MDWA: a model-driven Web augmentation approach—coping with client- and server-side support

Matias Urbieta[1,2] · Sergio Firmenich[1,2] · Gabriela Bosetti[1] · Pedro Maglione[1] · Gustavo Rossi[1,2] · Miguel Angel Olivero[3,4]

Abstract

Web augmentation is a set of techniques allowing users to define and execute software which is dependent on the presentation layer of a concrete Web page. Through the use of specialized Web augmentation artifacts, the end users may satisfy several kinds of requirements that were not considered by the analysts, developers and stakeholders that built the application. Although some augmentation approaches are contemplating a server-side counterpart (to support aspects such as collaboration or cross-browser session management), the augmentation artifacts are usually purely client-side. The server-side support increases the capabilities of the augmentations, since it may allow sharing information among users and devices. So far, this support is often defined and developed in an ad hoc way. Although it is clear that server-side support brings new possibilities, it is also true that developing and deploying server-side Web applications is a challenging task that end users hardly may handle. This work presents a novel approach for designing Web augmentation applications based on client-side and server-side components. We propose a model-driven approach that raises the abstraction level of both, client- and server-side developments. We provide a set of tools for designing the composition of the core application with new features on the back-end and the augmentation of pages in the front-end. The usability and the value of the produced augmentations have been evaluated through two experiments involving 30 people in total.

Keywords Model-driven Web engineering · Augmentation · Web development · Separation of concern

1 Introduction

Designing Web applications can be a challenging task; Web application owners would like to easily know the users' real needs and moreover being able to fast-and-cheap introduction of functionalities in order to satisfy as many people requirements as possible.

Web augmentation has emerged as a set of technologies allowing users to adapt third-party Web sites to cope with personal needs not covered by the original design, and without depending on developers. Web augmentation has an interesting community combining a broad range of efforts by the crowds (userscripts or browsers extensions) and by researchers promoting good software development practices, such as reuse [1] or robustness [2]. Most of the augmenta-

✉ Matias Urbieta
matias.urbieta@lifia.info.unlp.edu.ar

Sergio Firmenich
sergio.firmenich@lifia.info.unlp.edu.ar

Gabriela Bosetti
gabriela.bosetti@lifia.info.unlp.edu.ar

Pedro Maglione
pedroms@lifia.info.unlp.edu.ar

Gustavo Rossi
gustavo@lifia.info.unlp.edu.ar

Miguel Angel Olivero
miguel.olivero@iwt2.org; miguelangel.olivero@isti.cnr.it

[1] Facultad de Informática, Universidad Nacional de La Plata, calle 50 y 120, 1900, La Plata, Buenos Aires, Argentina

[2] CONICET, La Plata, Argentina

[3] Web Engineering and Early Testing Group, Computer Languages and Systems Department, University of Seville, Avda. Reina Mercedes S/N, 41012 Seville, Spain

[4] ISTI - CNR, Pisa, Italy

tion approaches rely on the client-side without the need for a back-end application for producing the augmentation. Such architecture limits the power of the augmentation because it does not profit from collaborative features and the limited resources provided by the browser (as the processing power or the storage alternatives). Other approaches have a traditional client–server architecture since certain services cannot be deployed only on the client-side. For instance, the synchronization of devices supports distributed user interfaces [3], the use of a recommender system [4]) and social Web content management tools, such as Diigo [5]. All these back-end counterparts are dedicated applications specifically designed and deployed for the particular kind of augmentation, but, to our knowledge, there are no approaches considering both client- and serve-side as a general-purpose approach.

A complex augmentation may require client- and server-side components. The client-side, running in a Web browser, is a set of artifacts that helps to extract concepts from a host application, enriching the page with new UI features and pulling/pushing data from another host. On the other hand, the server-side of the augmentation extensions can be seen as a Web application that is merged with a target site. The application provides dynamic content generation, APIs and vast CPU and storage resources. When these applications are developed using standard programming languages or frameworks, skilled developers are required with experience on publishing API, managing a high throughput of request and security flaws.

The development of server-side components for a Web augmentation can be solved using either ad hoc development (e.g., a developer writes most of the lines of code) or model-driven Web engineering (MDWE) approaches (an analyst models the solution and the code is then generated). In the former, the stakeholder interested in augmenting a system has to manage the development of server-side components by coding the content generation, providing CORS support, supporting integration mechanisms, etc. In the latter, the usage of MDWE allows resolving the augmentation requirements by abstracting source code aspects and focusing on functional requirements instead. Each approach has its strengths. For example, ad hoc development allows a fully customize solution, whereas MDWE increases the abstraction level. The MDWE approaches have reported benefits like performance improvements of the development requiring a considerable less effort and time [6,7], providing a better quality of the application [8], and developer satisfaction [9] despite of both approaches having the same maintenance challenges [10]. The MDWE community has performed researches for supporting empirically claimed model-driven benefits [6,9,11,12]. The contribution covers aspects like the practitioner perspective, productivity concerning efficiency, effectiveness, usefulness and learnability.

With this in mind, this paper presents a Web augmentation modeling approach contemplating a client–server application that hides both front-end and back-end complexity to users. To support this approach, we present a specialized model-driven and client-side tool that allows users to model application augmentations. As we will discuss later, this may be a benefit not just for end users but for the applications' owners to weave their applications and evaluate new functionalities without the need of modifying their source code.

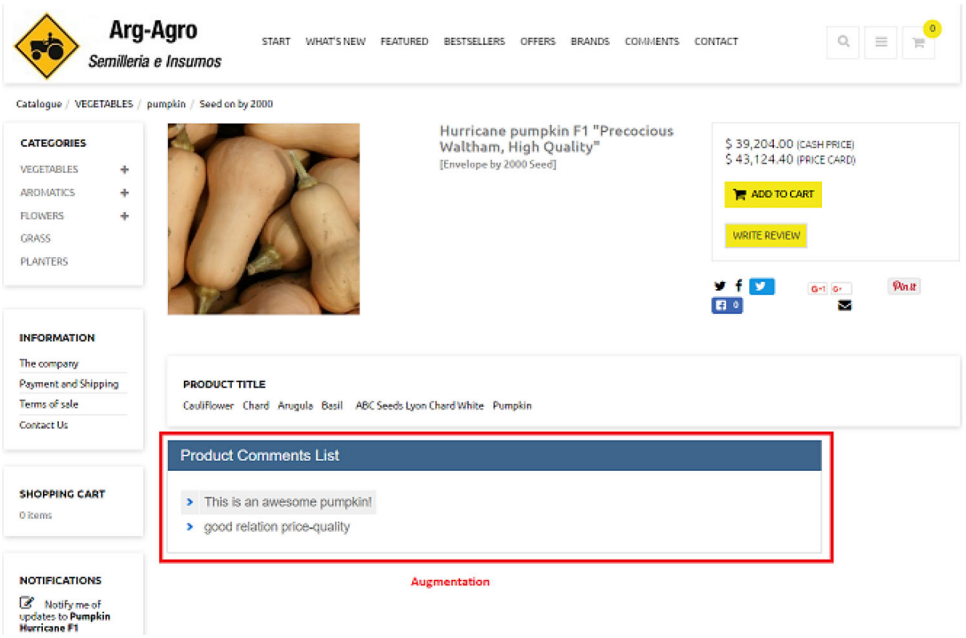
The main contribution of this work is a model-driven approach supporting the production of augmentations which benefits the combination of artifacts at both, client- and server-side. In the second place, we introduce a tool so-called WOA that supports the seamless introduction of this kind of augmentation. The tool is used to assess the applicability of the approach in two evaluations. The approach uses advanced principles of separation of concerns [13], where the user—who designs an augmentation—focuses on an augmentation—a concern of importance—for designing its artifacts and later grouping them into a module, ignoring other requirements. We used illustrative scenarios to exhibit our ideas with the WOA tool [14] and WebRatio [15] platform. The WOA tool has been extended with a full-fledged augmentation engine that was developed for this approach.

In a previous work [16], we introduced the foundations of our approach and basic examples. In this work, we present a comprehensive description of the approach, a discussion about its challenges and benefits, a set of descriptive scenarios using our approach and an evaluation section that reported preliminary results supporting our claims. The paper is organized as follows. Section 2 describes the background. Then, Sect. 3 introduces related works. Section 4 addresses an overview of the approach. Section 5 presents examples of use. Section 6 introduces the evaluation of our approach, and Sect. 7 presents the conclusions and future works.

1.1 Augmentation dimensions

In Fig. 1, we show how an e-commerce site is enhanced with a collaborative feature that allows users to post comments related to a product. Using an augmentation technique, the user interface can be enriched with a set of widgets that are not provided by the application. Although we can enrich a site using, for example, a browser plugin, the required server-side counterpart is complex because storage and CPU processing are required to handle the new extension.

So far so good, we have mentioned several real examples of Web extensions materializing some Web augmentation that requires (and use) server-side support. In the following list, we recall some dimensions that must be considered when designing an augmentation like the one shown in Fig. 1:



– **Execution Scope**: Typical Web augmentation artifacts [17] run in the Web browser once a particular Web page is loaded. This way limits the execution context, given that the augmentation artifact—so-called augmenter—is only running when the target Web page is already loaded. Some augmentations may require continuing working even if the target Web site is not being in use. For instance, when it requires to perform, information crawls from other sources or Web sites. In this case, the solution must be addressed under a server-side approach which allows, for example, to process big amounts of information whose results will be later displayed in the client-side augmentation layer. As an example of augmenter that would fit better running on server-side than in client-side are those based on Web scraping because of the required computing workload. For instance, Web scraper[1] is a browser extension that allows end users to define how to scrap information items from Web pages for further uses.
– **Cross-device integration**: In a client-side strategy, augmenters run into a single device, independently from other executions/installations of the same augmenter in other Web browsers. If a user requires any kind of integration/interaction among their devices, then a server-side back-end would be needed. For instance, this is the case of augmenters for distributed user interface interactions, where different Web browsers running in different devices need to be synchronized to offer the "distribution" effect [3].
– **Collaboration**: There are several collaborative tools (such as Evernote[2] or Diigo) that propose augmentation layers based on collaboration patterns, in which end users may annotate Web sites, add comments, etc. This kind of Web augmentation requirement needs a common database where all users' contributions will be stored. Considering other users for the same augmentation layers also enables the possibility of going beyond UI adaptation, because other aspects around personalization, such as collaborative filtering [18] could be achieved.
– **High computing and big storage requirements**: if the augmentation aims at satisfying requirements that demand high workload (i.e., image processing or deep learning), the Web browser may not be best option to execute the augmenter and, because of the limited resources, the navigation experience of navigation Web may be negatively affected. In cases like this, the artifacts requiring high-performance resources can be deployed at the server-side, and then its outcome is rendered at the client-side. For clarification, let us consider again the Grammarly extension; at the server-side's end, the sentences are processed using natural language processing which demands CPU resources, and storage for the text under processing.

It is important to note that these four dimensions are not examples of augmenters but representations of different concerns that may be involved in several kinds of augmentation requirements. This work proposes a set of abstractions to

[1] Web Scraper, https://chrome.google.com/webstore/detail/web-scraper/jnhgnonknehpejjnehehllkliplmbmhn?h1=es.

[2] Evernote, https://chrome.google.com/webstore/detail/evernote-web-clipper/pioclpoplcdbaefihamjohnefbikjilc?hl=es-419.

generate both client- and server-side components but without requiring advanced programming skill neither complex deployment process. With this in mind, it is clear that we are not focusing on a particular target of end users, because each particular augmentation application (and its underlying domain) has a specific users target and may vary among them in terms of amount and required skills. Some augmentation approaches proposes end-user development (EUD) tools for end users without programming skills [4], while others propose a collaboration method in which users with and without some programming skills (hobbies programmers knowing Web technologies such as HTML, CSS and JavaScript) cooperate [19].

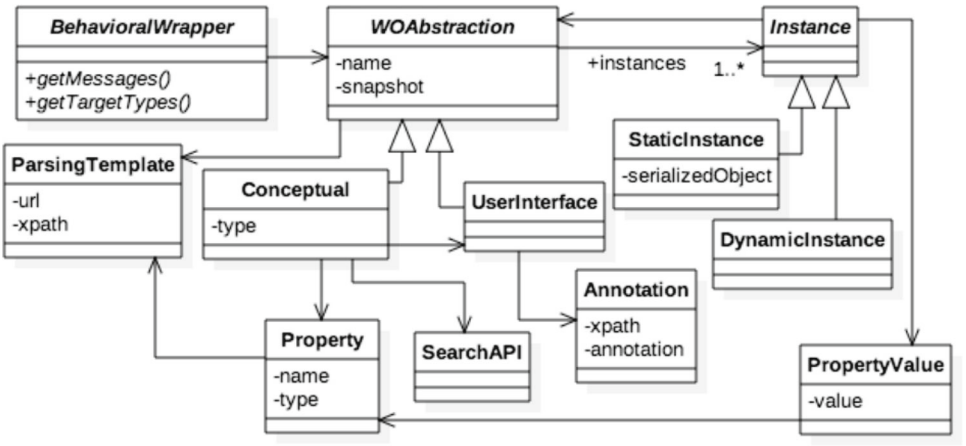
2 Background

Our approach aims to empower existing end-user development approaches with the possibility of defining a back-end counterpart of these augmentations by modeling these components rather by programming in a specific back-end technology. Several aspects of augmentation have been addressed through modeling activities, like the requirements specification [19]. Other approaches use model-driven technology for composing augmentations in the context of end-user tasks support [20,21]. These aspects are usually aimed to model the presentation layer, leaving aside the behavioral specification that may take advantage of the server-side components; such models are not enough to represent the back-end logic of an application. Web augmentation could be considered as a foreign application mechanism, since it may adapt third-party Web sites on the client-side. However, client-side adaptations could also be seen as a core concern during application design. An approach under this look used such a mechanism in an e-learning system [22]. The work proposes a modeling layer for client-side adaptation, but it is not used from the augmentation point of view, and their use is restricted to the application owners.

