A Semi-Automated Approach to Redesign Web Applications with UWA

Mario Luca Bernardi
Dept. of Engineering, Univ. of Sannio, Italy
Viale Traiano, 1
Benevento, Italy
mlbernar@unisannio.it

Giuseppe A. Di Lucca
Dept. of Engineering, Univ. of Sannio, Italy
Viale Traiano, 1
Benevento, Italy
dilucca@unisannio.it

Damiano Distante
Fac. of Economics, Tel.M.A. University, Italy
Via S. Caterina da Siena, 57
Rome, Italy
distante@unitelma.it

Francesco Mazzone
Dept. of Engineering, Univ. of Sannio, Italy
Viale Traiano, 1
Benevento, Italy
fr.mazzone@gmail.com

ABSTRACT

This paper presents a process and a set of supporting tools for the semi-automatic redesign of web applications, based on the Ubiquitous Web Applications (UWA) design methodology. The process starts by analyzing the client side HTML pages of the legacy web application and applies clone detection and clustering techniques to abstract from them a UWA conceptual model of the application. The recovered model is then used as input for a model-driven forward engineering phase leading to a lower level design model that can be used to implement the application adopting an MVC architectural pattern. The paper describes both the reverse modeling and the forward design phases of the proposed process, the adopted models and underlying techniques, and the tool support provided for each phase. A concrete example of the application of the approach to redesign a real world web system is also described.

1. INTRODUCTION

In the last ten years the development, diffusion, usage, and popularity of Web based Applications (WAs) have exponentially grown: they have become the underlying engine of many e-business, including e-commerce, e-government, service, and data access providing. The complexity of WAs is grown according to the growth of the sophistication of the functions provided by them.

A main consequence has been the growing of the requests for modifications for a WA to meet new functional and non-functional requirements of the evolving context in which they are used. Thus WAs are characterized by continuous maintenance and evolution operations to meet and satisfy these requests. Usually, most of the required modifications are related to perfected, adaptive and evolutive maintenance operations for the adoption/integration of, or the migration towards, new technologies, the improvement of the front-end (client side) of the WA to better presents the WA’s contents and services to its users with increased performances. Thus most of the required modifications do not require to alter the existing functional behaviours or business rules of a WA, but just the operational/technological structure used for its execution or the way the contents and services are presented to users. In these cases the modifications entail mainly the design of the WA that has to be revised and modified, rather than the conceptual models describing the domain objects and services characterizing the WA. Thus, a re-engineering [5] (and mainly a re-design) of the WA has to be carried out to meet the needed modifications.

The availability of up-to-date documentation (such as the models describing the application’s contents structure, and its design) has a key role to successfully maintain/evolve these systems in the short time usually available to accomplish these tasks. Unfortunately, due to the short time-to-market that often constrains the development and maintenance of WAs, such documentation is often lacking. This causes maintenance and evolution to become difficult and risky tasks, potentially compromising the effectiveness and correctness of the whole system.

In particular the availability of models produced by Model Driven Engineering and User Centered methods in the development of a WA can reduce the complexity of re-engineering process effectively, mainly when this entails the front-end part of the WA, because these models well support the comprehension of the system to modify. The usage of techniques and tools to recover such models and documentation, when these lacks, from the system to modify is indispensable.

User-centered methods model an application from the point of view of its final users, highlighting the ways users will interact with the application and the concepts and services of their interest. Model-driven methods allow to model the
application structure, behavior, and requirements by means of formal modeling languages and the usage of transformation engines and generators that analyze certain aspects of models and then synthesize the needed types of artifacts.

Hence in MDE, models, instead of source code, represent the first class artifacts of the software development process. As a consequence they are not anymore only an informal way for merely communicating design choices and tradeoffs in the development; rather they are the formal descriptions of all requirement specifications and the coherent set of design decisions implementing them that can be used to automatically generate the final system implementation for several target platforms.

