Web Augmentation as a Promising Technology for End User Development

Iñigo Aldalur, Marco Winckler, Oscar Díaz, Philippe Palanque

Web Augmentation as a Promising Technology for End User Development

Iñigo Aldalur[*], Marco Winckler[§], Oscar Díaz[*] & Philippe Palanque[§]
[*]University of the Basque Country (UPV/EHU), San Sebastián (Spain)
{inigo.aldalur, oscar.diaz}@ehu.eus

[§]ICS-IRIT, University of Toulouse, Toulouse (France)
{winckler, palanque}@irit.fr

Abstract: This chapter presents Web Augmentation (WA) technologies as tools and techniques for end-user development. WA technologies differ from other web development technologies as they target at improving existing Web pages and not at creating new Web sites. These improvements can deeply alter the way users use and interact with Web sites. This chapter revisits the concept of WA and provides an overview of the main features that characterize WA technologies. This characterization is used to position and compare the various contributions that have been made in WA. To make things more concrete we provide an illustration of WA technology through a case study using a dedicated tool called WebMakeup. Despite all their advantages, WA technologies present some limitations that might result in challenges on the user side. These aspects are also presented and discussed, highlighting directions for future work in that domain.

Keywords: End-User Development, Web Augmentation, Web Adaptation,

1. Introduction

Nowadays, many applications which, formerly, would have been designed for the desktop such as calendars, travel reservation systems, purchasing systems, library card catalogs, maps viewers or even games have made the transition to the Web, largely successfully. Many Web sites are created every day to help users to find information and/or to provide services they need. However, there are cases where rather than a new Web site, what users need is to combine information or services that are already available but scattered on the WWW. Some examples follow: (1) users who want to have additional links on a Web page to improve the navigation (for example to create a personalized menu that gathers in one location multiple personal interests), (2) users who need to integrate contents from diverse Web sites (for example to include a Google's map into a Web page that originally only shows addresses as flat text) in order to improve their performance in identifying distance from their personal location or (3) simply to remove content from Web pages (such as contact details they consider irrelevant) to improve reading and selection performance as identified by Hick's law [43]. Because these needs might be perceived as idiosyncratic, volatile (being short-lived or occasional) or dissenting with the interests of the Web site, they might well not be considered (or even not known) by Web developers [33]. This is because Web sites are, by definition, designed for the masses and that at design time only few users are available.

Previous work on End-User Development (EUD) [48][54] has demonstrated that, if appropriate tools are provided, end users might be able to create what they need (or at least define more precisely part of what they need). DENIM is a pioneer example that illustrates how tools can be used for involving users into the design of the Web sites to be developed [58]. A more demanding scenario is when the target is not in-home Web sites but Web pages that have already been created by third parties. The options are here, either to redevelop what has already been done by the third party or to try to convince the third party to tune its development to fit a particular user need. This deeply collides with the principle of Web development that target the masses and not the individual.

The term Web Augmentation (WA) is used to describe tools that can be used to improve (hence the word "augment") existing Web pages (found for instance whilst browsing the Web) to create better fit user's needs and activities. Some of the most popular WA tools work by extending the functionalities of the Web browser used by the user via plug-ins that can run client-side scripts to manipulate the structure of Web pages loaded in the browser. In that case the augmentation will be applied to all the visited Web page featuring specific characteristics. The potential of WA techniques can be illustrated by some advanced applications such as lightweight integration of information extracted from the Web, context-sensitive navigation across diverse Web site, context-dependent multimodal adaptation [36] or refactoring Web sites for accessibility [35]. Another example is a spellchecking plug-in that would automatically check the text entered by the user on any Web page. The degree of expertise required for using WA tools varies dramatically [42]. For example, some tools only require basic knowledge of how to install plug-ins in the Web browsers while others may require integrating sophisticated scripting code created by the user.

In this chapter, we examine the potential of WA technology for supporting end-user development for the Web. In section 2, we discuss the relationship between WA and end-user development. Section 3, proposes a classification of WA technologies, positions existing tools with respect to this classification and provides a study of research contributions for each main category of the classification. To make things concrete, section 4 illustrates how the WebMakeup WA tool relates to the classification using a case study based on augmentation of the dblp computer science researchers' publications repository. In section 5, we explain some of the users and usage difficulties specific to the adaptation of Web applications. Section 6 concludes the paper and highlights possible directions for future work.

2. Web Augmentation and End-User Development

Web Augmentation (WA) is *not* End-User Development (EUD) for the Web but some of the features provided by WA tools can be used for that purpose. To highlight similarities and differences, we revisit their definitions.

Many authors have tried to define precisely the term end-user programming [13][74]. In this chapter, we adhere to the definition provided by Ko et al. [50] who state that "*end-user programming is programming to achieve the result of program primarily for personal, rather than public use*". That definition has many implications. First, it is important to note the absence of any reference to an application domain and/or technology highlighting the large scope for the use of EUD tools. Next, the term "programming" refers to a general activity, which might encompass the development of software from scratch and/or making modification to an existing software. Finally, the term "end-user" does not refer to the user's skills in so for as a professional developer is engaged in end-user programming when writing code to fulfill a personal need, such as visualize the data structure to help diagnose a bug. Moreover, even if the definition implies a particular intention behind the development of the program, it does not exclude the possibility of sharing the program with other users.

There are fewer attempts to define precisely the term Web Augmentation. This term was originally coined by Bouvin in 1999 [11] to describe a tool that "*through integration with a Web browser, a HTTP proxy or a Web server adds content or controls not contained within the Web pages themselves to the effect of allowing structure to be added to the Web page directly or indirectly, or to navigate such structure. The purpose of such a tool is help users organize, associate, or structure information found on the Web. This activity may be done by a single user or in collaboration with others*". More recently, Díaz [24] said that "*WA is to the Web what Augmented Reality is to the physical world: layering relevant content/layout/navigation over the existing Web to customize the user experience*". These definitions highlight WA as a non-intrusive approach: augmentations are "layers" on top of an existing content. These augmentation layers might be needed to cater for situational and idiosyncratic needs, difficult for designers to foresee. Technically, augmentations do not need the participation of the Web sites used for the augmentation since the augmentation occurs on the Web browser. Web augmentation technology only acts on the user interaction and does not change the original Web page stored on the Web server. It is interesting to note that whilst Bouvin does not assign any particular intention for the use of WA tools, Díaz explicitly mentions that augmentation layers might aim at improving the user experience with the Web page.

For our purposes, WA describes *tools that allow people to modify Web pages to improve user performance and satisfaction*. This definition connects WA to EUD as EUD "*is programming to achieve the result of program primarily for personal, rather than public use*". Indeed, WA realizes this vision in the web sphere as far as it helps to support users' needs that have not been originally been identified or taken into account during the design of the Web site.

3. Overview of EUD tools for the Web

The evolution of Web technology is changing the way users interact with Web sites. At first, users could only consume contents provided by Web sites. Later, users could actively contribute with content by using tools such as CMS and wikis. More recently, WA tools empower people in different ways making these tools real EUD tools: (1) to create their own web sites, (2) to combine information from diverse Web sites into a single hub (using mashups), and even (3) to modify Web pages created by others (using WA tools e.g. MADCOW [10] and DiLAS [3]). This highlights the broad range of approaches that Web-centered EUD tools explore. Figure 1 introduces a set of dimensions to classify these tools while the positioning of existing tools with respect to this classification is shown in Table 1.

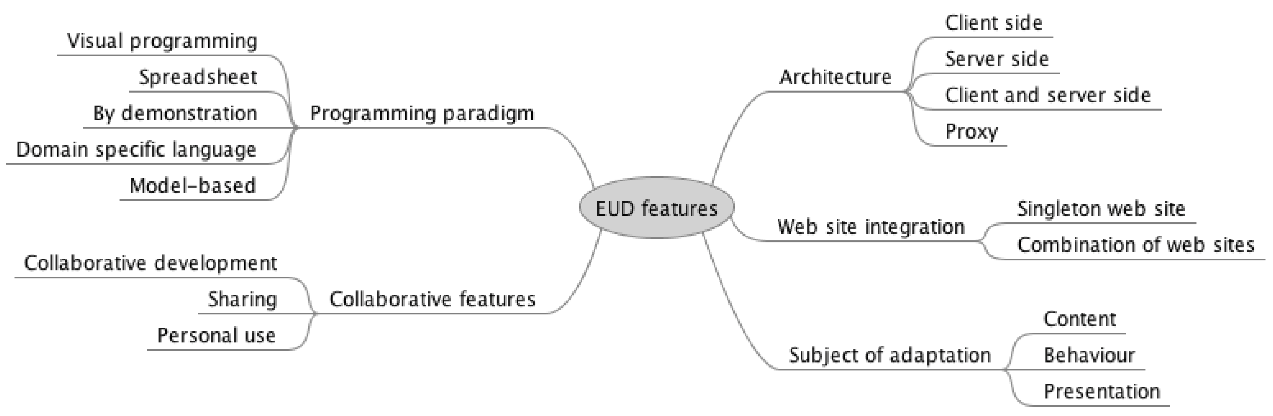


***Figure 1. Five EUD features** of WA tools and their attributes*

Tools	Year	Type	Architecture			Subject of adaptation			Web site Integration	Collaboration features	Programing Paradigm	Ref.
			C	S	P	Co	Be	Pr				
Marmite	2007	M	C			Co			Combination	Personal use	Visual program.	[73]
MARGMASH	2007	**WA**	C			Co			Combination	Personal use	By demonstration	[25]
CoScripter	2008	M	C			Co	Be		Singleton	Collaborative dev.	By demonstration	[53]
Reform	2009	**WA**	C			Co			Combination	Personal use	By demonstration	[69]
SemanticWebPipes	2009	M		S		Co			Combination	Sharing	Visual program.	[61]
Mashroom	2009	M	C			Co			Combination	Personal use	Spreadsheets	[71]
Deep	2010	M	C			Co		Pr	Combination	Personal use	By demonstration	[41]
MashSheet	2010	M	C			Co			Combination	Collaborative dev.	Spreadsheets	[44][45]
Atomate	2010	M	C			Co			Combination	Collaborative dev.	Model-based	[49]
RUMU	2010	**WA**		S		Co		Pr	Singleton	Personal use	Visual program.	[62]
CSN framework	2011	**WA**	C			Co	Be		Combination	Sharing	By demonstration	[31]

