Model Driven Evolution of Web Applications

Mario Luca Bernardi[1], Giuseppe Antonio Di Lucca[2], Damiano Distante[3], Marta Cimitile[4]
[1] [2] Department of Engineering - RCOST, University of Sannio, Italy
[3] Faculty of Economics, Unitelma Sapienza University, Italy
[4]Faculty of Jurisprudence, Unitelma Sapienza University, Italy
[1]mlbernar@unisannio.it, [2]dilucca@unisannio.it, [3]damiano.distante@unitelma.it, [4]marta.cimitile@unitelma.it

***Abstract*—Reverse engineering is usually used to recover missing and up to date models of a software system to support its comprehension when changes are required to maintain or evolve it. Model driven engineering approaches have been recently proposed to develop more quickly web applications with a high design quality and maintainability. Integrating reverse engineering techniques with model driven web engineering methods originates evolution approaches that would reduce the evolution effort while improving the quality of the modified web application. Such an evolution process exploits the models recovered by reverse engineering as the inputs of a model driven web engineering approach to design and implement the modified/evolved version of the application. This paper describes a general process for the model driven evolution of web applications, suitable for any model driven web engineering method. An instance of such a process tailored for the Ubiquitous Web Applications (UWA) design methodology is also briefly summarized.**

Keywords: Web systems evolution, Web application reengineering, Reverse engineering, Model driven Web engineering

I. Introduction

Existing web applications (WAs) are subject to continuous maintenance and evolution. Sometimes the required modifications are driven by the need/opportunity to improve aspects characterizing the application and influencing its external quality, such as its usability. More often, maintenance and evolution is required to correct bugs or to introduce new functionalities. Aspects characterizing a WA from a user point of view include content, navigation, presentation, and behaviour (e.g., the behaviour associated to the implemented business processes) [11]. Changes such as the addition of new functionalities, new content types, new navigation structures, new user interfaces, or the modification of existing ones in order to enhance them and improve the user's experience can have a large impact on the conceptual design of the application. In other cases the required modifications do not need to alter the existing functional behavior of the WA, nor the way contents and services are presented to the user, but just the technological structure used for the application functioning. In these cases, the modifications entail mainly the implementation layer of the application, rather than its conceptual design model. In all cases, the WA has to be effectively analyzed and represented by means of appropriate models. Thus, the availability of up-to-date instances of conceptual and design models has a key role to successfully evolve a WA. Unfortunately, due to the limited budget and short time-to-market that often constrain the process of development and maintenance of WAs, such documentation is often lacking. This causes maintenance and evolution to become difficult and risky activities that can compromise the effectiveness and correctness of the whole system.

The usage of reverse engineering techniques and tools to recover such models is as useful as required. As a consequence, several reverse engineering approaches have been defined to recover models of an existing WA. The list of such approaches includes methods to recover architectural views of a WA, static, dynamic and behavioural models [15][9][17][21], functional requirements models [14], domain conceptual models [5], business process models [8][10], presentation models [20], just to mention a few among the most remarkable ones.

In the last years model driven approaches have been adopted to develop WAs [6][18][13][4] with the aim of increasing productivity (by reusing standardized models), simplify the design process (by using recurring design patterns), and promoting communication between individuals and teams working on the system. Studies have also been conducted to measure the effort reduction in adopting model driven development approaches [12].

Integrating reverse engineering and model driven web engineering can lead to model driven evolution approaches for WAs that would reduce the evolution effort while improving the quality of the modified application.

In such an approach, the models recovered by reverse engineering are used as the starting point of a model driven forward engineering phase to implement the required evolution changes. Overall, the evolution approach would lead to the implementation of an evolved version of the WA, or a (fully) new implementation of it [7], with reduced effort and higher design quality.

Based on authors' past research, this paper describes a general process for the evolution of existing WAs that integrates two main phases. In the first phase up-to-date models representing the design "as-is" of the application are recovered by reverse engineering; in the second phase model driven forward engineering is used to modify the recovered "as-is" design according to the new evolution requirements and to generate the evolved version of the application. An instance of the proposed process specialized for the Ubiquitous Web Applications (UWA) design methodology [19] is briefly described.

The paper is organized as follows. Section II describes the general integrated evolution process; section III presents an instance of this process, based on UWA models. Finally, some conclusive remarks are provided in Section IV.

