Web Applications Design Recovery and Evolution with RE-UWA

Mario Luca Bernardi¹, Marta Cimitile², Damiano Distante³

¹RCOST - Department of Engineering, University of Sannio, Italy
²,³Unitelma Sapienza University, Italy
¹mlbernar@unisannio.it, ²marta.cimitile@unitelma.it, ³damiano.distante@unitelma.it

SUMMARY

This paper presents a semi-automatic approach for the recovery and evolution of the design of existing Web applications. The proposed approach, structured in two main phases, is based on the Ubiquitous Web Applications (UWA) design framework, a methodology based on a set of models and tools, for the user-centered design of multi-channels context-aware Web applications. In the first phase a representative set of the application’s front-end Web pages are analyzed in order to abstract the “as-is” design model of the application according to the UWA methodology. In the second phase, the recovered design model is evolved to define the “to be” version of it. This evolution activity considers the up-to-date requirements available for the application and UWA design guidelines to identify shortcomings and opportunities of improvements in the “as-is” design. The reverse modeling phase exploits clustering and clone detection techniques and is supported by the RE-UWA tool platform, an Eclipse IDE customized to implement the reverse engineering process defined to extract formal UWA models expressed as instances of a MOF metamodel. The forward design phase is supported by a set of UWA modeling tools which are built on top of the Eclipse Modeling Framework (EMF) and the Eclipse Graphical Modeling Framework (GMF). The proposed design recovery and evolution approach is applied to four real-world Web applications and the obtained results are also presented in the paper. Copyright © 2012 John Wiley & Sons, Ltd.

KEY WORDS: Web applications design recovery, Web systems evolution, UWA, RE-UWA, clustering, clone detection, Eclipse, GEF, GMF

1. INTRODUCTION

Web applications are rapidly becoming the underlying engine of many e-businesses, including on-line shopping, on-line banking, e-government, and services such as news and data provision. As Web applications become more widespread, sophisticated and complex, their maintenance and improvement becomes a more difficult, critical and costly task. In fact as a consequence of the necessity of adding new features and due to the continuous change of their requirements, Web applications are subject to continuous maintenance and evolution. This often means that a considerable effort is necessary to maintain these applications [17]. In particular, the effort depends on the kind of improvements. Some modifications do not alter the existing functional behavior or business rules of the Web application, nor the way contents and services are presented to users, but just the operational or technological structure used for its execution. These modifications entail mainly the implementation layer, rather than the conceptual design models describing the domain objects and services. On the contrary, other modifications are not driven by new technology, but by the need/opportunity to improve some Web applications external qualities (such as their usability) and aspects (like content, navigation and presentation) [12]. Moreover, some maintenance
activities, like the modification of content types, the modification of access structures to content, the enhancement of the navigation structure with new navigation paths, and the improvement of the application user interface, have a great impact on the conceptual design of the application. Consequently they should be profitably analyzed and represented by means of suitable Web application design models.

The continuous Web application maintenance and evolution makes the availability of up-to-date instances of such models even more critical. Such models play a key role for a successful Web application evolution task in response to new requirements. Unfortunately, Web applications development is, contrary to other domains, not characterized by rigorous software engineering methods and has a short time-to-market. These aspects constrain the development and maintenance of Web applications, and together with the lack of documentation, make maintenance and evolution harder tasks, compromising the effectiveness and correctness of the whole system. Techniques and tools for recovering, editing and keeping up-to-date such models from the application code can be very useful in supporting both comprehension and evolution tasks.

This paper proposes a semi-automatic approach for Web applications design recovery and evolution which adopts the Ubiquitous Web Applications (UWA) design framework [24] as a reference modeling methodology. The approach is based on a two-phase process: (i) the recovery of the UWA models representing the “as-is” design; and (ii) the use of recovered models together with the available up-to-date requirements to identify needs and opportunities of design improvement. The models obtained at the end of the process describe the design “to-be” of the application and can be used by developers to introduce the required changes in the source code. The models produced may also be used as a starting point of a model-driven development process based on UWA, which would lead to a new implementation of the application [7].

This paper is an extension of the work presented in [3] and, compared to it, adds several new contributions: (i) a bigger case study discussing the application of the approach and involving four real-world Web applications (of different domains and size); (ii) a discussion of the results obtained in the case study; and (iii) a more detailed description of the tools supporting the entire design recovery and enhancement process.

The remainder of the paper is structured as follows: Section 2 synthesizes the overall design recovery and evolution process and provides some details on the underlying techniques and technologies for both the reverse and the forward design phases. Section 3 presents the set of tools to support the approach and discusses their features. Section 4 reports on a case study showing the application of the approach to four real world Web applications. Section 5 discusses the validity of the approach and its limits, while Section 6 synthesizes related work. Finally, Section 7 provides final remarks and outlines future work.

2. THE REDESIGN PROCESS

The proposed redesign process consists of two main phases: a reverse engineering phase, and a forward model-driven design phase. The first phase refers to the UWA Web applications design methodology and to a number of code analysis techniques (including clone detection and clustering techniques) to abstract the user-centered conceptual model of the application to redesign. The second phase takes as input the UWA model recovered by the first phase and enables refining and evolving it, e.g., to respond to new application requirements. Tool support is provided for both phases which are semi-automatic and require user intervention mostly to drive the different process activities and validate the produced results.

In the remainder of this section we first introduce the UWA methodology and, more specifically, its design models; then we describe the two phases in which the redesign process is articulated.

2.1. The UWA Web Design Methodology

The UWA design framework includes a complete design methodology and a set of models and modeling tools for the user-centered conceptual design of data and operation intensive ubiquitous (i.e., multi-channel, multi-user and generally context-aware) Web applications [24].
Similarly to other well-known web engineering methods proposed in the literature, such as OOHDM [20], WebML [5] and UWE [13], UWA specifies the design of a Web application by means of three main models: the Information Model (a.k.a., content or domain model), the Navigation Model, and the Publishing Model [23]. Additional models proposed by UWA include: the Transaction Model, which models the business processes the application is intended to support; the Operation Model, which is used to specify the elementary operations the application will provide to its users; the Customization Model, which specifies, by means of customization rules, how the application will adapt to different usage contexts.

The UWA Information Model comprises two sub-models: the Hyperbase Model and the Access Structures model. The Hyperbase Model describes the contents of the applications in terms of base information classes (Entities), their structure (Components and Slots) and their relationships (Semantic Associations). The Access Structures model defines subsets (Collections) of the application content, each based on a selection criterion derived from a specific information access user goal. The UWA Navigation Model assembles elementary information elements (slots from one or more entities, Association Centers and Collection Centers) into reusable units of consumption (Navigation Nodes) and defines navigation contexts (Navigation Clusters) by grouping nodes and defining navigation paths through them (using Navigation Links). Finally, the UWA Publishing Model specifies how the application is organized in terms of pages, which are the components of each page (Publishing Sections and Publishing Units), and which node is published in each publishing unit.