In the context of re-engineering (or re-designing) a system, if such models are not available, the main effort is on recovering them, from the existing system to drive a Model Driven re-engineering process.

The higher degree of automation in model driven software development makes more efficient the entire software lifecycle, higher the average quality (improving reusability and maintainability attributes) of the resulting software products.

In order to adopt a Model Driven Re-Engineering (MDRE) two main components are required: (i) a modeling language definition standard such as the Object Management Group’s (OMG) Meta Object Facility (MOF) [OMG04] that allows to represent concepts in a standardized and formal way is required and (ii) a model to model transformation technology (e.g., like Atlas transformations - ATL or Query View Transformation - QVT [OMG05a]) used in each node of the modeled-driven process to generate output models starting from other input models.

In this paper we propose a MDRE approach in which the REUWA reverse engineering process proposed in [1] is integrated with the MDE process highlighted in [7] in order to generate MOF-based UWA models, extracted from existing (legacy) WAs, and then derive from them a low-level MVC-based design implementing the same requirements, domain model and navigational structure of the original legacy application, that can be furtherly and easier revised and modified to meet new requests, such as the migration towards new platforms and languages.

The proposed approach enables recovering the “know-how” embedded in the legacy application by abstracting “as-is” its conceptual model and reusing this model to redesign the application using a proper Web engineering method in a model-driven fashion and adopting the MVC architectural pattern.

The rest of the paper is organized as follows. Section 2 discusses some of the works related to ours. Section 3 describes the overall redesign process and provides some details on the underlying techniques and technologies for both the reverse and forward design phases. Section 4 presents the set of tools developed to support and semi-automatize the overall approach and discusses their functionalities. A practical application of the approach to redesign a real-world web site is reported in Section 5. Section 6 concludes by summarizing its contribution and future work we are conducting.

2. RELATED WORK

Both reverse-engineering of existing Web Applications and model-driven web system development approaches are relevant to our redesign approach.

For the reverse engineering of a WA, several approaches have been proposed in the last years. They differ in the aspects they focus on, the level of abstraction of the recovered information and the formalism they adopt to represent it. The works presented in [12, 8, 15] focus on recovering an architectural view of the WA depicting its components (i.e., pages, page components, etc.) and the relationships among them at different levels of detail. In [11], an approach for abstracting a description of the functional requirements implemented by the WA is proposed. UML use case diagrams are used to represent the abstracted functional requirements. A technique and an approach for reverse engineering UWA Web Transactions models representing the business processes implemented by a WA from a user centered perspective are presented in [18]. The VAQUISTA [21] system by Vanderdonckt et al. allows the presentation model of a web page to be reverse engineered, in order to migrate it to another environment. The TERESA tool presented in [14] produces a task-oriented model of a WA by source code static analysis, where each task represents single page functions triggered by user requests. The resulting model is suitable for assessing WA usability, or for tracing the profile of the users of the analyzed WA.

The reverse engineering approach proposed in this paper differs from the works cited in this section, and others proposed in literature, mainly because it refers to a robust and complete methodology, specific for the conceptual design of WAs to abstract models which features a user-centered perspective on the application. No other work, to the best of our knowledge, deals with the recovering of such user-centered conceptual models. Moreover, being our approach based on client-side source code analysis, it is applicable to any WA producing HTML pages as front-end, regardless of the technologies used server-side.