2.1 Web application augmentation

Web augmentation is a set of techniques used by a vast number of users; thousands of extensions for adapting Web content can be found at the Web browser stores, and significant communities support some of these tools. End users and other stakeholders with programming skills may interact in such communities for the creation, sharing, and improvement of specific augmentation artifacts. For instance, the Userstyles community (http://userstyles.org) offers a vast number of scripts that augment Web sites by adding further CSS specifications to change the presentation of the content. Userscripts communities, such as Greasyfork (https://greasyfork.org/), offer repositories of scripts with a broader spectrum of purposes since they support different weavers of JavaScript code (e.g., GreaseMonkey or TamperMonkey); therefore, it is possible to change not just the style but the content and behavior of a Web page. In all these communities, no matter which tool they support, there is a dependency between users with and without programming skills, since not all of them can implement the solutions they need, and sometimes they ask others for help.

This collaboration between end users and more skilled users is also considered in scientific works. For instance, as a motivation for the Sticklet [23] approach, authors differentiate between producers and consumers. Sticklet proposes a DSL that empowers hobby programmers (producers) by providing them with a JavaScript DSL that makes augmenter development more faster and less error-prone. From the end users point of view (the consumer), Sticklet benefits them by providing more controllable augmenters, given that Sticklet-based augmenters are more understandable. Other approaches, such as Pocket and Scenario [24], propose different development levels. While more skilled stakeholders may construct low-level blocks, the end user may encompass these blocks to arrange a particular augmentation effect. Although Web augmentation is still a very promising technology for EUD approaches [25], which is enforced because the Web is a natural platform for this discipline, there are already some interesting EUD approaches that directly empower end users without any programming skills. For instance, this is the case of WebMakeUp [26], a tool that allows end users to define augmentation by visual programming and programming by example techniques. However, these approaches usually are client-side standalone applications, without considering collaboration features. In general terms, there are interesting trade-offs between end users and augmenters developers [27]. Our approach is defined with the philosophy of empowering both end users and developers. In this way, our approach split the augmentation layer definition into different points for contribution. On the one hand, in some cases, the same end user could act as a developer for both client- and server-side counterparts. On the other hand, very complex server-side counterparts could be defined by more experienced stakeholders, while end users without advanced programming skills can maintain the augmentation control at client-side, by defining other augmentation concerns, such as content extraction and new UI widgets weaving. Either built using an EUD environment or not, Web augmentation artifacts are usually somehow coupled (by means of an XPath expression or a CSS selector) to the target Web page DOM. This problem is also shared in other kinds of artifacts, such as selenium Web tests. If a Web page changes its underlying DOM structure, these XPath expressions may not work anymore or spoil the results of an adaptation. This problem has been studied before, and although there are approaches to make them more robust [28], it is still possible that a



substantial change in the target Web page's DOM breaks the references. In spite of more robust expressions (either XPaths or CSS selectors), we also consider that this dependency must be minimized as much as possible. In this work, we encapsulate this dependency only in the extraction template defined by the user, while the rest of the application is unaware of it. In this sense, when a reference becomes old, only the extraction template must be redefined.

2.2 Web objects ambient

In a previous work, we presented a tool called WOA (Web Objects Ambient) [19,29], as the technological support of an approach that contemplates end users annotating and abstracting content—from diverse Web sites—as information items instances that later can be used—even in combination—to create diverse personal Web experiences, such as Web site augmenters and mashups. In this paper, we focus on its Web augmentation capabilities. WOA is implemented as a browser extension, and technical details about it can be found in the site of the project.[3]

In that approach, the end user should abstract DOM elements into *Conceptual* items. These conceptual items are obtained through DOM extraction templates, which are defined in terms of the WOA's metamodel, shown in Fig. 2. Both WOA concept and *Property*, which is related to a concept, are bound to a particular ontology ("type" attribute) to allow stakeholders to define different *Parsing Templates* for a same ontological concept, but materialized in different Web pages. Note that these conceptual items are defined by selecting the target DOM element and defining a specification that includes the XPaths matching that DOM element and its relevant properties. The main purpose of WOA is modeling how a concept and its properties are matched to a particular DOM element and how it would behave under different UI events.

Figure 3 shows the Web browser with the WOA extension installed, so the main WOA menu (top right corner) is available. Suppose that an end user is interested in the same agriculture Web site from Fig. 1. In this case, the user may select a DOM element that represents the concept of interest. Once selected, it is previsualized in the WOA panel (at the left of the figure). This panel lets users specify different properties of the concept being defined (such as the name), but also allows users to define which instance (concrete information items) will be extracted for this Web page. WOA finds several XPath expressions that are presented in a select menu. When the user selects one of this option, the preview changes to show him which are the concrete instances that will be retrieved with that selector.

As aforementioned, the specification is composed of a selector matching a container for each information item, and as many selectors as properties the item has.

In further steps, the user can collect multiple information items from different Web sites and have them instantiated in a global space of information, where those can be enhanced through specialized decorators that add some behavior to these objects [30]. The decorated instances can be used by the users to solve different domain-specific tasks by different means: through a default viewer presented in a sidebar, through an augmentation menu offered by some decorators or by specialized applications that use the WOA API to retrieve the collected data and wrap it. Summarizing, the users choose which information to collect and which behavior they want to associate with. Later, they can interact with those objects in multiple ways.

In this work, we are reusing the abstraction mechanisms of WOA to define a conceptual model of a given Web site. Moreover, the tool was enhanced to support back-end components.

[3] https://sites.google.com/site/webobjectambient/.

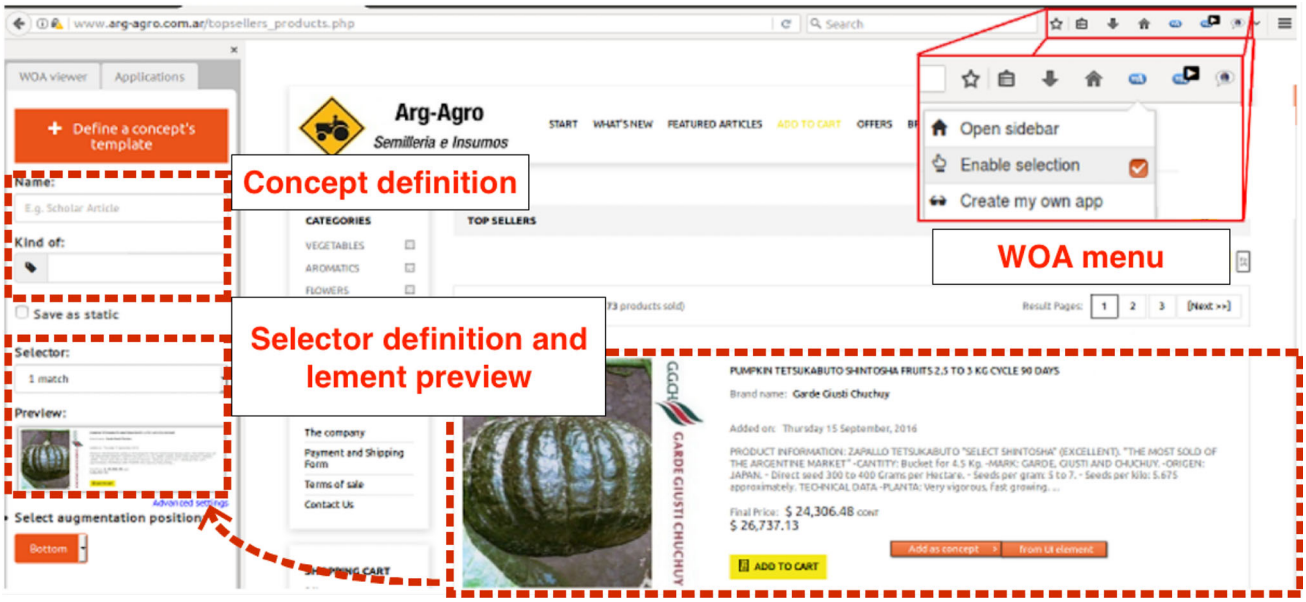


Fig. 3 WOA tool: main menu and concept definition

2.3 Interaction flow modeling language

In most mature Web design approaches [31], such as UWE, WebML, UWA, Hera, OOWS or OOHDM, a Web application is designed with an iterative process comprising—at least—the conceptual and navigational modeling. According to the state-of-the-art on model-driven Web engineering techniques [32,33], these methods produce an implementation-independent model that can be later mapped to diverse runtime platforms. For the sake of clarity, we will concentrate on the IFML approach.

Fortunately in 2013, the Object Management Group (OMG) [34] adopted the Interaction Flow Modeling Language (IFML) [35] as the standard approach for describing the interaction features of Web—and other kind of interactive—applications, making it part of the set of Software modeling standards like the Unified Modeling Language (UML) or the Business Process Modeling Notation (BPMN). The reader would find a complete description of the language elements, examples and a metamodel description in the IFML site [35].

IFML implements Model-View-Controller (MVC) metaphor [36] and inherits the simplicity and expressibility of its "parent" WebML. Additionally, the language supports the same kind of extension mechanisms that UML already possesses (e.g., stereotypes or tagged values), and it is itself described by a clearly defined metamodel. During this work, we will use the current IFML version supported by the WebRatio tool [15]. The use of this tool provides a solution to all the technical server-side aspects, such as configuration and management of hosting, storage and deployment. The user only interacts with the high-level platform, abstracted of all the technical issues, where only needs to configure basic storage parameters. In this version, each Web application designed using this approach must define at least the following models:

- The entity–relationship model of a Web application (a.k.a., application, domain, or content model) is focused on defining the contents of the application with their attributes and associated behavior.
- The navigational design of a Web application is aimed at defining *siteviews*, pages and components to enable the user to easily access, handle and navigate the information. Most of the Web engineering methods base their navigational model on two modeling primitives, namely Node and Link. IFML [37] is no exception to this, as it defines pages as logical views on application model classes and links as the hypermedia realization of application model associations that define either browsing capabilities or triggering system behavior.

IFML defines a broad set of model elements that can be used to generate a Web application. In the following sections, we will apply IFML for describing our approach considering a subset of available language elements in real-life examples.

3 A model-driven Web augmentation approach

In this section, we introduce our approach and present a brief characterization of the augmentations' requirements that demand client- and server-side supports.

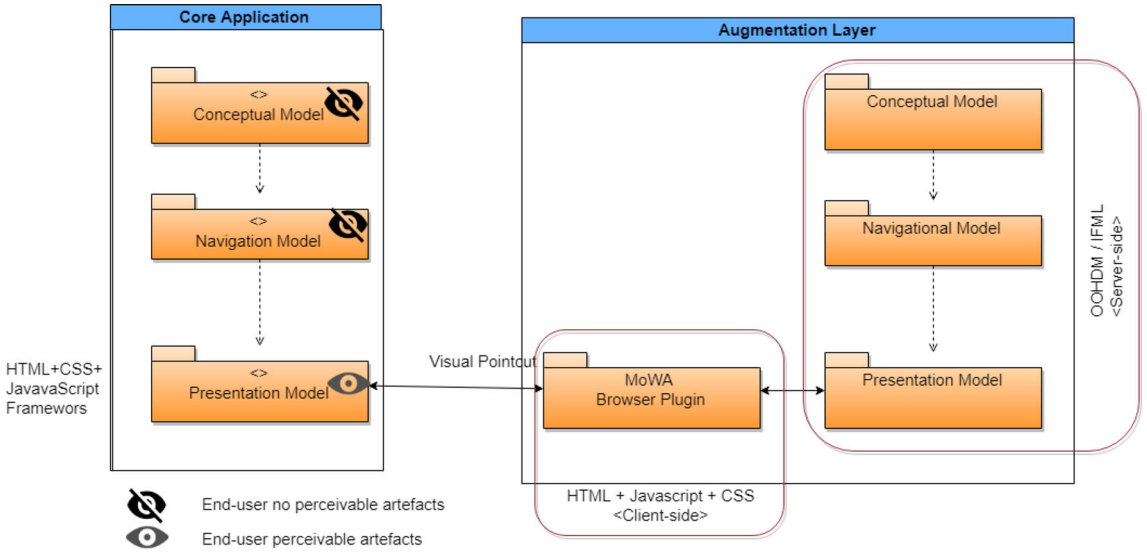


Fig. 4 Approach schema

3.1 Our approach in a nutshell

Our approach is based on the idea that even the most straightforward behavior (e.g., a new community comment feature) should be considered as a first-class functionality and, therefore, it should also be designed accordingly. Its design and implementation should be separated from the host site (from now on, the "core application"), and decoupled as much as possible from the stable functionalities of the core application. The outcome of the approach is a new application layer, so-called augmentation layer, which comprises packages of artifacts for client- and server-side features as shown in Fig. 4. In the following sections, we explain how the aforementioned principles have been put into practice.

Augmentations cannot be introduced in the core application since the augmenter analyst is not part of the application development; he or she will contribute with new features that were not already contemplated by its owners. Augmenter analysts may not be software engineers, but they need to have at least some technical knowledge of the Web. That is to say, he/she must be familiar with the class, variable and relationship concepts from UML to model the business domain, and he/she must make sense of the structure of the Web, like pages, links, and the basic HTTP request consequences to design the augmentations. Conversely, he/she does not need to have basic programming skills such as SQL, JavaScript and any other general-purpose programming required to develop the augmentation in an ad hoc way. The augmenter designer has limited comprehension of the system based on what he/she perceives in the Web site ignoring if there are other system modules available, what other system roles are allowed to do and low-level artifacts such as source code and database schema. As the augmenter does not have access to resources like APIs or database, the augmentation will be built on top of perceivable pieces of information neither composing nor reusing assets. This constraint is shown in Fig. 4 where the presentation model is only perceivable. The browse plugin manages the weaving of the augmentation artifacts into the page application.