II. The Model Driven Evolution Process

The Evolution process consists of two integrated main phases: a reverse engineering phase, and a model-driven engineering phase. In the first phase conceptual/design models are recovered by reverse engineering from the WA to evolve. The second phase takes as input the recovered models and enables refining and evolving them (to meet new/changed requirements) by a model driven engineering approach. Transformation engines and generators are applied to the recovered models to produce artifacts, at a lower level of abstraction and a higher level of detail, and source code, at the final stage.

Methodological, technical, and tool support has to be provided for both the two phases which usually 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 discuss the main steps of the phases composing the integrated evolution process.

A. The Reverse Engineering Phase

The Reverse Engineering phase includes the following main steps:

- Selection/definition of the models to be recovered
- Selection/identification of tools able to recover the selected models
- Analysis/comprehension of the recovered models

The selection of the models to be recovered can be addressed by the type of the improvements required, which give rise the evolution of the WA. The types of models can be the ones that were used along the development (or previous maintenance/evolution interventions) of the WA, or new ones according to the new methods/techniques developers are going to use for implementing the new version of the WA. Usually, the former case is when the existing documentation of the WA is no longer reliable and consistent with its actual implementation (e.g., the models were not updated when the WA has been maintained/modified) or no model was produced along the WA development/maintenance. The latter case is when developers are going to adopt a new methodology to develop (or re-develop/re-engineer) a WA; thus it is useful to recover models of the WA, making up the organization portfolio, according to the ones used/defined by the new methodology.

After models have been selected, reverse engineering tools to perform the recovering activity have to be identified. According to the recovering requirements to meet, a tool will be implemented or existing commercial/free tools can be used.

So, the selection of which models are to be recovered could be affected by the available existing reverse engineering tools, as well as if the tool is free or not.

The recovered models are then analysed to comprehend the WA, mainly with respect to the features to change.

Usually, the models to be recovered are the ones depicting the design (both at architectural and detailed level) of the WA by the several modules/components implementing it (such as the WA's pages), as well as the models depicting the WA at a more abstract level, such as:

- a domain model, also called information model, describing the concepts of the main domain objects the WA deals with;
- a navigational model, describing the navigation available to a user across the WA components;
- a publishing model, describing what and how 'things' are presented to users.

As we said, several reverse engineering approaches have been developed to recover these types of models.

B. The Model Driven Phase

The models recovered along the reverse engineering phase provide a representation of the design "as-is" of the analyzed application. By examining the recovered models and by considering the (eventually new) requirements available for the application, as well as the design guidelines provided by the (new) design methodology to use in the evolution phase, the analyst can:

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

The evolution operations can be based on a Model Driven web engineering approach (such as the one proposed in [7]) where the recovered models are used as a starting point to modify/re-design the application.

In Model Driven approach every artifact, including source code, is a model element, and the overall development process can be seen as a chain of transformations from one model to the next one enabling the automated implementation of a system starting from its requirements. Models are formalized in a way to be generative by means of meta-models, and each transformation takes models as input and produces some other models. The high degree of automation allowed by model driven approaches makes more efficient the entire software life-cycle, and makes higher the overall quality of the resulting software products. Thus, applying model driven engineering to the evolution of WAs would lead to a reduced effort, reduced delivered time, higher software quality with respect to traditional approaches.

The main components of a Model Driven Evolution approach are: (*i*) a modeling language definition standard, such as the Object Management Group's (OMG) Meta Object Facility (MOF) [16], to represent formally standardized concepts, and (*ii*) a way to model transformation technology (e.g., like Atlas transformations - ATL or Query View Transformation - QVT [16]) to generate output models starting from the input models. This requires the selection of a modeling language (e.g., MOF) to define a meta-model to describe the WA according to the desired models, as well as a language to

