Trade-offs in user interface adaptation using Web augmentation techniques: towards a negotiated approach

Diego Firmenich[1,3], Sergio Firmenich[1,2], Gustavo Rossi[1,2], Marco Winckler[4], Damiano Distante[5]

[1]LIFIA, Facultad de Informática, Universidad Nacional de La Plata
[2] CONICET, Argentina
{sergio.firmenich, gustavo}@lifia.info.unlp.edu.ar

[3] DIT, Facultad de Ingeniería, Universidad Nacional de la Patagonia San Juan Bosco
dfirmenich@tw.unp.edu.ar

[4] ICS-IRIT, University of Toulouse 3, 118, route de Narbonne
31062 Toulouse Cedex 9, France
winckler@irit.fr

[5] COST – Research Centre on Software Technology Department of Engineering, University of Sannio, Italy
distante@unisannio.it

Abstract. This paper is concerned by social and collaborative aspects of Web augmentation tools which are aimed at extending and adapting Web sites features. It is clear that Web augmentation techniques have an impact on many actors that interact with the Web site, which ultimately include the owners of Web sites (and the corresponding development team), developers of external scripts and the end users. Because the Web site can be modified by external scripts, one of the risks for Web site's owners is to lose control about the way Web site contents is delivered to end-users. To prevent this situation, Web site owners might be tempted to regularly modify the DOM structure of Web pages thus making harder to execution external scripts. Nonetheless, communities of developers dedicated to coding Web augmentation scripts are increasing and many scripts they produce are useful and can affect positively the user experience with Web sites. However, end-users might have special needs that are not yet covered by original Web sites, nor supported by existing Web augmentation scripts. In this paper we analyze the trade-offs of the introduction of Web augmentation scripts. In order to mitigate some of negative effects, such as the loose of control, we propose an approach based on negotiation and coordination between actors involved in the process. As far as collaboration is an important aspect of our approach, we present in this paper a set of tools we have developed to facilitate the integration of scripts and to foster the dissemination of Web augmentation scripts for the benefit of all actors involved.

Keywords. Web augmentation, client-side adaptation, communities of Web developers.

1 Introduction

Web augmentation techniques have been proposed as a suitable way for extending Web sites features without changing the code of existing applications on the server-side [3]. Most of the popular Web augmentation tools extend the Web browser functionalities via dedicated plugins that can run dedicated client-side scripts to manipulate the structure of Web pages. Once Web pages are loaded on the Web browser, these augmenters can not only change the way information is delivered to the users but they can also modify the way users can interact with Web sites; for example by adding a button with a shortcut to Skype every time a phone number is detected on a Web page [10].

The potential of Web augmentation techniques for adapting existing Web sites according to a variety of users' need is huge and this can be easily illustrated by some advanced applications such as lightweight integration of information extracted from the Web [19], generalization of social features allowing user to vote and share information with other users whilst browsing Web sites [12], and support to context-sensitive navigation across diverse Web site [13] and refactoring Web sites for accessibility [18]. It is interesting to notice in all these examples that Web augmentation techniques are used to adapt contents and functionalities according to users' needs that have not been originally take into account during the design of the Web site.

However, because the Web site can be modified by external scripts, developers of the so-called Web augmenters do not necessarily need to collaborate with the developers of the existing Web sites to accomplish their job. The flexibility provided by Web augmentation techniques motivates individual and communities of creative coders to develop scripts, and some of them become popular [6]. For example, YouTube center [29], which has been conceived to add several new functionalities (ex. download videos, repeat videos, change the default quality of the video to be played by default) for improving the user experience while navigating Web page containing videos, has been proved very popular with more than 15K downloads; which is an impressive number for a script that provide similar features already supported by others similar augmenters with even more downloads! It is interesting to notice that the very existence of communities might also allow end-users to provide direct feedback to the developers. Thus, in front of requirements that are not satisfied, end-users might formulate their requests for new augmenters instead of addressing the Web site's owners to change the original Web site [15]. Moreover, some of the tools delivered by the community such as "*modding interface*" [1][9] and WebMakeUp [7] explore the possibility of supporting the development of simple Web augmenters by end-users. Whilst these are still early attempts for supporting end-user programming over the Web, at least in terms of coordination of adaptation tasks to be performed over Web sites, are promising and should be considered in a longer run [14][22].

In this context, Web augmentation might compete with other existing techniques for adapting contents and functionalities over existing Web sites. On one hand, when compared with closed adaptation techniques that are broadly-used by large Web applications (such as Netflix, Amazon, etc.) for supporting personalization of Web site contents [4], Web augmenters present the advantage of allowing users to go beyond of a limited number of adaptation features predefined by the Web site's owner. Some Web sites (such as Google drive, Facebook, etc.) provide APIs that can used to build extension-based adaptation of contents. From the point of view of developers, APIs extensibility usually implies to follow specific guidelines and to comply with constraints. Moreover, there is no guarantee that the API will provide all the sought adaptation mechanisms, which might be frustrating for both coders and end-users. On the other hand, without prior commitment of the participants involved in the process, the development of Web augmenters can also be challenging and frustrating. For example, Web augmenters might stop working when owners decided to change the Web site design [8]. Moreover, coders can see their efforts wasted if their augmenters are not adopted by users, which is often the case when they are not able to communicate with end-uses to understand their needs [6]. As for the Web site owners, neglecting users' need for adaptation and the creative potential of community of coders, might make of them less competitive in the market.

In this paper we analyze the trades-off of Web augmentation approaches and we claim that benefits can be shared among actors involved in the process. An analysis of different strategies for Web site adaptations (including Web augmentation) and the actors involved in the process is presented in section 2. In order to mitigate some of negative effects, such as the loose of control due to actors that work in isolation, we propose in section 3 a negotiated approach for Web augmentation adaptation. The main idea is to capitalize the Web augmentation-based adaptation in order to delegate to the crowd of users the specification of what are the changes they are looking for, delegate to coders (users with programming skills) the implementation of the augmenters and finally let to the Web site's owner to integrate augmenters developed by external coders into their Web sites. Such as an approach is duly supported by a platform which is described in the section 4. At section 5 we propose an assessment based on the cost estimation. In section 6 we discuss the contribution at the light of existing work and lately in section 7 we present conclusions and future work.

2 Actors, strategies and trade-offs for adapting Web interfaces

Traditionally, the adaptation of user interfaces require the definition of what are the goals to be reached with the adaption, what is going to be adapted, when events trigger the adaptation, and which programming techniques are used to perform the adaptation [4]. In the context of this work, the goal of adaptation is to modify any aspect of the client-side user interface (either rendering and/or behavior) to cope with the needs of a user or a crowd of users. For that purposes adaptations might change the way users perform tasks (ex. replacing scroll navigation between the top/bottom parts of a Web page by adding navigation buttons), the contents in display (ex. enriching the Web page with information obtaining from other sources), and/or the structure and layout of Web page (ex. adapting the contents for improving accessibility of the Web page or adapting contents to screen size of devices). These adaptations of the user interface should be triggered by DOM events in the client-side. As for the techniques, the main focus of this study is on Web augmentation.

The development, deployment and usage of Web adaptation techniques affects many actors, including the *Web site owner*, the *coders of external scripts* and the *end-users* who might have special needs. The term *Web site owner* is used hereafter to designate the development team that are involved in the constructions and have full control of the original Web site. In opposition, *coders of external scripts* refer to developers who implement augmenters to support adaptations on the client-side. *End-users* refer to the user populations that consume Web site contents (being adapted or not to their profile). **Fig. 1** illustrates the relationship between these actors according to four different strategies for adapting Web applications, as follows:

- *Closed adaptation* refers to adaptation techniques that are embedded as part of the original Web site and, therefore, totally under the control of *Web site owner*. This kind of adaptation might encompass adaptations that might be processed either on the server-side or on the client-side. In any case, developers don't have any constraint for accessing to the code source of the Web site. Moreover, adaptations can be built upon users' characteristics that have been obtained by explicitly collecting users' profile (via Web forms) or implicitly (by tracking the behavior of ordinary visitors) [23]. So In closed adaptation, the Web site is modified as the result of a direct relationship between the *Web site owners* and the *end-users*. On one hand, end-users contribute with information that can be used to personalize the adaptation. On the other hand, end-users only have access to adaptations that have been predefined by the Web site owners. Typically, recommendation systems and collaborative filtering systems [26] as well well-known works on adaptive hypermedia such as Navigation Support [4] belong to this category.
- *Extension-based customization* encompasses adaptation techniques that are built by extending the Web site with the help of dedicated APIs which allow external developers to adapt the user interface. This kind of adaptation strategy is very well known in applications using APIs such as Google Drive [30] or Facebook apps [31]. By using APIs, *external coders* can create new forms of adaptations that have not been considered yet by *Web site owners*. Overall, adaptations require a triangulation between what *Web site owners* make adaptable on the Web site through an API, what *coders* can do with the API and the *end-users'* expectations. Quite often, advanced programming skills, deep knowledge about the original Web site and the functions provided by the API are required to create adaptations. Thus, *coders* are somewhat dependent of the availability of API delivered by the original Web site. Besides that, end-users need to be supported with some tool for browsing and installing the available extensions.
- *Web augmentation* is a term used to address a large set of techniques that allow the adaptation of existing third-party Web applications in the client-site in such a way that no prior authorization from *Web site owners* is required [10]. Web augmentation techniques can be tuned to work on a specific Web site but they can also be generic enough to work in any kind of Web site. There are alternative solutions for implementing *Web augmentation* techniques, but currently the most common one is via Web browser extensions (i.e. plugins) that once installed inside the Web browser are able to modify the Web sites visited by the users. There are communities around these augmenters, and then, end-users without the necessary skills may take advantage of the existing scripts, which are uploaded by their creators to public repositories such as GreaseFork [33] and UserScripts [34]. Most of the popular augmenters are implemented as JavaScript scripts, which mean that coders need to have advance skills in imperative programming. As we shall, the relationship between actors is simplified by excluding *Web site owners* from the adaptation process.
- *End-User Web augmentation* is a concept built upon *Web augmentation techniques* for empowering end-users with tools allowing them to program their own scripts [6][7][14]. The underlying idea is that users with little program skill can use end-user programing environments for creating specific kinds of augmenters. Quite often, users are guided through a set of visual programming languages that hide the complexity of the inner JavaScript code used to perform the adaptations on a Web site. This kind of techniques assume that end-users can manage by their own their experience with Web site so that they do not longer depend on *coders* and/or *Web site owners* for having their Web sites personalized according to their will. For example, in [15] authors propose an environment where, by augments, users can modify on the fly Web pages as a mean to express requirements and/or to personalize the interaction with Web site. The developers of the end-user programming environments are excluded in the relationships show in **Fig. 1** because they have no control on the kind of adaptations users do with the tools.