Building on the above ideas, our approach can be summarized with the following design steps, which are also shown schematically in Fig. 5; first, the front-end components (e.g., containers) of the augmentation are created; then, the conceptual model of the core application is gathered through DOM annotations; after that, the requirements are modeled using a MDWE approach; and finally, the modeled solution is connected to the augmentation components. Note that, after Step 2, which defines how a domain object is extracted from a Web page), the Step 3 includes that object definition as part of the conceptual model layer. This third step generates a new application based on a back-end whose results (an HTML widget or a set of HTML widgets) are woven at client-side by the WOA plugin each time that the target Web page is loaded.

(1) The augmentation is decoupled from the core application by introducing a design layer (called augmentation Layer), which comprises additional conceptual, navigational and interface models in addition to the ones of the core application.
(2) The basic conceptual model is captured by tagging the information presented in the core application pages, in a similar fashion to WOA [14]. The lack of access to the underlying models of the core application requires the generation of a second conceptual model; the one inferred by the user perception. In this process, data

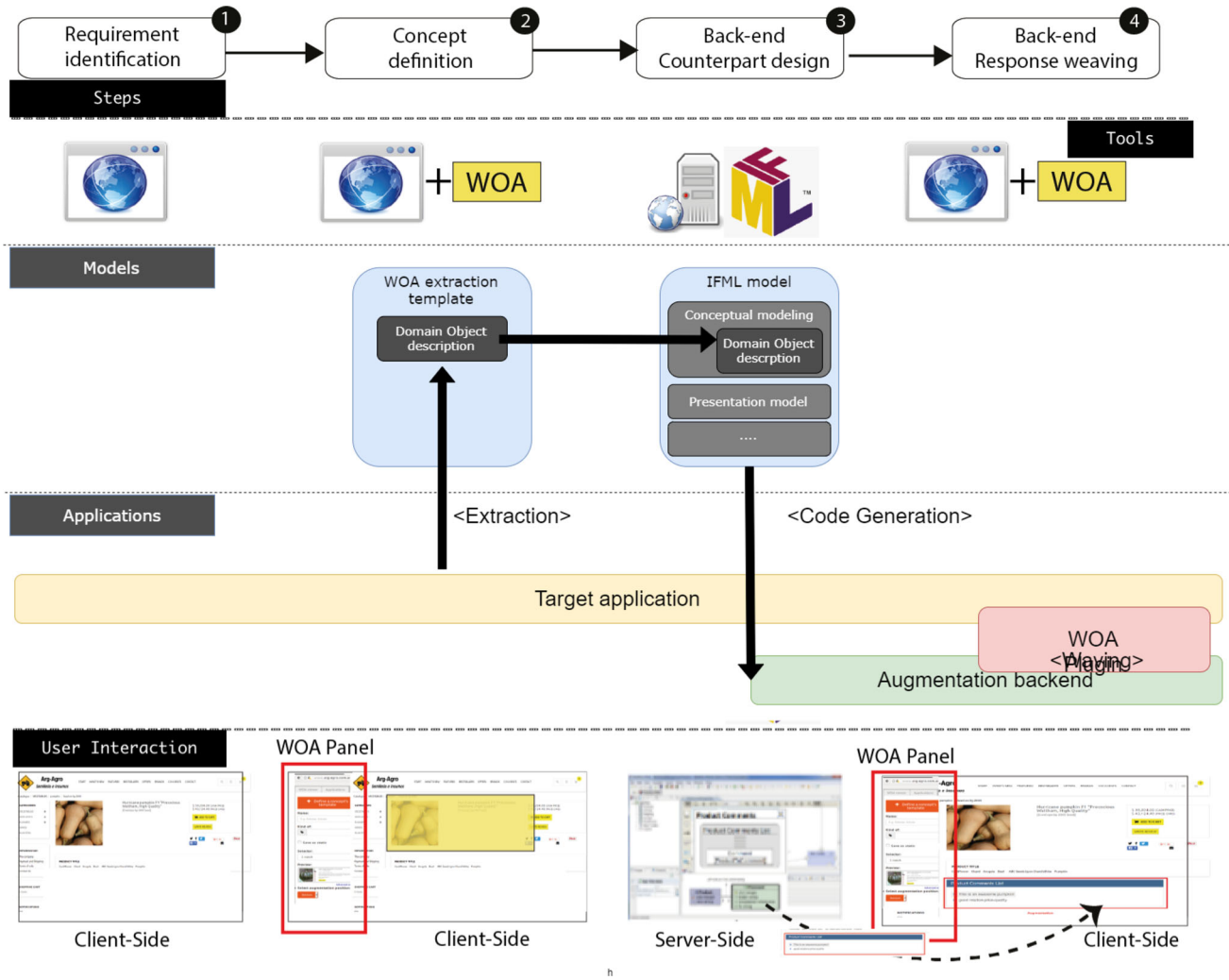


Fig. 5 MDWE process: steps, tools, models, applications and user interactions

elements in the page are tagged and grouped by an *augmentation analyst* into an entity definition to obtain a simplified conceptual model. The *augmentation analyst* is a skilled end user with advanced knowledge of Web applications, whose goal is to improve a core application. In further steps, the model instantiation in a particular user session is used to provide contextual information to the augmentation engine by providing model instances information when triggering the augmentation.

(3) Augmentation requirements are modeled using Web engineering notations (e.g., use cases or user interaction diagrams) and separately mapped onto the following models using the heuristics defined by the design approach (e.g., [38]). Notice that as shown in Fig. 4, the augmentation requirements are not integrated into the core requirements model, leaving their integration to further design activities.

(a) New behaviors, i.e., those belonging to the augmentation layer, are modeled as first-class objects in the augmentation's conceptual model. Such a model defines all the objects and behaviors corresponding to the new requirements. Additionally, it may include the core application conceptual classes (captured in step 2) perceived by the *augmentation analyst*, allowing defining relationships between the augmentation business model and the core application. Notice that this strategy can be applied to any object-oriented method, i.e., any method using a UML-like specification approach.

(b) Nodes and links belonging to the augmentation's navigational model may or may not have links to the core navigational model. The core navigational model is, in contrast, oblivious to the augmentation's navigational classes, i.e., there are no links or other references from the core to the augmentation layer.

This principle can be applied in any Web design approach.

(c) We use a separate specification for the connection between the core and the augmentation nodes. As we show later in the paper, the integration is achieved at runtime as part of a client-side weaving engine. Conversely, the integration can be performed during model transformation in other model-driven approaches [31].

(d) Optionally, we design (and implement) the interfaces corresponding to each concern (core and augmentation) separately; the interface design of the core and augmentation concerns are described using Mockups—widely accepted by the agile community.

(4) Core and augmentation interfaces are woven by evaluating an integration specification, which is realized using DOM transformations. Again, the idea of model weaving is generic and, therefore, different approaches can produce the same result.

Once the augmentation requirements are modeled in Step 3, the augmentation of another site may be quite straightforward, since it is just a matter of mapping the virtual classes (Step 2) and specifying how to wave the augmentation UI artifacts (Step 4).

3.2 Supporting toolset

Our approach has been instrumented on top of WOA and IFML approaches which were described in Sect. 2. The WOA approach has a tool available in a public repository[4] that includes source code and setup manual. Moreover, different tutorials have been published to YouTube [5] to train users. Regarding IFML, we use IFML notation to model server-side artifacts required for augmenting a Web site and generate the code using WebRatio (the official supporting tool for IFML method). In WebRatio's Web site, the reader will find tutorials, and a forum to discuss any doubt. To ease the replication of the case study presented in this article, a step-by-step guide is provided,[6] that can be followed to run the augmentation. The used tools are referenced in the description of the examples.

[4] https://github.com/pmaglione/mdwa_comments.

[5] https://www.youtube.com/playlist?list=PLHuNJBFXxaLCXzK_0pTkANEZ9RXsUsCSO.

[6] Examples documentation and videos https://drive.google.com/drive/folders/1PVSGmClH8MbVvJpeGq7eeuo7Yojqf9CE.

3.3 The core application and the conceptual model extraction (Step 2)

Our approach is based on the creation of objects that are specified by abstracting the Web page's content [19]. In this regard, a visual programming tool has been developed which extends the capabilities of the browser. It allows users to select and abstract any DOM element on the Web by associating them with a conceptual class. The properties of the DOM element can be defined as well as the population by using the values of its children elements. The process is shown in Fig. 6, and it implies (1) the user enabling the selection of elements in a given Web page, (2) choosing a concrete element through a contextual-menu and (3) filling a form at the sidebar. The final step using WOA is performed once the back-end counterpart is defined. At this point, the augmentation analyst defines how the augmentation layer will be woven in the Web site by using a select menu showing different available insertion strategies (4).

The "ArgAgro" site has been used as a target core application to introduce the four steps of this process. Enabling the DOM selection (1) involves clicking a button at the browser's toolbar. Such interaction enables the DOM element selection, which applies a special CSS class to highlight the DOM element that is currently pointed, so the user can clearly appreciate which is the current target element that can be collected with the tool. As a result of such interaction, a particular context menu is enabled, which allows defining a concrete element in the DOM as a concept or as a property (2). Once the DOM element is selected, a UI form is opened at the sidebar (3), which lets the user select:

1. a name for identifying the concept. e.g., "ArgAgro seed."
2. a conceptual class to associate with. e.g., "Seed."
3. the number of occurrences that could be extracted from a single Class. It is transparent to the user, who just sees a combo with occurrences, but internally it is about an XPath selector matching one or multiple UI elements. e.g., "//*[@class='product-container']" in such page can be used for retrieving up to 28 instances.
4. a position for the augmentation concerning the UI element matching the concept, e.g., "top" or "bottom."

There are no restrictions on which DOM element the users can tag or which name they can use for the conceptual class. If the latter does not exist, it can be defined in further steps. The benefit of having concepts associated with conceptual classes is the possibility to reuse the existing behaviors for similar concepts. For example, if the user has already abstracted the concept "ArgAgro seed," it is likely that some behavior can be reused for the creation of the "Amazon seed."

Regarding the occurrences, a combo is automatically filled with multiple options matching different XPaths applicable

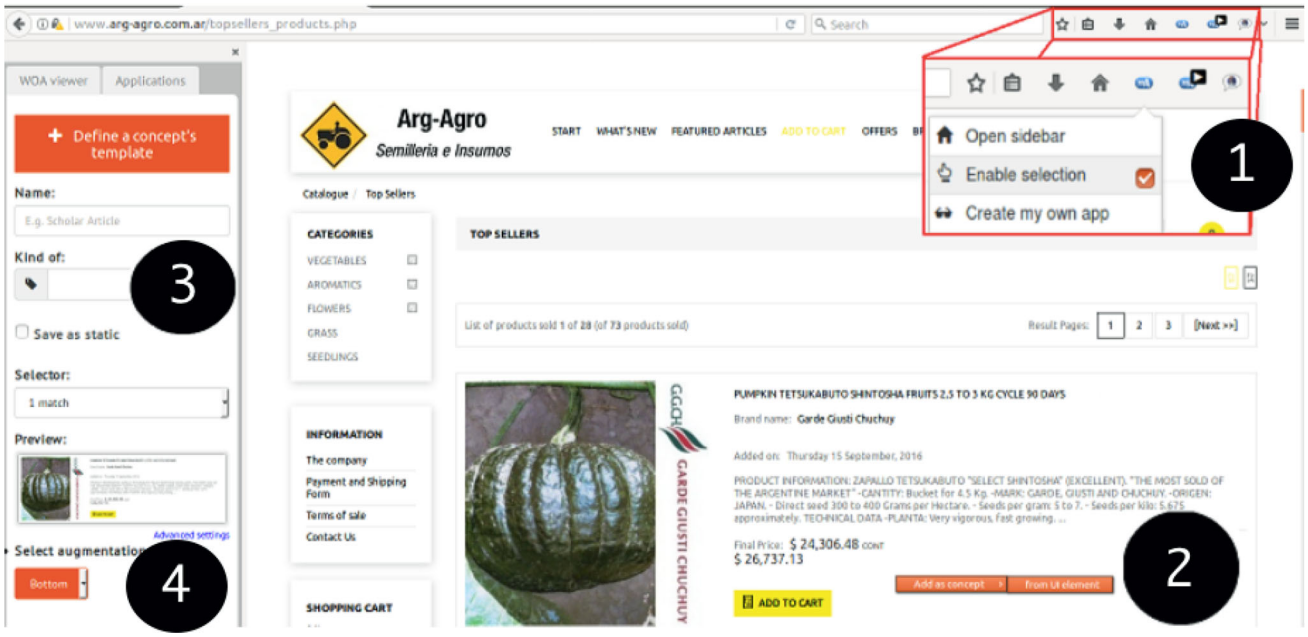


Fig. 6 Concept definition

to the selected DOM element. It is possible to unequivocally reference such a single element, or a set of similar elements instead (e.g., the ones with the same set of CSS classes, the ones at the same level in the DOM tree). The user should choose the occurrences according to his needs.

Properties can be collected in the same way; the only difference is the addition of a combo for linking such property to any of the existing concepts defined for the current Web application.

At the end of this stage, users may see the collected classes and instances in a special panel in the sidebar, from where the user may also export the specifications in JSON format. Further aspects of the abstraction of Web content as domain objects may be found in the previous work [14].

The outcome of the conceptual model extraction is the definition of the "user-perceived conceptual model" and the "augmentation configuration" (e.g., where it will be positioned). Both specifications will be used in the following steps: the first one is used as the starting point for the augmentation layer modeling. The latter one is contrast used to instantiate the concrete concepts at runtime.

