Poster: End-User Software Engineering for the Personal Web

Sergio Firmenich[1,2], Gabriela Bosetti[1],
Gustavo Rossi[1,2]
[1] LIFIA, Facultad de Informática, UNLP
[2] CONICET
La Plata, Argentina
{gbosetti, sfirmenich, gustavo}@lifia.info.unlp.edu.ar

Marco Winckler
ICS-IRIT, Université Toulouse III
Toulouse, France
winckler@irit.fr

***Abstract*—In this work we present an approach for creating Personal Web applications by reusing existing content that can be extracted even from third-party Web sites. Our approach starts with the harvesting of content from diverse Web sites, by DOM manipulation. Users without programming skills are empowered with tools for transforming DOM elements into meaningful classes of objects that can be reused to build other domain-specific applications, such as mashups, Web augmentations, PIM systems, etc.**

Personal Web; End-User Software Engineering; Mashups.

I. INTRODUCTION

During the last decade, the term End User Development (EUD) [1] became popular in the field of the Web, since it is a natural environment for daily life activities nowadays. In this regard, there is a survey [2] conducted with 800 information workers reporting that 68% of them created some Web artefact, 65% wrote HTML pages and around 40% already coded in JavaScript.

The Personal Web (PW) also arose in the last years, providing a first class support to the end-users' needs [3]. EUD clearly benefits the PW, but currently it usually covers a single stage in the software lifecycle. End-User Software Engineering (EUSE) [4, 5] is a wider research aimed at providing end users with solutions to create good quality software. However, we found that the related literature in PW by end users is mainly focused in EUD.

The PW offers the means for combining content and services to facilitate the execution of cross-site tasks, and making possible the interaction between Web applications. There are different kinds of applications materializing the PW, like Personal Information Management systems (PIMs) [6], Web Augmentation (WA) [7] based, mashups [8] or annotation tools [9]. In [6], authors present a study about the use of Web-based PIMs. There are also approaches in the fields of WA [10,11] and Distribution of UI [12]. Some approaches automate recurrent users' tasks [13,14] or integrate different Web sites [8,15]. All of them have two points in common: they support a concrete kind of PW application and do not address the whole software lifecycle.

This led us to a first problem: the lack –to our knowledge– of a mean for empowering end users to achieve a comprehensive PW experience. PW applications are treated in isolation, impacting negatively in the adoption of the PW. E.g. consider the diversity of tools that can be used for bibliography searching: one for integrating content from different sources, another one for augmenting such sources with extra behaviour, a third one to collect the relevant works into a PIM, etc. In this setting, it is not possible to reuse the same data model, which can improve the reuse and make the maintenance easier. A similar approach presents a very generic three-layer meta-model for the PW that allow users to define operations internally mapped to concrete objects [3]. However, there is no explanation about how end-users create both, their own data model and applications.

A second problem is that although there are several approaches in EUD, they suffer from over-siloing [4] in the context of EUSE. For instance, WebMakeUp [11] and AccessMonkey [9] let end-users to adapt Web sites through WA, but they are purely focused on development. Full support in the PW application development lifecycle is important due to the natural change of the Web over time, which raises the need for constant maintenance or resilient solutions (as explained in [3]). By analysing more than 4500 artefacts shared in a WA community called Greasyfork.org we found that 56.7% of them were upgraded at least once, and just the 25% have done it in the last five months [16]. This maintenance is in charge of developers and without their collaboration, users may have frustrating experiences.

To address these problems, we present a general-purpose approach supporting EUSE in the context of the PW. A support tool allows end users to transparently model a common data layer over existing (usually third-party) Web content, which is reusable in different PW applications. Such applications can be built, tested and maintained by end users, covering not just the development stage in their life cycle.

II. EUWE FOR THE PERSONAL WEB

Our approach is based on the principles of abstraction and structuring of Web content. The underlying idea, illustrated in Fig. 1, is to take DOM elements from existing Web pages and transform them into first-class application

objects that can be reused for building new PW applications. Users can collect any DOM element they consider of their interest, and store it as a User Interface Object (UIO) or a Conceptual Object (CO). UIOs are treated as snippets of HTML code, while COs keep the underlying HTML structure but are treated as objects with properties. Both UIOs and COs should be associated with an entity in an ontology, so the tool can find related behavioural wrappers that enhances UIOs and COs. For instance, a CO matching a film entity may be enhanced with the behaviour of showing a related trailer on YouTube. All these objects are stored in a common space of information called Web Objects Ambient (WOA) [17] and became available for every PW application. There is an API that developers can use in their own productions, fostering the reuse of data and behaviours. End-users can create applications by using a visual editor and, as maintenance is a critical issue in applications manipulating non-APIfied third-party Web content [7], we also provide them the means for allowing corrective maintenance.