OntoCompo	2011	M	C			Co	Be		Singleton	Personal use	Model-based	[12]
Mixer	2011	**WA**	C			Co			Combination	Sharing	By demonstration	[34]
IVO	2011	M	C	S		Co	Be		Singleton	Sharing	By demonstration	[65]
MashupEditor	2011	M			P	Co			Combination	Sharing	By demonstration	[37][38]
DashMash	2011	M	C/S			Co	Be		Combination	Personal use	Visual program.	[14][16]
MAIDL	2011	M	C/S			Co			Combination	Personal use	By demonstration	[17]
VisPro	2011	M	C/S			Co	Be		Combination	Personal use	Visual program.	[9]
SOA4All Studio	2011	M	C/S			Co	Be		Combination	Sharing	Visual program.	[70]
Cowpath	2012	**WA**	C				Be		Combination	Sharing	**DSL**	[26]
WebCrystal	2012	**WA**	C			Co		Pr	Combination	Personal use	By demonstration	[19]
Baya	2012	M	C			Co			Combination	Sharing	Visual program.	[20][22]
ResEval Mash	2012	M	C/S			Co			Combination	Sharing	Visual program.	[47]
CrowdDesign	2012	M	C/S			Co	Be		Combination	Sharing	Visual program.	[57]
Chudnoskyy et al.	2012	M	C			Co			Combination	Sharing	Visual program.	[21]
MOWA	2013	**WA**	C			Co			Combination	Sharing	Model-based	[18]
Sticklet	2013	**WA**	C			Co			Combination	Sharing	**DSL**	[7][24]
Social Overlays	2013	**WA**	C			Co		Pr	Singleton	Sharing	Visual program.	[27]
openHTML	2013	**WA**		S		Co		Pr	Singleton	Collaborative dev.	By demonstration	[60]
Ardito et al. (a)	2013	M		S		Co		Pr	Combination	Sharing	Visual program.	[4]
MobiMash	2013	M		S		Co	Be		Combination	Personal use	Visual program.	[15]
DireWolf	2013	M		S		Co	Be		Combination	Collaborative dev.	Visual program.	[51]
Rana et al.	2013	M		S		Co			Combination	Personal use	Visual program.	[64]
CapView	2013	M		S		Co	Be		Combination	Personal use	Visual program.	[63]
WebMakeup	2014	**WA**	C			Co	Be	Pr	Combination	Sharing	Visual program.	[23]
CrowdMock	2014	**WA**	C			Co	Be		Combination	Collaborative dev.	Visual program.	[29]
Ardito et al. (b)	2014	M		S		Co			Combination	Sharing	Visual program.	[5][6]
MultiMasher	2014	M		S		Co			Combination	Sharing	Visual program.	[46]
NaturalMash	2014	M	C	S		Co			Combination	Sharing	By demonstration	[2]
SmartComposition	2014	M	C	S		Co			Combination	Sharing	Model-based	[52]
Tayeh et al.	2014	**WA**	C			Co			Singleton	Personal use	Visual program.	[67][68]
FaceMashup	2015	M		S		Co			Singleton	Personal use	Visual program.	[55]
IWC	2015	M		S		Co	Be		Combination	Sharing	By demonstration	[59]
MAMSAAS	2015	M		S		Co			Combination	Sharing	Visual program.	[72]
EasyApp	2016	M	C	S		Co	Be		Combination	Personal use	Visual program.	[75]
MOWA/WOA	2016	**WA**	C			Co	Be	Pr	Combination	Collaborative dev.	By demonstration	[8][28]
Miján et al.	2016	**WA**	C			Co	Be		Singleton	Sharing	Visual program.	[56]

Legend: M: Mashup, **WA**: WA | C: client side, S: server side, C/S: both client and server sides, P: proxy | Co: content, Be: behavior, Pr: presentation | DSL: domain specific language|

Table 1. EUD tools for the Web positioned with respect to the classification in **Figure 1**.

Although the focus is on WA tools, we also introduce mashup tools because this provides some elements of comparison between the existing approaches for EUP of the Web.

Mashup technology is an interesting alternative for final users to combine existing resources and services in a new Web application [1]. Mashups are often very specialized and only operate with specific types of contents (quite often structured data sources). For example, FaceMashup [55] is a EUD tool for mashup that allows users to manipulate social network APIs to combine data and sharing them with other users through the social networks. It is interesting to notice that some WA tools such as CSN Framework [31] borrow from mashups the ability to integrate contents but they are even more flexible allowing to compose any kind of DOM element from a Web page.

Tool wise, **Figure 2** highlights how mashups (66%) have received more attention throughout w.r.t. WA tools (34%). This seems to suggest that integrating different data sources is being considered more important than customizing existing Web sites. Though this might be true in a general sense, when it comes to empowering end-users, data integration might be more costly and hence, more difficult to end users to achieve. By contrast, WA is not so demanding, and hence more affordable to end-users. This makes WA tools more likely to be adopted by end users.

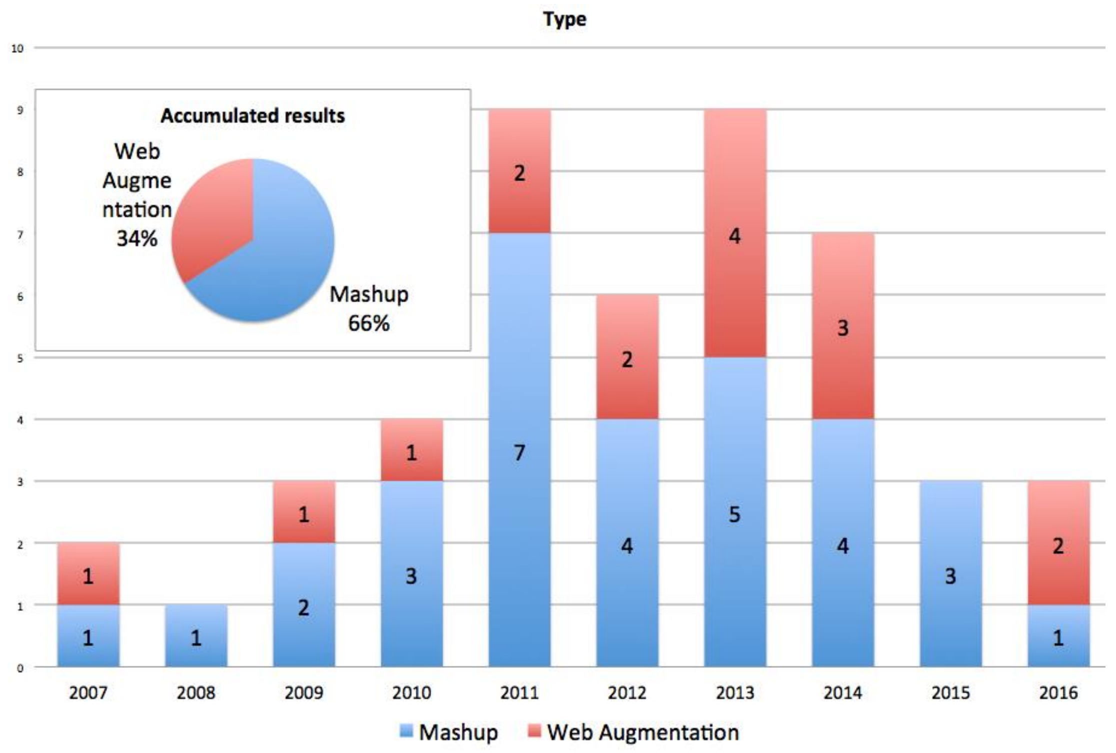


Figure 2. Contributions presenting tools: Mashup versus WA technology

The rest of this section explains the classification presented in **Figure 1** and provides examples of the corresponding Web technology.

3.1 Architecture

Tools might rest on the client side, the server side or both. Client-side tools are executed as Web browsers' extensions (or plug-ins) and processing happens on the user's local computer. Common programming languages used to implement client-side applications include HTML, CSS, and Javascript. Conversely, server side technology runs on a remote machine, and only the outcome of the execution returns to the user's local computer. Common programming languages include Ruby, Python, PHP, C#... Server side technologies can store persistent data. However, data can only be accessed than through HTTP requests for a particular URL.

Miján et al. [56] and WebCrystal [19] illustrate the client-side approach. WebCrystal is a Firefox plug-in that allows the inspection of code corresponding to visual objects. WebCrystal provides users feedback using a textual description and a customized code snippet that can be copied-and-pasted to rebuild the user-selected properties. Additionally, Miján et al. resort to a set of personalization rules to be applied in the client-side with minimum alterations defined without requiring either advanced programming skills or advanced configuration.

Whilst Web browsers can store data in the local cache, server-side technology is used by many tools such as DireWolf [51], FaceMashup [55], Ardito et al. [4] and MultiMasher [46] as a means to support data persistence. DireWolf provides several extensible components for adapting Web sites and it implements a service for data persistence such as user device profiles and shared application states.