3.4 The augmentation layer modeling

The augmentation request is triggered by our Web browser extension when the augmentation requirements are met (e.g., the URL matches a given pattern) once the target core application is accessed. The extension asks an application server to build the content that represents the augmentation and embeds its response in a specific position of the core application visited page. The application server runs application models (conceptual, navigational, and interface) to resolve the request. Next, we introduce the steps involved in designing each model.

3.4.1 Conceptual model (step 3a)

Augmentation functionalities, like customized recommendation systems, may involve brand new content classes (e.g., the class modeling a Comment from a user) or the enhancement of existing core application classes and its behavior that are persisted at server-side. In our approach, a class diagram is designed defining new classes and the enrichment for virtual classes extracted in Step 2. Such classes are used to augment the core application's behavior. Additionally, the virtual classes arise from those concepts detected in the previous step and are representations of the information perceived by the augmentation analyst. Virtual classes are enhanced with new attributes (e.g., a product may be enriched with a ranking attribute) or relationships to other virtual or augmentation classes (e.g., a product may be enhanced with a relationship to a review class). Any new feature defined in a virtual class will be woven automatically and must be considered as a decorator [30], since it allows adding new features (properties and behaviors) to an application in a non-intrusive way.

In our approach, augmentation functionalities might be new behaviors that are added to the conceptual model (and which might encompass many classes) or full-fledged navigation models, containing new nodes, links and even relationships with conceptual classes. Each augmentation

functionality is treated as a self-contained sub-system and modeled using the selected MDWE approach. The notation is similar to symmetric approaches for separation of concerns such as the one described in [31]. In the case of IFML, the business model is specified in the domain model diagram using the WebRatio tool. The tool generates the Java code related to the modeled entities and relationships. Moreover, the tool generates the ORM descriptors for Hibernate framework which allows persisting and retrieving objects from/to most of the database engines (i.e., MySQL, Microsoft SQL, and Oracle).

3.4.2 The navigational design (step 3b)

The augmentation and the core navigational components are connected using a specification for integration at the navigational layer called *Affinity*. It is a formal specification that indicates, for instance, whether the augmentation features are "inserted" in the core node or if they are connected with a hyperlink. This specification also includes a query indicating which core nodes will be considered in the extension. Nodes matching the query are affected by (or enhanced with) the augmentation functionality and represent the affinity of the augmentation functionality. It is possible to define one or more affinities for the same augmentation functionality, i.e., the same functionality might be incorporated in different parts of the application, by following different rules. The affinities of an augmentation functionality are specified with the same query language used in OOHDM to define nodes [39] which is supported by our tool, which is presented in the next section. The language is based on object queries. The affinity is a model-level specification that is supported by the WOA tool for both defining and interpreting the definition. Other MDWE approaches can borrow the concept for specifying crosscutting behaviors. By using this query language, the definition of an affinity assumes the following form:

```
AFFINITY: AffinityName
FROM C1..Ci
WHERE Predicate
INTEGRATION: Extension | Linkage(V1..Vi)
```
(1)

In the affinity template (1), *AffinityName* is the name associated with the affinity, $C_1 \ldots C_i$ indicates core node classes involved in the query. *Predicate* is a logical expression defined in terms of properties of model objects. It determines the instances of the core node classes $C_1 \ldots C_i$ that will be affected by the augmentation functionality. *Extension / Linkage* indicate the way the augmentation functionality is integrated into core nodes through the augmentation nodes $V_1 \ldots V_i$. An extension indicates that the core nodes are enhanced to contain the new functionality information (and operations). In a linkage integration, the core nodes "just" allow navigation toward the augmentation nodes $V_1 \ldots V_i$ which contain the augmentation functionality, and therefore does not support new behaviors. In the case of linkage integration, we can also specify additional features such as attributes or anchors that have to be added to the extended node (e.g., to make navigation more clear).

3.4.3 Structural weaving of an augmentation (step 3d)

As a consequence of inserting augmentation functionalities into the conceptual model or the navigational model, new interface elements must be added into the interface model, therefore introducing new fields with data or control interface objects, like anchors or buttons. Though we described this process in [40], we briefly review it here for completeness and readability reasons. Each concern (core and augmentation) will comprise Mockups for its corresponding navigational nodes. The style of the final page is defined by specifying how the augmentation interface will be inserted into the core Mockup. This design is made during the interface design stage when a node should exhibit some augmentation functionalities. More specifically, we indicate the relative position of the added interface objects concerning the core interface objects. A simple specification language has been defined to express the integration. It allows indicating point-cuts and insertions at the abstract interface level, (i.e., the position) where the augmentation's Mockup has to be inserted in the core's Mockup (2).

```
Integration: IntegrationName
Target: MockupTargetName
Add: MockupSourceName | InsertionSpecification
[Relative to: Widget name]
Position:
   [above | bottom | left | right]  |
   [ float [ left | center |right] [ top | middle | bottom ] ]
```
(2)

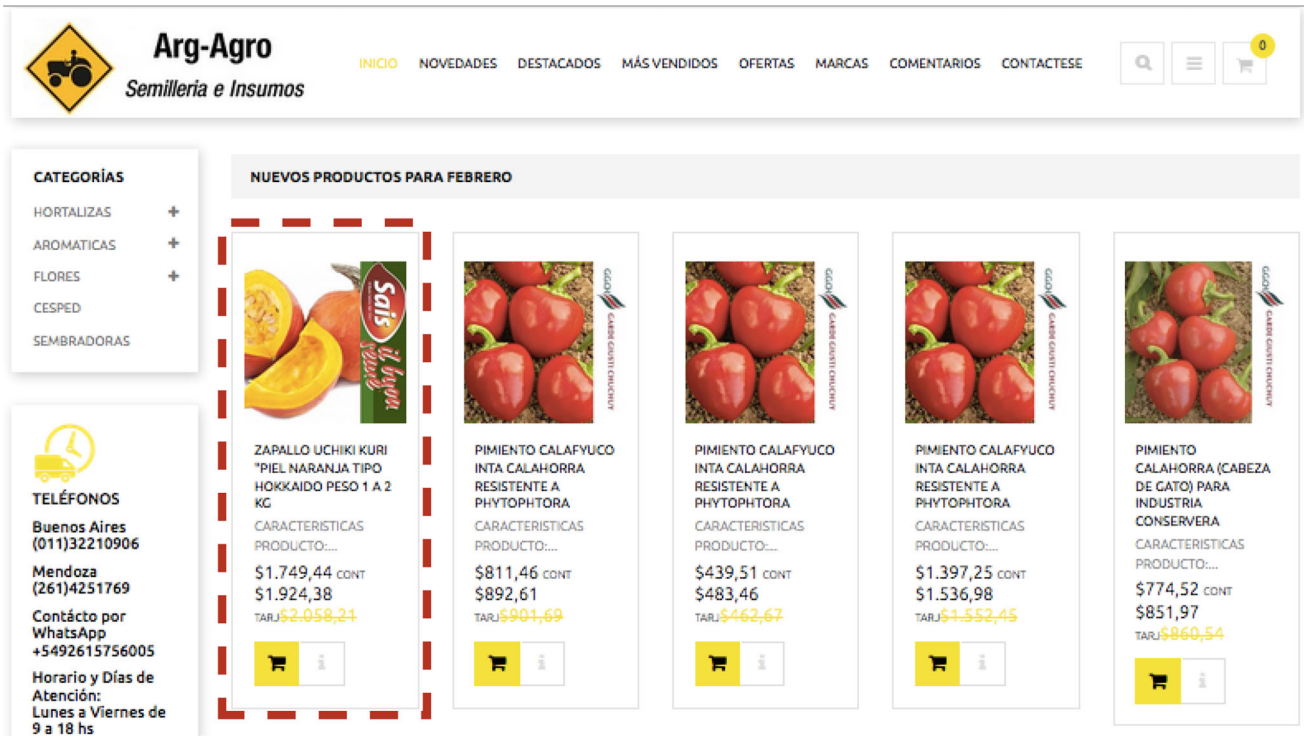


Fig. 7 Concept and instance identification in existing Web content

The field *Integration* is an identification for this specification. It may refer to a navigational affinity since the same UI integration specification can be used with many navigational affinities. The field *Target* indicates the names of the Mockups (one or more), which will host the augmentation interface code. Inner Mockups may be specified using a "." character. As an example, we will write a product. Reviews reference to indicate that the insertion will take place in the Reviews' Mockup, which is a part of the product's Mockup. The *Add* field indicates which elements must be inserted in the target, either a Mockup or an immediate specification which is used when the inserted field is simple enough to avoid the specification of another (auxiliary) Mockup. Finally, we indicate the insertion position by using the *Relative* and *Position* fields. It is worth noting that the specification is still "abstract," enabling the fine-tuning for the following stage, during implementation. Like the affinity case, the integration specification is a formal definition that must be instrumented to weave the user interface, for example, using the WOA tool or any other augmenter script. Often, the script runs at the client side but it can be executed at the server side in a proxy server.

4 A descriptive scenario

In this section, we describe two augmentation examples performing all the steps proposed in the approach. We will use the agriculture business domain specifically enhancing Web sites that sell different products required for sowing. The examples explore the dimensions introduced in Sect. 1.1. These augmentations attach a business model and user interface components. The execution scope is enlarged including the server-side environment where available resources are used for high computing and big storage. In the examples, collaboration requirements are instrumented using augmentation allowing several users to interact. The Web browsers and user scripts plugin (i.e., Tumpermonkey[7]) ease the cross-device integration.

4.1 Implementing a collaboration feature

In the first place, we extend an application with the possibility of posting comments stored in a server for sharing opinions about products as shown in Fig. 1.

4.1.1 Extracting conceptual model from core application

The first step in the approach is to capture the underlying model perceived by the end user that will be used later to enhance the Web application. For instance, Fig. 7 highlights a particular part of the DOM containing information about an instance of the product class. It is also possible to see that other DOM elements are representing more instances of the

[7] Tumpermonkey—https://www.tampermonkey.net/.

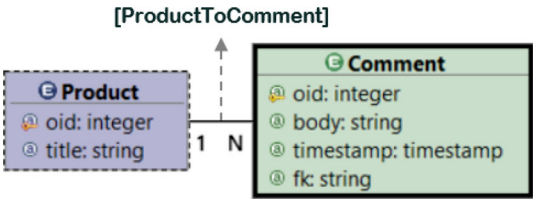


Fig. 8 Entity–relationship model

same class. These instances seem to present similar information related to different properties of the product virtual class, such as its name, its price and a brief description. The main idea is to define the product concept as a class, with its properties and their relations to the DOM elements, so it is possible to extract the properties' values for each concrete instance.

4.1.2 Modeling augmentation layer

We have designed a Web application using IFML approach to support the augmentation aspects at the server-side. It comprises the conceptual, navigational and interface models. The application supplies the back-end support to the whole augmentation application.

Once the virtual classes are extracted from the page, we designed the conceptual model documenting how the augmentation enriches the model of the core application. In our example, a product virtual class is enriched with a set of comments that represent the community's reviews. It is important to note that the virtual class being modeled in IFML only requires an OID which denotes a unique identifier, the attributes from the page which are going to be handled at the server side, and the new attributes that enhance its definition. In IFML, the first step is describing the conceptual model using entity/relationship modeling. In Fig. 8, we show a model, which was designed using the WebRatio tool, where a product is associated with its comments. The product virtual class (with dotted lines) is augmented with a multi-valuated reference to the Comment entity; the reference is automatically named ProductToComment by WebRatio. It is noteworthy that the Comment entity does not need to state all the information available on the page; indeed, it is just required a base set of information (e.g., any information that identifies the object such as an id or a name) that let a basic lookup implementation.

Actions on the client-side require—in addition to the conceptual model enhancement—improvements at navigational models to describe the behavior of the application. It is modeled using units, components and links as part of an IFML site view. In Fig. 9, a page is defined listing all the available comments for a given product browsing the ProductToComment relationship. Furthermore, a product's id is required for resolving the comments when browsing the relationship

Fig. 9 An IFML navigational model

shown in Fig. 9. At the bottom of the figure, there is a URL responsible for triggering the server-side behavior which sends the required id valued "pumpkin." The augmentation extension requests the URL at the browser-side once the user has visited the target page.

The next step in the augmentation design is the specification of how to achieve its composition with the core application navigational model. To do that, we use the affinities introduced in Sect. 3.4.2. In the specification (3), the affinity points out that the ProductDetail node (the one perceived by the augmentation analyst) must include the behavior required for listing the product's comments, such as retrieving the products list and generating the required hypermedia links.

```
AFFINITY: IncludeProductComment
FROM ProductDetail
WHERE Predicate
INTEGRATION: ProductComments
```
(3)

Finally, the designer must study the best UX design to present the list of comments. The visual point-cut is declared in the specification (4) based on the integration rules (presented at Sect. 3.4.3). This determines the user interface point-cut and corresponding transclusion position to be done. Each time a user accesses a page, the captured augmentation model instance (in this case, a Product) is sent to the server for its processing and further rendering. The outcome will be transcluded into the core application page using the configuration data. As a result of augmentation execution on the client side, the Web page is modified listing the product's comment as expected.

```
Integration: Community review
Target: ProductDetail
Add: ProductComments
Relative to: ProductTitle
Position: bottom
```
(4)

In Fig. 1, the augmented comment section was inserted above the title of the product. The weaving of the Web content is achieved with the Web Object Ambient (WOA) extension [19], which then calls the proposed service for performing the augmentation on the client side. It is important to highlight the importance of a good annotation process to get the proper instances (at content-extraction time). Otherwise, the augmentation may not present any results.

