiWAS 2008). ACM, New York, NY, USA, 33-40.
13. Leshed, G., Haber, E., Matthews, T., Lau, T. CoScripter: automating & sharing how-to knowledge in the enterprise. In.: Proc. of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (Italy, 2008), 1719-1728.
14. Miller, C. S., and Remington, R. W. Modeling an Opportunistic Strategy for Information Navigation. In: 23th Conf. of the Cognitive Science Society, 2001.
15. MozillaUbiquity, At: http://mozillalabs.com/ubiquity/ (last visit on Jun. 7, 2011)
16. Yu, J., Benatallah, B., Casati, F., and Daniel, F. Understanding Mashup Development. IEEE Internet Computing, 12:44–52, 2008.
17. Wong, J. and Hong, J. I. Making Mashups wit Marmite: Towards End-User Programming for the Web. ACM, City, 2007.