As for client-server tools, most requests a kept in the client with sporadic calls to the server. For example, DashMash [16] has a client-side module for mashup creation and a server module responsible for integrating and storing data from different types of services. In the mobile world, IVO [65] follows a similar architecture. For mashups, MashupEditor [37][38] allows for adaptations to be created on the client (using a dedicated editor). Next, a proxy server store those adaptations that can be later reused during the creation of the mashup.

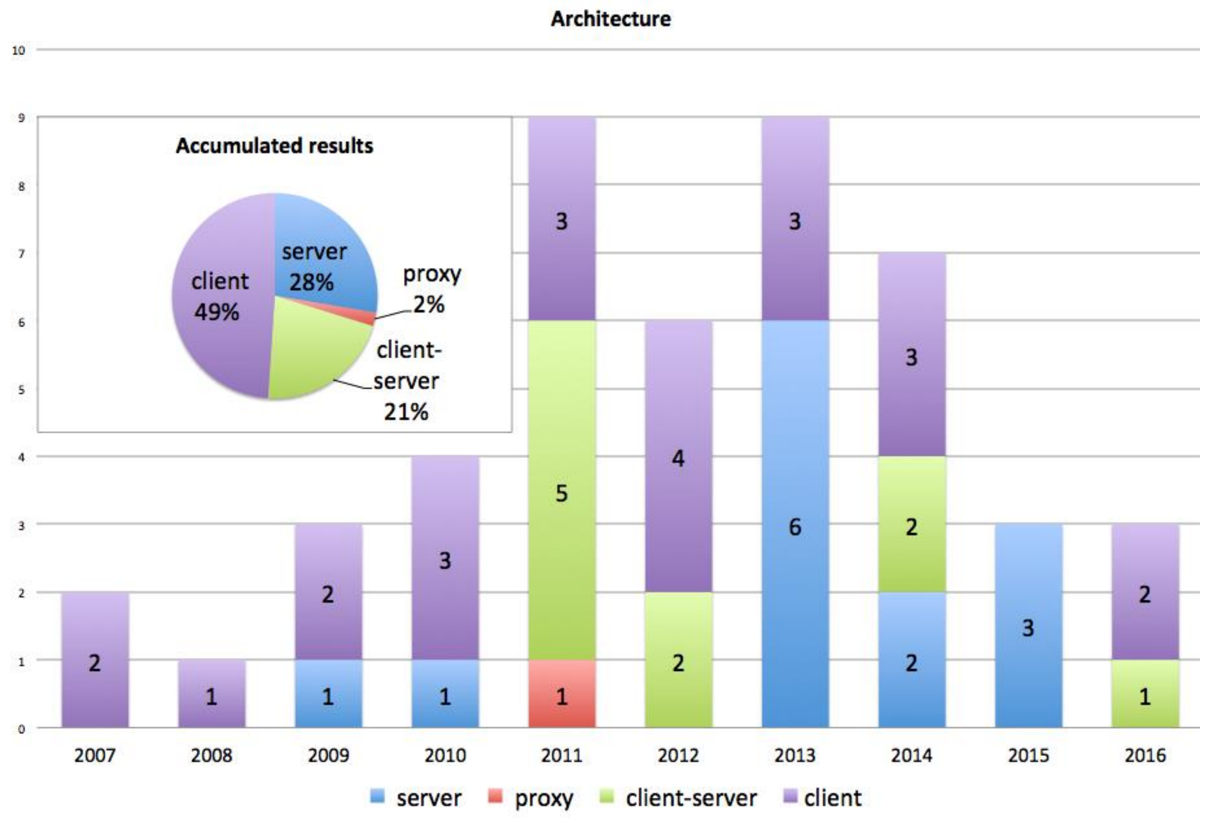


Figure 3. Distribution over the years of tools and what part of the Web architectures they were exploiting

From the accumulated results in **Figure 3**, it is clear that the client-side approach is the most popular architecture (49%). The Client-server option (21%) boosted in 2011, presumably due to the popularity of the Web 2.0 and the focus on sharing and the need to have common repositories. The server-side option (28%) rose from 2013 onwards, arguably on the search for a business model for mashup platforms.

3.2 Subject of adaptation

Web sites might be adapted in different ways: including brand-new content, changing the behavior associated to DOM elements or altering the appearance (style and layout). Most tools provide functions to add/remove/replace contents. Adding content from other sources is often used as a means for making information readily available whilst removing content is useful to improve focus, preventing users from distraction. Mixer resorts to WA to improve the organization of Web pages simply by letting users to move contents around and include/exclude contents needed. Mashups are also used to add content from different websites. SmartComposition [52] is another content-based approach that is primarily used to build mashups but it also features unique functions that allow to reorganize contents to fit into different screen sizes. Chudnoskyy et al. [21] take a step forward by assisting users with recommendations and automatic composition.

Whilst modifying CSS code (color, font, etc.) is relatively simple, few tools account for this kind of adaptation. RUMU [62] is a web-based WYSIWYG editor that resorts to a semantic language to change the page style and simplifying web design. OpenHTML [60] is also a web editor to introduce laymen into HTML and CSS.

Finally, changing the behavior of Web sites is far from trivial. It often requires adding some Javascript code to DOM elements like show or hide web nodes, click on certain button, change the content of an element, etc. Changing the behavior of web sites might be necessary, for example, for automating repetitive tasks. Inter-Widget Communication (IWC) [59] is a semi-automatic, end-user friendly approach to extend widgets employing the programming-by-demonstration paradigm. IWC is built by composing interactive widgets. IWC leaves users with the tedious task of manual wiring widgets to create mashups. SOA4All [70] is a visual development environment that addresses adaptation of Web applications through the connection of different service components into an assembly line.

3.3 Web site integration

This dimension tells if users work with one (singleton) or more (combination of) Web sites in a single project. Whilst many EUD tools are designed to augment a particular type of singleton Web site (e.g. OpenHTML), some tools allow to mix content from diverse Web sites.

Mashups tools like Baya [20], Deep [41], MamSaas [72] and Marmite [73] are typical examples of tools that allow to extract data from different Web sites and recombine them in a form that better fulfill user's needs. Nonetheless, other strategies combine Web sites that don't necessarily involve structured data sets. For example, Ardito et al. [6] is a platform for end users to compose personal information spaces by assembling pieces of information from different sources. Such personal information spaces can be enacted in different devices and shared with other users. MamSaas is a layered architecture to deploy and identify mashup components as well as link and execute mashups for quick application development. MOWA [8] is another EUD tool for WA that enables end users to create a custom guided tour of a city based on contents collected from diverse Web sites. Its aim is to augment existing Web applications with mobile features. Using MOWA end users can pinpoint in a map content from a different Web site and then generate a custom script. This mobile Web application prompts the users add points of their interests while they move around the city.

Finally, CrowdDesign [57] can also be classified as a EUD tool in so far as it supports mashup based on the integration of scripts coming from diverse sources. CrowdDesign works as a storage for scripts and user interface components shared by a community of developers. CrowdDesign also features a visual authoring environment that allows users to combine contents and scripts available at the platform to create a more personal version of Web sites.

3.4 Collaborative features

Whilst a WA strategy can be adopted only for personal purposes, sharing is an important aspect of end-user development [54][66]. We distinguish between sharing and collaborative development.

Sharing. Some tools focus on personal use, i.e. results cannot be reused and/or shared with other users. Tayeh et al. [68] is a case in point. These authors provide a tool for the linking and the integration of arbitrary documents and multimedia content dynamically. Rana et al. [64] and EasyApp [75] are also tools for personal use. Both tools provide a systematic way of designing, developing and deploying personalized apps. Reform is a Firefox extension that contributes with architecture for web enhancement that allows end users to integrate existing enhancements with new websites. Despite the fact that it allows end users to communicate with developers for requesting new features, they do not allow sharing developments. CapView [63] is a mashup platform that provides instant feedback for user development actions. CapView helps non-programmers form components with recommendations provided by the system and it manipulates a mashup through visually composing component features.

Moving away for the personal realm, Social Overlays [27] and the CSN framework [31] illustrate the use of repositories for script sharing. Social Overlays focuses on repairing either the behavior or the appearance of Web sites. Updates made by individuals are visible to the community which use a voting mechanism to decide if the updates are relevant and if so, be incorporated as part of the Web site offerings. CSN features a plug-in that allows users to adapt Web pages by triggering different types of scripts. It has different features depending on the user profile: developer or end user. Developers can write new augmentation scripts to extend the set of original sets of scripts available in the framework. Such scripts can then be obtained by other users who on their turn can execute them to adapt the Web sites. Finally, it is interesting to notice that a few tools allow to publish the code in social networks (e.g. Sticklet [7][24]) whilst others allow to export files for personal use on an individual basis (e.g. WebMakeup).

Collaborative development. CrowdMock [29] does not provide a voting mechanism but it permits to amend/complete augmentation script by people other than the author. CoScripter [53] resorts to programming by demonstration to enable users to record all the information related to user interaction to edit a Website. The outcome is a script macro that can be automatically stored in a Web server from where they can be delivered to other users and they can use a collaborative scripting environment for recording, automating, and sharing web-based processes.