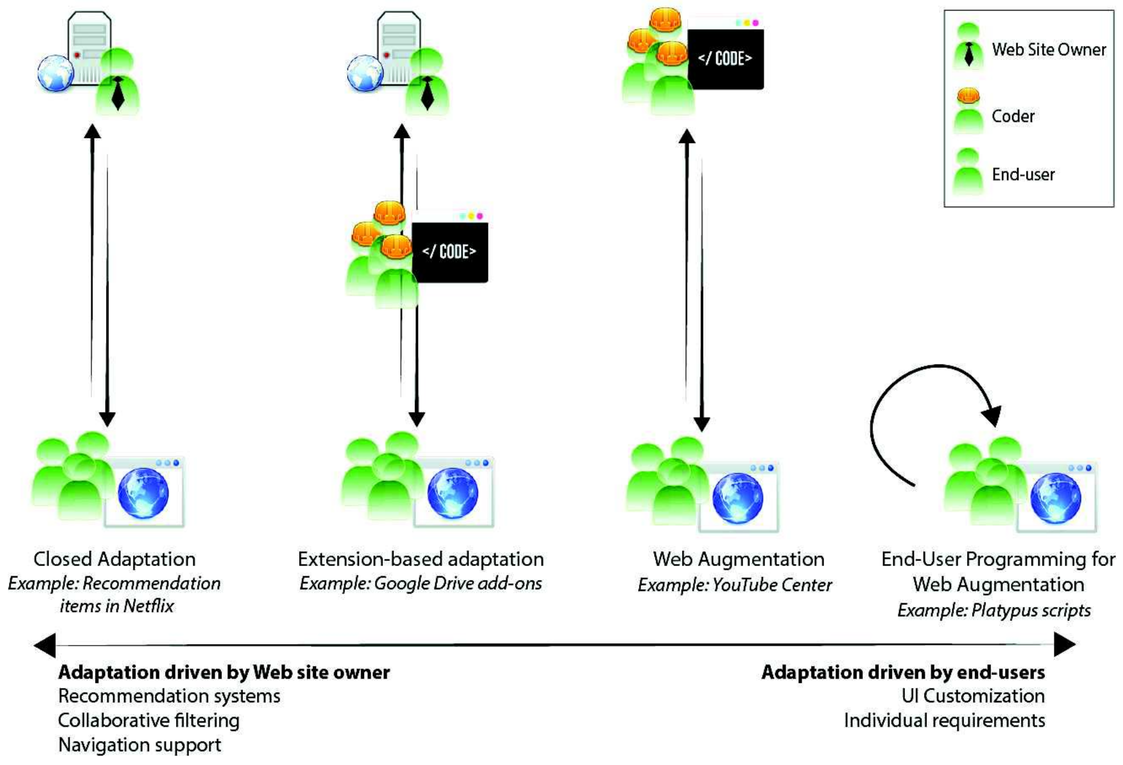

Web Site Owner
Coder
End-user
</ CODE>
</ CODE>
Closed Adaptation
Example: Recommendation items in Netflix
Extension-based adaptation
Example: Google Drive add-ons
Web Augmentation
Example: YouTube Center
End-User Programming for Web Augmentation
Example: Platypus scripts
Adaptation driven by Web site owner
Recommendation systems
Collaborative filtering
Navigation support
Adaptation driven by end-users
UI Customization
Individual requirements

Fig. 1. Relationship between actors involved in the development of adaptation strategies for Web applications.

Adaptation strategies define different types of relationship between actors. For example, in *closed adaptation* only *Web site owners'* and the *end-users* are involved. This is the most traditional approach for Web adaptation nowadays. For that, users often have to provide lots of personal information (explicitly via a user account, or implicitly by collecting user footprints whilst navigating the Web site). Personal information has a cost for the users in terms of time (for providing information) and privacy but for the Web site owner this is a basic requirement for creating a user model to drives the personalization of the Web site [23]. Closed adaptation, by definition, excludes the possibility of collaboration with external coders so all the costs of adaptation are supported by Web site owners. But site owners have at least full control on the adaptations provided to end-users.

By exposing an API to the community *Web site owners* allows the collaboration with an external community of *coders*. Deploying an API implies significant costs for the Web site owners but is might also contribute to the popularity of the Web site, for the benefits of *external codes* and the *Web site owners* alike. To ensure protection on the backend, adaptability through APIs is often limited to a few functions which do not necessary grant access to information about the user profile. If one hand such strategy allows some level of control, it might also limit creativity of *external coders*. For the users, *extended adaptation* will only work in a few Web sites and do not necessary cover all users' needs for personalization.

In the case of *Web augmentation* strategies, it is the *Web site owner* who is excluded from the process. The relationship occurs only between *end-users* who become the clients of *coders* who develop augmenters that operate on client-side without the supervision and control of the *owners'* of the Web. In such cases, *Web site owners* are excluded from the adaptation process and have no control in how the contents are delivered to end-users. *Web site owners* might be tempted to regularly modify the DOM structure of Web pages thus making harder to execution external scripts; which is a hard blow for *external coders* and *end-users*. Adaptations are often volatile and should be reapplied every time users return to the Web site. The kind of adaptation supported by *Web augmentation* strategies can be limited, but end-users can particularly benefit from adaptations that work alike in a variety of Web sites and might support user's tasks whilst navigating several Web sites [13][17].

The situation is more complex in the case of *end-user Web adaptation,* because both *Web site owners* and *external coders* have no control about which kind of adaptation are performed by *end-users* in the client-side. Moreover, *end-users* still have to develop some level of programming skill, download and install end-user development environments etc.

Table 1 provides a summary of advantages and limitations for actor involved in all the adaptations strategies previously described.

Table 1. Tradeoffs for actors involved in strategies for Web site adaptations.

	Web site Owner	External Coder	End-user
Closed adaptation	- Implementation of adaptations models on the Web server is expensive	- No involvement	+ Users do not need to install anything on the browser - Costs limit the availability of adaptation to large and specialized Web sites
	+ User can provide personal information for portraying a user profile, which can be used to tune adaptations and orient the business model		+ Can support personalization via recommendation and collaborative filtering systems - Users must to have create an account and a profile to get the benefits of adaptations - Adaptations that require user profile/feedback raise privacy issues
	+ Full control of adaptation mechanisms proposed to users		+ Adaptations can be tune to the usual tasks with the Web site - Might not be enough to support the adaptations required by users
	- Adaptation mechanism are dependent of a Web site		- Users have to learn how adaptation works in every Web site
Extension-based adaptation	- Implementation of APIs has direct costs, including for exposing them to external coders and users	+ Often free of costs for coders + Support and documentation might be free of charges + Coders can build they applications on the top of existing Web sites, which is often cheaper than starting from scratch	+ Users do not need to install anything on the browser +/- There is no guarantee that applications using APIs are free of charges for the end-users - Availability of adaptation is limited to Web sites that provide APIs supporting adaptation
	+ Control of functions that are made available through the API	- Coders creativity can be limited to functions available in the API - Might not be enough to support the adaptations envisaged by coders - Adaptations might stop working if the APIs is updated	- Might not be enough to support the adaptations required by users
	+ Code provided by external coders might contribute to the popularity of the owners' Web site - Still requires programming on the top of APIs to implement the required adaptations - Web site owner don't have control of adaptations implemented with APIs	+ Coders can contribute to create new forms adaptations	+ Let users to customize the application by installing extensions
Adaptation via Web Augmentation	- No involvement	+ Coders creativity is not limited by APIs - Adaptations might stop working if the DOM of Web page are updated	+ Users are exposed diverse alternatives for adaptation Web sites - Might not be enough to support all the adaptations required by users
		+ By tracking downloads of extensions proposed to the users coders can assess popularity and infer user needs	- Require users to install extensions
		+ The independence of APIs allows the development of augmenters focused on specific tasks that might be generic and thus be applied in many Web sites	+ Users can reuse adaptation tools in a seamlessly way whilst navigating the Web - It is hard to take into account the user profile in generic augmenters, so that users might lose the benefits of recommendation systems or collaborative filtering systems
		+ There is no limitation about what aspects of the client-side application the coder may adapt on the Web site	- Adaptations might be volatile and should be trigger by the user every time s/he visits the Web site
		- Different augmenters developed by different coders might spoil the alterations between them	- Adaptation mechanism provided natively may be spoiled - Since augmenters are not verified, some of them may be malicious
EUP for Web augmentation	- No involvement	- No involvement	+ Users are empowered with tools to perform the adaptations they want
			- End-users must to develop programming skills for using EUP environments
			- Adaptations can be limited by users knowledge and skills, as well as the functions delivered in EUP environments

Legend: Advantage (+) *and inconvenient* (-) *for actors involved in the adaptation process*

3 Towards a negotiated Web adaptation approach

All existing adaptation strategies remarkably fail to provide seamlessly collaboration between actors involved. For that, we describe in this section a negotiated approach for Web adaptation which relies in three basic principles: first of all, all actors must find advantages in the collaboration; secondly, actors must to collaborate; last but not least, appropriate tool is essential to incite actors to collaborate. In section 3.1 we present a view at glance of the approach. Section 3.2 provides a more detailed description of the distribution of tasks among the actors and how the execution of individual tasks can contribute to advantages for all. A first presentation of tool is introduced with the description of tasks.

3.1 The approach in a nutshell

Fig. 2 illustrates some advantages actors can find in this kind of relationship. For example, *Web site* owners can stablish a trustful relationship with *coders* that guarantee that augmenters are not malicious. The negotiation between *Web site owners* and *coders* also benefit *end-users* who, by extension, can trust that third-party augmenters have been checked by owners of the Web site they trust. By keeping *end-users* in the loop benefits both the *Web site owners* who can continue to collect information about users and even share such as information with *coders* for improving their augmenters according with user needs.

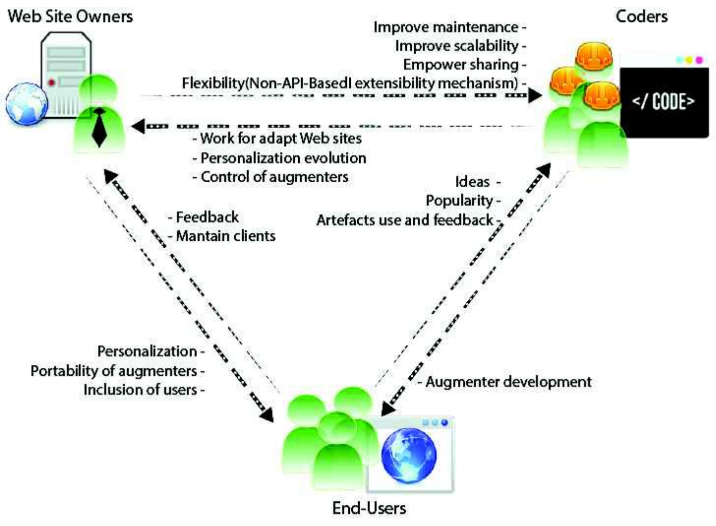


Fig. 2. Relationships between actors in a negotiated adaptation approach.