Each UWA model specifies a specific aspect (e.g., navigation) of the designing application from the perspective of the final user. As a consequence, the UWA model of a Web application describes how the application will be perceived by its users, rather than how it is implemented. Different models of the same kind (e.g., different navigation models) can be drawn to define how the application will appear and behave for different user types. In the above sense, UWA is a user-centered design methodology, and so are its models.
Entities, entity components, semantic associations, collections, navigation nodes, navigation clusters, pages, sections, publishing units and other UWA modeling primitives can be “typed” (the most common case), thus representing classes of objects, or “untyped”, thus representing singletons.

Examples of UWA models, including excerpts of the information, navigation and publishing models are reported in figures 4 and 7.

2.2. The Reverse Engineering Phase

Aiming at describing a Web application at a high level of abstraction and from a user-centered perspective, UWA models are independent of any technology chosen to implement the application. This characteristic makes UWA models suitable to represent potentially any Web application producing HTML pages as front-end. Moving from the above consideration, we have developed a semi-automatic reverse engineering approach which is able to abstract UWA models from existing Web applications [4][23]. We shortly synthesize this process and the underlying techniques while more details can be found in [4].

The reverse engineering process is made up of five steps which enable the recovery of the following UWA models:

- UWA Information model
- UWA Navigation model
- UWA Publishing model

Figure 1 shows the complete activity diagram showing all the steps in detail.

2.2.1. UWA Information Model Abstraction

The UWA information model of the considered application is recovered by analyzing its client side Web pages to abstract UWA entities, UWA semantic associations and UWA collections.

The identification of UWA entities is carried out by searching for groups of related attributes (we refer to these attributes as keywords) in the client-side HTML pages (static and dynamically generated) of the WA. A group of keywords involved in the same user input or output operation and included in the same HTML form or output report is considered as a possible group of slots characterizing a UWA entity. The rationale behind this assertion is that the set of data items that a user enters into an input form, or that are shown to a user by an output report, usually represents a concept of interest for the user in the domain of the application. Similar considerations apply to groups of keywords characterizing a set of cloned client pages, i.e., a group of client pages characterized by the same HTML control structure but different content. In this case keywords can be identified by considering labels associated to content items (such as text, images, multimedia objects, etc.), text appearing in table headings, titles appearing in page sections, etc. From each group of cloned client pages, a HTML page template is produced. This template has the same control component and the same set of keywords that are common to all the pages in the set of cloned pages. Each identified keyword is candidate to be a UWA slot and the keywords in a group are candidate to be an entity component. Edit distance metrics and clustering techniques have been defined to identify groups of cloned pages and extract from them associated groups of keywords. A validation phase is carried out manually with the keywords clusters already automatically identified to finally obtain a validated set of UWA entities associated to the considered application.

A candidate UWA semantic association is assumed to exist between pairs of entities having some slots in common. If different entities are shown in the same HTML page, a candidate association between them is also considered to exist. Semantic associations are also derived from hyperlinks connecting pages showing different entities mainly when one of their slots is used as an anchor for the hyperlink. Similarly to candidate entities, candidate associations automatically found in this step have to be validated by a human expert knowledgeable of the application domain.

The identification of UWA collections is based on the ways they are usually implemented in a WA. These include: (i) the usage of a table where each row reports a different instance of a given
entity or association; and (ii) a list of hyperlinks pointing to pages showing different instances of the same entity. As for entities and semantic associations, the automatically recovered collections will undergo a validation phase conducted by a human expert knowledgeable of the application domain.

2.2.2. UWA Navigation Model Abstraction The recovery of the UWA navigation model is carried out by identifying nodes and clusters for the analyzed application. Nodes are identified by associating them with structural sections in the pages of the application, displaying/requiring information to/from the user. The client pages related to entities, associations, and collections are selected and analyzed to: (i) identify which attributes of each of the entities, associations, or collections are referred in the page; (ii) associate a node to each group of attributes; and (iii) identify hyperlinks connecting nodes in the same page or in different pages. Links between nodes are used to identify navigation clusters. A list of nodes and their organization into clusters is the result of this step. Each node and each cluster is assigned a unique name derived from the elements of the information model they are associated with.

2.2.3. UWA Publishing Model Abstraction The UWA publishing model of the analyzed application is abstracted by identifying publishing pages, publishing sections and publishing units (PU) from the set of templates obtained during the phase of information model recovery. A page is associated to each template contributing to the identification of at least one entity. To identify sections and PUs associated to each page we assume that a section includes only one PU and associate a PU to each of the nodes recovered in the navigation model. By tracing the association between nodes and templates it is possible to associate PU to pages.

2.3. The Forward Design Phase

The UWA model recovered in the reverse engineering phase provides a representation of the design “as-is” of the analyzed application. By examining this model and by considering the (eventually new) requirements available for the application, as well as the design guidelines provided by the UWA design methodology, the analyst can:

- identify weaknesses of the current design;
- define changes in order to overcome them;
- evolve the design to meet new or changed requirements.

An example of such analysis and of the evolution interventions that can be applied is reported in Section 4.

The data gathered during the reverse engineering phase by the RE-UWA tool platform are exported using the RE-UWA Tool as well-formed instances of the UWA MOF metamodel. The UWA model obtained in this way is then imported in the UWA graphical editor described in Section 3. Figure 7 and 4 show excerpts of the UWA model recovered for the Exibart.com web site considered in our case study.

By means of the UWA modeling tools, we can introduce changes in the recovered model to overcome weaknesses identified in the “as-is” design of the application and we can evolve it to better meet new or changed requirements.

The new model can be used as a reference to implement changes in the application. As an additional option, the new model can be used as a starting point to re-implement the application by applying the UWA model-driven Web application development approach detailed in [7].

3. TOOL SUPPORT

3.1. The RE-UWA Tool Platform

RE-UWA Tool Platform (RTP) is described in Figure 2 (a). As shown by the figure, the lowest level of the architecture is composed by the RTP Core layer providing project integration and supporting
builders in the comprehension of UWA resources and a project nature enabling RE-UWA process workflow for Eclipse WTP projects. Some basic services for the entire RTP are located in this layer: HTML/XML parsers, along with similarity distance calculators between HTML documents and core platform services.

At this level are also located the platform services. They permit to import the pages of the Web application into a UWA project. The import phase extracts structural information about: (i) the downloaded client pages; (ii) the inner components of each page (e.g. forms, scripts module, frame, applet, etc.); and (iii) the hyperlinks connecting the pages. This information is stored into a repository located in the analyzed project.