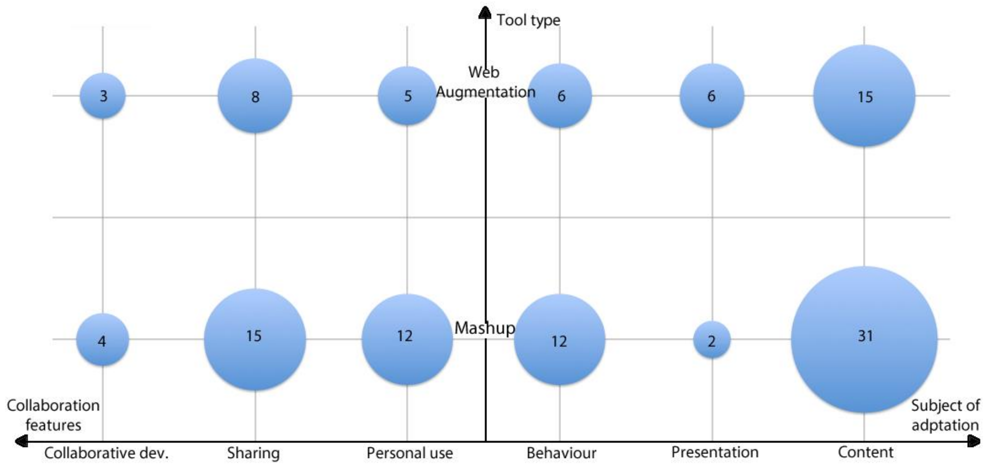


Figure 4. Mapping WA tools and Mashups across "Collaboration features" and "Subject of adaptation"

Figure 4 helps to apprehend differences and similitudes between WA and mashups as for "collaboration features" and "subject of adaptation" support. As for the former, both scenarios (i.e. WA and mashups) pay attention to the idiosyncratic scenario ("Personal use"), while the potential of reuse (i.e. sharing) is felt to be more intensive for mashups than for WA developments. Also, mashups and WA coincide in their interest in handling content (31 vs. 15) while WA underscores in addressing presentation concerns (2 vs. 6). This is according to expectations since WA adapts existing web sites whose presentation might need to be tuned to better meet users' needs. By contrast, behavior modification has received more attention in the mashup realm.

3.5 Programming paradigm

EUD tools resort to diverse programming paradigm: visual languages, spreadsheets, programming by demonstration, domain specific languages (DSL) and model-based automation [1].

Visual programming is mainly found in mashup tools that allow drag-and-drop to connect components to create a mashup. Examples include VisPro [9], ResEval Mash [47], MobiMash [15], SemanticWeb Pipes [61] and WebMakeup [7][23]. VisPro creates mashups by dragging and dropping widgets from a library. ResEval Mash is a domain-specific mashup tool that explores dedicated mashuping, in this case in the domain of research evaluation. MobiMash resorts to visual notations to create mobile mashups. The particularity of SemanticWeb Pipes is to blend mashups and the Semantic Web. Here, ontologies are used for better matching widgets parameters that build up the mashup. WebMakeup is an editor that delivers Chrome plugs-in for augmentation purposes. A DSL is defined that sets the expressiveness of the augmentation. WebMakeup helps construct DSL expressions on top of the page being augmented. Once constructed, WebMakeup generates and installs the corresponding Chrome extension.

Programming by demonstration is most popular for data extraction and visualization, where service composition and orchestration play an ancillary role. NaturalMash [52], WOA [28], Margmash [25] and MAIDL [17] illustrate this approach

NaturalMash is a WYSIWYG mashup tool. NaturalMash stands out for its formative support where the tool is able to collect user feedback. WOA enables users to create/extract Web contents in the form of objects that they can manipulate to create Personal Web experiences. Margmash creates augmentations out of personalized information, which are gathered from diverse Web sites. Margmash behaves as a lightweight wrapper that guides end users on both data gathering and data recombination. MAIDL permits the rapid creation of mobile mashup out of components.

Model-based Automation is concerned with the automatic creation of mashups out of knowledge about the user and the context of use. This technique's weakness is the risk of generating irrelevant mashups w.r.t. the given requirements. Ontocompo [12] and Atomate [49] illustrate this approach. Ontocompo makes use of an ontology to generate new applications based on existing ones. Atomate is a personal information assistant engine that automatically carries out tasks for the user. Atomate combine RSS/ATOM feeds from social networking into a simple RDF model representing people, places and things.

DSLs strive to abstract from general-purpose programming language. The challenge here is to find a compromise between expressiveness and learnability. DSLs in the augmentation realm can be illustrated by Cowpath [26] and Sticklet [24]. Cowpath focuses on "Web trails", i.e. recurring navigation paths across distinct Web sites. Rather than switching between tabs and typing once and again the same URLs, Cowpath augments the affected websites with additional hyperlinks that "pave the way" of these Web trails. On the other hand, Sticklet explores the use of a dedicated assistant that help users to come with Sticklet expressions to augment Web sites.

Spreadsheets-like programming are often considered ease-of-use, intuitive and with enough expressive power to represent and manage complex data. When it comes to mashups, Mashroom [71] and MashSheet [44][45] explore this approach. Mashroom builds Web applications by combining content coming from different Web sites. To this end, it resorts to an expressive data structure and a set of defined mashup operators. The data structure allows users to express complex data objects while mashup operators are visualized in the formula bar. MashSheet extends conventional spreadsheet paradigms to facilitate Web services "mashup" in a spreadsheet environment. MashSheet is a collection of operators that supports orchestrating Web services, manipulating and visualizing data created by the services.

Figure 5 depicts the distribution of research contributions with respect to the "programming paradigm" feature over the years. Visual programming is by far the most popular approach (53%), where the other approaches fall behind: programming by demonstration (30%), Model-based (9%), DSL (4%) and spreadsheets (4%). Worth mentioning, the boost of programming-by-demonstration in 2011 although it faded over the years.

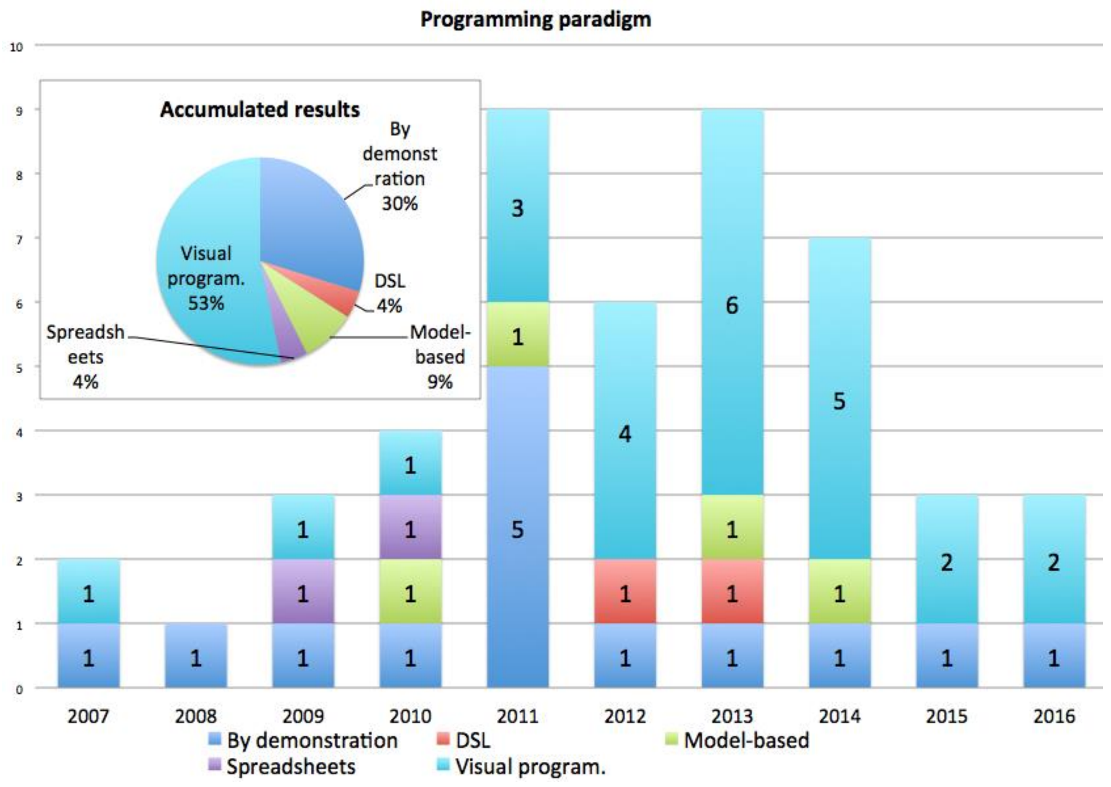


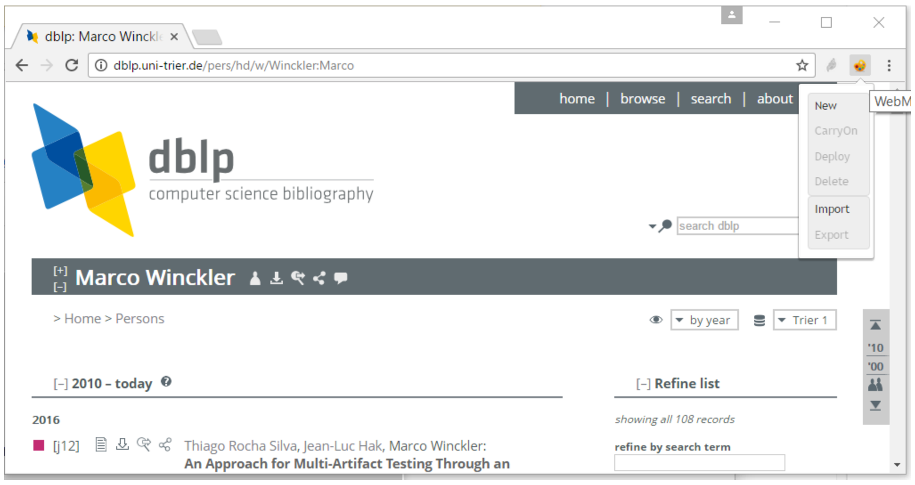
Figure 5. "Programming paradigm" in research contributions over the years

4. Web Augmentation: a case study with WebMakeup

This section illustrates WA at work using WebMakeup [23]. This tool supports the modification of the content, the presentation, and the behavior of Web pages. Moreover, it also supports the integration of dynamic content from other web sites. So far, WebMakeup only work for the Chrome browser. A video is available at `https://vimeo.com/204338864`.