4.2 Orchestrating complex inter-augmentation augmentations

In [41], we introduced the definition of concern-sensitive navigation (CSN) for a Web application when the content, links and operations exhibited by the core application pages are not fixed for different navigation paths, but instead can change when accessed in the context of various navigation concerns. To make this definition concrete and practical, we assume that users navigate through navigation objects which are the realization of hypermedia nodes. Suppose a navigation object N_j, which is an instance of a navigation object type N, while in the conventional Web navigation, this object will exhibit the same content and links regardless of how it was reached. In CSN, its properties can be slightly adjusted according to the current user's concern. Web augmentation can be used to implement CSN in Web applications.

An example of CSN for *Keeping Navigational Concerns persistent through different applications* [41] let end users to develop a complex task by browsing the nodes from different Web applications. The applications' nodes are enhanced with new content, navigation and links according to the task purpose, supporting the underlying user's activity that motivates the navigation. As a result, the user's navigations may help to improve the user's experience while performing the activity.

Let us consider now another scenario in the domain of agriculture: planning a vegetable production by a farmer. One of the minor tasks of the planning activity is the research of different product alternatives used during vegetable production. During this task, the user needs to gather information about seeds, pesticides and fertilizers (i.e., name, details and cost), as well as to prepare a final report summarizing the information of the preselected products. Conversely, without a CSN support, the users must be aware that each time they jump from one site to another (while browsing different Web sites), they lose the information present in the pages, and they may want to keep notes of the relevant information for easy comparison. As an alternative, nowadays the users tend to leave open a dozen tabs with product information. This sort of practice limits the research activity done by the user to a specific browser because the information is not shared among other devices (i.e., smartphones, tablets and notebooks) owned by the user. To support the product research activity in the planning of vegetable production, we will model CSN using our approach, relying on the client- and server-side components.

4.2.1 Extracting conceptual model from core application

In Fig. 11a, we present the conceptual model composed by three entities:

- **Seed** to capture alternative species of a given product.
- **Pesticide** to identify the requirements and constraints of the products and the vegetables.
- **Fertilizer** to identify the product and used technique.

This model is used for persisting a pre-selection of seeds, pesticides and fertilizers that will be evaluated to define which vegetable will be produced by a farmer.

4.2.2 Modeling the augmentation layer

Modeling the navigation layer in this example is pretty similar to the transclusion example presented in Sect. 4.1.2 on page 18, because the navigation must be modeled and the affinities and integration specifications must be declared. This example introduces the challenge of aggregating information gathered from the navigation of several sites and their persistence. Therefore, the back-end component plays a primary role since it allows to centralize the user's activity information storing the navigation's state defined by navigating several Web site. Moreover, the back-end component can be used to provide a liquid [42] and multi-device computing environment, where data and applications seamlessly move between multiple devices and screens and are transparently synchronized.

Figure 11b presents the navigation model required for supporting the products' information gathering task. Each time a user access a Web site that contains a seed, a pesticide or a fertilizer concept, a bucket (selecting seed, selecting fertilizer and selecting pesticide) of such entity lists the product's details (concept attributes such as the name or the price). Besides, it is possible to add the current visited product to the list or delete one product from the list. At any time, the end user can review the current task advance by accessing the planning report page through a link present in each bucket.

The *flow* from selecting seed, selecting fertilizer and selecting pesticide pages to the report one allows such navigation.

The integration of specifications supporting the selection task of products (seed, fertilizer, pesticide) are listed next to the specifications (5), (6) and (7), respectively:

```
Integration: Selecting Seed
Target: SeedDetail
Add: Selecting Seed task
Position: float right top
```
(5)

```
Integration: Selecting Fertilizer
Target: Fertilizer Detail
Add: Selecting Seed task
Position: float right top
```
(6)

```
Integration: Selecting Pesticide
Target: PesticideDetail
Add: Selecting Pesticide
Position: float right top
```
(7)

The result of enriching a Web site with features that allow the user to gather information—to be used later—during a process that involves several activities in different Web sites, as shown in Fig. 11c. The first step is the Seed selection at Lowe's[8] which will be later used for augmenting the site considering the integration specification (5). Then, several fertilizers are selected by a farmer from Amazon[9], where the augmentation is based on such specification (6). As a third step, support for saving a list of pesticide is added to the TractorSupply[10] website honoring the specification (7). The user can access a summary page where all the gathered information is shown and can be used to make a decision for the vegetable to be produced. Note that the navigation must not be performed necessarily in the presented order. Indeed, the user can browse all the applications (back and forth) until he/she obtains the information he/she needs for making a decision. As long as new augmentations appear, they can coexist in the applications being run on the cloud. That is to say, a given app can run different augmentations at the same time. The examples described in this section can be deployed to enhance apps on the internet as shown in Fig. 10. Amazon site can host both vegetable production planning augmentation as well as comment reviews. On the other hand, the ArgAgro site can also run the augmentation for planning a production.

[8] https://www.lowes.com, last accessed May 17th 2019.

[9] https://www.amazon.com, last accessed May 17th 2019.

[10] http://www.tractorsupply.com, last accessed May 17th 2019.

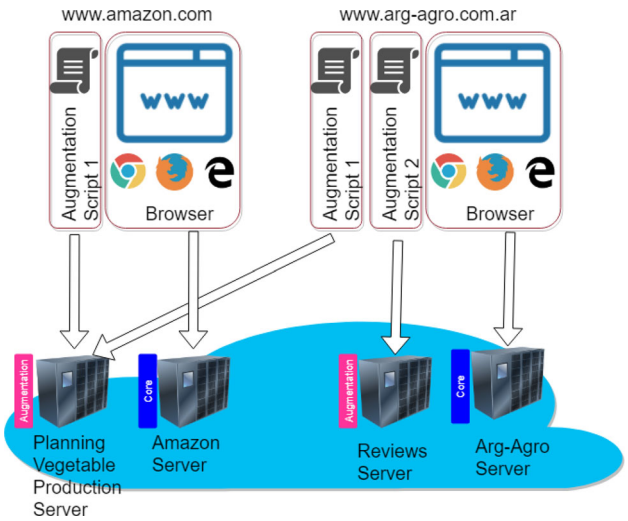
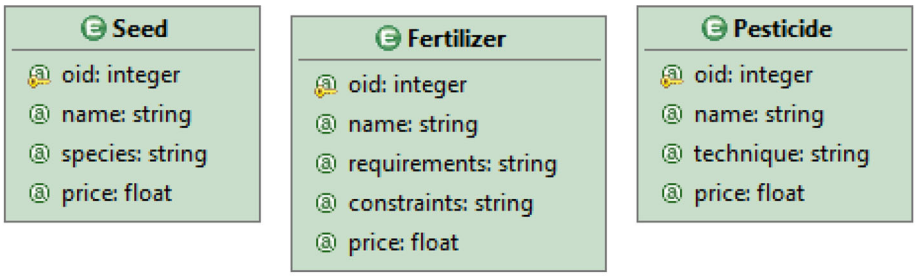


Fig. 10 Applications and servers schema

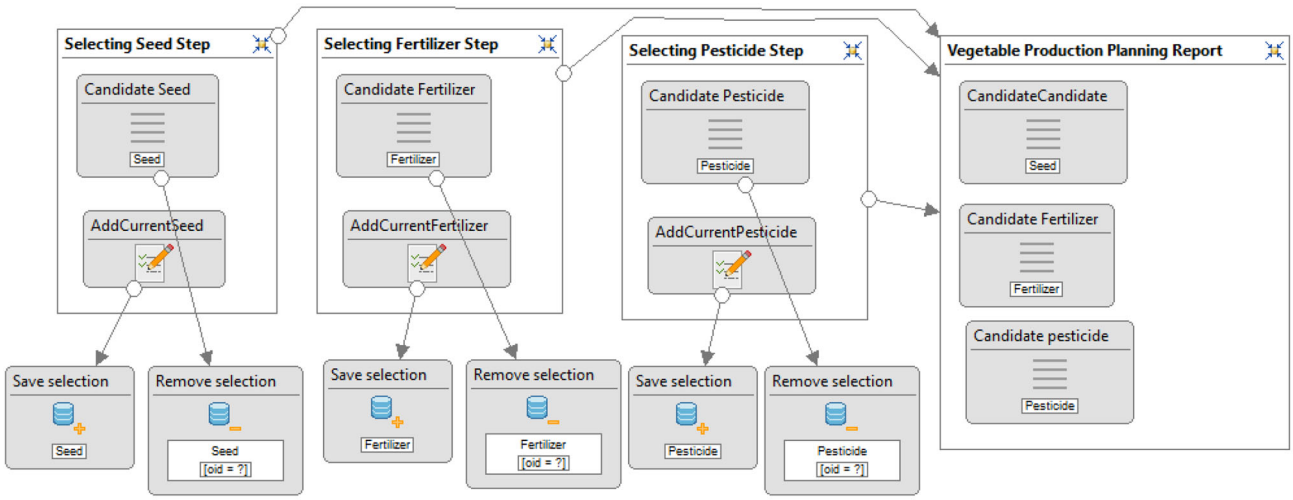
5 Rationale about the use of our approach

Augmentations can be solved at the client-side or by combining client- and server-side resources, using different approaches. In this section, we introduce a brief discussion including the pros and cons of each strategy. In the discussion, we outline the potential system modification for each Web application layer that could be introduced when implementing an augmentation requirement by using 1) a standard Object-Oriented approach, 2) a standard augmentation technique on the client-side, and 3) a client- and server-side approach.

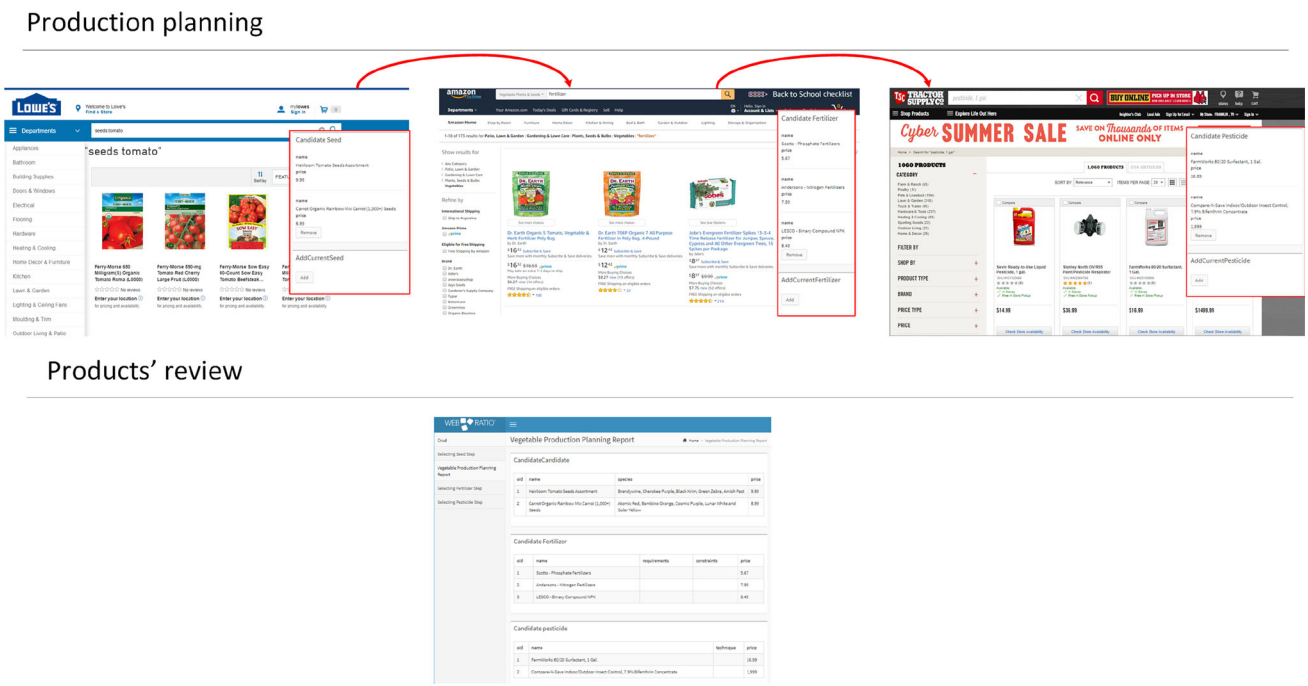
– **Ad hoc**: In this setting, augmentations are managed as a new requirement, so the application source code is modified honoring its development process (agile or RUP). When using a "bare" object-oriented (OO) approach to implement augmentation, new elements such as variables, relationships, and accessors must be coded inside existing classes in which the augmentation analysts do not have access at all. The changes compromise all Web application layers: conceptual, navigation and interface. In this case, the extensions are new features that have full access to the application resources, such as the databases' connection, the static resources or the business processes. However, the change life cycle must be manually managed, including the development, testing and deployment tasks of the new feature. Additionally, if the feature is no longer needed, it must be removed from the source code, the application needs to be tested again and finally released. The users are only able to contribute when they are involved in the development process as part of user experience studies, and they do not collaborate in the software development process. When using a tradi-



(a) Conceptual



(b) Navigation



(c) User interfaces

Fig. 11 Supporting concern-sensitive navigation

tional OO approach, volatile functionality development does not only compromise the coding task, but it also demands additional effort on testing, since the affected features must be tested twice, both for the introduction and removal of the augmentation requirement, if the community does not adopt it, like augmentations. Maintaining volatile code has been studied in our previous works [10,43–45] where we pointed out the detrimental consequences of maintaining this kind of features.