The static analysis on the HTML client Web pages of the application and the identification of pages that are clones is performed by a clone detector module. The clone detector component traverses HTML/XML DOMs to generate distance matrices for the HTML pages under analysis. This component can be configured by independent modules that gain access to the DOMs of Web application pages to calculate the distance matrix with several distance algorithms. Currently, two distances are supported: the edit distance and a maximum sub-tree matching distance. The data extracted are made accessible to the entire RTP environment.

Returning to the figure, at the second level, the RTP Process layer is summarized. This layer implements the process logic through a workflow engine that follows the RE-UWA process specification. A component implements each step of the process there. The engine takes the process instance and transfers the control between the steps as specified in the process definition.

The process can be structured as a direct graph with several kind of nodes and edges. Nodes can be distinguished in simple nodes or composite ones. Simple nodes can be process (executing recovering process logic) or predicate nodes (to structure the control and data flow). Compared to simple ones, composite nodes have an inner structure and can be of several types depending on the policy of execution (i.e., all nodes must be executed; only one must be executed; any of inner nodes can be executed). For edges we can distinguish different types according to the needs for interaction on the transition and on the routing policies (auto or manual routing). The software components instantiate the framework the framework by inheritance and composition: they can be added, removed or modified in flexible ways. Each process has a customizable configuration phase.
Figure 3. A Screenshot of the RE-UWA tool platform taken while analyzing the VisitTrentino.it Web site where control and data flow dependencies, among the steps involved in the process, can be specified. In the highest level of the figure there is the presentation layer that allows the interaction with users driving process execution. It is organized as a set of Eclipse editors and views interacting with the engine and the concrete components. It supports the analyst to execute the step logic providing the needed and related information useful for her/his choices. The RTP IDE layer introduces several perspectives each one related to the recovering of a well defined portion of the UWA model:

- **Forms and Clones Perspective** It contains all the views related to HTML page clustering, templates generation and group of keywords extraction and validation. It also includes some editors defined to handle the recovered elements.
- **Entities and Associations Perspective** It contains all the views related to UWA entities and semantic associations. Entities are added to this view from: (i) the output of the algorithm discussed in Section 2; and (ii) the entities specified by performing the semi-automatic analysis using the Web Page Designer (WPD) editor embedded in Eclipse. The validated and refined entities and associations are finally saved into the internal repository.
- **Collections Perspective** It groups together the views used to drive the recovering of collections and their centers. The collection view shows, for each identified collection, the pages containing it. Moreover, analyst can validate and refine the identified collections using the collection editor.
- **Navigation Model Views** They are used by the analyst to generate and to browse the UWA navigation model from the information regarding entities, semantic associations and collections stored in the RTP repository. Clusters and related nodes, along with links among them and the attributes, are also shown.

A screenshot of the RTP tool in the perspective for the recovery of UWA entities and associations captured during the analysis of the Web site Visit Trentino.com is shown in Figure 3.
Figure 4. A screenshot of the UWA editor showing some of the Semantic Associations, Collections, Navigation Clusters and Publishing Pages recovered for Exibart.com

The reverse engineering process implemented in the RE-UWA platform is semi-automatic. The user is invoked by the tool at each validation step (refer to the Figure 1) in order to assess the models extracted automatically and optionally improve them by changing algorithm parameters or direct editing. In particular the entity recovery is performed by two different steps: automatic and manual. The automatic mode recovers the keywords inside cloned pages by comparing them as specified in Section 2.2. While this mode is completely automatic it cannot be always used: as already pointed out in Section 2, when keywords are not used in the WA the recovery cannot be done. In those cases the manual step allows the user to mark the pages with semantic hints representing keywords: the tool automatically tries to apply the provided hints to the corresponding HTML nodes of the other pages in the same cluster (such pages are clones). This requires more effort since it forces user to tag at least a page for each cluster but allows the process to continue with keywords extraction and template generation steps as in the automatic case. Manual and automatic modes can also be used together and results are merged by the tool: this is useful since often applications use keywords only in some cases and not consistently in all sections.

3.2. The UWA Modeling Tool

Starting from the UWA conceptual meta-model, created with the Eclipse Modeling Framework (EMF) tool and the Ecore meta-language, a visual editor has been developed to graphically instantiate conceptual models and in order to replace the default tree editor offered by the EMF tool.

As shown in Figure 2 (b), to define the UWA meta-model we used EMF and the Ecore meta-language to support the meta-modeling process, while for realizing the UWA editor starting from
WEB APPLICATIONS DESIGN RECOVERY AND EVOLUTION WITH RE-UWA

Figure 5. A screenshot of the UWA editor showing the UWA Hyperbase recovered by the gold standard for VisitTrentino.it

the meta-model, we used GMF. The output of this phase has been a graphical editor with the features of drag and drop, copy and paste, undo and redo that allows to draw UWA diagrams using the tools palette and to connect, re-size or move the model elements.

The UWA Modeling tool is a fully graphical UWA editor that supports in place editing of all UWA diagrams and concepts. Using such editor the modeler can edit the models produced by the reverse step changing them accordingly to new design requirements or evolution tasks. The editor supports exporting UWA models in MOF/EMF format. This allows to use them with other tools or generators opening the road to future work in model-driven code generation approaches.

4. CASE STUDY

In order to validate our redesign approach we have applied it to a number of real world Web sites. The selected applications, are:

- **Exibart**, an art news portal ([www.exibart.com](http://www.exibart.com)), which provides information on art exhibitions, events, performances, with reviews and comments by art curators and visitors.
- **VisitTrentino**, a tourism portal ([www.visittrentino.it](http://www.visittrentino.it)), which provides information on locations, facilities, restaurants and activities for people visiting the Region of Trentino in Italy.
- **AirBnB**, an online marketplace and community portal ([www.airbnb.com](http://www.airbnb.com)), that allows users to book and list spaces they are interested in renting. Features like galleries, notifications and classifications are also provided.
- **ClassicalM**, an online music store ([www.classicalm.com](http://www.classicalm.com)), which provides information on classical music compositions, composers, and interpreters, and which enables listening to samples of each composition or collection before purchasing and downloading it.
Figure 6. A screenshot of the UWA editor showing some of the entities, collections, navigation clusters and publishing pages obtained by the experts conducting a manual analysis on VisitTrentino.it

Figure 7. A screenshot of the UWA editor showing the UWA Hyperbase Model recovered for Exibart.com
For each Web application a preliminary similarity analysis was performed on the downloaded HTML pages and a subset of pages was selected in order to cover the different sections of the site. Table I reports the number of pages selected for each Web application.