III. SUPPORT TOOL

To demonstrate the technical feasibility of our approach we implemented a Firefox extension. Fig. 2 illustrates the use of this tool for creating a CO from the DBLP Web site. First, the plugin should be enabled (1) so DOM elements can be selected by right-clicking (2). The user creates a class for describing the DOM element (3), and then its properties in a similar way. The user can apply a behavioural wrapper (4) from a list automatically extracted based in the artefacts' tags. A set of messages is available in (5) for the selected

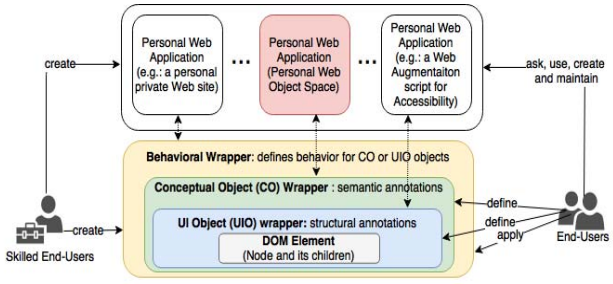

Figure 1. Overview of the proposed Personal Web Architecture

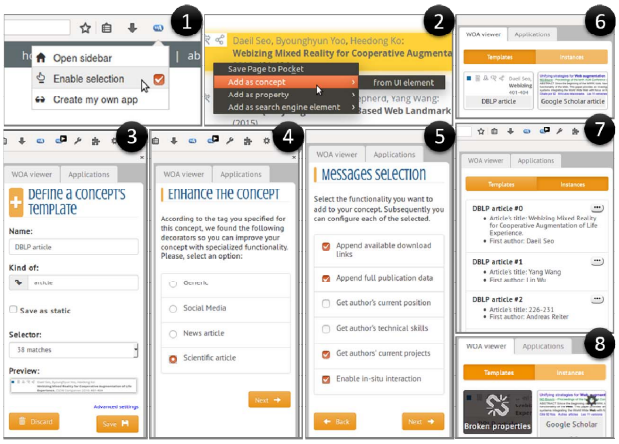

Figure 2. Steps for creating a class to a CO object

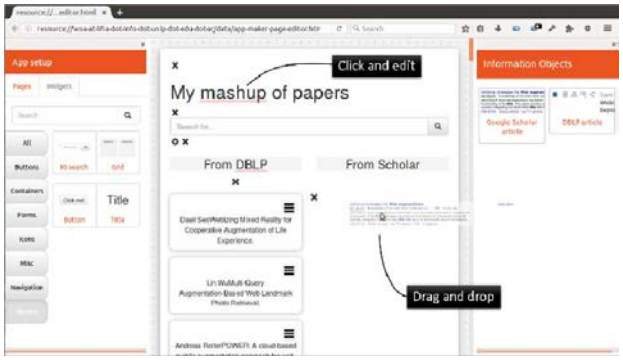

Figure 3. Creating a new personal application using CO objects

decorator. After this operations, the objects become available in the platform as classes (6) and instances (7), and users can interact with them or create their PW applications.

To create new applications, like mashups or WA-based, the user can launch a visual editor shown in Fig. 3. The editor has three parts: a left-side menu that allows the user to compose a Web page using basic widgets such as title, buttons or layout elements. The main editor area in the middle can be loaded with an existing Web site or a blank Web page. On the right-side, there is a menu containing the COs and UIOs previously collected. Both widgets and collected objects can be dragged and dropped into the central area to populate the application. Concretely, Fig. 3 illustrates the creation of a mashup application encompassing a title and two classes of CO: google scholar and DBLP articles. Note that when these CO are dropped in the middle of the application, it is populated with all instances of DOM elements that correspond to the CO type. Finally, applications created with this editor are available in a dedicated tab (right tab in step 6 of Fig. 2).