– **Client-side Web Augmentation (CS Aug.)**: In this case, skilled end users with strong knowledge of programming in cutting-edge technologies, such as HTML, CSS, JavaScript frameworks and Web browser extensions development, undertake the Web application augmentation development by manipulating the DOM. The outcome of this task is a set of JavaScript assets packaged in a Web browser plugin (i.e., Chrome extensions or GreaseMonkey plugin's extensions[11]) tied to the application. Notwithstanding the above, the augmentation's features are limited to the browser extensions APIs capabilities and the content available on the Internet which can be mashed up for producing a new experience. When augmentation is no longer required, the user deactivates the script, and the augmentation stops working. User experience teams can profit from this feature for designing application improvements, tests and finally promote them as core features by merging the augmentation artifacts with the application's source code.
– **Code-Based Client- and Server-side Web Augmentation (CB C&S Aug.)**: As a regular client- and server-side augmentation demands, as stated in the previous case, highly qualified developer is able to understand client and server technologies (i.e., CSS, HTML, AJAX, Java or Python) as well as HTTP protocols to implement an augmentation. Although there is an active Web augmentation community, it is working under a "code-based approach" which excludes many of the end users because of the requirement of technical knowledge. Often low-level solutions reduce the abstraction level and the augmentations become handicrafts with a limited reuse capacity. Overall, one of the major drawbacks of using conventional approaches (e.g., OO approaches) for handling augmentation concerns is the fact that their life cycle has to be supported in an ad hoc fashion, by hard-coding the associated logic. That is not possible in the context of Web augmentation because of its nature: users enhancing sites seeking a new unsupported goal.

In Table 1, we highlight the advantages and disadvantages of the diverse augmentation strategies currently available and our approach.

[11] http://www.greasespot.net.

Table 1 Web augmentation approaches comparison

Aspect	Ad hoc	CS aug.	CB C&S aug.	MDWA
Access to resources	**Full**	Limited to browser	Limited to browser and independent back-end	Limited to browser and independent back-end
Programming skills	High	High	High	**Low**
Abstraction level	Code	Code	Code	**Models**
Reuse	Low	Low	Low	**High**
Productivity	Low	Low	Low	**High**
Management	High	**Low**	**Low**	**Low**
Client-side technology	HTML, CSS and JS frameworks	HTML, CSS and JS frameworks	HTML, CSS and JS frameworks	–
Server-side technology	Web frameworks (based on Java, Python, etc)	Web frameworks (based on Java, Python, etc)	Web frameworks (based on Java, Python, etc)	IFML

In the light of the mentioned strategies, it is noteworthy that these require augmentations are to be coded without any support. Moreover, the development tasks have limited support for design approaches, tools and framework. Next, we present an approach which combines design artifacts at the client- and server-side abstracting developers from most of the technical details related to the Web development.

6 Evaluation

This work promotes handling augmentations as a first-class requirement and proposes a strategy to design, implement and deploy the required augmentations. As the Web applications comprise several artifacts that are designed and implemented using different kinds of tools, the impact of introducing augmentations poses different challenges to the augmenter analyst to deal with diverse technologies and concepts. We realize that a Web augmentation has two main phases in its life cycle. First, it must be designed considering a set of requirements to meet, and then it is used by end users. So, we evaluate both, the approach experience and the augmentation usability and performance perceived by the end user. In the first place, we did a usability study of the toolset which is described in Sect. 6.1. This study is preliminary, and it is intended to collect usage metrics of the supporting toolset and discover what factors influence its use. In the second place, we evaluated the usefulness of the augmentations produced under our approach introduced in Sect. 6.2, to understand how useful the end users perceive them. Both experiences are presented in detail below.

6.1 A usability study

To assess the feasibility of our approach, we conducted a usability study of augmented applications with ten people. This preliminary study was carried out by people who daily use Web applications; they were trained and then asked to use the tools supporting our approach for creating an augmentation with back-end support.

The approach was applied to augment an e-commerce application to validate it and assess its benefits. We asked the participants to implement the example deeply described in Sect. 4 where a site was enriched with a stateful behavior, CSN requirements and sophisticated content integrations. The participants were asked to use our tool for capturing the main application concepts (Step 2) and implementing the back-end behavior with WebRatio (Step 3).

6.1.1 Subjects

The participants were computer scientists, partially or fully dedicated as teachers, and PhD students. Two of them also worked as freelancers. Almost all the participants had an Ecuadorian nationality; just one of them was Argentinean. Regarding their ages, we segmented them according to decades: two of the participants fell in the range of their 20s, six in their 30s and two in their 40s. One-third of them were females, and the remaining percentage was males.

Concerning their experience in the use of Web applications, only one of the participants started using the Web from 2 to 5 years ago; the remaining participants were users for more than 5 years. There was also a single case matching the participants without knowledge of Web extensions. Regarding the use of browser extensions—regardless of the browser they use—four users said they used them daily, another four said they did so only a few times a week. A single user said she used the extensions a few times a month, and another one said she never used extensions. Concerning their preferred browser, only one of them said he/she uses Firefox, while the remaining participants chose Google Chrome.

6.1.2 Training

The subjects were all software engineers and had experience browsing the Web. None of them had previous experience neither modeling Web applications nor modeling augmentations. So we trained the subjects in two days in sessions of 4 h each (8 h in total) on IFML and WebRatio tool. After the training, the subjects were able to model and run a Web application. On the third day, we also trained the subject on the augmentation concept and tools for 4 h where the subjects acquired the skills for identifying concepts in Web Sites, and designing the most common aspects of an augmentation. Finally, on the fourth day, they performed the experiment.

6.1.3 Setup

At the beginning of the experiment, we asked participants to fill out a demographic questionnaire. Then, we provided the participants with a copy of a virtual machine, in order to avoid differences in the environment in which the subjects worked on. The virtual machine was provided with Mozilla Firefox, our browser extension and the Community Edition [15] of WebRatio Web Platform.

We asked the participants to take note of the precise time in which they start and finish the experiment.

6.1.4 Protocol

The subject received a set of requirements describing the expected functionality introduced in Sect. 4 where the page presenting information about a given product is enhanced with the possibility of posting comments. The presentation used to document the augmentation spec is provided in [6]. The process started with the participants being asked to annotate

the DOM with our WOA tool. We expected them to improve a Web site that sells seeds for agriculture proposes [12]

As the existing Web application lacks end-user community features—like collaborative patterns that let them share their experience with a given product—they were asked to enhance the view of the products on sale in the site with a new section with comments.

To do so, they had to annotate a "product" and then use WebRatio to model the extra properties and components required by the augmentation application to support the business logic and the persistence at the back-end.

The study was executed in the same way with all the subjects and, lastly, we asked them to fill a final questionnaire which included the system usability scale (SUS) questions [46] among others. The SUS is a reliable tool for measuring the usability. It is based on ten questions that can effectively differentiate usable systems. The surveys used in the experiment are outlined in Appendix 1.

6.1.5 Results

The average points of the SUS questionnaires were 73.25, exceeding the general average of the system usability scale (68 points). Regarding the times, the full process took them 35 min, with a standard deviation of 12 min.

Besides, we asked the participants the extra questions—presented below—to get extra feedback. These questions were optional for the participants.

- **Do you consider it difficult to use the tool? Why?** The first question was closely related to the intention of the SUS, but we wanted to explore the reason behind their responses for understanding the "why." Six users replied using different metaphors that there was no issue with the approach and the supporting tool; one of them said that "the tools might be improved," and the last one said "The tool is friendly. I would like to have improvements in the time for the compilation or generation in WebRatio. The app running in the Browser is somewhat slow. I liked the experience of enhancing the Web application." This last comment is related to the time required by WebRatio to perform the cloud deployment, which often took time.
- **What additional use you find for the approach?** We also asked about other uses of our approach, and we got answers from four users. One of them mentioned "small prototypes to reinforce functional requirements with users for a company." The second one mentioned integrating applications implemented with different technology, and the third one was interested in augmenting the Web with graphics or videos. The remaining user mentioned he/she would like to create robots for getting user feedback on different Web pages.
- **Did you detect any problem in the use of the tools? Which one(s)?** This question was related to problems in the use of the tools. A participant replied that "It freezes a lot and it takes time to load when exploring." Another one complained about the "little support," but he/she did not mention in which moment of the augmentation creation process he/she would like to have it. The last user mentioned "modeling is important" as a problem.
- **Which aspects would you like to improve?** Two participants asked to improve the generation time of the site because it took considerable time to build the application, and the other two participants asked integrating more controls during the modeling process.

The usability evaluation showed that the users were satisfied with the experience. Moreover, the qualitative analysis reported opportunities to improve the toolset in the future.

6.1.6 Threats to validity

We have conducted a usability evaluation of our approach using as a case study an augmentation requirement in an e-commerce site. Next, we present the threats to validity faced during the usability evaluation according to the one proposed by Wohlin [47].

Conclusion Validity

We have sampled ten subjects that have been reported as a representative sample size [48] for presenting preliminary results which tackle *Low statistical power*. The low statistical power is a threat to the ability to reveal a true pattern in the data. Moreover, an experiment is necessary for comparing the approach with, for example, the ad hoc development of augmenters. The experiment may lead to stronger conclusions and should be supported with statistical tests.

To avoid *Fishing* threat and only focus on the expected results, we defined a protocol for the evaluation which was strictly followed by subjects and researchers who conducted the activity. *Reliability of treatment implementation* and *Random irrelevancies in experimental setting* were prevented by performing the evaluation at the same time, in the same conditions, and at the same place. The context was controlled without disturbs and distractions.

The *Random heterogeneity of subjects* is a threat when subjects are so heterogeneous that not comparison could be made. During the evaluation, the subjects were software developers with an identical educational background. However, the industry experience was heterogeneous.

Internal Validity

As we conducted an evaluation, some internal validity threats were not considered like the *Histrory*, *Maturation* and *Testing* threats because the candidates are not being

[12] http://www.arg-agro.com.ar/.

affected on such facts. In contrast, it is quite difficult to avoid *Instrumentation* threat. During the production of assessment material, we performed several internal tests like the smoke test with the subject to detect and adjust errors (these samples were discarded). Regarding the *Mortality* threat, no subject dropped out.

Construct Validity

The experiment may suffer *Mono-operation bias* threat as we used only one problem that was attempted to be resolved when applying the approach.

The experiment may suffer *Mono-method bias* threat as we only used the SUS as a main measure. To avoid *Interaction of testing and treatment* threat, the researcher who observed the experiment validated each measurement.

External validity

All subject had the same educational background and experience using MDWE before the experiment, but they are not a complete representation of all final users to which this approach is thought. So this may lead to *Interaction of selection and treatment* threat. Regarding *Interaction of setting and treatment*, we used WebRatio which is a mature actual platform with more than 20 years of usage in the industry and the latest version of Firefox Web browser. Moreover, the use case was a real application which was available online.

6.2 A controlled experiment with the resultant augmentations

In this section, we present the results of an experiment comparing the results of end users performing a task in multiple Web sites by using our approach and conventional navigation.

To set up the experiment, we applied our approach to augment the e-commerce application used as a running example, to validate our approach and assess its benefits. The site was enriched with stateful behavior and sophisticated content integrations.

We previously applied our approach by using our tool for capturing the main application concepts (Step 2) and implementing the back-end behavior with WebRatio (Step 3). Now, we performed a high-level impact analysis of the main advantages and challenges of different alternatives for implementing Web augmentations that comprise client- and server-side features.

The goal is to analyze Web augmentations that rely on server-side features to measure how the usability is improved in contrast to the same task performed without the augmentation. Researchers conducted the study, and the target subjects were end users who daily use Web applications. The study is preliminary, and it is intended to assess the usability and performance of the generated augmentations under our approach. Therefore, we did not consider evaluating the user experience related to another approach usage.

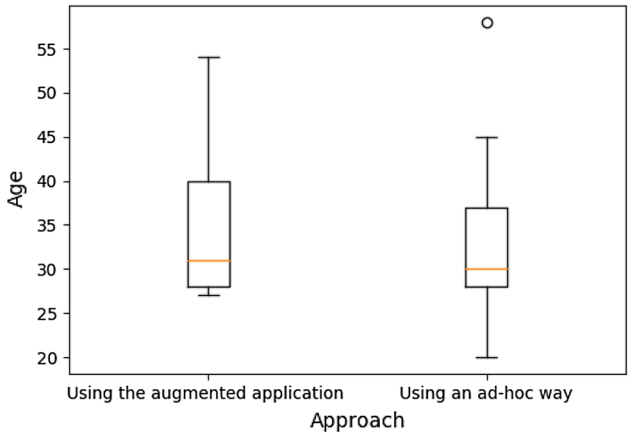


Fig. 12 Age of subjects

The reader must note that the development process which combines WOA and WebRatio is not studied in this section.

6.2.1 Subjects

The participants were 20 end users that include professionals, scientists, bookkeepers and lawyers among others. The age of participants is shown in a box plot in Fig. 12. Regarding the education, one got a secondary degree, six of them are undergraduate, and the remaining got a bachelor's degree in their respective professions.

6.2.2 Materials

The experimental units used during the experiment comprise the Web sites with information about seeds, pesticide and fertilizer. The used augmentation enhances the mentioned sites with a feature that helps users in planning vegetable production. The e-commerce site are: ArgAgro,[13] Dinkos,[14] Easy[15] and AgroPoints.[16] Additionally, we asked them to fill out two questionnaires: a demographic questionnaire and a SUS questionnaire.

We packaged the Web browser with the WOA plugin extension in a virtual machine based on the Oracle Virtual Box product using a new image of Microsoft Windows 10. The use of virtual machines allowed us to ensure there is no difference in the environment in which the subjects work. The virtual machine was provided with a Mozilla Firefox, the browser extension and any augmentation script required for the experiment.

[13] http://www.arg-agro.com.ar/, last accessed Feb 12th 2018.

[14] http://www.semillas-organicas.com.ar/, last accessed Feb 12th 2018.

[15] https://www.easy.com.ar/webapp/wcs/stores/servlet/es/easyar, last accessed Feb 12th 2018.

[16] http://www.agropoints.com/, last accessed Feb 12th 2018.