As an example, around 30K pages were downloaded for the VisitTrentino Web application and from these pages 8.5K were selected by the similarity analysis, with an average of 212 pages per section.

Clone analysis and clustering techniques were used in order to group similar pages. For the Exibart Web application this analysis was executed several times with different clustering thresholds in order to obtain the best results possible. For each threshold value, precision and recall were calculated and discussed. Precision was calculated using the valid groups extracted from templates with respect to the total number of generated groups. Conversely, recall was calculated by manually inspecting the application in order to compute the ratio between the number of identified UWA entities and the total number of the entities defined in the application.

Degenerate clusters containing a single page or useless pages were discarded before the model abstraction steps. From each cluster, a template was generated and for each valid template, a group of attributes containing potential keywords was extracted using the content extraction algorithm detailed in [4]. These groups were then validated by the human expert using the RE-UWA tool perspective that allows discarding invalid groups and refining the valid ones (by means of merge, split and modify operations). The resulting groups of keywords represented the candidate UWA entities recovered for the application. Following the process described in Section 2.2 UWA semantic associations and collections were recovered, and together with entities formed the information model of the application; navigation nodes and navigation clusters were recovered to abstract the navigation model, and publishing pages, sections, and publishing units were abstracted to obtain the UWA publishing model for the application.

In order to validate the approach, the selected Web applications were analyzed by two groups of analysts, each composed of two members working in pair. The first group recovered the conceptual design of each application using the RE-UWA Tool and applying the semi-automatic reverse engineering process described in Section 2.2; the second group conducted the analysis manually, by directly inspecting the Web applications considered in the study. The resulting models are compared in order to assess the validity and effectiveness of the semi-automatic approach. Finally, in order to provide a measure of the quality of the overall design recovery process, the results were validated by direct inspection and a complete and correct model was created. These models have been used to calculate precision and recall for entities, semantic associations and collections.

Next we report on the results obtained by applying the design recovery approach on the selected Web applications using the RE-UWA tool, and compare them with the results obtained by the experts manually, i.e., without applying the approach and the RE-UWA supporting tool. Then we present some of the design improvements that the recovered UWA models suggest for each Website.

### 4.1. Recovering the Conceptual Design of the Applications

Table II reports for each application and for the best achieved clustering threshold, the total, valid, and rejected number of groups of keywords that were identified with the semi-automatic analysis. The computed groups represent the candidate UWA entities. The abstraction algorithm defined in [6]
Table II. Total, valid, and rejected groups of keywords and entities identified for each Web application

<table>
<thead>
<tr>
<th></th>
<th>#Valid Templates</th>
<th>#Analyzed Templates</th>
<th>#Groups of Keywords</th>
<th>#Groups of Keywords to Validate</th>
<th>#Automatically Discarded Groups of Keywords</th>
<th>#Entities identified with RE-UWA</th>
<th>#Entities Manually Identified by the Experts</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirBnB</td>
<td>160</td>
<td>77</td>
<td>390</td>
<td>189</td>
<td>201</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>ClassicalM</td>
<td>105</td>
<td>68</td>
<td>215</td>
<td>124</td>
<td>101</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Exibart</td>
<td>102</td>
<td>66</td>
<td>149</td>
<td>109</td>
<td>40</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>VisitTrentino</td>
<td>374</td>
<td>143</td>
<td>844</td>
<td>619</td>
<td>619</td>
<td>8</td>
<td>13</td>
</tr>
</tbody>
</table>

Table III. Entity types and slots identified in VisitTrentino, ClassicalM, AirBnB and Exibart.

<table>
<thead>
<tr>
<th>WA</th>
<th>Entity</th>
<th>Slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>AirBnB</td>
<td>Guest</td>
<td>Name, Age, Photo, School, Work, About me</td>
</tr>
<tr>
<td></td>
<td>Member</td>
<td>About, Acceptance Rate, Work, Response Rate Name, Member Since, Photo, Languages, Groups, School</td>
</tr>
<tr>
<td></td>
<td>Room</td>
<td>Name, Description, Size, Location, Room-Type, Extra people, Minimum Stay, Neighborhood, City, Amenities Description, Photos, Maps, Weekly Price</td>
</tr>
<tr>
<td></td>
<td>Group</td>
<td>Name, Countries, Members, Cities, About</td>
</tr>
<tr>
<td>ClassicalM</td>
<td>Artist</td>
<td>Name, Photo, VoiceOrInstrument</td>
</tr>
<tr>
<td></td>
<td>Collection</td>
<td>Title, CD Cover Image, Year, Description, Track-list</td>
</tr>
<tr>
<td></td>
<td>Composer</td>
<td>Name, Photo, Country, Period, Biography, Birth Date, Death Date</td>
</tr>
<tr>
<td></td>
<td>Composition</td>
<td>Title, CD Cover Image, Genre, Opus, Composer, Conductor, Performers, Track-list</td>
</tr>
<tr>
<td></td>
<td>Discography</td>
<td>Title, CD Cover Image, Performers</td>
</tr>
<tr>
<td></td>
<td>Conductor</td>
<td>Name, Photo, Birth Date, Death Date</td>
</tr>
<tr>
<td>ExibArt</td>
<td>Author</td>
<td>Name.</td>
</tr>
<tr>
<td></td>
<td>Comment</td>
<td>Author, Release Date, Text.</td>
</tr>
<tr>
<td></td>
<td>Event</td>
<td>Title, Subtitle, City, Date, Photo, Description, Time, Entrance, Supports, Note, Kind, Email, Website, Press Release.</td>
</tr>
<tr>
<td></td>
<td>News</td>
<td>Title, Release Date, Text, Author, Note, Photo.</td>
</tr>
<tr>
<td></td>
<td>Person</td>
<td>Name.</td>
</tr>
<tr>
<td>VisitTrentino</td>
<td>Attraction</td>
<td>Information, Hours, Tel, Fax, Email, Name, Photo, Map, Location</td>
</tr>
<tr>
<td></td>
<td>Event</td>
<td>Type Brochure, Information, Name, Short Description, Long Description, Location, Place, Period</td>
</tr>
<tr>
<td></td>
<td>Ski Area</td>
<td>Well-Being, Services, Information, Photo Gallery, Interests, Accommodation, Mobile, Phone, Fax</td>
</tr>
<tr>
<td></td>
<td>Tourist Area</td>
<td>Name, Short Description, Contacts, Directions, Photos, Videos, Map</td>
</tr>
<tr>
<td></td>
<td>Eating and Drinking</td>
<td>Hours, Details, Days closed, Winter Special, Multimedia, Services, Languages Spoken, Information, Accommodation, Location, Address and Location, Brochure, Email, No, of Covers, Closing Day, Short Description, Prices, Tel, Fax, Features, Information, Video Gallery, Credit Cards Accepted,</td>
</tr>
<tr>
<td></td>
<td>Sport-Centre</td>
<td>Altitude, Location, Category, Multimedia, Brochure, Location, Video Gallery, Information, Tel, Accesses, Difficulty, Features, Photo Gallery, Interests, Gradients, Category, Name Departure point Fax Email</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>Location Name, Information, Map</td>
</tr>
<tr>
<td></td>
<td>Accommodation</td>
<td>Home, Phone, Fax, Notes, Activities and leisure, Amenities, Family Amenities, Parking Information, Details, Short Description, Long Description, Days closed, Suitable for, Multimedia, Fax, Length, Address and Location, Mobile, Days closed, Rooms, Equipment, Prices, Accommodation, Location, Credit Cards Accepted, Brochure, Name, Contact, Altitude, Prices per person night, Languages Spoken, Hours, Services, Supplements, Facilities, Video Gallery, Photo Gallery, Amenities for the children, Information, Features, Email</td>
</tr>
</tbody>
</table>
rich UWA information model. For each entity the corresponding slots are reported in the table. For instance, in VisitTrentino for the Accommodation entity (a key concept within the information model of the Web site) there are 40 slots.