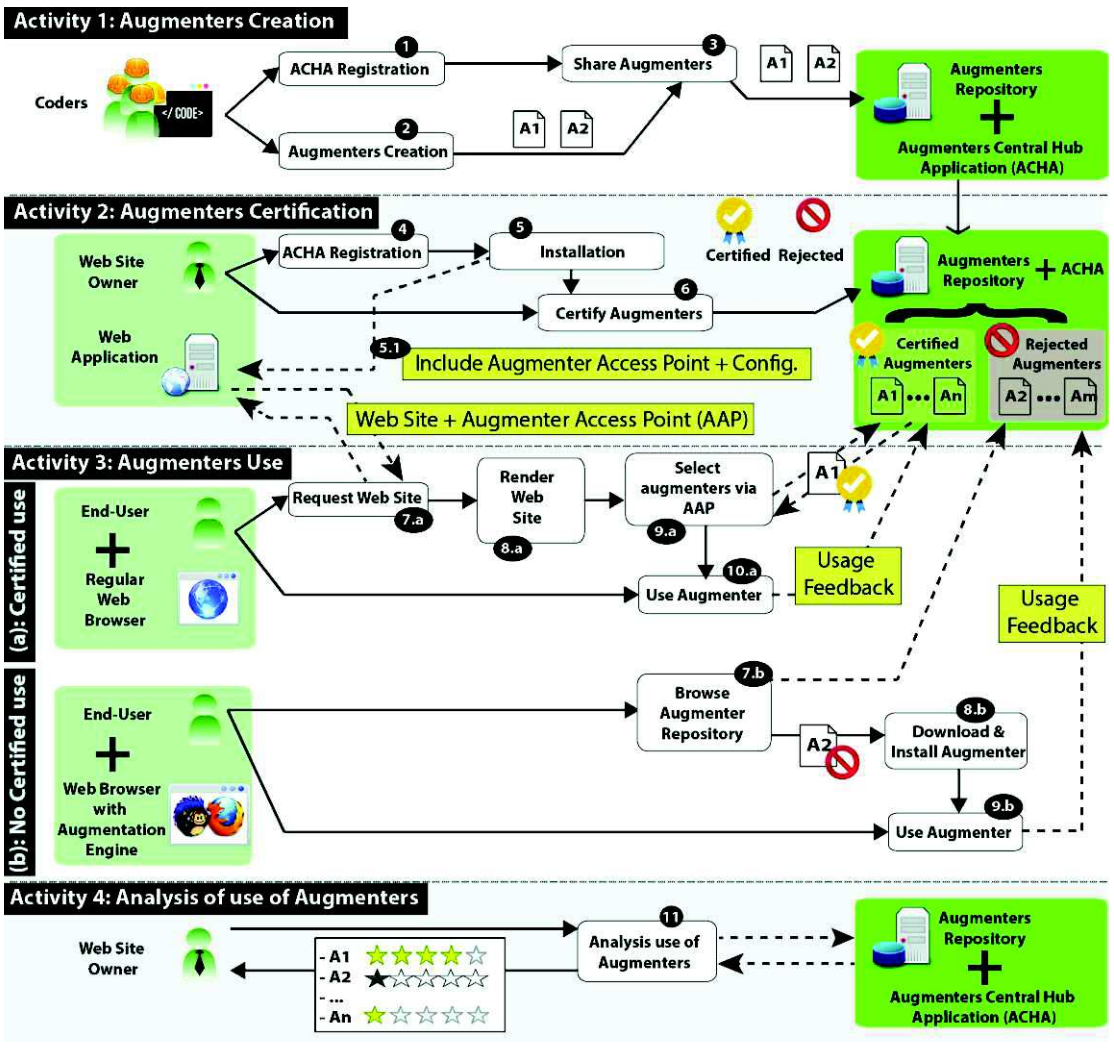
The negotiated approach also implies that a kind of commitment can be reached and for that, actors must collaborate. Close collaboration and commitment often demand the implementation of coordination and communication tasks [24], which require additional resources (in particular in terms of time and cognitive effort to maintain relationships running). To prevent that additional coordination and communication tasks come to plunge the advantages of such collaboration, the negotiated approach proposes that actors can work independently (as much as possible) and only perform the tasks for which they might foresee a direct advantage. To support such as a light-tight collaboration, we rely on a distribution of task among the participants and the existence of appropriate tool support as presented in **Fig. 3**. Task allocation per actor is presented below.

3.2 Analysis of actors' tasks

In order to better analyze the implications of approach for individuals, we present herein tasks involved in each activity as illustrated by Fig. 3. Notice that these tasks cannot be performed with appropriate tool support. For that a set of tool have been developed, including:

- *Augmenter repository:* is a Web site that contains the augmenters developed by coders;
- *Augmenters Central Hub Application (ACHA):* is the front-end application that allows the management (search, inclusion, etc.) of augmenters into the repository.
- *Augmenters Access Point (AAP):* is a client-side component embedded on the Web sites registered in ACHA thus providing direct access to certified augmenters.

A full description of these tools is provided in section 4. Nonetheless, we make explicit reference to these tools whilst describing the tasks the different actors have to perform in a negotiated approach.



Activity 1: Augmenters Creation
Coders
ACHA Registration
Augmenters Creation
Share Augmenters
A1
A2
Augmenters Repository
Augmenters Central Hub Application (ACHA)
Activity 2: Augmenters Certification
Web Site Owner
Web Application
ACHA Registration
Installation
Certify Augmenters
Certified
Rejected
Include Augmenter Access Point + Config.
Web Site + Augmenter Access Point (AAP)
Certified Augmenters
Rejected Augmenters
A1 ... An
A2 ... Am
Activity 3: Augmenters Use
(a): Certified use
End-User + Regular Web Browser
Request Web Site
Render Web Site
Select augmenters via AAP
Use Augmenter
Usage Feedback
(b): No Certified use
End-User + Web Browser with Augmentation Engine
Browse Augmenter Repository
Download & Install Augmenter
Use Augmenter
Usage Feedback
Activity 4: Analysis of use of Augmenters
Web Site Owner
Analysis use of Augmenters
- A1
- A2
- ...
- An

Fig. 3. Tasks allocation in a negotiated adaptation based on a Web augmentation approach.

3.2.1 Augmenters Creation: Coders

The main task of *coders* is to create and share augmenters. To share their creations, coders must to register at ACHA (see task (1) in Fig. 3. Then, registered coders can create (2) and share (3) augmenters into the Augmenter Repository. Indeed, we can foresee many types of adaptations supported by augmenters including generic adaptation that could seamlessly in different Web sites, for example replacing phone numbers in Web pages with a shortcut to Skype; augmenters that exploit user profile; for example by using user history of navigation, the augmenter might propose links to most recent searches; or advanced augmenters that allows user customize user interface, for example letting the user to rearrange the layout.

3.2.2 Augmenters Certification: Web Site Owners

As for *Web Site owners*, they have three main tasks, as follows:

- *Registration*: in first place, owners need to register themselves in the repository of Web augmenters as the owners of a particular web domain (see task (4) in Fig. 3), for instance dblp.org. They also must to provide a security file that is use to authenticate its Web site.
- *Installation*: to make augmenters available through their Web sites, owners have to include in their HTML responses the AAP component, task (5). This component is the responsible of allowing users to select augmenters without installing Web browser plug-ins. The inclusion of AAP, may also implies some more tasks, see (5.1) in Fig. 3. *Config AAP, adapting look & feel*: the tools included in AAP component have a specific look & feel by default. However, it is possible to envisage that *Web site owner* could tune it to make it to fit in Web site's design.
- *Augmenters Certification*: once the registration process is finished and *Web site owner* is connected to the repository, s/he can navigate and inspect the augmenters available. Suitable augmenters can be certified as indicated by task (6) in Fig. 3.

3.2.3 Augmenters Use: End-Users

The negotiated approach gives to *end-users* a major role as they ultimately have full control about the adaptation that is going to be performed on the Web site. This activity implies many duties and tasks with respect to the selection of augmenters. As we mentioned, selection of augmenters may be made in two ways. On the one hand the selection is made from the AAP (for the certified augmenters), which is the Activity 3.a in Fig.3. On the other hand, augmenters may be selected as it happens nowadays in Web Augmentation communities, which is illustrated in Fig. 3 such as the Activity 3.b:

Use of certified augmenters (a): when users visit a Web site – task (7.a) – for which there are certified augmenters, the Web is rendered in the client (task 8.a) embedding the AAP tool. By using the AAP tool (task 9.a) *end-users* can select the desired augmenters which are then downloaded and executed transparently in the client-side (task 10.a). At the same pace, the ACHA record in the repository the information that a user has downloaded and used a given augmenter.

Use of non-certified augmenters (b): *end-users* can also decide to use augmenters that do not have been certified by *Web site owners*. Non-certified augmenters do not automatically appear through the AAP when visiting a Web site. However, by using ACHA, users may browse non-certified augmenters – task (7.b) –. If some these rejected augmenter is relevant, the user may download it (task 8.b), and use with some external Web Augmentation engine such as GreaseMonkey – task (9.b) –. Note that this activity is not necessarily carried out by all the Web application users, but by those that are aware of the existence of the mechanisms for adapting third-party existing applications.

3.2.4 Analysis of use of Augmenters: Web Site Owners

Finally, Web Site Owners may obtain feedback of use of the existing augmenters; it is the task (11) in Fig. 3. It is interesting to notice that the information about users using non-certified augmenter (as part of the task 9.b) of a Web site becomes part of the knowledge base of ACHA. This information is available for *Web site owners* who, thereupon, can decide to investigate (or not) why users are using such augmenters and what are they need for adapting the Web site.

4 A platform for Web adaptation based on augmentation techniques

In this section we present further details about the tool support that we have developed to demonstrate the feasibility of our negotiated approach. Section 4.1 present a few underlying requirements that we have identified as essential for automating (as much as possible) user tasks. This follows with the presentation of the set of tools that have been developed to support the approach and concrete example of tool usage.

4.1 Underlying requirements

These components were defined in based to response to several aspects we believe that are really important for a negotiated adaptation approach:

- *Easy to install*: tools installation should be as simple as possible for the *Web application owners*. With this in mind all the necessary is to register as owner of the corresponding Web application in ACHA and also to add a JavaScript library on the Web pages (which contains the AAP among others). In the development of current applications supported by Web frameworks, it usually would means to add a line of code in the main template of the application.
- *Customizability* of the look & feel for augmenters: besides to be easy to install, we allow Web application developers to define specific styles and behavior to the end-user tool (Augmenter Access Point tool) in order to make it compatible with the look & feel of the application.
- *Plug & Play*: it is essential for *end-users* to be able to select, activate and deactivate augmenters. Users must feel in control of the usage of adaptation but should guide them in the process.
- *Compatible and extensible*: the negotiated approach and the corresponding tool set should be compatible with existing augmenters in the community. In this way, our current implementation is compatible with existing user scripts, which are probably the most popular kind of artefacts. For that, our tool set must also provide an *Augmenter Engine Emulators* to make possible to execute any kind of scripts featuring augmenters.
- *Independence of Augmenter Repository*: the external repository shown in Fig. 3 is proposed as a public standalone Web application. However, if *Web site owners* don't want to consume augmenters from the public repository, they instantiate a version of both the Augmenter Repository and ACHA in a private Web server accessible to a small community of *coders*.