The virtual environment was hosted on our private servers, and the Windows sessions were accessed remotely using TeamViewer.[17] The user sessions were recorded to perform an in-depth analysis of the application's user experience. The application was running on Microsoft Windows 10, and we screen-recorded their sessions with a default tool provided by such operating system. We extracted the time consumed by each participant from the recorded sessions. To give support the server-side features, we used WebRatio Web Platform Community Edition [15]. As mentioned in Sect. 4.2, WebRatio was used to model the application that allows for the back-end support to manage the business logic and persist the information.

6.2.3 Protocol

The experiment protocol was executed in the same way for all the subjects. The subjects were divided into two groups: one using one approach (solving the tasks with the support of the augmentation generated with our toolset), and another one without any augmentation (an ad hoc way).

First of all, the participants filled out a demographic questionnaire. Then, they were asked to perform the study tasks. We divided the Web sites into bookmark folders, according to the tasks the users should perform (by the different product types that the user should pick): seeds, fertilizers and pesticides. The ArgAgro and Dinkos sites were provided for choosing the seeds, AgroPoints for fertilizers and Easy for pesticides. The tasks were simple requirements emulating being a farmer who plans his planting; for this, they need to browse the sites and select one tomato seed, one fertilizer and one pesticide using their criteria:

In ArgAgro's or Dinko's site:

- Task 1: Locate and choose one tomato seed;
- Task 2: Browse the application and add the product to your production planning

In AgroPoint's site:

- Task 1: Locate and choose one fertilizer;
- Task 2: Browse the application and add the product;
- Task 3: Check if the selected tomato seed and fertilizer are available in your report;

In Easy's site:

- Task 1: Locate and choose one pesticide;
- Task 2: Browse the application and add the product;
- Task 3: Check if all your products are available in your report;

[17] https://www.teamviewer.com/en/, last accessed Feb 20th 2018.

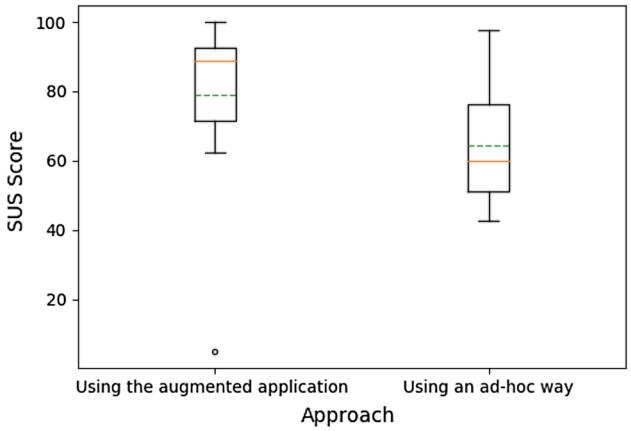


Fig. 13 SUS scores

Finally, we asked them some questions about their perception of the usability of the generated augmentations. They were provided with an online SUS questionnaire [46].

6.2.4 Results, analysis and implication

The evaluation of the SUS questionnaires [46] corresponding to the planning task is shown using a box plot in Fig. 13. The SUS scores denoted that there is an improvement in user satisfaction when using the plugin over the same task but performed in an ad hoc way. Because the samples' distribution is not parametric, the Mann–Whitney [49] test has as the outcome a p value 0.0113 which is lower than the alpha level 0.05, and the effect size is 0.565 which means high meaningfulness. On the other hand, Fig. 14 shows a box plot graph reporting the time spent performing vegetable production planning. When studying the measures using the Mann–Whitney test, the p value 0.05 (which is equal to the alpha) supports that there is a better performance of those end users using the approach.

The results of this work present preliminary evidence that augmentation with server-side support introduces a statistically significant improvement. However, a throughout study is carried out comprising either a controlled experiment evaluating the development approach with a broader set of a subject and power analysis and a complete user experience study of the augmentations developed using the approach.

6.2.5 Threats to validity

As in the first evaluation, there are several threats to validity that were considered during the experiment design.

Conclusion validity This kind of threat conditions the drawing of the correct conclusion.

To avoid *hypothesis fishing*, the search of a specific result, both objective metrics and a strict protocol, was stated during the experiment planning. The *low statistical power*, the

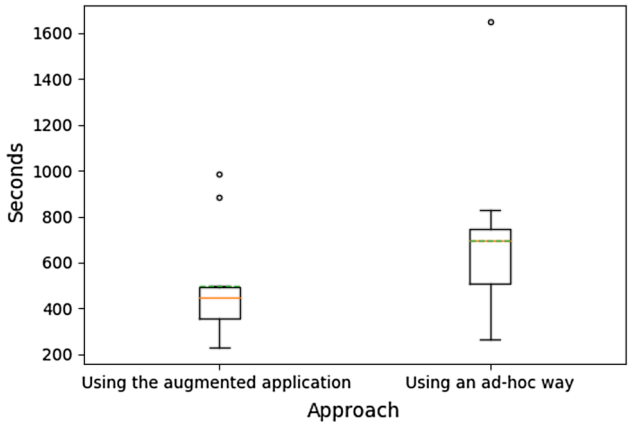


Fig. 14 Time spent performing the task

risk of a wrong conclusion, has been addressed with a high effect size (r = 0,565) accordingly to Pearson's correlation. However, a broader study needs to be performed with a more extensive set of samples that help to dilute the experiment's irrelevances. Due to the conclusion drawing highly depends on the quality of the measurements, and its instrumentation, the *reliability of measures* and *reliability of treatment implementation* treats are not considered as a threat because we used SUS tool which has proofed to be effective. As its name describes, *heterogeneity of subjects* threat is minimal as the subjects share the same profile having a similar academic degree, experience and age.

Construct validity This sort of threat focuses on aspects related to social factors and the design of the experiment. The experiment was designed to measure how the use of augmented UIs with a client and server augmentation improves the usability of the application uses cases. There is no prior relationship between subjects and the augmentation, so there is no evident reason to experience an increase in subject sensitivity or receptive to the treatment, or *Interaction of testing and treatment*. *Inadequate preoperational explication of constructs* refers to the improper definition of experiment constructors. To reduce the experiment's complexity and bias introduction possibility, we defined the method (with or without augmentation) as the only independent variable. So the constructs map perfectly to a simple mean comparison testing rather than other complex experiment settings like many groups or repeated measures. Due to only one application has been evaluated, more tests using other applications could help to get a full picture of the approach benefits (*Mono-operation bias*). The SUS metric is computed using the result of ten questions which help to cross-check the responses. This reduces the chance of bias when only one measurement is available, or *Mono-method bias*. The spent time metric which measures the effort on performing the test was registered by the subject and a observer who took notes of the experiment events. So any divergence on the values was reviewed considering both values.

Internal validity The independent variables can be affected by factors without the researchers' awareness, or internal validity treats. The subjects were selected randomly mitigating the consequences of an arbitrary selection (*selection* threat). And all of the subjects participated in the experiment only once. This aims at reducing the likelihood of responding differently because the subjects know how the test is conducted (*testing* threat). The fact that the experiment only lasts 1 h approx. limits the *maturation* threat where the subject performance can change as long as time goes by (i.e., get boring). The SUS tool has been widely adopted and has proofed to be an appropriate tool for user experience testing. The provision of SUS questionnaire in a standardized Web-based form reduces the impact of the instrument in the test because of the selected tools' maturity (*instrumentation* threat). We also checked that all the users had basic knowledge of Web application usage. Additionally, the usage of a virtual machine for providing a desktop helps to ensure homogeneous instrumentation when the subjects perform the test. It allowed avoiding the use of heterogeneous operative systems or hardware among the different participants. However, there could be a threat in the case of poor connectivity which detriment the virtual machine usage experience. The candidates needed training before the evaluation. Thus, a potential threat regarding the testing is whether the candidate did not pay attention to the training before the evaluation. This could wrongly throw bad results on the results of adopting this approach.

External validity This sort of threats affects the generalization of the conclusions. The subjects were mixed-end users. A broader experiment considering subjects of different cultures who have worked on diverse business domains will improve the generality of our claims (*Interaction of selection and treatment*). Nevertheless, regarding academic education, the population was mixed since the subjects belonged to different educational levels. This provided us a wider perspective on the usability of the presented approach. As both the Web site on which the experiment was based and the Web form to gather information are wide adopted tools, nowadays, there is no reason to conclude that these make the subjects uncomfortable, or *Interaction of setting and treatment* threat.

7 Discussion

Our approach is supported by model-driven techniques promoting the solution design rather than "coding" software artifacts. It also simplifies the documentation of augmentations because the models are self-documented. The augmentation behavior is limited to the resources handled by the augmentation layer. That is, augmentation can only manage (instantiate

and modify) objects defined in the augmentation model, and it can only reuse the perceived behavior accessible from the page; it is not aware of other core underlying routines available when the core system is maintained. The business model is modeled in a new UML package which is isolated from the target application. Any class enhancement, such as variables and methods, are modeled in a target-class decorator which is afterward composed with the core concept by the Web browser extension.

From the navigational point of view, adding new nodes in the navigational layer does not introduce any core application modification because they are defined detachedly in its scope and later weaved by the Web browser extension. Therefore, our approach promotes the packaging of new nodes in such a way that they are easily plugged and unplugged.

Regarding the user interface layer, new structural features (such as widgets, layout configuration or complex compositions of UI elements) are specified and later weaved by a transformation engine. This technique avoids the development of complex UI code within the core application code (since the augmentation analyst does not have access and may not be able to code because he/she is not a programmer). The base user interface remains unchanged in the source code.

The MDWE approaches allow engineers abstracting low code issues which focus all the development effort on the design of the application. Mature MDWE approaches [31] provide the transformation definitions and toolset for generating the Web application source code that meets the models. In this way, our approach is not tied to IFML because approaches such as UWE or OOHDM can be used to design models and generate the Web application. The outcome is similar to the application generated with WebRatio. Note that it is similar in the client-side counterpart in our approach. In this paper, we have adapted an existing tool (WOA) that is based on a specific metamodel. In base of this metamodel, an augmenter artifact is created that runs on top of a Web browser. Nevertheless, other tools fit perfectly with our approach and they can be used to generate augmenter artifacts, such as Sticklet, or directly generate more basic JavaScript code to be run in weaving engines such as Greasemonkey.

During this process some requirements should be considered, for example, the interaction between the user interface widget and the server component which relies in a different host with a different domain, can bring the problem of cross-origin resource sharing (CORS). One way to prevent this problem is to add a CORS proxy for the UI calls (such as crossorigin.me[18]) which allows to access resources from others domains preventing that the client Web block any call. The management of this kind of Web augmentation is pretty straightforward because with our approach, the core behavior is not modified and, thus, the regression testing is not needed. When the user does not need to use the functionality provided by the augmentation anymore, he/she can just deactivate it in the Web browser.

Finally, the productivity should be better than code-based approaches supported on the claimed productivity of WebRatio of 900%, presented in [6], which even though not being an academic paper (there are no academic performance evaluations yet on IFML) is a result based on a large number of projects.

Another interesting discussion is about the stakeholders around Web augmentation artifacts. We share the philosophy of the existing literature that differentiates roles among stakeholders of Web augmentation software. We believe that our MDWA approach offers a first step toward the definition of server-side counterparts by end users. However, the expressivity used for defining these back-end applications is very rich, since it is based on IFML, a very powerful language for specifying full-stack Web applications. For simple augmentations requiring some simple information sharing among users of the same augmenter, straightforward models are required which can be modeled by an end user. Nevertheless, our approach could be used in scenarios where the augmentation requires complex behavior and then a more expert stakeholder appear. Our approach could be applied in situations where new functional requirements that require complex server-side counterparts could be tackled without modifying the application to test them.

8 Conclusions

In this work, we have presented a novel approach for designing Web augmentation coping with client-side and server-side behaviors. The augmentations are modeled by users, who do not belong to the application development team, using IFML, but our approach can be instantiated with other approaches [31] such as UWE or OOHDM. The approach relies on the separation of concerns principles; thus, we provide the composition mechanism for each model (conceptual, navigational and user interface).

To depict the possibilities of our approach, some running examples were described supporting different user's needs and complexities in an agriculture e-commerce site (i.e., the design of a community-review feature and a concern-sensitive navigation for planning sowing). The implementation of these examples was aimed at improving decision-making activities. Then, we presented a discussion comparing three approaches to provide some insights about the advantages and disadvantages of producing augmentations: the ad hoc implementation (evolutionary requirements), the code-based development at both, client- and server-side and our model-driven approach. The evaluation was addressed in

[18] https://crossorigin.me.

two experiments. The first one assessed the usability of the toolset with ten analysts, and it allowed to validate that the toolset can be used to produce augmentations. In the second place, we evaluated the usability of the augmentation produced with our approach. Both evaluations report the benefits of our approach in terms of user satisfaction and the performance of the subjects. We also study how to improve the reuse of the produced augmentations. Finally, we study the use of model-driven Web augmentation in diverse business domains (such as financial, banking or logistic). We are planning to continue the evaluation studies to support stronger claims as well as studying some correlation influences between the subject profile and the resultant augmentation quality.

Acknowledgements Authors of this publication acknowledge the contribution of the Project 691249, RUC-APS: Enhancing and implementing Knowledge based ICT solutions within high Risk and Uncertain Conditions for Agriculture Production Systems (www.ruc-aps.eu), funded by the European Union under their funding scheme H2020-MSCA-RISE-2015.

Surveys used in the experiments

The next surveys were used on both experiments. Only a subset of question was used only in the toolset evaluation for assessing the perception about the approach.

Initial survey

This survey aims at gathering information about subject user.

- Nationality
- Age
- Gender
- Current level of education:
- My training focuses mainly on the field of:
- Occupation/job position:
- How often do you use the Web?
- How many years have you been using the Web?
- How often do you use a Web browser in your daily activities?
- Do you know about browser extensions?
- How often do you use extensions for Web browsers in general?
- What *desktop* Web browser do you use frequently?