Concerning the forward approach, the Model-Driven Development paradigm is applied successfully by a number of web engineering methods, namely UWE, OO-H, OOHD-MDA, and WebML. These methods use models to separate the platform-independent design of web systems from the platform-dependent implementations as much as possible. They have associated development environments that support code generation from model specifications, either fully or partially automated. The UWE [9] process to developing web systems follows the MDA principles and uses the OMG standards [13]. The process makes use of model transformations defined at metamodel level and specified in general purpose transformation languages, such as QVT [13] and graph transformations. A variant of the UWE approach is UWE4JSF [10]. It builds upon a set of meta-models (both for conceptual and low level design) and targets the JSF platform. It shares with our approach the transformations of conceptual models (by means of ATL rules) and provides fully automatic code generation. It however is only a forward approach and hence lacks the support to reverse engineer models from existing applications. OOHDMDA [16] generates servlet-based web applications from OOHD models. The OOHDMDA approach follows MDA principles by employing the OOHD conceptual and navigational scheme of a web application as the basic PIM for the MDA process, using any UML-based design tool which produces an XMI-file as output. WebML [4, 3, 9] is a model-driven method for the development of data intensive web applications, with an associated supporting CASE tool called WebRatio. WebML
follows an MDD approach for mapping its modelling elements onto the components of the MVC Model 2 architecture, which can be transformed into components for different platforms.

Similar to our approach, UWE, OO-H and OOHDMDA adopt an MDD process that follows MDA principles for the models. Differently from our approach they do reported, WebML differs from our and other considered approach in that its process is MDD but not MDA. Similar to our approach, WebML uses MVC as architectural pattern for its PIMs. All the considered approaches enable different technologies to be used for the implementation of the PSMs. Our choice of adopting MVC as architecture for the PIM logical model guarantees the availability of a wide range of open-source and commercial technology frameworks to choose from for the different platforms, such as J2EE, .Net and PHP.

3. 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 Ubiquitous Web Application (UWA) design methodology and a number of code analysis techniques (including clone detection and clustering techniques) to abstract a UWA conceptual model of the application to redesign. The second phase takes as input the UWA model recovered by the first phase, enables refining and evolving it (e.g., to respond to new requirements), and finally produce transform a low level design model enabling the implementing the new design adopting a Model-View-Controller (MVC) architectural pattern and the JavaServer Faces (JSF) technology framework. Tool support is provided for both the two phases which are semi-automatic and require the user intervention mostly to drive the different process activities and validate the produced results. In the following of this section we first introduce the UWA methodology and, more specifically, its design models, and then describe the two phases in which the redesign process is articulated.

3.1 The UWA Web Design Methodology

The Ubiquitous Web Applications (UWA) design framework includes a complete design methodology and a set of models for the user-centered design of data and operation intensive ubiquitous (i.e., multi-channel, multi-user and generally context-aware) Web applications [20].

Similarly to other well known web engineering methods proposed in the literature, such as OOHDMDA [17], WebML [2] and UWE [9], UWA specifies the design of a web application by means of three main models: the Information Model (n.k.a., content or domain model), the Navigation Model, and the Presentation Model [19]. 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 contents, 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 presentation 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, indeed, 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 will behave for different user types. In the above sense, UWA is a user-centered design methodology, and so are its models.

Entities, entities’ 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. An example of UWA model, including a portion of the Information, Navigation and Publishing models is reported in Figure 6.

3.2 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. In the MDA jargon, such models are also known as Platform Independent Models (PIM). 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-automated reverse engineer approach which is able to abstract UWA models from existing web applications [1][19]. Here we shortly synthesize this process and the underlying techniques. More details can be found in [1].

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

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

Figure 1 shows the activity diagrams modelling the defined Reverse Engineering process. The process for a given WA is carried by analysing a significant amount of client side HTML pages of the application that are downloaded from it by means of a Web crawler.

3.2.1 UWA Information Model Abstraction

The UWA Information model of the considered application is recovered by analyzing its pages to abstract Entities, Semantic Association and Collections.
The identification of UWA Entities is carried out by searching for groups of related attributes (we refer to these attributes as to 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 manually carried over the automatically identified groups of keywords to finally obtain a validated set of UWA Entities associated to the considered application.

A candidate 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 a Slot is used as an anchor to set 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; (ii) a list of hyperlinks pointing to pages showing different instances of the same Entity. As for Entities and Semantic Associations, the automatically recovered UWA Collections will undergo to a validation phase conducted by a human expert knowledgeable of the application domain.