4.2 Set of tools supporting the approach

In order to address the requirements and support the tasks of all actors involved in our negotiated approach of Web augmentation, we have developed a bipartite system.

At the server-side, we have developed two highly coupled components:

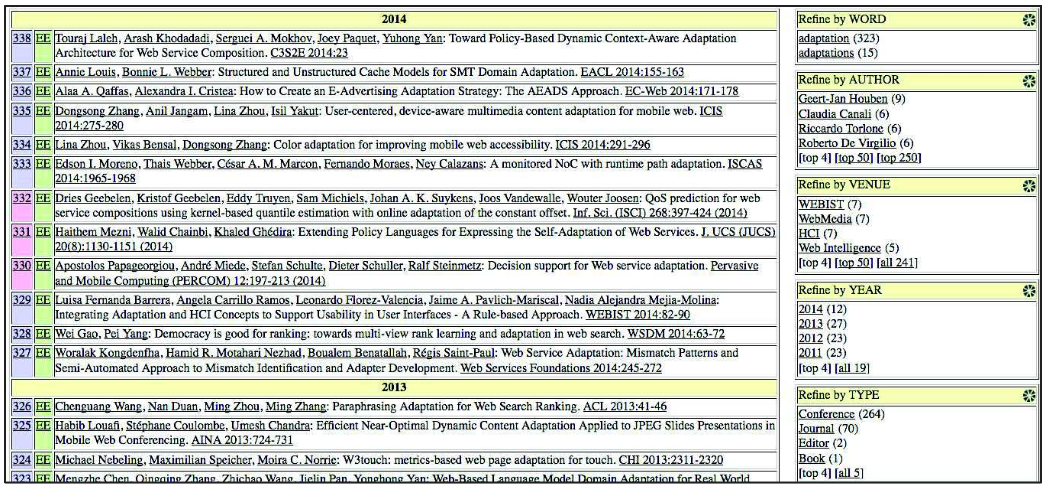
- *Augmenters Repository and Augmenter Central Hub Application*: Augmenter Repository centralizes all the augmenters created by coders. The Augmenter Central Hub Application (ACHA) allow to manage the repository according with each role, i.e., that ACHA exposes different views and functionality for repository accordingly with the responsibilities of coders, owners and end-users.

At the client-side, our Augmenter Access Point component (AAP) is a library written in JavaScript, which is composed by three subcomponents cooperating with each other:

- *End-User Augmenter Selection Tool*: this tool was implemented to help end-users to select augmenters they want to apply in the Web site. This tool takes into account the current context, i.e. which Web page of the application is loaded; this is because an augmenter not necessary works for the whole application but just one or a set of nodes.
- *Augmenter Injector*: When the user selects one of the certified augmenters by using the Augmenter Selection Tool, it delegates to the Augmenter Injector the tasks of downloading and executing the corresponding augmenter. Also the current selection state is saved by this component, then, the following time that the user visits the same Web page, augmenters previously selected are executed automatically.
- *Augmenter Engine Emulator*: augmenters uploaded to the Augmenter Repository may depend on different APIs. For instance GreaseMonkey scripts may use different functions provided by GreaseMonkey engine for facilitating some programming aspects. Then, in order to make our approach compatible with existing Web augmentation artefacts, we have implemented our system to be extensible with specific APIs emulators. As an example, we have developed an emulator for GreaseMonkey script.

4.3 Illustration of tools in a case study

The case study presented in this section is based on the Web site dblp.org. For the sake of illustration, examples we adopt the perspective of actors whilst presenting tools and the corresponding tasks. Moreover, all the examples below make reference to adaptions of the original page shown in Fig. 4.



		2014
338	EE	Touraj Laleh, Arash Khodadadi, Serguei A. Mokhov, Joey Paquet, Yuhong Yan: Toward Policy-Based Dynamic Context-Aware Adaptation Architecture for Web Service Composition. C3S2E 2014:23
337	EE	Annie Louis, Bonnie L. Webber: Structured and Unstructured Cache Models for SMT Domain Adaptation. EACL 2014:155-163
336	EE	Alaa A. Qaffas, Alexandra I. Cristea: How to Create an E-Advertising Adaptation Strategy: The AEADS Approach. EC-Web 2014:171-178
335	EE	Dongsong Zhang, Anil Jangam, Lina Zhou, Isil Yakut: User-centered, device-aware multimedia content adaptation for mobile web. ICIS 2014:275-280
334	EE	Lina Zhou, Vikas Bensal, Dongsong Zhang: Color adaptation for improving mobile web accessibility. ICIS 2014:291-296
333	EE	Edson I. Moreno, Thais Webber, César A. M. Marcon, Fernando Moraes, Ney Calazans: A monitored NoC with runtime path adaptation. ISCAS 2014:1965-1968
332	EE	Dries Geebelen, Kristof Geebelen, Eddy Truyen, Sam Michiels, Johan A. K. Suykens, Joos Vandewalle, Wouter Joosen: QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset. Inf. Sci. (ISCI) 268:397-424 (2014)
331	EE	Haithem Mezni, Walid Chainbi, Khaled Ghédira: Extending Policy Languages for Expressing the Self-Adaptation of Web Services. J. UCS (JUCS) 20(8):1130-1151 (2014)
330	EE	Apostolos Papageorgiou, André Miede, Stefan Schulte, Dieter Schuller, Ralf Steinmetz: Decision support for Web service adaptation. Pervasive and Mobile Computing (PERCOM) 12:197-213 (2014)
329	EE	Luisa Fernanda Barrera, Angela Carrillo Ramos, Leonardo Florez-Valencia, Jaime A. Pavlich-Mariscal, Nadia Alejandra Mejia-Molina: Integrating Adaptation and HCI Concepts to Support Usability in User Interfaces - A Rule-based Approach. WEBIST 2014:82-90
328	EE	Wei Gao, Pei Yang: Democracy is good for ranking: towards multi-view rank learning and adaptation in web search. WSDM 2014:63-72
327	EE	Woralak Kongdenfha, Hamid R. Motahari Nezhad, Boualem Benatallah, Régis Saint-Paul: Web Service Adaptation: Mismatch Patterns and Semi-Automated Approach to Mismatch Identification and Adapter Development. Web Services Foundations 2014:245-272
		2013
326	EE	Chenguang Wang, Nan Duan, Ming Zhou, Ming Zhang: Paraphrasing Adaptation for Web Search Ranking. ACL 2013:41-46
325	EE	Habib Louafi, Stéphane Coulombe, Umesh Chandra: Efficient Near-Optimal Dynamic Content Adaptation Applied to JPEG Slides Presentations in Mobile Web Conferencing. AINA 2013:724-731
324	EE	Michael Nebeling, Maximilian Speicher, Moira C. Norrie: W3touch: metrics-based web page adaptation for touch. CHI 2013:2311-2320
323	EE	Mengzhe Chen, Qingqing Zhang, Zhichao Wang, Jielin Pan, Yonghong Yan: Web-Based Language Model Domain Adaptation for Real World

Refine by WORD
adaptation (323)
adaptations (15)

Refine by AUTHOR
Geert-Jan Houben (9)
Claudia Canali (6)
Riccardo Torlone (6)
Roberto De Virgilio (6)
[top 4] [top 50] [top 250]

Refine by VENUE
WEBIST (7)
WebMedia (7)
HCI (7)
Web Intelligence (5)
[top 4] [top 50] [all 241]

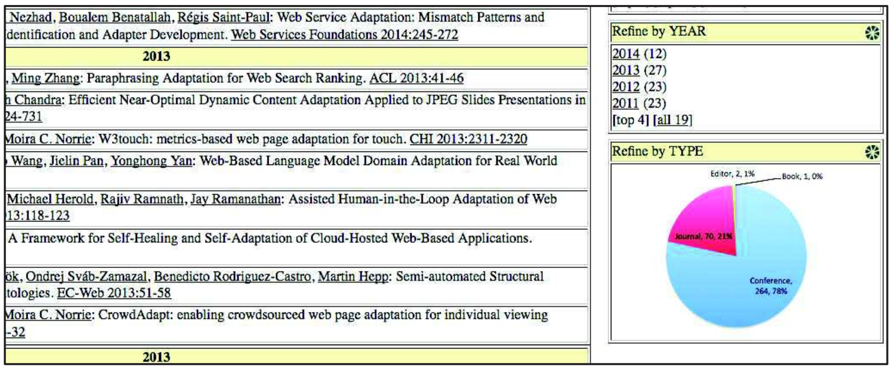
Refine by YEAR
2014 (12)
2013 (27)
2012 (23)
2011 (23)
[top 4] [all 19]

Refine by TYPE
Conference (264)
Journal (70)
Editor (2)
Book (1)
[top 4] [all 5]

Fig. 4. Original search page of the DBLP web site.

4.3.1 Coders

We assume that a *coder* has developed two augmenters that are aimed to improve the user experience of users of DBLP by implementing adapting features that are not yet available there such shown by Fig. 5. The first is an augmenter that is able to parse a DBLP Web page, extract information about publications and create a pie chart graph that can be injected into the Web site for depicting publication as shown at Fig. 5.a.The second augmenter is aimed at modify the layout of Web page responsive to screen size. Fig. 5.b shows the original Web page when visualized in a small screen and how visualization problems are fixed by the augmenter called *responsiveDBLP* Fig. 5.c.


Nezhad, Boualem Benatallah, Régis Saint-Paul: Web Service Adaptation: Mismatch Patterns and
dentification and Adapter Development. Web Services Foundations 2014:245-272

2013

, Ming Zhang: Paraphrasing Adaptation for Web Search Ranking. ACL 2013:41-46

h Chandra: Efficient Near-Optimal Dynamic Content Adaptation Applied to JPEG Slides Presentations in
24-731

Moira C. Norrie: W3touch: metrics-based web page adaptation for touch. CHI 2013:2311-2320

Wang, Jielin Pan, Yonghong Yan: Web-Based Language Model Domain Adaptation for Real World

Michael Herold, Rajiv Ramnath, Jay Ramanathan: Assisted Human-in-the-Loop Adaptation of Web
13:118-123

A Framework for Self-Healing and Self-Adaptation of Cloud-Hosted Web-Based Applications.