After the abstraction of UWA entities, existing semantic associations among them were identified. For each Web application, the set of semantic associations recovered with RE-UWA are shown in Table V (associations rows, third and fifth column). As shown in the table, 9 semantic associations were found for Exibart, 19 for VisitTrentino, 8 for ClassicalM, and 8 for AirBnB. The table also reports, in the fourth column, the associations that were identified only by the human experts performing the reverse modeling process manually.

Table IV reports the collections identified for the four Web applications with RE-UWA. For each collection the set of slots included in its center are reported in the second column. For Exibart, the RE-UWA tool identified 12 collections. Many of them have only one associated slot and the collection with the highest number of slots (News of a City) includes 5 slot in its collection center. Among the analyzed Web applications, VisitTrentino was found to have the highest number of collections (17), with several of them presenting in their centers a large number of slots. For ClassicalM 11 collections were identified. The number of slots corresponding to these collections is low. Finally, in AirBnB only 3 collections were found.

Table V reports the comparison between the design recovered by the first two analysts using the RE-UWA tool, with that recovered by the human experts performing a complete manual analysis of the Web sites. The first column reports the Web application while the second reports the elements taken into account in the corresponding rows. The third column shows the common elements between the semi-automatically recovered and the manually recovered models, while the fourth and fifth columns are, respectively, the elements identified only by human experts and that recovered by the tool.

As we can observe, in two cases the experts found some entities that were not identified by with the tool (the Exhibition Area entity for Exibart, and 5 more entities for VisitTrentino). We found by inspection, that while for the first case (Exibart) this is related to missing keywords, in the VisitTrentino case this was due to the lack of support for inheritance. The mining approach, in fact, is not capable of recognizing specialization among entities and, because of the common attributes and very similar structures, all specialized entities regarding specific types of Resorts and Hotels were recovered as Accommodations. Conversely, the human experts were easily able to identify and model the sub-concepts as separate entities. We can also observe that while the tool has an acceptable precision in recovering entities, it is very precise in recovering association and collections. This is the case because the associations and the collections identification algorithm exploits mainly information on entities and on the pages containing them. Moreover the recall measured for the associations and collections is lower only when there are missed entities (if an entity is not identified then all associations and collections involving it will be missing too).

The last two columns of Table V report precision and recall calculated for the design models abstracted with the RE-UWA tool by considering as gold standard the models obtained by merging the results obtained with the tool-supported analysis with that obtained manually by the human experts. These values are also reported in Figure 8. Looking at the graph it can be seen that precision and recall are higher for sites with simpler domain models and lower (but still quite good) for sites with more complex ones. As an example, the VisitTrentino application scored p={$0.63,0.72,0.7$} and r={$0.61,0.63,0.7$} for entities, associations and collections respectively. Still looking at Table V some observations can be made for the elements recovered only by the tool. In several cases, mainly for applications with a very rich domain model like VisitTrentino, the experts missed several associations and some kind of collections as they were unable to reach the pages with instances of those elements. The analysis made on a high number of pages by the RE-UWA tool greatly reduces this risk. This is confirmed by the results reported in the table: the cases in which the associations and the collections are missed by the tool are almost all related to missed entities.

Figure 7 shows a screenshot of the UWA Editor with an excerpt of the information model recovered for Exibart. We can recognize in it entities with their internal components and slots.
Table IV. Collections identified in VisitTrentino, ClassicalM, AirBnB and Exibart.

<table>
<thead>
<tr>
<th>Collection Name</th>
<th>Slots of Collection Centre</th>
<th>WA</th>
<th>Collection Name</th>
<th>Slots of Collection Centre</th>
<th>AirBnB</th>
<th>Collection Name</th>
<th>Slots of Collection Centre</th>
<th>ClassicalM</th>
<th>Collection Name</th>
<th>Slots of Collection Centre</th>
<th>ExibArt</th>
<th>Collection Name</th>
<th>Slots of Collection Centre</th>
<th>VisitTrentino</th>
<th>Collection Name</th>
<th>Slots of Collection Centre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recent Added Members</td>
<td>Name</td>
<td></td>
<td>All Composers by Name Initial</td>
<td>Title, CD Cover Image, Composer</td>
<td></td>
<td>All Collections by Title Initial</td>
<td>Title, CD Cover Image, Composer</td>
<td></td>
<td>All Attractions by Category</td>
<td>Image, Name, Description, Address, Details</td>
<td></td>
<td>Artistic and Cultural Attractions by Type</td>
<td>Image, Name, Description, Address, Details</td>
<td></td>
<td>Natural Attractions by Type</td>
<td>Image, Name, Description, Address, Details</td>
</tr>
</tbody>
</table>

Figure 4 shows a subset of the abstracted UWA model also showing collections, navigation clusters and publishing pages. In figures 5 and 6 the analogous screenshots for VisitTrentino are shown.
Table V. Comparison between results obtained by RE-UWA Tool and by the Experts.