Final survey

This survey is twofold: system usability scale questions and additional inquiring questions:

- SUS
  - I think that I would like to use this system frequently.
  - I found the system unnecessarily complex.
  - I thought the system was easy to use.
  - I think that I would need the support of a technical person to be able to use this system.
  - I found the various functions in this system were well-integrated.
  - I thought there was too much inconsistency in this system.
  - I would imagine that most people would learn to use this system very quickly.
  - I found the system very cumbersome to use.
  - I felt very confident using the system.
  - I needed to learn a lot of things before I could get going with this system.
- Additional questions (only for the toolset usability evaluation)
  - Start time of the experiment
  - End time of the experiment
  - Do you think the tool was hard to use? Why?
  - What other uses of the tool do you come up with?
  - Did you detect any problem in the use of the tool? Which one(s)?
  - Do you think there are aspects of the tool that can be improved? Which ones?
  - Do you have any other comments about the tool, the approach or the experiment?

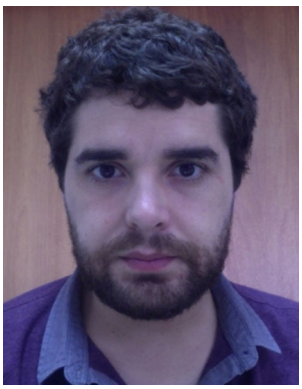
References

1. Garrido, A., Firmenich, S., Rossi, G., Grigera, J., Medina-Medina, N., Harari, I.: Personalized web accessibility using client-side refactoring. IEEE Internet Comput. **17**(4), 58 (2013)
2. Díaz, O., Arellano, C., Iturrioz, J.: In: Web Engineering, 10th International Conference, ICWE 2010, Vienna, Austria, July 5–9, 2010. Proceedings, pp. 233–247 (2010)
3. Firmenich, S., Bosetti, G., Rossi, G., Winckler, M.: In: Current Trends in Web Engineering - ICWE: International Workshops, DUI, TELERISE, SoWeMine, and Liquid Web, Lugano, Switzerland, 6–9 June 2016. Revised Selected Papers, pp. 200–207 (2016)
4. Wischenbart, M., Firmenich, S., Rossi, G., Wimmer, M.: In: Proceedings of the Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, IntRS 2015, co-located with ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, 19 Sept 2015, pp. 53–60 (2015)
5. Diigo (2017). https://www.diigo.com/
6. DDway: Calculation of the functional size and productivity with the IFPUG method (CPM 4.3.1). The DDway experience with WebRatio (2016). http://www.webratio.com/website/documentation/Case_Study_Productivity_with_WebRatio.pdf
7. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice, vol. 1. Morgan & Claypool Publishers, Los Altos (2012)
8. Panach Navarrete, J.I., Dieste, O., Marin, B., Espana, S., Vegas, S., Pastor, O., Juristo, N.: Evaluating model-driven development claims with respect to quality: a family of experiments. In: IEEE

Transactions on Software Engineering (2018). https://doi.org/10.1109/TSE.2018.2884706
9. Martínez, Y., Cachero, C., Meliá, S.: MDD vs. traditional software development: a practitioner's subjective perspective. Inf. Softw. Technol. **55**(2), 189 (2013)
10. Urbieta, M., Frajberg, D., Rossi, G.: Assessing the impact of volatile functionality removal in web applications: model-driven vs code-based approaches. Softw. Pract. Exp. (2017). https://doi.org/10.1002/spe.2503
11. Martínez, Y., Cachero, C., Meliá, S.: Empirical study on the maintainability of web applications: model-driven engineering vs code-centric. Empir. Softw. Eng. **19**(6), 1887 (2013)
12. Panach, J.I., España, S., Dieste, S., Pastor, S., Juristo, N.: Assessing data analysis performance in research contexts: an experiment on accuracy, efficiency, productivity and researchers' satisfaction. Inf. Softw. Technol. **62**, 164 (2015)
13. Tarr, P., Ossher, H., Harrison, W., Sutton, S.M.: In: Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No.99CB37002), pp. 107–119 (1999)
14. Firmenich, S., Bosetti, G.A., Rossi, G., Winckler, M., Barbieri, T.: In: Web Engineering—16th International Conference, ICWE 2016, Lugano, Switzerland, 6–9 June 2016. Proceedings, pp. 77–95 (2016)
15. WebRatio. WebRatio Platform (2017). http://www.webratio.com/site/content/en/web-application-development
16. Urbieta, M., Firmenich, S., Maglione, P., Rossi, G., Olivero, M.A.: In APMDWE. INSTICC, ScitePress (2017)
17. Díaz, O., Arellano, C.: The augmented web: rationales, opportunities, and challenges on browser-side transcoding. TWEB **9**(2), 8 (2015)
18. Herlocker, J.L., Konstan, J.A., Riedl, J.: In Proceedings of the 2000 ACM conference on Computer Supported Cooperative Work—CSCW '00, pp. 241–250 (2000)
19. Firmenich, D., Firmenich, S., Rivero, J.M., Antonelli, L., Rossi, G.: CrowdMock: an approach for defining and evolving web augmentation requirements. Requirements Eng. (2016). https://doi.org/10.1007/s00766-016-0257-3
20. Díaz, O., Sosa, J.D., Trujillo, S.: In: 24th ACM Conference on Hypertext and Social Media (part of ECRC), HT '13, Paris, France—02–04 May, 2013, pp. 69–78 (2013)
21. Firmenich, S., Garrigós, I., Wimmer, M.: In: Web Engineering, 14th International Conference, ICWE 2014, Toulouse, France, 1–4 July 2014. Proceedings, pp. 359–369 (2014)
22. Ceri, S., Dolog, P., Matera, M., Nejdl, W.: In: Koch, N., Fraternali, P., Wirsing, M. (eds.) Web Engineering: 4th International Conference, ICWE 2004, Munich, Germany, 26–30 July 2004. Proceedings, pp. 201–214. Springer, Berlin (2004)
23. Díaz, O., Arellano, C., Azanza, M.: A language for end-user web augmentation: caring for producers and consumers alike. ACM Trans. Web **7**(2), 9:1 (2013)
24. 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 (2014)
25. Aldalur, I., Winckler, M., Díaz, O., Palanque, P.: In: Paternò, F., Wulf, V. (eds.) New Perspectives in End-User Development, pp. 433–459. Springer International Publishing, Cham (2017)
26. Díaz, O., Aldalur, I., Arellano, C., Medina, H., Firmenich, S.: Web mashups with webmakeup. In: Communications in Computer and Information Science (2016). https://doi.org/10.1007/978-3-319-28727-0_6
27. Firmenich, D., Firmenich, S., Rossi, G., Winckler, M., Distante, D.: User interface adaptation using web augmentation techniques: towards a negotiated approach. In: Cimiano, P., Frasincar, F., Houben, G.J., Schwabe, D. (eds.) Engineering the Web in the Big Data Era, pp. 147–164. Springer International Publishing, Cham (2015)
28. Aldalur, I., Diaz, O.: In: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems (ACM, New York, NY, USA), EICS '17, pp. 45–50 (2017)
29. Bosetti, G., Firmenich, S., Gordillo, S.E., Rossi, G., Winckler, M.: An end user development approach for mobile web augmentation. Mob. Inf. Syst. (2017). https://doi.org/10.1155/2017/2525367
30. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc, Boston, MA (1995)
31. Rossi, G., Pastor, S., Schwabe, D., Olsina, L.: Web Engineering: Modelling and Implementing Web Applications, vol. 12. Springer, London (2008)
32. Aragón, G., Escalona, M.J., Lang, M., Hilera, J.R.: An analysis of model-driven web engineering methodologies. In: International Journal of Innovative Computing, Information and Control, vol 9, pp 413–436 (2013)
33. Domínguez-Mayo, F.J., Escalona, M.J., Mejías, M., Ross, M., Staples, G.: Towards a homogeneous characterization of the model-driven web development methodologies. J. Web Eng. **13**(1&2), 129 (2014)
34. Object Management Group. http://www.omg.org
35. Interaction flow modeling language. http://www.omg.org/spec/IFML/
36. Lowe, D., Henderson-Sellers, B., Gu, A.: In: International Conference on Conceptual Modeling, pp. 105–119. Springer (2002)
37. Brambilla, M., Fraternali, P.: Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML. Morgan Kaufmann, Burlington, MA (2014)
38. Popovici, A., Gross, T., Alonso, G.: In: Proceedings of the 1st International Conference on Aspect-Oriented Software Development, AOSD '02, pp. 141–147. ACM, New York, NY (2002)
39. Rossi, G., Nieto, A., Mengoni, L., Lofeudo, N., Silva, L.N., Distante, D.: In: 2006 Fourth Latin American Web Congress, pp. 179–188 (2006)
40. Ginzburg, J., Distante, D., Rossi, G., Urbieta, M.: Oblivious integration of volatile functionality in web application interfaces. J. Web Eng. **8**(1), 25 (2009)
41. Firmenich, S., Rossi, G., Urbieta, M., Gordillo, S., Challiol, C., Nanard, J., Nanard, M., Araujo, J.: Engineering concern-sensitive navigation structures, concepts, tools and examples. J. Web Eng. **9**(2), 157 (2010)
42. Mikkonen, T., Systä, K., Pautasso, C.: In: Cimiano, P., Frasincar, F., Houben, G.J., Schwabe, D. (eds.) Engineering the Web in the Big Data Era: 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, 23–26 June 2015, Proceedings, pp. 134–143. Springer International Publishing, Cham (2015)
43. Ginzburg, J., Rossi, G., Urbieta, M., Distante, D.: In: Baresi, L., Fraternali, P., Houben, G.J. (eds.) Web Engineering: 7th International Conference, ICWE 2007 Como, Italy, 16–20 July 2007 Proceedings, pp. 152–166. Springer, Berlin (2007)
44. Urbieta, M., Rossi, G., Distante, D., Ginzburg, J.: Modeling, deploying, and controlling volatile functionalities in web applications. Int. J. Softw. Eng. Knowl. Eng. **22**, 129 (2012)
45. Frajberg, D., Urbieta, M., Rossi, G., Schwinger, W.: In: Bozzon, A., Cudre-Maroux, P., Pautasso, C. (eds.) Web Engineering: 16th International Conference, ICWE 2016, Lugano, Switzerland, 6–9 June 2016. Proceedings, pp. 59–76. Springer International Publishing, Cham (2016)
46. Brooke, J.: SUS - A quick and dirty usability scale. Usability Eval. Ind. **189**(194), 4–7 (1996). https://doi.org/10.1002/hbm.20701
47. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction, vol. 15. Kluwer Academic Publishers, Norwell (2000)
48. Hwang, W., Salvendy, G.: Number of people required for usability evaluation: the 10±2 rule. Commun. ACM **53**(5), 130 (2010)

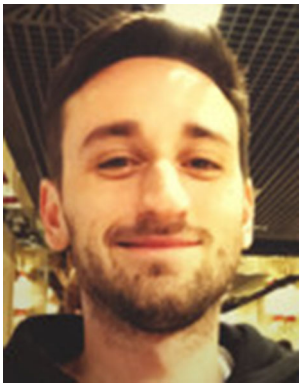
49. Juristo, N., Moreno, A.M.: Basics of Software Engineering Experimentation, 1st edn. Springer Publishing Company, Berlin (2010)

Matias Urbieta was born in Formosa, Argentina, in 1981. He received this BS and PhD in Computer Science from the National University of La Plata (UNLP). Since 2008, he is a Research Assistant at the LIFIA Institute, La Plata, Buenos Aires, Argentina. He is the author of several conference and journal articles. His research interests include separation of concerns in Web Applications, Agile requirement engineering and Web Augmentation. He teaches Object-Oriented Programming concepts and Web engineering at bachelor, and doctoral levels at UNLP and Universidad Abierta Interamericana.



Sergio Firmenich obtained a CONICET grant in 2009 with which he could finish his PhD on Computer Science, in 2013. Since 2015, he is a researcher at CONICET and LIFIA. He has taught at universities for over 14 years, and currently, he is professor in different subjects at the Facultad de Informática, Universidad Nacional de La Plata.

Gabriela Bosetti is a Ph.D. in Computer Science graduated from the UNLP, Argentina. Her research interests are Web Augmentation, End-User Development, Web Engineering, User Experience, Mobile Web and Machine Learning.



Pedro Maglione got his bachelors degree from the UNLP, Argentina. He Olivero is PhD candidate in Computer Science at the University of Trento, Italy.

Gustavo Rossi received his PhD diploma in PUC-Rio, Brazil, in 1996. His PhD thesis was the development of the OOHDM design approach, one of the mature methods for the development of Web Applications. He is currently full professor at Facultad de Informática, Universidad Nacional de La Plata. He has been a visiting professor at the Universities of Lyon and Montpellier in France and has received the Habilitation pour Diriger Recherches (HDR) at INSA-Lyon. He has been part of the PC committee of the most important conferences of his research field such as ACM WWW, ICWE, and ACM Hypertext. He is member of the editorial board of IEEE Internet Computing, IEEE IT Professional, WWW Journal, Journal of Web Engineering, Journal of Internet Services and Applications and the International Journal of Cooperative Information Systems.

Miguel Angel Olivero is PhD candidate in Computer Science at the University of Seville. He has participated in various projects as a researcher as member in the Web Engineering and Early Testing Group (IWT2). He has been part of the organizing committee of different international conferences. He has made national and international stays and participated in both national and international projects. His current research interests are related to Model-Driven Engineering, Security and the System of Systems context. Further information about his research activities and his list of publications can be found at https://investigacion.us.es/sisius/sis_showpub.php?idpers=25279.