ök, Ondrej Sváb-Zamazal, Benedicto Rodriguez-Castro, Martin Hepp: Semi-automated Structural
tologies. EC-Web 2013:51-58

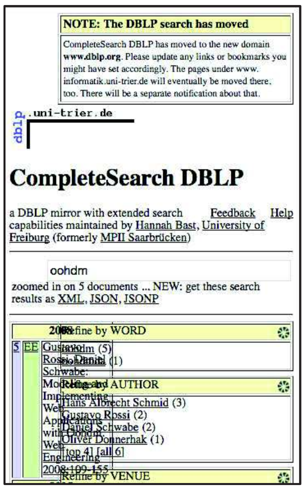
Moira C. Norrie: CrowdAdapt: enabling crowdsourced web page adaptation for individual viewing
-32

2013

Refine by YEAR
2014 (12)
2013 (27)
2012 (23)
2011 (23)
[top 4] [all 19]

Refine by TYPE

a) Pie chart augmenter featuring publications per type at DBLP.


NOTE: The DBLP search has moved

CompleteSearch DBLP has moved to the new domain **www.dblp.org**. Please update any links or bookmarks you might have set accordingly. The pages under www.informatik.uni-trier.de will eventually be moved there, too. There will be a separate notification about that.

dblp.uni-trier.de

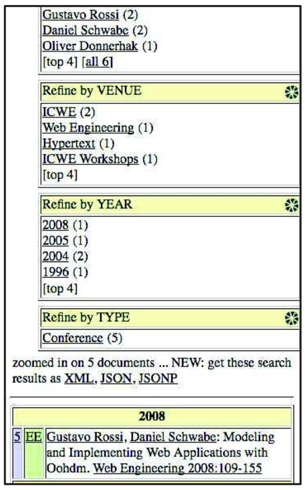
CompleteSearch DBLP

a DBLP mirror with extended search capabilities maintained by Hannah Bast, University of Freiburg (formerly MPII Saarbrücken) Feedback Help

oohdm

zoomed in on 5 documents ... NEW: get these search results as XML, JSON, JSONP

2008 Refine by WORD
5 EE Gustavo Rossi, Daniel Schwabe: Modeling and Implementing Web Applications with Oohdm. Web Engineering 2008:109-155
oohdm (5)
Refine by AUTHOR
Hans Albrecht Schmid (3)
Gustavo Rossi (2)
Daniel Schwabe (2)
Oliver Donnerhak (1)
[top 4] [all 6]
Refine by VENUE


Gustavo Rossi (2)
Daniel Schwabe (2)
Oliver Donnerhak (1)
[top 4] [all 6]

Refine by VENUE
ICWE (2)
Web Engineering (1)
Hypertext (1)
ICWE Workshops (1)
[top 4]

Refine by YEAR
2008 (1)
2005 (1)
2004 (2)
1996 (1)
[top 4]

Refine by TYPE
Conference (5)

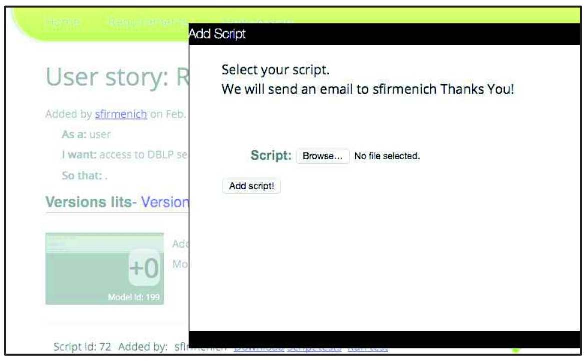
zoomed in on 5 documents ... NEW: get these search results as XML, JSON, JSONP

2008
5 EE Gustavo Rossi, Daniel Schwabe: Modeling and Implementing Web Applications with Oohdm. Web Engineering 2008:109-155

b) original DBLP page *d) adapted page using responsiveDBLP augmenter*

Fig. 5. Example of augmenters created by *coders* and waiting to be shared with the community.

The augmenters illustrated by Fig. 5 are ready to be used but they did not have been certified yet by the owners of the DBLP Web site. In order to get a certification and improve the visibility of these augmenters, we assume that the coder decides to share them via a public instantiation of both the Augmenter Repository and the ACHA, hosted at *UserRequirements*.org. For that, *coders* he must to create a user account on ACHA, define a user story describing the adaptations provided by each augmenter and finally upload it using the Web form as shown by **Fig. 6**.


User story: R

Added by sfirmenich on Feb.

As a: user

I want: access to DBLP se

So that: .

Versions lits- Version

+0

Model Id: 199

Script id: 72 Added by: sfi

Add Script

Select your script.
We will send an email to sfirmenich Thanks You!

Script: Browse... No file selected.

Add script!

Fig. 6. User interface allowing uploads of scripts into the repository *UserRequirements.org*.

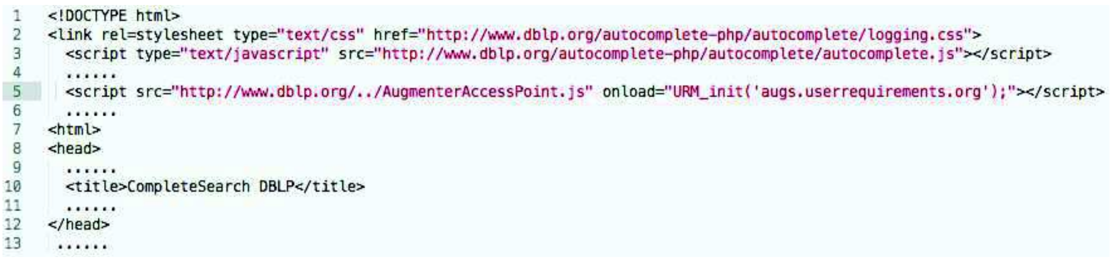
4.3.2 Web site owners

Let's assume that the owners are particularly interested by the augmenter *responsibleDBLP* shown Fig. 5, mainly because such augmenters might save lots of work for making the DBLP responsive. For including the *responsiveDBLP* into the DBLP Web site, the owners have at first to register at *UserRequirements.org* which can be accomplished by following these steps:

- Create a user account in ACHA, hosted in *UserRequirements.org*
- Register to themselves as owners of the domain *dblp.org*

a. Certificate ownership: in order to demonstrate that they are actually the owners of dblp.org, they must download from ACHA a file containing a security token for *dblp.org* and upload the security token file to the web application root
b. Log in in *UserRequirement* and validate domain. ACHA will check that the security token file is already in the owners' Web server.

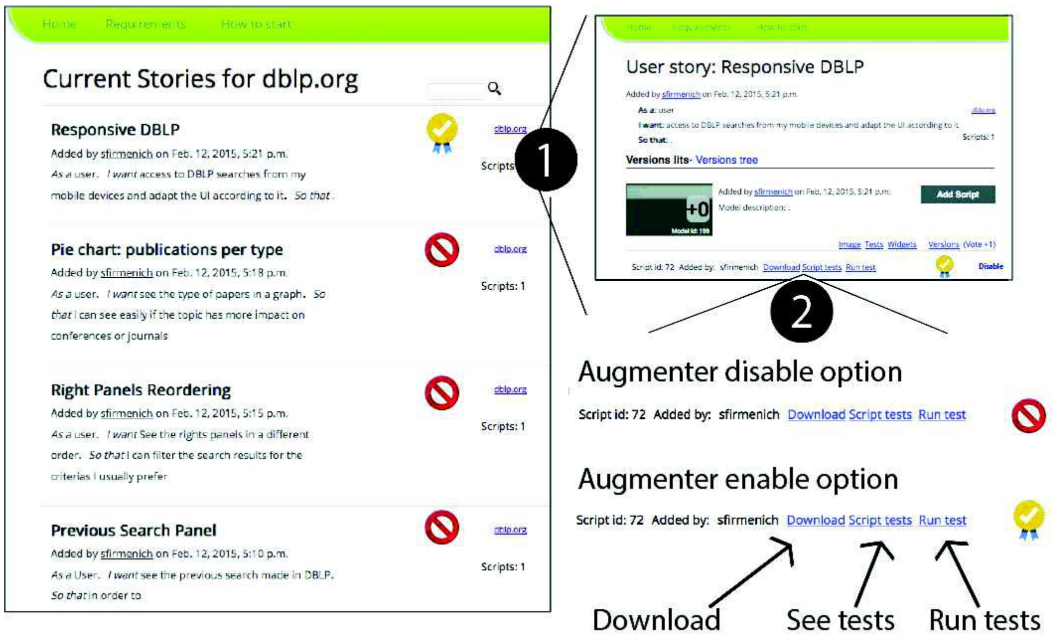
With these steps, ACHA accepts the association between that user account and the specified domain, in this case dblp.org. Once affiliated, owners need to add the Augmenter Access Point component into their main HTML. This only implies to add one line of code for adding a JavaScript file, line 5 in the code shown in **Fig. 7**.



```
1   <!DOCTYPE html>
2   <link rel=stylesheet type="text/css" href="http://www.dblp.org/autocomplete-php/autocomplete/logging.css">
3     <script type="text/javascript" src="http://www.dblp.org/autocomplete-php/autocomplete/autocomplete.js"></script>
4     ......
5     <script src="http://www.dblp.org/../AugmenterAccessPoint.js" onload="URM_init('augs.userrequirements.org');"></script>
6     ......
7   <html>
8   <head>
9     ......
10    <title>CompleteSearch DBLP</title>
11    ......
12  </head>
13   ......
```

Fig. 7. Example of DBLP Web page (HTML) featuring the links binding it to the augmenters repository.

Now is time to certify augmenters. For that *Web site owners* can navigate the Augmenter Repository looking for augmenters that are available for the DBLP Web site as shown by Fig. 8. Once the sought augmenter *responsiveDBLP* is found, further details can be obtained and by selecting the actions enable/disable is possible to certify this augmenter for the DBLP web site. It is also possible to download locally the augmenter for inspecting the code source, run it to see how it works. These tests are addressed mainly to check if augmenters are compatible with the current of the Web site DOM, however, owners may add further tests about the augmenter execution in order to prove if the adaptation is not spoiling relevant original content or functionality. Also from the ACHA, Web site owners can monitor user's feedback on this augmenter.