<table>
<thead>
<tr>
<th>WA K</th>
<th>Common Elements</th>
<th>Only by Expert</th>
<th>Only by RE-UWA Tool</th>
<th>P</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airbnb</td>
<td>Entities</td>
<td>Guest, Member, Room, Group</td>
<td></td>
<td>0.40</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Associations</td>
<td>Guest ( \rightarrow ) (Member, Room), Member ( \rightarrow ) (Guest, Room, Group), Room ( \rightarrow ) (Guest, Member, Group), Group ( \rightarrow ) Member</td>
<td>Member ( \rightarrow ) Member, Room ( \rightarrow ) Room, Group ( \rightarrow ) Room</td>
<td>1</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>Collections</td>
<td>Recent Added Members, Rooms By Group, Rooms By Type</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ClassicalIM</td>
<td>Entities</td>
<td>Artist, Collection, Composer, Composition, Discography, Conductor</td>
<td></td>
<td>0.54</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Associations</td>
<td>Composition ( \rightarrow ) Composer, Composition ( \rightarrow ) Conductor, Composition ( \rightarrow ) Artist, Discography ( \rightarrow ) Artist,</td>
<td>Composition ( \rightarrow ) Composition, Discography ( \rightarrow ) Discography, Collection ( \rightarrow ) Collection</td>
<td>0.73</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>Collections</td>
<td>All Composers by Name, All Composers by Period, All Composers by Genre, All Artists by Name, All Artists by Genre, All Collections by Title, All Collections by Genre, All Compositions by Title, All Compositions by Genre, All Discographies by Title, All Discographies by Genre</td>
<td>Composer’s Compositions by Genre, Composer’s Compositions By Period</td>
<td>1</td>
<td>0.84</td>
</tr>
<tr>
<td>Exibart</td>
<td>Entities</td>
<td>Author, Comment, Event, News, Person</td>
<td>Exhibition Area</td>
<td>0.61</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>Associations</td>
<td>Author ( \rightarrow ) (Comment, Event, Person), Comment ( \rightarrow ) (Author, News, Person), Event ( \rightarrow ) (Author, Person), News ( \rightarrow ) Comment</td>
<td>Author ( \rightarrow ) Author, Person ( \rightarrow ) (Event, News, Person), News ( \rightarrow ) News, Event ( \rightarrow ) Exhibition Area, Exhibition Area ( \rightarrow ) Event</td>
<td>1</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>Collections</td>
<td>Related Authors, Comments, Last Comments, Events, Events of the Day, News, Last News, Artist Parade, Curator Parade</td>
<td>Persons</td>
<td>News of The City, News of an Author, Most Clicked News</td>
<td>1</td>
</tr>
<tr>
<td>VisitTrentino</td>
<td>Entities</td>
<td>Attraction, Event, Ski Area, Tourist Area, Location, Accommodation, Sport Centre, Eating and Drinking</td>
<td>Holiday Offer, Well-Being Centre, Thermal-Water, Vacation Package, Themed Holiday</td>
<td>0.63</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>Associations</td>
<td>Location ( \rightarrow ) Sport Centre, Accommodation, Attraction, Tourist Area ( \rightarrow ) (Eating And Drinking, Accommodation, Attraction, Sport Centre ( \rightarrow ) Location, Accommodation, Attraction, Event ( \rightarrow ) Location</td>
<td>Location ( \rightarrow ) Vacation Package, Thermal-Water ( \rightarrow ) Location, Themed Holiday ( \rightarrow ) Hotel Lodging, Hotel Lodging ( \rightarrow ) Vacation Package, Holiday Offer ( \rightarrow ) Location</td>
<td>0.72</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>Collections</td>
<td>All Attractions By Cat., Sport Centres By Type, Events By Cat., Events of the Month, Accommodations By Type &amp; Category, Accommodations By Type &amp; Feedback, Ski Areas, Locations, Tourist Areas</td>
<td>Top Vacation Package Offers, Ski Districts, Holiday Offers of a Tourist Area, All Well being Centres, All Thermal Water and Spas, All Themed Holidays, All Holiday Offers</td>
<td></td>
<td>0.7</td>
</tr>
</tbody>
</table>
4.2. Identifying Possible Design Improvements

By analyzing the recovered UWA models and taking into account the requirements defined for the specific application, a domain expert can identify weaknesses of the current design (if any) and derive possible design improvements.

In the following section we synthesize some of the weaknesses that the experts were able to identify for the Web applications of our case study and possible design improvements to overcome them.

4.2.1. AirBnB

Entities, semantic associations and collections identified for the AirBnB Web site show that the current design of this application well supports the room-renting community it aggregates. However, the analysis of the recovered models and their comparison against characteristics common to Web sites belonging to the same domain of AirBnB (i.e., tourist and accommodation social network Web sites) can reveal weaknesses and/or room for possible improvements of the current design of the application. As an example, one of the concepts which seems to be missing in the current design of AirBnB is the concept of Rating. Guests can provide Reviews on Rooms and Members (room owners) and Members can provide Reviews on their Guests. The creation of trust in this community is based on these Reviews which, from the modeling point of view, corresponds to semantic associations between the Room, Guest and Member entities, with Date and Review Content as information slots. A Review have no Rating slot associated. This causes the need for a user visiting the site to entirely read a Review before understanding if it represents a positive or negative feedback for the entity instance to which it is associated, i.e., a Room, a Guest, or a Member. A simple but effective design improvement for the AirBnB Web site could be the introduction of a Rating slot in the centers of the following semantic associations: Guest-providesReviewfor-Room, Guest-providesReviewfor-Member, Member-providesReviewfor-Guest. A related but more invasive change in the design of this application could be the introduction of a Review entity substituting the above listed semantic associations and collecting slots such as the Content of the Review, the submission Date and the Rating. In this case, a Review could itself receive feedback from other Guests/Members which could confirm or disprove it and assess its usefulness. Moreover, each entity that can receive Reviews could be associated with an additional slot named Overall Rating, calculated as the average Rating of the received Reviews.

Such design improvements are depicted in Figure 9. In particular, Figure 9 (a) shows an excerpt of the “as-is” hyperbase model recovered for the AirBnB Web site, and Figure 9 (b) shows the improved “to-be” version of it. For simplicity, slots that are not involved in any change are omitted. The new entity introduced in the “to-be” design of AirBnB requires new nodes, clusters and publishing pages to be added to the navigation and publishing UWA design models, respectively, in
Figure 9. The Hyperbase Model (Entities and Semantic Associations) recovered for the AirBnB Web site (a), and an improved version of it (b) including the Review Entity and Rating Slots
order for the new content to be reachable and visible. For the sake of brevity, these changes are not shown in the figure but are fully supported by the UWA Editor.