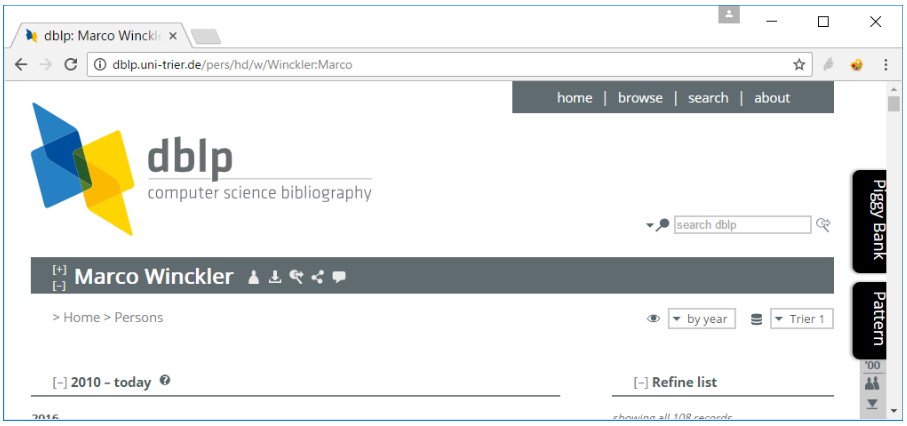
4.1 Architecture

WebMakeup is a plug-in freely available at the Chrome Web Store[1]. Once installed, it can be activated at any time by selecting the icon in the top-right side of the address bar as shown by **Figure** *6*.a. By selecting the option "New" from the pop-up menu, two vertically aligned tabs called "Piggy Bank" and "Patterns" appear (see Figure 6.b).


dblp: Marco Winckl
dblp.uni-trier.de/pers/hd/w/Winckler:Marco
home | browse | search | about
New
CarryOn
Deploy
Delete
Import
Export
WebM
dblp
computer science bibliography
search dblp
Marco Winckler
> Home > Persons
by year
Trier 1
[–] 2010 – today
[–] Refine list
2016
showing all 108 records
refine by search term
[j12]
Thiago Rocha Silva, Jean-Luc Hak, Marco Winckler:
An Approach for Multi-Artifact Testing Through an

a) Launching WebMakeup to create a new augmentation layer on top of DBLP.

[1] Available at: https://chrome.google.com/webstore/detail/alnhegodephpjnaghlcemlnpdknhbhjj



b) Tab menus vertically aligned at right-side (collapsed)

Figure 6. WebMakeup main menus.

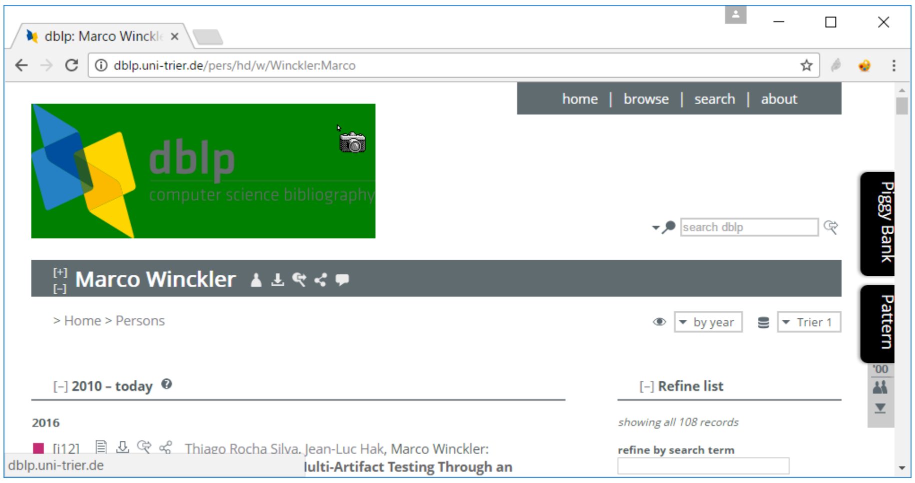
WebMakeup is a client-side application developed using JavaScript. Scripts created by the user are stored in the Web browser, so persistence can be ensured as far as the user does not clear the local cache.

4.2 Subject of adaptation

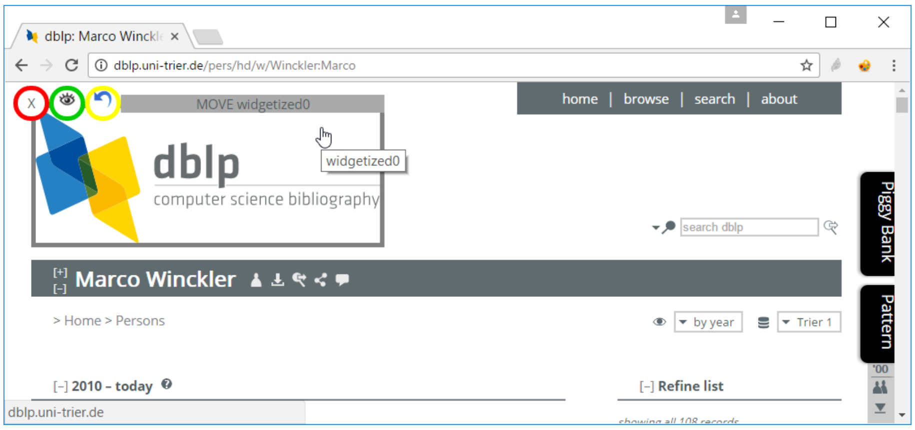
WebMakeup allows users to modify the contents, the presentation and the behavior of existing Web pages through the manipulation of the DOM elements that conform the Web page. Only after selecting a DOM element, it is possible to manipulate it: remove, re-arrange or change its behavior.

4.2.1 Selecting DOM elements in a Web page

As shown in **Figure 7** WebMakeup highlights the underlying DOM through two visual elements: the pointer, which becomes a small camera, and the background color, which is turned into green. By clicking on the green zone, the corresponding DOM element is selected and transformed into a widget. Widgets are framed by "decorators", i.e. frames that include three button (see Figure 7.b): the red-circle button removes the DOM element at hand; the green-circle button changes the visibility of the DOM element from hide to show, and vice versa; finally, the yellow-circle button unselects the DOM element, removing the decorator frame..



a) Selection of DOM elements using mouse over operation on a Web page.



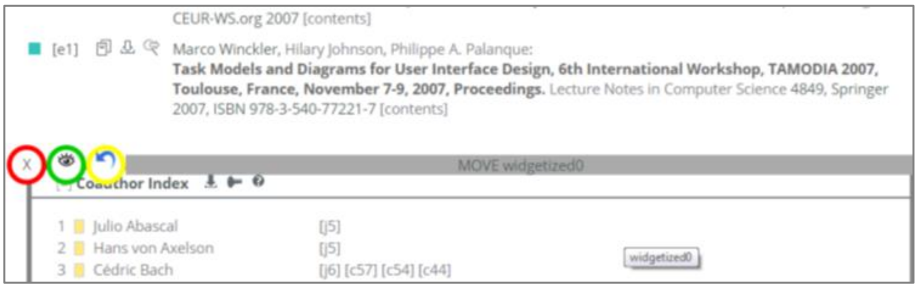
b) DOM element selected (after click) showing options for inspecting it.

Figure 7. Selection of the DOM element using WebMakeup.

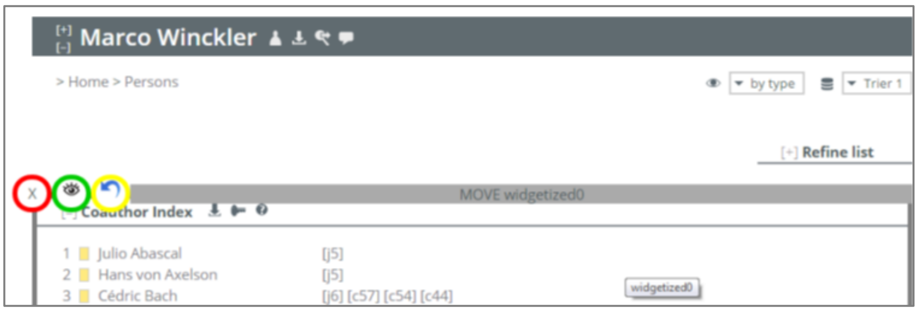
In this way, users can remove elements from Web pages to accomplish diverse personalization needs. For example, removing short papers from the DBLP page might help highlight other types of publications. However, the red-circle button (see Figure 7.b) only removes the corresponding DOM element from the current session. For changes to become permanent (i.e. enforceable in future visits to the DBLP Web site), users should "deploy" the WebMakeup script by clicking on the namesake option in the scrollable menu (third item at **Figure 6**.a).

4.2.2 Re-arranging contents around the Web page

Another way to highlight content is to place it in a more suitable position. Figure 8 provides an example. Here, the DOM elements accounting for the coauthor index is moved upwards from the bottom section of the page. This operation is achieved by selecting the corresponding DOM node (see Figure 8.a), click on the MOVE legend and next drag & drop to the new position (see the resulting page at Figure 8.b). The new position might prevent scrolling for users that mind co-authors.