Current Stories for dblp.org

Responsive DBLP
Added by sfirmenich on Feb. 12, 2015, 5:21 p.m.
As a user. *I want* access to DBLP searches from my mobile devices and adapt the UI according to it. *So that* .
dblp.org
Scripts

Pie chart: publications per type
Added by sfirmenich on Feb. 12, 2015, 5:18 p.m.
As a user. *I want* see the type of papers in a graph. *So that* I can see easily if the topic has more impact on conferences or journals
dblp.org
Scripts: 1

Right Panels Reordering
Added by sfirmenich on Feb. 12, 2015, 5:15 p.m.
As a user. *I want* See the rights panels in a different order. *So that* I can filter the search results for the criterias I usually prefer
dblp.org
Scripts: 1

Previous Search Panel
Added by sfirmenich on Feb. 12, 2015, 5:10 p.m.
As a User. *I want* see the previous search made in DBLP. *So that* in order to
dblp.org
Scripts: 1

User story: Responsive DBLP
Added by sfirmenich on Feb. 12, 2015, 5:21 p.m.
As a: user
I want: access to DBLP searches from my mobile devices and adapt the UI according to it
So that: .
dblp.org
Scripts: 1
Versions lits- Versions tree
+0
Model id: 199
Added by sfirmenich on Feb. 12, 2015, 5:21 p.m.
Model description: .
Add Script
Image Tests Widgets Versions (Vote +1)
Script id: 72 Added by: sfirmenich Download Script tests Run test Disable

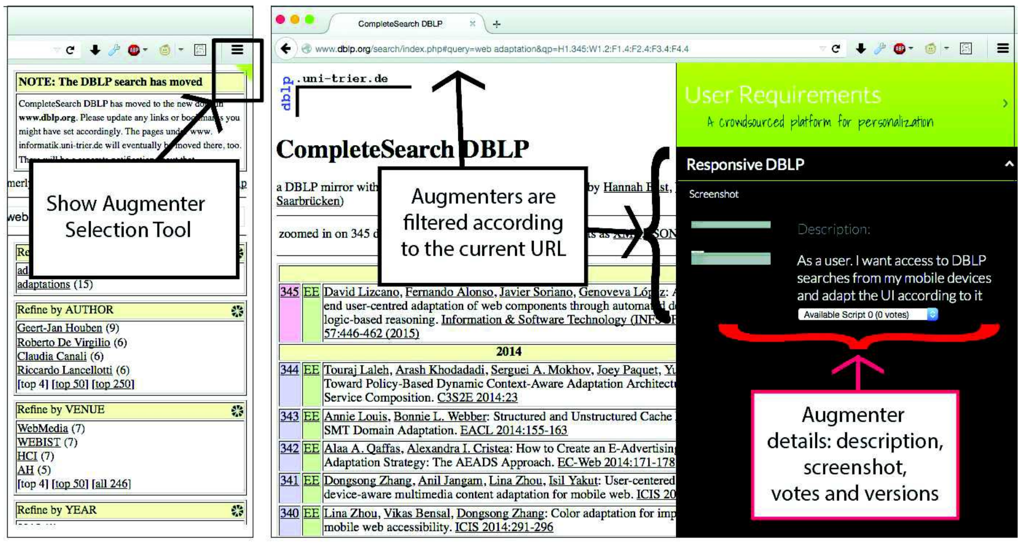
Script id: 72 Added by: sfirmenich Download Script tests Run test

Script id: 72 Added by: sfirmenich Download Script tests Run test

Fig. 8. Web site owner manages augmenters available for his Web site at *UserRequirements.org*.

4.3.3 End-users

From this point, when end-users of DBLP visit the Web site they will be notified that certified augmenters exist by a green binding point at the up corner of the screen, as the left picture from Fig. 9 shows. This is rendered by the Augmenter Access Point component, which was embedded in the Web site. This interaction between the end-user, the Web browser and our components is shown in Figure 10.


CompleteSearch DBLP
www.dblp.org/search/index.php#query=web adaptation&qp=H1.345:W1.2:F1.4:F2.4:F3.4:F4.4
Show Augmenter Selection Tool
Augmenters are filtered according to the current URL
User Requirements
A crowdsourced platform for personalization
Responsive DBLP
Screenshot
Description:
Available Script 0 (0 votes)
Augmenter details: description, screenshot, votes and versions

NOTE: The DBLP search has moved

CompleteSearch DBLP has moved to the new domain **www.dblp.org**. Please update any links or bookmarks you might have set accordingly. The pages under www. informatik.uni-trier.de will eventually be moved there, too.

dblp.uni-trier.de

CompleteSearch DBLP

a DBLP mirror with … by Hannah Bast, … Saarbrücken)

zoomed in on 345 d… …ts as XML JSON

adaptations (15)

Refine by AUTHOR
Geert-Jan Houben (9)
Roberto De Virgilio (6)
Claudia Canali (6)
Riccardo Lancellotti (6)
[top 4] [top 50] [top 250]

Refine by VENUE
WebMedia (7)
WEBIST (7)
HCI (7)
AH (5)
[top 4] [top 50] [all 246]

Refine by YEAR

345 EE David Lizcano, Fernando Alonso, Javier Soriano, Genoveva López: … end user-centred adaptation of web components through automated d… logic-based reasoning. Information & Software Technology (INFSOF… 57:446-462 (2015)

2014

344 EE Touraj Laleh, Arash Khodadadi, Serguei A. Mokhov, Joey Paquet, Yu… Toward Policy-Based Dynamic Context-Aware Adaptation Architectu… Service Composition. C3S2E 2014:23

343 EE Annie Louis, Bonnie L. Webber: Structured and Unstructured Cache … SMT Domain Adaptation. EACL 2014:155-163

342 EE Alaa A. Qaffas, Alexandra I. Cristea: How to Create an E-Advertising… Adaptation Strategy: The AEADS Approach. EC-Web 2014:171-178

341 EE Dongsong Zhang, Anil Jangam, Lina Zhou, Isil Yakut: User-centered… device-aware multimedia content adaptation for mobile web. ICIS 20…

340 EE Lina Zhou, Vikas Bensal, Dongsong Zhang: Color adaptation for imp… mobile web accessibility. ICIS 2014:291-296

As a user. I want access to DBLP searches from my mobile devices and adapt the UI according to it

Fig. 9. Using augmenters at the DBLP web site.

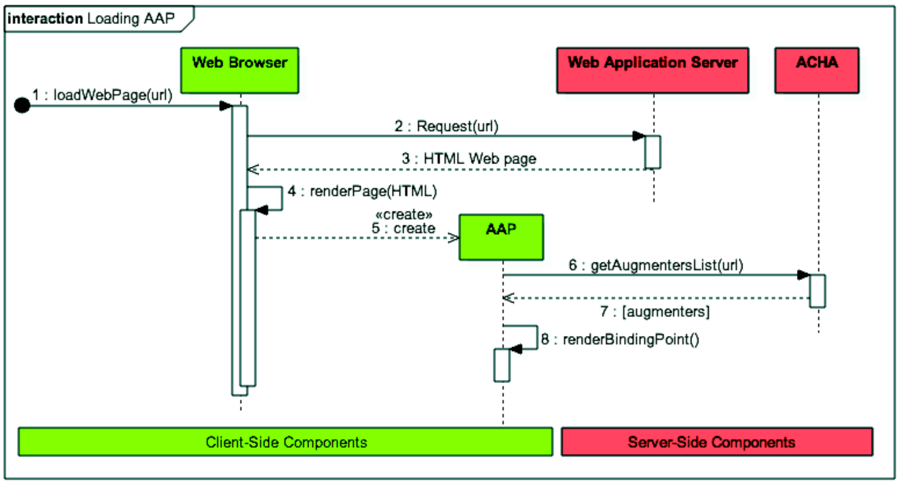


Fig. 10. Interaction when a Web site embedding AAP is rendered.

When the user clicks on it, a menu is deployed showing augmenters available (this is the Augmenter Selection Tool). To active/deactivate an augmenter, end-users only need to click on the corresponding name in the list. Via this menu, end-users can also see the description of the augmenter left by the coders and further details about its popularity. Note that, as Fig. 8 shows, the only augmenters available are those certified by the Web site owners in this case Responsive DBLP.

When a user selects an augmenter from the AAP, s/he gets it from the Augmenter Repository which delegates the execution to the Augmenter Injector. This interaction is shown in Figure 11.

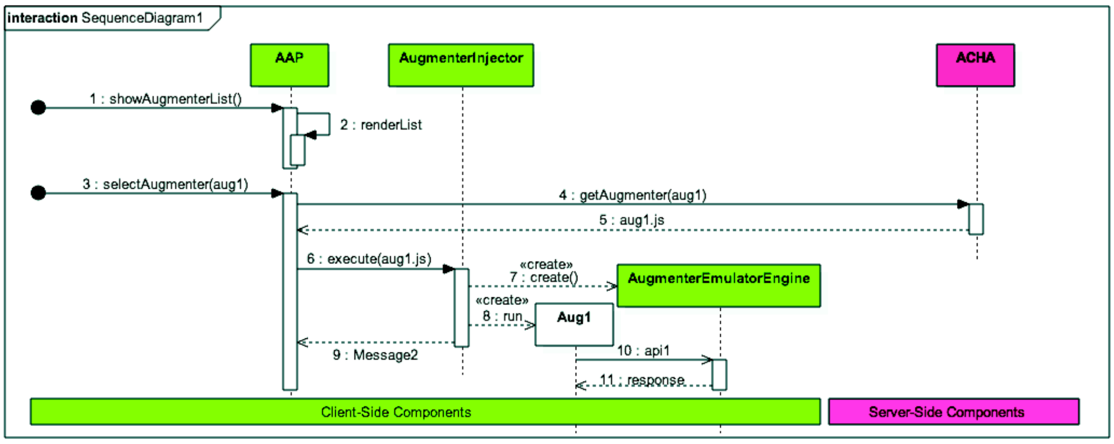


Fig. 11. Interaction when the end-user wants to enable an augmenter.

Although the non-certified augmenters do not appear in the Augmenter Selection Tool, if a user navigate the Augmenter Repository from ACHA, he may find also those rejected by the *Web site owner*, for instance "*Pie chart: publications per type*" among others. In these cases, the user may download and install it, in this case, with GreaseMonkey engine. Besides that, if available augmenters do not satisfy a particular user's need, he may ask to *coders* for new scripts by the addition of new user stories. For the sake of conciseness these functions are not described here but the interested reader can find further information at [15].

5 Preliminary assessment of tools

In this section we present a preliminary assessment of the platform and the negotiated approach.

5.1 Existent augmenters compatibility

To determine whether (or not) *existing augmenters in public repositories are fully functional and compatible with our platform* we have assessed the compatibility of 15 augmenters from public repositories[1], listed in Table 3. The augmenter selection was addressed to test different features:

- Generality: report if the augmenter works for any Web site or it is Web site-specific.
- Popularity: in terms of number of users (thousands of users *versus* a few known users).
- Programming effort: in terms of lines of code.
- API use: altogether, the augmenters selected use most of the API provided by the corresponding engines (User Script engines), in this way, we could show that Augmenter Engine Emulator is feasible to be built.

Since several of the augmenters are executed in very well-known Web sites that we are not able to manipulate at server-side, we have attached our platform to these applications via a bookmarklet, which is just a bookmark that executes JavaScript code when the user clicks on it. This JavaScript code is executed with the same privileges that native JavaScript code, then it is a sufficient prove of that augmenters may work from inside the application if the platform is also loaded. This also explains the case study presented in Section 3.1. The result was that the 100% of augmenters listed in Table 3 ran successfully with our platform. We have compared the result with the execution via Web Augmentation engines, and the augmentation effects were the same. Every feature of the augmenters (for instance, some personalization options that some of them support) also worked.