4.2.2. ClassicalM Let us consider the list of entities and associated slots recovered for the ClassicalM Web site as reported in Table III. It can be noted that a biography information slot is only available for the Composer entity. An easy design improvement would consist in providing this information also for Conductors and Artists, by enriching the respective entities with a Biography slot*. Going further, if we consider the goals of a user visiting the ClassicalM Web site, we can identify another major lack in the current conceptual design of the site. It concerns the slot Genre of the Composition entity. This slot is currently just an attribute used to classify and group Compositions (e.g. it is used in the collection Compositions by Genre). Additional information that could be usefully provided to the user regarding a Genre are: a description of its salient characteristics, historic and geographic outlines on where and when the genre was born and spread, the list of its most famous composers, and some example compositions. This design improvement can be accomplished by introducing a new entity in the model, named Music Genre, with the just mentioned information slots. Another possible improvement for the design of ClassicalM concerns the slot Period of the Composer entity. This slot is currently an information element characterizing a Composer by the musical period to which he/she belongs (e.g., Baroque, Romantics, Modernism, etc.) and enables grouping of Composers (used in the collection of Composers by Period). An evolved version of the ClassicalM Web application design could have the Musical Period being modeled as an entity including the slots Name, Description, Epoch of Origin and Spread. The new entity should be related to the Composer entity with a bi-directional semantic association named Composer-belongs-to-Musical_Period. Finally, another design improvement concern the Collection entity. In the current version of the ClassicalM application, this entity is completely isolated from other entities of the model. This means that even navigating from a Collection to one of its performers or Composers is not possible. To address this weakness, semantic associations from Collection to Artist, Composer, Conductor and Music Period could be added along with their centers showing relevant preview information.

To better highlight the described design improvements, we show the relevant excerpt of the design “as is” in Figure 10-a and the corresponding excerpt of the design “to be” in Figure 10-b.

4.2.3. Exibart Let us consider Figure 4 which shows a detailed view of a small portion of the hyperbase, navigation and publishing models for the Exibart.com Web site. In these diagrams, semantic associations and collections are presented with their centers and included slots. By analyzing these centers, the analyst can evaluate if a collection or an association is defined properly or if it can be better presented. In fact, UWA design guidelines suggest that slots to be included in collections and associations centers should be chosen based on their purpose to “announce” the entity they will link to. Looking at the model we found that a very useful collection (called Most Clicked News) could be added to the Web site in the home page, in order to improve the access to News. In the collection center, the slots Publication Date, Short Description, and Authors were selected as preview attributes for the News. By using this collection, the users of Exibart can be informed on which News are the most viewed ones and click to visit them directly from the home page.

Moving onto the navigation model, among other aspects, a UWA navigation cluster specifies the navigation pattern used within the cluster to navigate between nodes included in it. Typical navigation patterns are Index, All-All, Guided tour, Index/guided tour. All of the collections identified in the Exibart Web site can be navigated by index, i.e., navigating from the center of the collection to its members. This is represented by the recovered UWA navigation model. However the collection of events that took place in an exhibition area should instead support a guided tour

---

*Adding a slot to an entity requires the corresponding data to be available in order to be published. The unavailability of these data could be the reason why the slots mentioned above are missing in “as-is” design of the application. Nevertheless, in the context of this paper, we aim to show that the recovered UWA models enable the analyst to identify possible shortcomings of the current design and define possible solutions to overcome them.
Figure 10. The Hyperbase Model (Entities and Semantic Associations) recovered for ClassicalM.com (a), and an improved version of it (b) including the Music Genre and Music Period entities and related Semantic Associations.
navigation pattern, so to enable the user navigating from an event to the next one, in a temporal order.

Finally, the analysis of the UWA publishing model can highlight pages or sections of the site that use a layout different from the rest, or publishing units which are repeated (for convenience or by mistake) throughout the site. In Exibart, this is the case of the publishing unit “Speed News”, which is used to publish the collection of latest news in the home page and is also included in almost all other pages of the site.

4.2.4. VisitTrentino

The VisitTrentino Web site is very rich in content, very well organized in navigation structure, and very effective in presentation. This not only appears from a direct inspection of the site, but is also shown by the UWA models representing the as-is design of this application. Nevertheless, by analyzing these models in detail, it is possible to identify some venues for design improvements. Let us consider the UWA hyperbase, navigation and publishing model recovered for this application and shown in figures 5 and 6. It can be noted that in the hyperbase model the two entities, namely Ski Area and Ski District located in the bottom-right part of the diagram, are mostly “isolated” from the others (i.e. they are not involved in any semantic associations with other entities). In particular, a Ski District is related only to the Ski Areas it contains, and a Ski Area is related only to the Sport Centres located in it. In fact, users visiting the page presenting a Ski Area in the VisitTrentino Web site have no direct way to find Accommodations (Hotels and Lodgings), Restaurants (Eating & Drinking), Attractions, Events or Holiday Offers for that Ski Area. Conversely, such associations and the related navigation paths exist for the entities Tourist Location and Tourist Area. The above considerations suggest that the hyperbase of VisitTrentino.it could be enriched with additional semantic associations between the Ski Area and/or the Ski District entities and the other entities of the site, in order to support navigation between them.

5. DISCUSSION ON VALIDITY AND LIMITS OF THE APPROACH

In order to better validate the results obtained in the conducted case study, we report, for the Exibart Web application, precision and recall for each clustering threshold.

Table VI and Figure 11 report discarded and valid groups and templates at each clustering threshold while Figure 12 reports a graph of precision and recall at different clustering thresholds. As we can see, with a threshold of 0.08 only 1 entity out of 6 were correctly identified (with a low precision, around 0.34). Decreasing the threshold to 0.06, 3 entities were identified giving a recall of 0.5 and with a better and acceptable precision (p=0.43). The best values were obtained at a threshold of 0.03; in this case 5 entities were automatically recovered (improving recall to r=0.83) with a much better precision (p=0.61). Continuing to decrease the threshold did not allow the identification of
more entities since the remaining two were in pages in which no keywords are used and hence a completely manual identification is needed). We observed a similar trend also in the other Web applications.

The results obtained in the case study indicate that the redesign approach is feasible and valid. It enables taking the benefits offered by a design methodology specifically suited for Web applications design, such as UWA, on existing Web applications. Our redesign approach makes it possible (i) to recover a model of the current design of the application, (ii) to identify its weakness and (iii) to evolve the current design in order to meet new or changed requirements. Additionally, as UWA provides tools for the model-driven development of the designed applications [7], our redesign process can be followed by a forward engineering phase.
The conducted case study also highlighted some limits of the approach, mostly related to the reverse engineering phase. One of them is related to the difficulty to abstract UWA entities automatically when the Web site does not make use of keywords to qualify attribute values. In this case, as pointed out in Section 3, manual inspection must be used, thus increasing the effort required to perform the reverse engineering step. Another limit of the approach is related to the identification of UWA entity components. At the moment the RE-UWA tool groups all the slots identified for an entity into a default entity component; this requires some additional effort when refining the recovered model using the UWA Editor in the forward design phase. As previously mentioned, in some cases, the templates generated for the identified page clusters originated synonyms or duplicated and thus useless groups of keywords (e.g., keywords originated from navigation bars and page footers). To improve the quality of the groups of keywords extracted from templates, some techniques of the information retrieval domain can be adopted. Such techniques include: (i) filtering keywords with a stop word lists, thus omitting words commonly used in menus and navigation bars, or specific for the considered application; and (ii) techniques allowing the identification of synonym groups, thus reducing the number of groups to validate.