CEUR-WS.org 2007 [contents]
[e1]
Marco Winckler, Hilary Johnson, Philippe A. Palanque:
Task Models and Diagrams for User Interface Design, 6th International Workshop, TAMODIA 2007, Toulouse, France, November 7-9, 2007, Proceedings. Lecture Notes in Computer Science 4849, Springer 2007, ISBN 978-3-540-77221-7 [contents]
MOVE widgetized0
Coauthor Index
1 Julio Abascal [j5]
2 Hans von Axelson [j5]
3 Cédric Bach [j6] [c57] [c54] [c44]
widgetized0

a) Initial position of the co-author index.

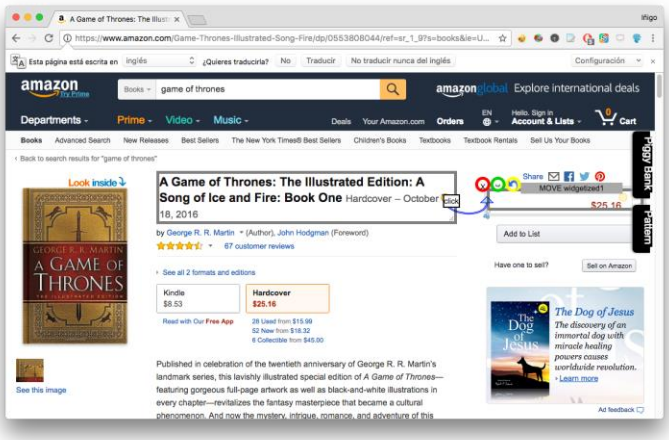


b) Final position of the co-author index.

Figure 8. Moving DOM elements around the Web page using WebMakeup.

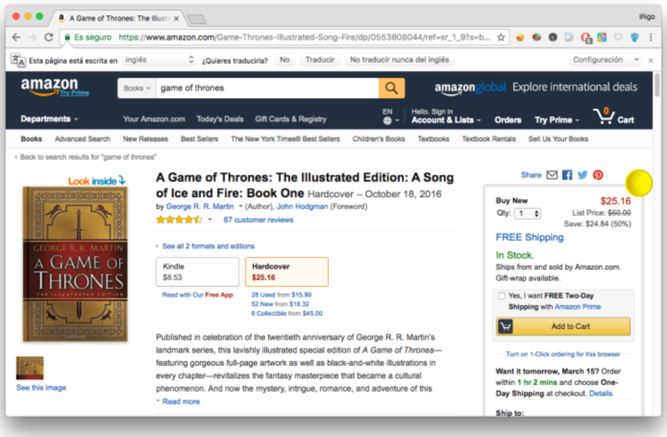
4.2.3 Creating new behaviors

WebMakeup allows supporting new behaviors (e.g. setting blink relationships between DOM elements). As an example, consider the Amazon page of the book “A Game of Thrones”. Two widgets are created after two DOM nodes: the *title* DOM and a widget with information of the book price and how it can be bought. Both widgets joined through the yellow point from the triggering widget to the triggered widget (see Figure 9.a). It is possible to choose which event (ex. click, doubleclick and mouseEnter) will trigger the show/hide behavior. At the end, the user can decide if the current book will be bought and clicking on the triggering element (the book title), the triggered widget will show the desired information (see Figure 9.b). Clicking on the book title again, the triggered element will be hidden.



A Game of Thrones: The Illustrated Edition: A Song of Ice and Fire: Book One Hardcover – October 18, 2016
click
MOVE widgetized1
$25.16
Add to List
Piggy Bank
Pattern

a) Associating between widgets.



A Game of Thrones: The Illustrated Edition: A Song of Ice and Fire: Book One Hardcover – October 18, 2016
Buy New
$25.16
Add to Cart

b) Resulting web site after deploying the adaptation.

Figure 9. Behavior definition in WebMakeup joining different widgets with wires

4.4 Collaborative features

WebMakeup scripts are stored locally in the Web browser. WebMakeup does not support collaborative development. Nonetheless, users can export scripts into a file and next share them through email or other means. Consumers should have WebMakeup installed and use the “import” option (Figure 6.a). Also in the scrollable menu, the entry “CarryOn” permits consumers to tune imported scripts to their own likes.

4.5 Programming paradigm

WebMakeup does not require users to write a single line of code to modify Web pages. All programming is achieved through selecting DOM elements and interacting with widget decorators. For that, WebMakeup is classified as visual programming**. Figure 10** highlight this and other EUD features of WebMakeup, w.r.t. those presented at **Figure 1,** by shadowing those not covered.

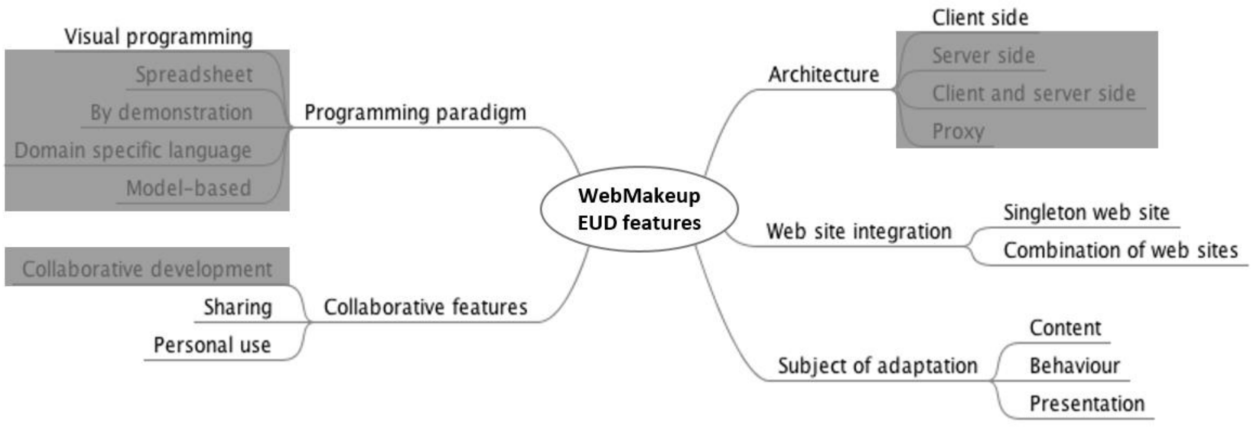


Figure 10. Summary of WebMakeup with respect to the classification presented in Figure 1.

5. User and usage challenges with WA tools

Each tool cited in this chapter has its own idiosyncrasies and their use will reveal very specific challenges. But beyond the use of a particular tool, WA challenges users to revise what they know about the web and how to program applications. When it comes to WA, users should be aware of a number of aspects, namely:

- WA is mainly a browser-based technology. Regardless of the technology employed to store and run the augmentation scripts, the adaptation only affects how a web site is displayed in the user's personal machine. Users must understand that their adaptation is personal and that will not be visible by other visitors of the same web site.
- WA is mainly a single browse technology. Changes performed by the user will only occur on the browser where the augmentation has been performed. The same user performing the same actions on another computer will not see the augmentation. It thus requires replication of the augmentation multiple times if the users are using multiple execution platforms (e.g. desktop computers, smartphones.
- Similar to other EUD technologies, WA require the adaption of the code produced by someone else. This has multiple implications for assessing the code of Web pages before to adapt them [39][40].
- WA is constricted within the DOM hierarchy. Users should be aware of manipulation of DOM elements imposes a certain order of access to contents. For instance, elements might appear visually together but be arranged in separated DOM nodes. This might imply having different ancestors. This, in turn, prevents these "alongside elements" from being selected as a single DOM element. This constrain is imposed by the DOM element hierarchy [8]. Notice that the DOM hierarchy itself does not need to be made visible but manipulated through metaphors and witty interactive tools. But no matter the tool, it is constricted within the DOM hierarchy
- WA is fragile upon Web-site upgrades. Web sites evolve overtime and with the evolution of a web site some elements resulting from the augmentations may disappear and/or be replaced by other elements that directly affect the way WA scripts operate. Thus, whilst some scripts will be resilient to maintenance of web sites, other scripts will stop working once a Web site is upgraded. This makes the use of WA a more suitable technique when user's needs are volatile [33]. WebMakeup illustrates the feasibility of having dynamic updates for contents but the bindings between WA scripts and the web site remain fragile and prone to become obsolete

when the underlying web site evolves. This a major challenge as, by definition, web applications are meant to evolve. Beyond, as the users do not own the web application, the loss of a web augmentation is not predictable.

- WA does not create brand-new applications but enhances existing ones. The inclusion of contents from other web sites raises some pragmatic questions about the type of relationship created between web sites [30]. The simplest approach is the clone&own of elements. This implies that changes in the source element will not propagate to its clones. Alternatively, it is also possible to keep a dynamic binding with the source element so that changes in the source ripple throughout its clones.

6. Conclusion

This chapter has presented the principles behind Web Augmentation and highlighted how this technology shares multiple similar objectives as End User Development. Indeed, as it allows users to recycle, reuse and exploit material that can be obtained from other web sites it supports the construction (by the end users themselves) of more usable and more adapted web application. One of the biggest challenges is how treat dynamic states of Web applications, which means contents that evolves over time. Whilst this remains an unsolved issue that should be addressed by future research, it is possible to envisage various copy and paste strategies to address the problem.

In our study of WA tools, we have observed a prominence of tools that run exclusively on the client side. This is not surprising as one of the advantages of using a client-side approach is the faster execution that has a huge impact on the user performance while interacting with the web application making it possible to provide immediate feedback to the users. Moreover, users do not need to understand sever side functioning and to deal with complex installations on a remote web server (for which they, most of the time, have no access rights). Whilst client-side approach is not a panacea, we suggest that this is still a suitable strategy for giving end users more autonomy on the scripts they want to develop.

As demonstrated in the chapter, there are multiple technologies for performing web augmentation. We have presented some precise examples through the use of a particular tool called WebMakeup. For sake of simplicity, we have only provided here simple examples that can be easily reproduced. Nonetheless, we have demonstrated that using such simple adaptation of contents, behavior and presentation, web sites can be profoundly modified to better fit with users' needs.