To allow the maintenance of the created artefacts, our tool also facilitates error detection and correction as shown in step 8 of Fig. 2. There, the abstraction defined previously in DBLP appears with a message alerting the user that some properties cannot be found in the DOM of the original Web site, so the reference to the element is broken. If the user clicks on it, the Web page is opened and he may fix this in a similar way as in steps 2 and 3. Once the user selects the new DOM element that fixes the reference, a set of tests is performed to determine if it satisfies specific conditions (e.g. if the property is still an anchor), ensuring that the returned value meets the expected one.

IV. CONCLUSIONS AND FUTURE WORK

The Web offers a big amount of content that can be reused to create PW applications and meet specific users' needs. To do so, we present an approach and a support tool enabling the creation, testing and maintenance of PW applications from existing Web content, by end-users with no need of programming skills.

We are improving our tool based on several case studies. We expect to conduct a formal experiment for understanding the potential adoption by end-users, and we are considering supporting collaborative tasks and cross-device interactions.

REFERENCES

[1] H. Lieberman, F. Paternò, M. Klann and V. Wulf. (2006). End-user development: An emerging paradigm. In End user development (pp. 1-8). Springer Netherlands.

[2] C. Scaffidi, A. Ko, B. Myers and M. Shaw. 2006. Dimensions characterizing programming feature usage by information workers, IEEE Symposium on Visual Languages and Human-Centric Computing, 59-64.

[3] M. Chignell,J. R. Cordy, R. Kealey, J. Ng, and Y. Yesha. (2013). *The Personal Web*. Springer Berlin Heidelberg.

[4] M. M. Burnett and B. A. Myers. (2014, May). Future of end-user software engineering: beyond the silos. In *Proceedings of the on Future of Software Engineering* (pp. 201-211). ACM.

[5] A. J. Ko et al.. (2011). The state of the art in end-user software engineering. ACM Computing Surveys (CSUR), 43(3), 21.

[6] M. Van Kleek, D. A. Smith and N. Shadbolt. (2012). A decentralized architecture for consolidating personal information ecosystems: The WebBox.

[7] O. Díaz, C. Arellano. The augmented web: rationales, opportunities, and challenges on browser-side transcoding. *ACM Transactions on the Web (TWEB)*, 2015, vol. 9, no 2, p. 8.

[8] F. Daniel and M. Matera. (2014). Mashups: Concepts, Models and Architectures. Springer.

[9] J. P. Bigham and R. E. Ladner. (2007, May). Accessmonkey: a collaborative scripting framework for web users and developers. In *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)* (pp. 25-34). ACM.

[10] O. Díaz, C. Arellano and M. Azanza. (2013). A language for end-user web augmentation: Caring for producers and consumers alike. *ACM Transactions on the Web (TWEB)*, *7*(2), 9.

[11] O. Díaz, C. Arellano, I. Aldalur, H. Medina and S. Firmenich. (2014, October). End-user browser-side modification of web pages. In *International Conference on Web Information Systems Engineering* (pp. 293-307). Springer International Publishing.

[12] Proxywork: Distributing User Interface Components of Web Applications. In *DUI@ EICS* (pp. 58-61).

[13] G. Leshed, E. M. Haber, T. Matthews and T. Lau. (2008, April). CoScripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1719-1728). ACM.

[14] M. Bolin, M. Webber, P. Rha, T. Wilson and R. C. Miller. (2005, October). Automation and customization of rendered web pages. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (pp. 163-172). ACM.

[15] S. Firmenich, G. Rossi, M. Winckler and P. Palanque. (2014). An approach for supporting distributed user interface orchestration over the Web. *International Journal of Human-Computer Studies*, *72*(1), 53-76.

[16] D. Firmenich, S. Firmenich, J. M. Rivero, L. Antonelli and G. Rossi. (2016). CrowdMock: an approach for defining and evolving web augmentation requirements. *Requirements Engineering*, 1-29.

[17] S. Firmenich, G. Bosetti, G. Rossi, M. Winckler and T. Barbieri. (2016, June). Abstracting and Structuring Web Contents for Supporting Personal Web Experiences. In International Conference on Web Engineering (pp. 77-95). Springer International Publishing.