A limitation of the RE-UWA tool is its lack of support for RIA (e.g., AJAX) Web applications. Even if the tool is capable of analyzing both statically and dynamically generated Web pages, it is not able to capture the dynamic asynchronous interactions between server and clients typical of RIA applications. This limitation can be overcome by implementing a RIA-aware crawler capable of capturing such interactions, as the subsequent steps of the approach are completely independent of how the Web pages of the application are generated and captured.

6. RELATED WORK

Recently, many studies focused on reverse engineering of Web applications have been conducted. They differ in the aspects they treat, the level of abstraction of the recovered information and the formalism they adopt to represent it. These studies contribute to distinguish among the suitable approaches, methods and tools supporting the Web application modeling processes.

Several works [16, 8, 19] focus on tool-supported methods recovering an architectural view of the Web application depicting its components (i.e., pages, page components, etc.) and the relationships among them at different levels of detail. Some studies [15, 11] aim at defining approaches for modeling and verifying of the Web application design and its functional requirements or for modeling and measuring the hyperlink structure; for example in [27] an approach to reconstruct and improve hyperlink structure, is discussed.

Moreover several UML-based approaches are introduced to support Web application modeling using standard UML constructs. UML use case diagrams are used to represent the abstracted functional requirements. UML diagrams are also used in [26] to depict the static, dynamic and behavioral aspects of the analyzed Web application. In [14] the authors present a tool for the reverse engineering of legacy PHP Web applications, generating static UML diagrams which show page components, functions and dependencies.

Approaches and techniques for recovering and abstracting business processes, by means of dynamic analysis of the Web applications implementing them, are presented in [22] and in [10]. The VAQUISTA [25] system, proposed by Vanderdonckt et al., allows the reverse engineering of the presentation model of a Web page, in order to migrate it to another environment. The TERESA tool presented in [18] produces a task-oriented model of a Web application by source code static analysis, where each task represents single page functions triggered by user requests. The resulting model is suitable for assessing Web application usability, or for tracing the profile of the users of the analyzed Web application.

In [21] a model driven approach for the development of context-aware Web applications is presented. This paper illustrates a conceptual framework that provides modeling capabilities for context-aware, multichannel Web applications; it also shows how high-level modeling constructs can be used to drive the application development process through an automatic code generation process. The approach is built on WebML (Web Modeling Language) and is also accompanied by
a development method and a CASE tool. However, the authors claim that their approach can be adopted independently of the chosen model, method, and tool.

Web-based system evolution in the context of model driven architecture is also discussed in [9]. This research presents a unified solution to Web-based system evolution, which consists of three components: (i) Web-based systems understanding; (ii) a Web-based systems representation tool and (iii) an extensible Web application framework to support the evolution task. The authors have validated the approach with a case study, highlighting the feasibility of the proposed solution.

The approach proposed in [2, 24, 3] aims to adopt a methodology and a set of models and modeling tools for a user-centered conceptual design of ubiquitous Web applications redesign process based on methods for UWA Web design. In this respect our approach differs from the the works cited in this section and others proposed in the literature because it refers to a robust and complete methodology specifically for the conceptual design of Web applications. As such, the recovered models feature a user-centered perspective on the analyzed application. To the best of our knowledge, no other work deals with the recovering of such user-centered conceptual models. Moreover, our approach is based on a client-side source code analysis. As such it is applicable to any Web application producing HTML pages as a front-end, regardless of the technologies used on the server-side.

7. CONCLUSIONS

This paper presents an approach to recover the user-centered conceptual design from an existing Web application and to support its evolution. In particular, the approach is able to abstract a conceptual model representing the content, relationships between content, and views, as perceived by the final users of the application. The models are formalized according to the Ubiquitous Web Application (UWA) design framework in terms of entities, semantic associations, collections, navigation nodes and clusters, publishing pages and sections.

The abstracted models, by providing an up to date and user-centered representation of the Web application, can be used to reason about possible design improvement and evolution tasks aimed at satisfying new user requirements or to better meet existing ones. Additionally, the recovered models can also be used as a starting point of a forward engineering process aimed at migrating the application towards new technologies and implementation frameworks.

A tool, based on Eclipse IDE, has been developed to support all the phases of the design recovery and evolution process.

The case study, involving the four Web applications, showed that the approach is feasible and valid, and highlighted some possibilities of improvement.

The approach allowed a quite good extraction of UWA entities, associations, and collections comparable with the ones obtained by two human experts conducting the analysis manually (but with reduced effort and in less time).

A first consideration is that the process is sensitive to the presence, within pages, of structures reporting explicit labels. Improvements are possible mainly in the extraction of groups of related keywords from clusters of HTML page-clones. Information retrieval techniques can provide useful support to this aim.

A very useful extension of the proposed work is its integration with a model-driven semi-automatic re-engineering approach which enables introducing design improvements and implementing them with a Model-View-Controller (MVC) architectural design pattern (especially for legacy Web application that are not designed according to modern architectural patterns). The advantage is that the extracted and refined models can be used directly to drive the automatic reimplementation of the original (or an improved) conceptual design targeting modern and updated environments and technologies.

Future work will also consider the extension of the approach and the supporting tools. In particular the tools will be extended with:
- **New extractors/transformers** - Components providing new analyses for both Web application artifacts and UWA resources which can be easily integrated into the environment. For instance, definition and validation of new structural distance algorithms among HTML documents as well as the analysis of JavaScript code elements will be added.

- **A workflow engine** - In particular in this context, some work will be devoted to extend the process with the logic needed to abstract the transaction model (extracting a business transaction model from the Web application).

- **RIA Support** - Introducing in RE-UWA the support for Rich Internet Applications in order to extract model from sites based on the Ajax architectural pattern. The approach can be extended using an AJAX-enabled crawler to download dynamically generated contents. The crawler should be able to capture asynchronous UI states and the transitions among them during WA usage sessions.

- **Model-driven approach** - Another worthwhile extension to the approach is the definition of a model-driven code generation process taking the extracted and evolved UWA models as input in order to generate the complete application source code. We plan to build intermediate low level design models specific to each target platform and a code generation step to generate the actual code.

**REFERENCES**