### 3.2.2 UWA Navigation Model Abstraction

The recovery of the UWA Navigational Model is carried out by identifying Nodes and Clusters for the analyzed application. Nodes are identified by associating them to structural sections in the pages of the application, displaying requiring information from/to the user. The client pages related to Entities, Associations, and Collections are selected and analyzed to: (i) identify which attributes of each Entities, Associations, or Collections are referred in the page; (ii) associate a Node to each group of attributes; (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 to.

3.2.3 UWA Publishing Model Abstraction

The UWA Publishing Model of the analyzed application is abstracted by identifying 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 an 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.

3.3 The Forward Design Phase

The activity diagram shown in Figure 2 depicts the model-driven forward design phase of our redesign approach.

Overall, this process takes as input the UWA model recovered with the reverse engineering phase described in Section 1 and, through an automatic transformation phase, produces a low level design model (MVC-JSF Model) that the developer can use to re-implement the application adopting an Model-View-Controller (MVC) architectural design pattern and the JavaServer Faces (JSF) technology.

Prior to proceed with the transformation phase the designer may decide to refine and extend the model in order to satisfy new requirements or to improve the design in response to user feedback. This refinement and evolution activity is supported by the UWA editor included in the UWAMDD tool platform described in Section 4.

The generated model is an instance of the MVC-JSF MOF metamodel that we have defined to represent a generic Web Application adopting the MVC pattern at an architectural level and the JavaServer Faces technology at the implementation level\(^1\). Being specific for the JSF implementation platform, the MVC-JSF model can be considered a Platform Specific Model (PSM) in the MDA jargon. This model has a dual purpose: first, it provides developers with precise specifications for implementing the Web Application using the JSF framework; moreover it enables to link the UWA conceptual models of the application to the code developed in order to implement it.

An example of MVC-JSF model is reported in Figure 7. The model is based on two diagrams: a Model Class Diagram (MCD) and a View Class Diagram (VCD), each purposefully defined to represent the Model and View/Controller components of the MVC design pattern. These diagrams are sufficiently detailed to be used by the development team.

In particular, the MCD defines the classes with methods and attributes (implemented by means of Javabeans in JSF) corresponding to the content types (UWA Entities) of the application; the VCD represents the presentation layer of the application and describes pages structure and content in terms of visual and interaction elements and the navigation between pages through the Controller component.

The automatic transformation of a UWA Conceptual model into the MVC-JSF design model is based on a set of well-defined mapping rules between the modeling primitives of the respective metamodels and the implementation of these rules using the Atlas Transformation Language (ATL). Broadly speaking, the UWA Information model maps to the MCD, while the UWA Navigation and Publishing Models merge into the VCD. Associations between visual and user interaction elements of the View pages with attributes and methods of the MCD, originate from the Navigation Model. Table 1 reports a synthesis of the defined mapping rules while Figure 3 reports the implementation of one of them with the ATL language. In particular, the reported rule reads as follows: for each Collection of the UWA model that has as member an Entity Type is generated in the MCD, originate from the Navigation Model. Table 1 reports a synthesis of the defined mapping rules while Figure 3 reports the implementation of one of them with the ATL language. In particular, the reported rule reads as follows: for each Collection of the UWA model that has as member an Entity Type is generated in the MCD, originate from the Navigation Model. Table 1 reports a synthesis of the defined mapping rules while Figure 3 reports the implementation of one of them with the ATL language. In particular, the reported rule reads as follows: for each Collection of the UWA model that has as member an Entity Type is generated in the MCD, originate from the Navigation Model. Table 1 reports a synthesis of the defined mapping rules while Figure 3 reports the implementation of one of them with the ATL language.