Despite our efforts, it is important to note that none of the references provided refer to studies with a large number of users. Because of that, we cannot measure the impact of such as a strategy on the end-user community. Nonetheless, the tools we have presented are functional and a dedicated community maintains most of them. We believe that these WA tools deserve more publicity and that a wider and more systematic communication towards end users would deeply impact usability of web application and, more generally, of the Web as a whole.

Acknowledgments.

This work is co-supported by the Spanish Ministry of Education, and the European Social Fund under contract TIN2014-58131-R and the stay scholarship EEBB-I-16-11126. Aldalur has a doctoral grant from the Spanish Ministry of Science & Education. This project is also supported by the STIC AmSud project WAMAW-OUR.

References

1. Aghaee, S., and Pautasso, C. End-user programming for web mashups - open research challenges. In Current Trends in Web Engineering - Workshops, Doctoral Symposium, and Tutorials, Held at ICWE 2011, Paphos, Cyprus, June 20-21, 2011. Revised Selected Papers, pages 347–351, 2011.
2. Aghaee, S., and Pautasso, C. End-user development of mashups with naturalmash. J. Vis. Lang. Comput. 25, 4 (2014), 414–432.

3. Agosti, M., Albrechtsen, H., Ferro, N., Frommholz, I., Hansen, P., Orio, N., Panizzi, E., Pejtersen, A. M., and Thiel, U. DiLAS: a Digital Library Annotation Service. Proceedings of the International Workshop on Annotation for Collaboration - Methods, Tools and Practices (IWAC 2005), La Sorbonne, Paris, France, November 23-24 2005, pages 91-101.
4. Ardito, C., Bottoni, P., Costabile, M. F., Desolda, G., Matera, M., Piccinno, A., and Picozzi, M. Enabling end users to create, annotate and share personal information spaces. In End-User Development - 4th International Symposium, IS-EUD 2013, Copenhagen, Denmark, June 10-13, 2013. Proceedings (2013), pp. 40–55.
5. Ardito, C., Costabile, M. F., Desolda, G., Lanzilotti, R., Matera, M., Piccinno, A., and Picozzi, M. User-driven visual composition of service-based interactive spaces. J. Vis. Lang. Comput. 25, 4 (2014), 278–296.
6. Ardito, C., Costabile, M. F., Desolda, G., Latzina, M., and Matera, M. Hands-on actionable mashups. In End-User Development - 5th International Symposium, IS-EUD 2015, Madrid, Spain, May 26-29, 2015. Proceedings (2015), pp. 295–298.
7. Arellano, C., and Díaz, O. Lightweight end-user software sharing. In End- User Development - 4th International Symposium, IS-EUD 2013, Copenhagen, Denmark, June 10-13, 2013. Proceedings (2013), pp. 241–246.
8. Bosetti, G., Firmenich, S., Rossi, G., and Winckler, M. Web Objects Ambient: an integrated platform supporting new kinds of Personal Web experiences. In proceedings of the International Conference of Web Engineering (ICWE 2016), Lugano, Switzerland, June 6-9, 2016. Proceedings. Lecture Notes in Computer Science 9671, Springer 2016, ISBN 978-3-319-38790-1, pages 563-566.
9. Bottaro, A., Marino, E., Milicchio, F., Paoluzzi, A., Rosina, M., and Spini, F. Visual programming of location-based services. In Human Interface and the Management of Information. Interacting with Information – Symposium on Human Interface 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part I (2011), pp. 3–12.
10. Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E., and Trinchese, R. 2004. MADCOW: a multimedia digital annotation system. In Proceedings of the working conference on Advanced visual interfaces (AVI '04). ACM, New York, NY, USA, 55-62. DOI=http://dx.doi.org/10.1145/989863.989870
11. Bouvin, N. O. Unifying Strategies for WA. In: Proc. of the 10th ACM Conference on Hypertext and Hypermedia, 1999.
12. Brel, C., Dery-Pinna, A., Renevier-Gonin, P., and Riveill, M. Ontocompo: A tool to enhance application composition. In Human-Computer Interaction - INTERACT 2011 - 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part IV (2011), pp. 588–591.
13. Burnett, Margaret M. and Scaffidi, Christopher (2011). End-User Development. In: Encyclopedia of Human-Computer Interaction. Soegaard, Mads and Dam, Rikke Friis (eds.). Available free online at http://www.interaction-design.org/encyclopedia/end-user_development.html
14. Cappiello, C., Daniel, F., Matera, M., Picozzi, M., and Weiss, M. Enabling end user development through mashups: Requirements, abstractions and innovation toolkits. In End-User Development - Third International Symposium, IS-EUD 2011, Torre Canne (BR), Italy, June 7-10, 2011. Proceedings (2011), pp. 9–24.
15. Cappiello, C., Matera, M., and Picozzi, M. End-user development of mobile mashups. In Design, User Experience, and Usability. Web, Mobile, and Product Design - Second International Conference, DUXU 2013, Held as Part of HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings, Part IV (2013), pp. 641–650.
16. Cappiello, C., Matera, M., Picozzi, M., Sprega, G., Barbagallo, D., and Francalanci, C. Dashmash: A mashup environment for end user development. In Web Engineering - 11th International Conference, ICWE 2011, Paphos, Cyprus, June 20-24, 2011 (2011), pp. 152–166.

17. Chaisatien, P., Prutsachainimmit, K., and Tokuda, T. Mobile mashup generator system for cooperative applications of different mobile devices. In Web Engineering - 11th International Conference, ICWE 2011, Paphos, Cyprus, June 20-24, 2011 (2011), pp. 182–197.
18. Challiol, C., Firmenich, S., Bosetti, G. A., Gordillo, S. E., and Rossi, G. Crowdsourcing mobile web applications. In Current Trends in Web Engineering - ICWE 2013 International Workshops ComposableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8-12, 2013. Revised Selected Papers (2013), pp. 223–237.
19. Chang, K. S., and Myers, B. A. Webcrystal: understanding and reusing examples in web authoring. In CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA - May 05 - 10, 2012 (2012), pp. 3205–3214.
20. Chowdhury, S. R., Rodríguez, C., Daniel, F., and Casati, F. Baya: assisted mashup development as a service. In Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume) (2012), pp. 409–412.
21. Chudnovskyy, O., Nestler, T., Gaedke, M., Daniel, F., Fernández- Villamor, J. I., Chepegin, V. I., Fornas, J. A., Wilson, S., Kögler, C., and Chang, H. End-user-oriented telco mashups: the OMELETTE approach. In Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume) (2012), pp. 235–238.
22. Daniel, F., Rodríguez, C., Chowdhury, S. R., Nezhad, H. R. M., and Casati, F. Discovery and reuse of composition knowledge for assisted mashup development. In Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume) (2012), pp. 493–494.
23. Díaz, O., Arellano, C., Aldalur, I., Medina, H., and Firmenich, S. End-user browser-side modification of web pages. In Web Information Systems Engineering - WISE 2014 - 15th International Conference, Thessaloniki, Greece, October 12-14, 2014, Proceedings, Part I (2014), pp. 293–307.
24. Díaz, O., Arellano, C., and Azanza, M. A language for end-user WA: Caring for producers and consumers alike. TWEB 7, 2 (2013), 9.
25. Díaz, O., Pérez, S., and Paz, I. Providing personalized mashups within the context of existing web applications. In Web Information Systems Engineering WISE 2007, 8th International Conference on Web Information Systems Engineering, Nancy, France, December 3-7, 2007, Proceedings (2007), pp. 493–502.
26. Díaz, O., Sosa, J. D., Arellano, C., and Trujillo, S. Web-based tool integration: A WA approach. In Web Engineering - 12th International Conference, ICWE 2012, Berlin, Germany, July 23-27, 2012. Proceedings (2012), pp. 431–434.
27. Dong, T., Ackerman, M. S., Newman, M. W., and Paruthi, G. Social overlays: Collectively making websites more usable. In Human-Computer Interaction - INTERACT 2013 - 14th IFIP TC 13 International Conference, Cape Town, South Africa, September 2-6, 2013, Proceedings, Part IV (2013), pp. 280–297.
28. Firmenich, S., Bosetti, G., Rossi, G., Winckler, and M., Barbieri, T. Abstracting and structuring web contents for supporting personal web experiences. In Web Engineering - 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings, pages 77–95, 2016.
29. Firmenich, D., Firmenich, S., Rivero, J. M., and Antonelli, L. A platform for WA requirements specification. In Web Engineering, 14th International Conference, ICWE 2014, Toulouse, France, July 1-4, 2014. Proceedings (2014), pp. 1–20.
30. Firmenich, D., Firmenich, S., Winckler, M., Rossi, G., Distante, D. User Interface Adaptation Using WA Techniques: Towards a Negotiated Approach. International Conference on Web Engineering 2015 (ICWE). LNCS vol:9114. p:147-164. Springer.
31. Firmenich, S., Winckler, M., and Rossi, G. A Framework for Concern-Sensitive, Client-Side Adaptation. In Web Engineering - 11th International Conference, ICWE 2011, Paphos, Cyprus, June 20-24, 2011. Springer, LNCS 6757, pages 198-213.