Table 3. Augmenters list

Site	Augmenter	Description	Users	Lines of Code
Youtube.com	*Aug1*: Download YouTube Videos as MP4	Adds a button to let users download YouTube videos.	3.711	765
Google.com	*Aug2*: Google Search Extra Buttons	Add buttons (last day, last week, PDF search etc.) to results of search page of Google	150	104
Imdb.com	*Aug3*: IMDB+	Add external links to IMDb. Every feature can be enabled/disabled in settings.	353	156
* (any Web site)	*Aug4*: Mouseover Popup Image Viewer	Shows larger version of thumbnails. Also supports HTML5 video.	8.580	1.207
* (any Web site)	*Aug5*: Google Translator Tooltip Expanded	Translates the selected text into a tooltip automatically.	493	1.227
Imdb.com	*Aug6*: Search IMDb Item on Netflix	Places a "Search for this on Netflix" button on the main page of any TV show/movie page on IMDb	141	34
Trello.com	*Aug7*: Trello-minimize lists	Minimize width of lists with toggle button	10	159
* (any Web site)	*Aug8*: Fixed Scroller Anywhere	Scroll by fixed pages	38	630
Wikipedia.org	*Aug9*: Wikipedia Inline Article Viewer	Adds a hover event to internal article links on wikipedia pages, which open the article inline in a dhtml frame.	3	512
Geocaching.com	*Aug10*: Geocaching Map Enhancements	Adds Ordenance Survey maps and grid reference search to Geocaching.com, plus other enhancements.	47.648	2.726
Twitter.com	*Aug11*: Twitter Instagram Cards - Photo Viewer	Now that Instagram have pulled their twitter support, this script adds back inline instagram photos.	361	46
* (all Web site)	*Aug12*: Universal Syntax Highlighter	It highlights plain text source code URLs in several languages. Based on the SpiralX auto highlighter	42	114
Google.com	*Aug13*: Endless google	Load more results automatically and endlessly.	4.732	142
Google.com	*Aug14*: Google Cache comeback	Brings back links to cached pages in the Google search results	15.208	248
Youtube.com	*Aug15*: Youtube to mp3	Convert youtube video to MP3	1.724	77

[1] http://greasyfork.org/

5.2 Cost and efforts estimation

In this section we discuss issues related to the estimation of costs and efforts needed for absorbing existing functionality implemented by augmenters taken into account three main schemas for Web adaptation, which are adaptation driven by Web site owners, adaptation driven by external scripts and negotiated adaptation. The underlying hypothesis is that: *using a negotiated adaptation, the cost and efforts for integrating existing functionality in Web augmentation communities are lower than implementing the same functionality natively in original Web site aimed to be adapted.*

For the analysis, we do not consider the costs of developing augmenters presented in Table 3, because we only know line of codes (LOC) and the amount of users. As it has been pointed by [18], it would not be correct to compare lineally the amount of LOC of each augmenter (coded with JavaScript) in comparison with the amount of LOC for the same functionality implemented natively, clearly, because the broad sort of technologies, which LOCs have different coding and debugging average times. However, we may analyses tasks' allocation in an owner-driven approach as well as in our negotiated approach as shown in Table 4. Particularly, the application must support a user profile (where the users selection of adaptations are stored), the augmenter implementation (which is the specific functionality or feature to be supported) as well the tool for allowing users to manage their profile. In this case, from the application owner point of view, if they want to incorporate natively *Aug1, Aug2,.., AugN*, the required effort may be stipulated as:

$$(i)\ OwnerEffortA = UserProfileModellingEffort + UserProfileImplementationEffort + \sum_{i=1}^{n} AbsorbEffort(Augi)$$

where the function *AbsorbEffort* calculates the effort required in a particular technology, which will vary for each augmenter given they have different complexity, as Table 3 shows. From the end-users point of view, the effort for use all the possible augmenters is reduced to:

$$(ii)\quad EndUserEffortA = \sum_{i=1}^{n} SelectAugmenterEffort(Augi)$$

Finally, since the coder role does not take part in this adaptation schema they have not an associated effort:

$$(iii)\quad CoderEffortB = 0$$

Table 4. Adaptation driven by Web site owners

	Server-Side	Client-Side	Repository
Owner	* User Profile Modelling * User Profile Implementation * Specific Augmenter Implementation	-	-
External community of coders	-	-	-
End-User	-	* Populate User Profile	-

In a second scheme for adaptation, where only a Web augmentation community can produce the adaptation, the development effort for owner is null.

$$(iv)\quad OwnerEffortB = 0$$

In a negotiated approach, coders make most of the implementation and advertising efforts. Supposing that there is only one coder, the effort he needs to make in order to create all the augmenters is:

$$(v)\quad CoderEffortB = \sum_{i=1}^{n} ImplementationAugmenterEffort(Augi) + UploadAugmenterEffort(Augi)$$

In this schema, we also need stipulate the effort made by end-users to finally use the augmenters. They need to do find, download and install each augmenter separately:

$$(vi)\ EndUserEffortB = \sum_{i=1}^{n} SearchAugmenterEffort(Augi) + DownloadAugmenterEffort(Augi) + InstallAugmenterEffort(Augi)$$

Table 5. Adaptation driven by external scripts only

	Server-Side	Client-Side	Repository
Owner	-	-	-
External community of coders	-	-	* Develop augmenter * Upload augmenter
End-User	-	* Install scripts	* Search augmenter * Download augmenter

Activities in a negotiated approach are summarized by Table 6. Notice that the effort for application's own encompasses the installation of tools and the activity for enabling augmenters:

$$(vii)\ \ OwnerEffortC = PlatformInstallationEffort + PlatformAffiliationEffort + \sum_{i=1}^{n} EnableAugmenterEffort(Augi)$$

On the one hand, the *PlatformInstallationEffort* is reduced to a line of code for adding the JavaScript library corresponding to our platform. On the other hand, both *PlatformAffiliationEffort* (the effort required for affiliate a Web site to the platform's repository) and the *EnableAugmenterEffort* are just administrative steps that can be done via a Web user interface. Estimation of effort for using a Web user interface is estimated by using GOMS-Keystroke (KLM) model [5]. The KLM model allows simulating the performance of a trained user proposing the average time to perform basic action (for instance, reach for mouse takes 0.40 sec). Thus, provided a detailed scenario of user actions including low-level user actions, it is possible to estimate user performance (i.e. speed). Using this model, we estimated that the *PlatformAffiliationEffort* is 21.7 seconds. To enable an augmenter, i.e. the *EnableAugmenterEffort*, cost 19.1 seconds regardless its complexity.With our platform, and how was described in section 5, end-users have a panel for enabling/disabling the augmenters that may be applied on the current Web site. In this way, their effort may be estimated as:

$$(viii)\ \ EndUserEffortC = \sum_{i=1}^{n} SelectAugmenterEffort(Augi)$$

As the reader may note from the Table, coders have the same activities in our negotiated approach than in the one based on existing Web augmentation communities, then, the coders' effort is equal to *(v)*:

$$(ix)\ \ CoderEffortC = CoderEffortB$$

Table 6. Negotiated Adaptation

	Server-Side	Client-Side	Repository
Owner	-	* Add JS Library	* Enable/Disable Scripts
External community of coders	-	-	* Develop script * Upload script
End-User	-	* Enable/Disable Scripts	-

5.3 Discussion

By analyzing the activities for each role involved in the three schemas of adaptation, we can say that our approach of negotiated adaptation based on Web augmentation techniques does not imply that coders have more efforts to do; in fact, it is the same for the approaches involving external coder communities (*CoderEffortC* and *CoderEfforB*).

End-users expend similar efforts in any adaptation approach for populating profile, enabling/ disabling or installing scripts. In current Web augmentation communities, end-users also are responsible of searching, downloading and installing the artefacts (besides the corresponding Web Augmentation Engine). In our approach, as well as in an owner-driven approach, users are guide to available augmenters. Thus *EndUserEffortA (iii)* and *EndUserEffortC (viii)* are almost equivalent.

Regarding to the efforts performed by owners (were owners do not interfere), we think that our approach has several advantages when comparing *OwnerEffortA* and *OwnerEffortC.*

For allowing end-users to select augmenters that adapt specific aspects of the Web sites, we have to take into account $(UserProfileModellingEffort + UserProfileImplementationEffort)$ from *OwnerEffortA* (i.e. without taking into account the *AbosrbEffort* function part) and $(PlatformInstallationEffort + PlatformAffiliationEffort)$ from *OwnerEffortC* (i.e. without *EnableAugmenterFunction*) we may clearly assume that *OwnerEffortC (vii)* would be lower since it only implies to add one line of code, meanwhile designing and implementing a User Profile and the corresponding mechanisms for populating it requires more than that.

If we consider that the mechanism recently described is working, and we only consider the effort needed to add a new adaptation artefact, we have to compare the *AbsorbEffort* (from *OwnerEffortA*) and *EnableAugmenterEffort* (from *OwnerEffortC*). When comparing both functions, it is essential to note that meanwhile the result of *AbsorbEffort* function depends on the augmenter complexity to be absorbed, the *EnableAugmenterEffort* function is constant for all the augmenters (19.1 seconds for each augmenter).

We don't analyze the discovery of augmenters. While the owner-driven approach requires inspect frequently the repositories in order to detect relevant augmenters (relevant from the end-user's point of view), with our approach owners get instant feedback from the same community.

6 Related work

The present work has interconnections with many relevant research questions for the development Web applications such as personalization and adaptation technique of Web applications, development of frameworks for supporting client-side adaptation and transcoding, end-user programming and communities of developers.

Since 1996, most of the papers related to adaptive hypermedia systems were focused on Web applications [4][27]. Clearly, the increase in the use of the World Wide Web was the main factor for which Web adaptation started to be an important research field, and this is still prevailing. Most of the well-known methods for design and develop Web applications have incorporated the design of adaptation mechanisms. User profile modelling [23] has become also an important concern in adaptive Web applications, as well as the design of recommendation systems, making together very good personalization systems. However, the use of the Web not only is still increasing, which was the main factor mentioned by Brusilovsky [4], but also the way in which the Web is used has been mutating.

Currently, the Web is for users a platform where a lot of tasks and activities are performed, this scenario makes the adaptation of these applications even more important, moreover if we take into account the huge and vary crowd of users, who have really different preferences, skills, background, needs, etc. In this scenario, and in front of users whose expertise is increasing, Web applications should be really customizable for satisfying all user requirements. With adaptation mechanism driven by owners, i.e. closed adaptation or extension-based adaptation, several users requirements are not taken into account, since those aspects of the systems that will be adaptable are those foreseen for the developers. This is understandable, because to make fully customizable an application is so difficult as foreseen all the possible requirements. However, this situation makes users contemplate other kind the requirements adaptation by extending Web applications through non-official mechanisms. Web augmentation is a clear example of that, and this is an example from the practice as well as from the academy.