4. TOOL SUPPORT

Figure 4(a) shows the architecture of the REUWA Tool Platform supporting the recovery process. The Core Layer enables project integration by providing builders aware of UWA resources and a project nature enabling REUWA process workflow for Eclipse WTP projects. This layer also provides
development.
provides basic services for the entire tool platform: HTML/XML parsers, distance calculators between HTML documents and core platform services. The Process Layer implements the process logic using a workflow engine and following the RE-UWA process specified in Section 3.2. For each step of the process there is a component implementing it. The engine takes the process instance and transfer the control between the steps as specified in the process definition. Finally, the IDE Layer represents the presentation layer of the REUWA tool platform: it enables the user interaction and the execution of the reverse modeling process and providing five perspectives, each devoted to the recovery of a well defined portion of the UWA model. The layer is structured as a set of Eclipse editors and views that interact with the process engine and the software components of the underlying architecture layers. More details on the REUWA tool platform can be found in [1].

Figure 4(b) depicts the architecture of the UWAMDD Tool Platform supporting the model-driven design process described in section 3.3. At the presentation level (IDE Layer) the tool platform exposes three main components: the UWA Editor and the MVC-JSF Editor which enable the creation, import and editing of UWA models and MVC-JSF models, respectively; the ATL Transformer which transforms UWA models to MVC-JSF models. The two graphical editors have been built using the Eclipse Modeling Framework (EMF) and the Graphical Modeling Framework (GMF). In particular, the EMF framework and the Ecore language have been used to define both the UWA and the MVC-JSF meta-models, while the GMF framework has been used to develop the base code of the two graphical editors. A screenshot of the two editors is provided by Figure 6 and Figure 7.

The transformation of a UWA model (an instance of the UWA MOF metamodel) into the corresponding MVC-JSF model (an instance of the MVC-JSF metamodel) is based on a Transformation Metamodel consisting of a set of transformation rules written in the ATLAS Transformation Language (ATL) and implemented by means of the ATLAS Development Tools (ADT).

Table 2 reports the UWA Entities identified by the approach, for the best achieved clustering threshold. For three Entities there was no threshold capable of identifying them, and hence they required manual identification. This was due to the lack of keywords in the pages containing the Entities: in these cases templates were empty (and hence discarded in the previous step) or contained invalid keywords and hence originated groups discarded later during validation.

### Table 1: Mappings between UWA and UML-MVC modeling concepts.

<table>
<thead>
<tr>
<th>UWA Conceptual Model</th>
<th>UWA modeling Concept</th>
<th>MVC Component</th>
<th>UML-MVC Modeling Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>Model</td>
<td>Class into MCD</td>
<td></td>
</tr>
<tr>
<td>Slot</td>
<td>Model</td>
<td>Private attribute and associated set and get methods into MCD</td>
<td></td>
</tr>
<tr>
<td>Semantic Association</td>
<td>Model</td>
<td>Association between Classes into MCD</td>
<td></td>
</tr>
<tr>
<td>Center</td>
<td>Model</td>
<td>Private Attribute and associated methods into MCD</td>
<td></td>
</tr>
<tr>
<td>Collection Type</td>
<td>Model</td>
<td>Class into MCD associated to the Collection Center</td>
<td></td>
</tr>
<tr>
<td>Navigation Node</td>
<td>View</td>
<td>Class into VCD</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2: Matrix of the identified UWA Entities and Semantic Associations.

<table>
<thead>
<tr>
<th>Semantic Associations</th>
<th>TV Guide</th>
<th>Director</th>
<th>Biography</th>
<th>Review</th>
<th>News</th>
<th>Curiosity</th>
<th>Quiz</th>
<th>Trailer</th>
<th>Filmography</th>
<th>Character</th>
<th>Opinion</th>
<th>Poster</th>
<th>SoundTrack</th>
<th>Film</th>
<th>PhotoGallery</th>
<th>Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>FimUp</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. CASE STUDY

In order to validate the approach we have applied it to redesign a real world website. The selected application is FimUp, a movie information portal, reachable at http://www.filmup.it, which provides films’ reviews and multimedia content (like trailers or photo galleries).

5.1 The Reverse Engineering Phase

An instance of the web application was mirrored and around 13000 pages were selected. A preliminary similarity analysis was performed on these pages and 4500 of them were selected to cover the different sections of the web site, with an average of 200 pages for section. These pages were analyzed using clone analysis and a clustering techniques.