32. Firmenich, S., Winckler, M., Rossi, and G., Gordillo, S. (2011) A Crowdsourced Approach for Concern-Sensitive Integration of Information across the Web. Journal of Web Engineering (JWE). Rinton Press., Vol.10 No.4, December 2011, pages: 289-315.
33. Frajberg, D., Urbieta, M., Rossi, G., Schwinger, W. Volatile Functionality in Action: Methods, Techniques and Assessment. In proceedings of the International Conference of Web Engineering (ICWE 2016), Lugano, Switzerland, June 6-9, 2016. Proceedings. Lecture Notes in Computer Science 9671, Springer 2016, ISBN 978-3-319-38790-1, pages 59-76.
34. Gardiner, S., Tomasic, A., Zimmerman, J., Aziz, R., and Rivard, K. Mixer: Mixed-initiative data retrieval and integration by example. In Human-Computer Interaction - INTERACT 2011 - 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part I (2011), pp. 426–443.
35. Garrido, A., Firmenich, S., Rossi, G., Grigera, J., Medina-Medina, N., and Harari, I. Personalized Web Accessibility using Client-Side Refactoring. IEEE Internet Computing 17(4): 58-66 (2013).
36. Ghiani, G., Manca, M., Paternò, F., and Porta, C. Beyond Responsive Design: Context-Dependent Multimodal Augmentation of Web Applications. MobiWIS 2014, LNCS Volume 8640, pp. 71-85, Springer Verlag.
37. Ghiani, G., Paternò, F., and Spano, L. D. Creating mashups by direct manipulation of existing web applications. In End-User Development – Third International Symposium, IS-EUD 2011, Torre Canne (BR), Italy, June 7-10, 2011. Proceedings (2011), pp. 42–52.
38. Ghiani G., Paternò F., Spano L.D., and Pintori G., An environment for End-User Development of Web mashups, International Journal of Human-Computer Studies Volume 87, March 2016, Pages 38–64, Elsevier.
39. Gross, P. A., and Kelleher, C. Non-programmers identifying functionality in unfamiliar code: strategies and barriers. J. Vis. Lang. Comput. 21, 5 (2010), 263–276.
40. Gross, P. A., Yang, J., and Kelleher, C. Dinah: an interface to assist nonprogrammers with selecting program code causing graphical output. In Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011 (2011), pp. 3397–3400.
41. Guo, J., Han, H., and Tokuda, T. Towards flexible mashup of web applications based on information extraction and transfer. In Web Information Systems Engineering - WISE 2010 - 11th International Conference, Hong Kong, China, December 12-14, 2010. Proceedings (2010), pp. 602–615.
42. Han, H., and Tokuda, T. Towards flexible and lightweight integration of web applications by end-user programming. IJWIS 6(4): 359-373 (2010).
43. Hick, W. E. (1952): On the rate of gain of information, Quarterly Journal of Experimental Psychology, 4:1, 11-26. Also available at: http://dx.doi.org/10.1080/17470215208416600.
44. Hoang, D. D., Paik, H., and Benatallah, B. An analysis of spreadsheet based services mashup. In Database Technologies 2010, Twenty-First Australasian Database Conference (ADC 2010), Brisbane, Australia, 18-22 January, 2010, Proceedings (2010), pp. 141–150.
45. Hoang, D. D., Paik, H., and Dong, W. Mashsheet: Mashups in your spreadsheet. In Web Information System Engineering - WISE 2011 - 12th International Conference, Sydney, Australia, October 13-14, 2011. Proceedings (2011), pp. 332–333.
46. Husmann, M., Nebeling, M., Pongelli, S., and Norrie, M. C. Multimasher: Providing architectural support and visual tools for multi-device mashups. In Web Information Systems Engineering - WISE 2014 - 15th International Conference, Thessaloniki, Greece, October 12-14, 2014, Proceedings, Part II (2014), pp. 199–214.
47. Imran, M., Soi, S., Kling, F., Daniel, F., Casati, F., and Marchese, M. On the systematic development of domain-specific mashup tools for end users. In Web Engineering - 12th International Conference, ICWE 2012, Berlin, Germany, July 23-27, 2012. Proceedings (2012), pp. 291–298.

48. 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.
49. Kleek, M. V., Moore, B., Karger, D. R., André, P., and m. c. schraefel. Atomate it! End-user context-sensitive automation using heterogeneous information sources on the web. In Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010 (2010), pp. 951–960.
50. Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M. B., Rothermel, G., Shaw, M., Wiedenbeck, S. 2011. The state of the art in end-user software engineering. ACM Comput. Surv. 43, 3, Article 21 (April 2011), 44 pages.
51. Kovachev, D., Renzel, D., Nicolaescu, P., and Klamma, R. Direwolf - distributing and migrating user interfaces for widget-based web applications. In Web Engineering - 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings (2013), pp. 99–113.
52. Krug, M., Wiedemann, F., and Gaedke, M. Smartcomposition: A component based approach for creating multi-screen mashups. In Web Engineering, 14th International Conference, ICWE 2014, Toulouse, France, July 1-4, 2014. Proceedings (2014), pp. 236–253.
53. Leshed, G., Haber, E. M., Matthews, T., and Lau, T. A. Coscripter: automating & sharing how-to knowledge in the enterprise. In Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008 (2008), pp. 1719–1728.
54. Lieberman, H., Paterno, F., and Wulf, V. (eds.) End-User Development. Kluwer/Springer, 2005.
55. Massa, D., and Spano, L. D. Facemashup: Enabling end user development on social networks data. In End-User Development - 5th International Symposium, IS-EUD 2015, Madrid, Spain, May 26-29, 2015. Proceedings (2015), pp. 204–210.
56. Miján, J. L., Garrigós, I., Firmenich, I. Supporting personalization in legacy web sites through client-side adaptation. In Web Engineering - 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings, pp. 588–592, 2016.
57. Nebeling, M., Leone, S., and Norrie, M. C. Crowdsourced web engineering and design. In Web Engineering - 12th International Conference, ICWE 2012, Berlin, Germany, July 23-27, 2012. Proceedings (2012), pp. 31–45.
58. Newman, M. W., Lin, J., Hong, J. I., & Landay, J. A. (2003). DENIM: An informal Web site design tool inspired by observations of practice. Human—Computer Interaction, 18(3), 259-324.
59. Nicolaescu, P., and Klamma, R. A methodology and tool support for widget based web application development. In Engineering the Web in the Big Data Era - 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings (2015), pp. 515–532.
60. Park, T. H., Saxena, A., Jagannath, S., Wiedenbeck, S., and Forte, A. Openhtml: designing a transitional web editor for novices. In 2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France, April 27 - May 2, 2013, Extended Abstracts (2013), pp. 1863–1868.
61. Phuoc, D. L., Polleres, A., Hauswirth, M., Tummarello, G., and Morbidoni, C. Rapid prototyping of semantic mash-ups through semantic web pipes. In Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009 (2009), pp. 581–590.
62. Poley, E. RUMU editor: a non-wysiwyg web editor for non-technical users. In Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Extended Abstracts Volume, Atlanta, Georgia, USA, April 10- 15, 2010 (2010), pp. 4357–4362.
63. Radeck, C., Blichmann, G., and Meißner, K. Capview – functionality aware visual mashup development for non-programmers. In Web Engineering - 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings (2013), pp. 140–155.

64. Rana, J., Morshed, S., and Synnes, K. End-user creation of social apps by utilizing web-based social components and visual app composition. In 22nd International World Wide Web Conference, WWW ’13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume (2013), pp. 1205–1214.
65. Realinho, V., Dias, A. E., and Romão, T. Testing the usability of a platform for rapid development of mobile context-aware applications. In Human-Computer Interaction - INTERACT 2011 - 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part III (2011), pp. 521–536.
66. Repenning, A., Ahmadi, N., Repenning, N., Ioannidou, A., Webb, D. C., and Marshall, K. S. Collective programming: Making end-user programming (more) social. In End-User Development - Third International Symposium, IS-EUD 2011, Torre Canne (BR), Italy, June 7-10, 2011. Proceedings (2011), pp. 325–330.
67. Tayeh, A. A. O., and Signer, B. A dynamically extensible open cross document link service. In Web Information Systems Engineering - WISE 2015 - 16th International Conference, Miami, FL, USA, November 1-3, 2015, Proceedings, Part I (2015), pp. 61–76.
68. Tayeh, A. A. O., and Signer, B. Open cross-document linking and browsing based on a visual plug-in architecture. In Web Information Systems Engineering - WISE 2014 - 15th International Conference, Thessaloniki, Greece, October 12-14, 2014, Proceedings, Part II (2014), pp. 231–245.
69. Toomim, M., Drucker, S. M., Dontcheva, M., Rahimi, A., Thomson, B., and Landay, J. A. Attaching UI enhancements to websites with end users. In Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009 (2009), pp. 1859–1868.
70. Wajid, U., Namoun, A., and Mehandjiev, N. Alternative representations for end user composition of service-based systems. In End-User Development - Third International Symposium, IS-EUD 2011, Torre Canne (BR), Italy, June 7-10, 2011. Proceedings (2011), pp. 53–66.
71. Wang, G., Yang, S., and Han, Y. Mashroom: end-user mashup programming using nested tables. In Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009 (2009), pp. 861–870.
72. Wang, S., and Wainer, G. A. A mashup architecture with modeling and simulation as a service. In Web Information Systems Engineering - WISE 2015 - 16th International Conference, Miami, FL, USA, November 1-3, 2015, Proceedings, Part I (2015), pp. 247–261.
73. Wong, J., and Hong, J. I. Making mashups with marmite: towards end-user programming for the web. In Proceedings of the 2007 Conference on Human Factors in Computing Systems, CHI 2007, San Jose, California, USA, April 28 - May 3, 2007 (2007), pp. 1435–1444.
74. Wulf, V., Paterno, F., Lieberman, H. 2006. End User Development. Kluwer Academic Publishers.
75. Zhai, Z., Cheng, B., Wang, Z., Liu, X., Liu, M., Chen, J. Design and implementation: the end user development ecosystem for cross-platform mobile applications. In Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume, pages 143–144, 2016.