In the practice, there are several Web Augmentation communities around of existing repositories. Most important communities (in terms of size) are related to two kinds of artefacts, userscripts and userstyles. The formers are JavaScript-based augmenters such as those described in this paper. Currently, there are several userscripts repositories: GreasyFork[2] (more than 5000 userscripts, most installed: 45.000 times), UserScripts.org[3] (more than 120.000 userscripts, most installed: more than 1 million times), OpenUserJS[4] (1475 userscripts, most installed: 198598), MonkeyGuts[5] (400 userscripts, most installed: 17095 times). The latters are CSS-based augmenters; consequently these are les powerful in terms of what may be adapted. However these are really popular as well. The main repository is UserStyles.org[6], hosting more than 60.000 userstyles. With all these existing repositories, it is clear that Web Augmentation is a current trend among the crowd of users. However, all these communities actually work without the intervention of Web site owners.

From the academy, in more recent years many work have investigate the potential of using End-User Programming techniques for allowing users to customize their applications [7][16]. Participation of the crowd of end-users is often presented as a suitable alternative for personalization, which often requires appropriate tool support [25] and methodological approach for personalization [11]. Indeed, many works such as [6] focus on tool support for allowing end-users to tune Web sites. The results are promising but the impact in terms of number of users that can be reached by such as an approach is limited, given the skill level required to build such applications [20].

Some studies [2][2827] have highlighted the importance of the involvement of communities of developers involved in the creation of scripts for adapting Web applications. Moreover, some authors [20] try to explain the role played by developers in the process.

Other approaches tackles frequent design and implementation issues that appear when developing Web augmentation artefacts. For instance, some authors have studied how augmenter may be more resilient to DOM changes or even to improve the augmenters' reusability (i.e. usable in several Web sites) [8][9]. The same authors have proposed a security model in order to control what augmenters could do in a Web site [1], which is clearly utilizable in our approach. Nonetheless, very few works

[2] GreasyFork: https://greasyfork.org/

[3] UserScripts.org: http://userscripts-mirror.org

[4] OpenUserJS: https://openuserjs.org

[5] MonkeyGuts: https://monkeyguts.com

[6] UserStyles: http://userstyles.org

have investigated the relationship and possible interactions between other actors involved in the process, namely the owners of Web sites being adapted by external scripts.

7 Conclusions and future works

In this paper we have presented a negotiated approach that involves end-users, owners of Web sites and external communities of coders specialized in the development of Web augmentation scripts that can be used to perform client-side adaptation of Web sites. The underlying idea behind of such as an approach is to share the tasks required for Web-side adaptation among the actors involved in the process and that, for the benefits of all. By exposing the advantages that all actors might found in process, our negotiated approach presents a new perspective for the research in the areas of Web scripts development and Web site adaptation. Indeed, we claim in this paper that a deep analysis of tradeoffs for all actors is an important activity for deciding design options for implementing Web applications. In this respect, the comparative analysis of advantages and drawbacks of adaptations approaches for each role should be understood as a contribution of this paper at its own right.

The negotiated approach is not a panacea and probably don't solve all adaptation problems that people might have in mind. Indeed, it is not aimed at replacing adaptation mechanisms that work pretty well and already fulfill a purpose. Nonetheless, we do claim that a negotiated approach opens up a new perspective for the research in the area in particular with respect to the way we involve actors in the adaptation process, in terms of tools required to support a highly distributed architecture for client-side adaptation and about mechanisms for observing the evolution of end-user needs for adaptation of Web site contents. Indeed, this work allows starting to investigate many interesting research questions that are of practical and theoretical importance for the Web engineering, for example: In which extension end-users are able to comprise and collaborate with Web site owners and communities of coders? How user's needs that require adaptation of Web sites evolve overtime? How communities of coders can make a bigger impact on existing Web sites? How to prevent those communities of coders can damage the presentation of Web site contents? How to improve trustful relationship between users that have different interests in the adaptation of Web applications? How to ensure long term compatibility between Web sites and external scripts?

It is evident that for supporting the approach, and ultimately the underlying research questions, appropriate tool support is necessary. For that we have developed a set of full-fledge tools that are publicly available and we invite the interested readers to take a look at the Web site http://UserRequirements.org for further information. The tools are fully functional and can be used either by end-users, community of coders and Web site owners as it was dully illustrated in the present paper. Since the availability of such tools is very recent, we still don't have collected enough material to make any assertion about the usability and/or user experience of people using these tools. Due to the inner nature of the negotiated approach, studies in a long run are required to make the necessary observation of all users and confirm if our hypothesis (dressed here merely as an estimation effort) hold on.

As part of our future work, we have already started to advertise the platform around the community so that we can have a substantial number of users (in different roles) for supporting further analysis. We are also planning to pursue the study about compatibility between scripts and Web sites.

References

1. Arellano, C., Díaz, O., Iturrioz, J. Crowdsourced Web Augmentation: A Security Model. In Proc. of WISE 2010. Springer LNCS 6488, pages 294-307.
2. Bigham, J., Ladner, R. Accessmonkey: a collaborative scripting framework for web users and developers. Proc. International Cross-Disciplinary Conference on Web Accessibility (W4A 2007), pp. 25-34.
3. Bouvin, N. O. Unifying Strategies for Web Augmentation. In: Proc. of the 10th ACM Conference on Hypertext and Hypermedia, 1999.
4. Brusilovsky, P. Adaptive Hypermedia. User Modeling and User-Adapted Interaction (UMUAI). Volume 11, Issue 1-2, pp. 87-110, 2001, Springer.
5. Card, S., Moran, T., & Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, NJ: Lawrence Erlbaum Associates. 448 pages.
6. Díaz, O., Arellano, C., Azanza, M. A language for end-user web augmentation: Caring for producers and consumers alike. TWEB 7(2): 9 (2013).
7. Díaz, O., Arellano, C., Aldalur, I., Medina, H., Firmenich, S. End-User Browser-Side Modification of Web Pages. In Proceedings of WISE (Thessalonoki, Greece), pp. 293-307, 2014.

8. Díaz, O., Arellano, C., Iturrioz, J. Interfaces for Scripting: Making Greasemonkey Scripts Resilient to Website Upgrades. ICWE 2010: 233-247.
9. Díaz, O., Arellano, C., Iturrioz, J. Layman tuning of websites: facing change resilience. WWW 2008: 1127-1128.
10. Diaz, O. Understanding Web Augmentation. In Proceedings of ICWE Workshops (Berlin, Germany), pp. 79-80, 2012, Springer.
11. Arellano, C., Díaz, O., Iturrioz, J. Opening Personalization to Partners: An Architecture of Participation for Websites. In Proceedings of the International Conference on Web Engineering (ICWE 2012), LNCS 7387, pp. 91–105, 2012.
12. Iturrioz, J., Azpeitia, I., Díaz, O. Generalizing the "like" button: empowering websites with monitoring capabilities. In Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14). ACM, New York, NY, USA, 743-750.
13. Firmenich, S., Winckler, M., Rossi, G. A Framework for Concern-Sensitive, Client-Side Adaptation. ICWE 2011, Paphos, Cyprus, June 20-24, 2011. Springer, LNCS 6757, pages 198-213.
14. Firmenich, S., Rossi, G., Winckler, M., Palanque, P. An approach for supporting distributed user interface orchestration over the Web. Int. J. Hum.-Comput. Stud. 72(1): 53-76 (2014).
15. Firmenich, D., Firmenich, S., Rivero, M., Antonelli, L. A Platform for Web Augmentation Requirements Specification. In proceedings of ICWE (Toulouse, France), pp. 1-20, 2014, Springer.
16. Firmenich, S., Rossi, G., Winckler, M. 2013. A domain specific language for orchestrating user tasks whilst navigation web sites. In Proceedings of the 13th International conference on Web Engineering (ICWE'13), Florian Daniel, Peter Dolog, and Qing Li (Eds.). Springer-Verlag, Berlin, Heidelberg, 224-232.
17. Firmenich, S., Gaits, V., Gordillo, S., Rossi, G., Winckler, M. Supporting Users Tasks with Personal Information Management and Web Forms Augmentation. International Conference on Web Engineering 2012 (ICWE). LNCS7387. p:268-282. Springer.
18. Garrido, A., Firmenich, S., Rossi, G., Grigera, J., Medina-Medina, N., Harari, I. Personalized Web Accessibility using Client-Side Refactoring. IEEE Internet Computing 17(4): 58-66 (2013).
19. Han, H., Tokuda, T. Towards flexible and lightweight integration of web applications by end-user programming. IJWIS 6(4): 359-373 (2010).
20. Jones, M. C., Churchill, E. F. 2009. Conversations in developer communities: a preliminary analysis of the yahoo! pipes community. In Proceedings of the fourth international conference on Communities and technologies (C&T '09). ACM, New York, NY, USA, 195-204.
21. Kennedy, K., Koelbel, C., Schreiber, R. Defining and Measuring the Productivity of Programming Languages. IJHPCA 18(4): 441-448 (2004).
22. Ko, A., Myers, B., Rosson, M., Rothermel, G., Shaw, M., Wiedenbeck, S., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M. The State of the Art in End-User Software Engineering. ACM Computer Surveys 43(3):21, pp. 1-44, 2011, ACM.
23. Kobsa, A. Generic User Modeling Systems. In The Adaptive Web, Methods and Strategies of Web Personalization, pp. 136 – 154, 2007, Springer.
24. Malone, T. W., Crowston, K. The interdisciplinary study of coordination. ACM Comput. Surv., 26(1):87–119, 1994.
25. Nebeling, N., Speicher, M., Norrie, M. C. 2013. CrowdAdapt: enabling crowdsourced web page adaptation for individual viewing conditions and preferences. In Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems (EICS '13). ACM, New York, NY, USA, 23-32.
26. Pazzani, M., Billsus, D. Content-Based Recommendation Systems. In The Adaptive Web, Methods and Strategies of Web Personalization, pp. 325-341, 2007, Springer.
27. Rossi, G., Schwabe, D., Guimarães, R. Designing personalized web applications. In Proceedings of the 10th international conference on World Wide Web (WWW'01). ACM, New York, NY, USA, 275-284.
28. Stolee, K.T., Elbaum, S., Sarma, A. 2013. Discovering How End-User Programmers and Their Communities Use Public Repositories: A Study on Yahoo! Pipes. Information and Software Technology 55(7):1289–1303. Retrieved October 9, 2014 (http://linkinghub.elsevier.com/retrieve/pii/S095058491200211X).
29. YouTubeCenter (2015). Available at: https://greasyfork.org/es/scripts/943-youtube-center, last access: 12/2/2015
30. The Drive Platform. Available at: https://developers.google.com/drive/ Last access: last access: 12/2/2015
31. The Facebook API. Available at: https://developers.facebook.com/ Last access: last access: 12/2/2015
32. Platypus. https://addons.mozilla.org/es/firefox/addon/platypus/, last access: 12/2/2015
33. GreasyFork. https://greasyfork.org, last access: 12/2/2015
34. UserScripts. http://userscripts-mirror.org, last access: 12/2/2015
35. GreaseMonkey. http://www.greasespot.net, last access: 12/2/2015