The reverse engineering phase was executed at increasing clustering thresholds to obtain best possible results. For each threshold value, precision and recall were calculated.

Clusters containing only one page or useless pages were discarded before continuing to 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 [1]. These groups were then validated by the human expert using the REUWA tool perspective that allows discarding invalid groups and refining the valid ones (by means of merge, split and modify operations).

The computed groups represent the candidate entities; the abstraction algorithm defined in [6] is used to recover the actual Entities.

Table 2 reports the UWA Entities identified by the approach, for the best achieved clustering threshold. For three Entities there was no threshold capable of identifying them, and hence they required manual identification. This was due to the lack of keywords in the pages containing the Entities: in these cases templates were empty (and hence discarded in the previous step) or contained invalid keywords and hence originated groups discarded later during validation.
After UWA Entities abstraction, UWA Semantic Association are identified among them. This step produced the 56 Semantic Associations showed in the matrix reported in Table 2 (of these 3 were discarded during validation since were originated by invalid links and are reported in red).

Using information on UWA Entities, UWA Semantic Association and the clustered pages, the UWA Navigational nodes and their clusters (Structural, Association and Collection) are recovered. An Association Cluster is recovered for each Semantic Association reported in 2.

For each UWA Entity a UWA Structural Cluster is generated (looking at the nodes generated from clusters containing that UWA Entity). The same is done for Collection clusters.

From the navigational clusters the publishing model is recovered as highlighted in Section 3.2, by generating a page for each structural cluster. For each page, a section is defined for each entity component and for both associated entities and collections.

Figure 6 shows the UWA Editor during an refinement session on a small excerpt of the entire recovered model. On top of the figure, three UWA Entities (Film, Actor and Trailer) are shown along with their internal components and slots. On the left, just below entities, is showed a UWA Semantic Association between the Entity Film and the Entity Trailer along with the Association Center. On the right is showed the collection Trailers modeling the list of incoming trailers. At the center of the diagram are showed two clusters: (i) the one on the left is the Association Cluster related to the Semantic Association between Film and Trailers. The internal node labelled Film-Trailer defines the selection of slots to be used as preview information in the navigation from a Film to its Trailer; (ii) the one on the right shows the collection cluster for the incoming trailers associated to the UWA Collection reported above.

5.1.1 Validation of results

In order to validate the results, we calculated precision and recall for each clustering threshold. 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 entities and the total number of the entities defined in the application.

Figure 5 shows the precision and recall for all the different clustering thresholds.

As we can see, with a threshold of 0.08 only 2 entities out of 17 were correctly identified (with an acceptable precision, around 0.75). Decreasing the threshold to 0.05, 14 entities were identified giving a recall of 0.82 but with a lower but still acceptable precision (p=0.55; these are the best values obtained). Continuing to decrease the threshold did not allow the identification of more entities since the remaining three were in pages in which no keywords are used and hence a completely manual identification is needed.

Due to space restriction we only report precision and recall for Entities. However, we have obtained comparable results also for Semantic Associations, Collections and for Navigation and Publishing models.

5.2 FilmUp ReDesign Phase
Figure 7 shows a portion of the final MVC-JSF model automatically generated for the Filmup.it application by following the process described in Section 2 with the support of the UWAMDD tool platform described in Section 4. The generated MCD in Figure presents the UML classes Actor, Film, and Trailer derived from each Entity of the UWA model and preceded by the IME (Information Model Entity) prefix with associations between them, each derived from a UWA Semantic Association. Particularly at the implementation level the three UML classes will result in three different Java Beans with own attributes and methods derived from the UWA Slots of the corresponding UWA Entity. The prefixes (IME, IMAC, etc.) are inherent to the mapping between the UWA conceptual model and the MVC-JSF logical model. If the MCD defines the informative content of a Web application and the Bean structure, the VCD, shown below the MCD in Figure, defines the content and structure of pages and the navigation between them. Each page of the Web application such as PMP_Film P, derived from a Page element of the UWA Publishing Model and preceded by the PMP (Publishing Model Page) prefix, is represented with a UML class, labeled with the stereotype <<view>>, which aggregates other classes such as NMI_Film, NMI_Film-Trailer and so on, that represent part of pages, identified by the stereotype <<subview>>. Each subview, preceded by the NMI (Navigation Model from Information model) prefix, is derived from a UWA Publishing Unit element and inherits the content from a Node Type element of the UWA Navigation Model. View and subview contain information about the elements of the Web page.
used in output, input and interaction elements such as links, buttons, etc. These items, identified by the \(<\text{coText}\>\), \(<\text{coList}\>\), \(<\text{cText}\>\) and \(<\text{link}\>\) stereotypes, relate directly to the attributes and methods of the Bean in the MCD. In this way the Controller component of the MVC architecture is embedded in the diagram VCD.

5.3 Discussion and Limits of the Approach

By running the complete process on FilmUp web application we were able, with reduced effort if compared to designing from scratch, to recover key data model abstractions and navigational paths and finally to generate a complete new MVC-based design for the analyzed application. The extracted knowledge can be used as is or evolved in order to extend the kind of domain objects and the navigational paths that are served to users. To assess the effectiveness of the approach, we’ve experimented to add two new collections to the web site: the most visited films and the last reviewed films (both were not available in the original web application). It has been just a matter of extending the UWA recovered model in order to satisfy the new requirements. The design models were then regenerated automatically by the transformation engine with no additional (and manual) effort needed. Another interesting point is on technological side. While actually only a single MVC-based design targeting JSF platform has been defined, more PSM can be defined in order to migrate the application from a platform to another with very reduced effort.

The validation of the approach has also highlighted some limits related to both reverse and forward processes. Concerning the reverse process, a limit is related to the identification of only a single component for UWA Entities; this increases the effort of the recovered model refinement step. Moreover the impossibility to identify entities in an automatic way, when keywords are not present, increases the effort required to perform the reverse engineering step requesting manual intervention. The forward meta-model also need further improvement since it have been defined with respect to a single PSM targeting JSF. We expect it to evolve in order to cover a much wider spectrum of possible design methodologies and technologies in order to make the migration of existing UWA-based applications easier.

6. CONCLUSIONS

This paper presents an approach and a supporting tool platform for the semi-automatic model-driven redesign of existing web applications by adopting the Ubiquitous Web Applications (UWA) design framework. The approach relies on a reverse engineering phase that exploits clustering and clone detection techniques to abstract UWA conceptual models from the existing application by analyzing the client side HTML pages. This phase is supported by the ReUWA tool platform and produces a model which is compliant with the UWA MOF metamodel. The recovered model is used to produce a lower level design model to re-implement the application adopting a MVC architectural pattern. This phase is supported by the UWAMDD tool platform and uses ATL transformations to represent MVC web applications. The main and new contribute of the proposed approach is that it, by means of the semi-automated reverse engineering process, allows redesign existing web applications in a model-driven style, while reordering their embedded knowledge (the “as-is” conceptual model), with an effective automatic support provided by the two integrated REUWA and UWAMDD tools.

Future works will be surely devoted to improve the MVC-based model by supporting more technologies and target platforms. Another interesting evolution of our approach is to take into account a set of common web application concerns like persistence, access control, customizability and business logic. Experiments will be carried out to keep such concerns separated both in the modeling and in the code generation steps by applying MDD in the context of Aspect Oriented Software Development.

7. REFERENCES

[14] L. Paganelli and F. Paterino. Automatic reconstruction of the underlying interaction design of web
Figure 7: An excerpt of the UWA MVC-JSF Model generated from recovered UWA Model for FilmUp.it applications. In *SEKE ’02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 439–445, New York, NY, USA, 2002. ACM.