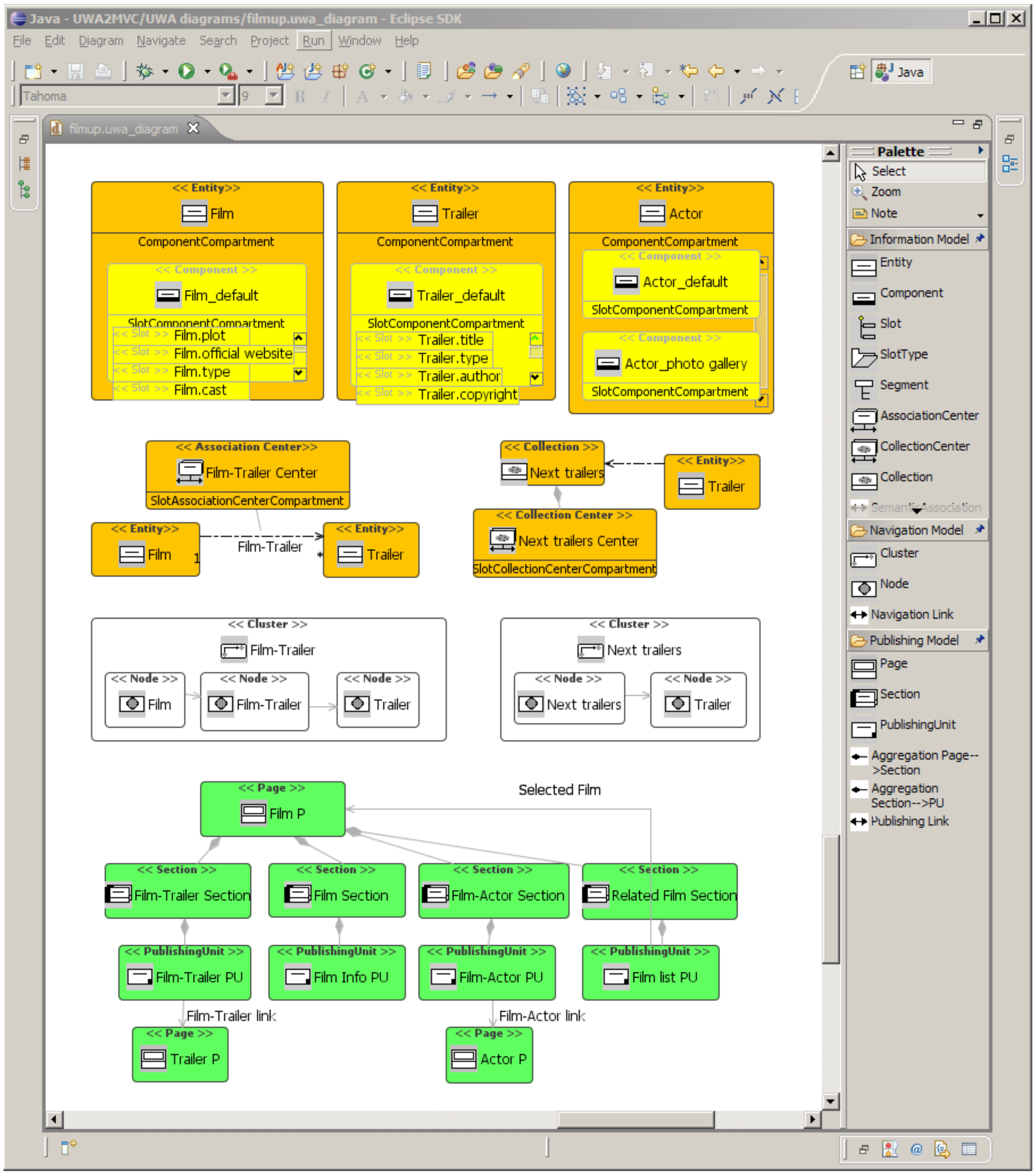

Fig. 1. Screenshot of the UWA graphical editor: recovered UWA domain models

define a set of transformation rules to be applied to the input models to automatically get the output models/artifacts. The availability of any existing modeling and transformation technology/language may affect both the selection of the models to recover and the target WA design.

In the next section we shortly describe a specialization of this general process tailored on the UWA models. It is based on our past research and experience about the recovery of models from existing WAs and model driven process for the fast prototyping of WAs [1][2][3].

III. The UWA Model Driven Evolution Process

The general proposed Model Driven Evolution approach has been specialized on the UWA models. The specialized process integrates the RE-UWA reverse engineering process proposed in [1][5] and the MDE process described in [7]. MOF-based UWA models are generated starting from the UWA models recovered from existing (legacy) WAs. A low-level MVC-based design (implementing the same requirements, domain model and navigational structure of the original legacy application) is then derived from them. It can be furtherly

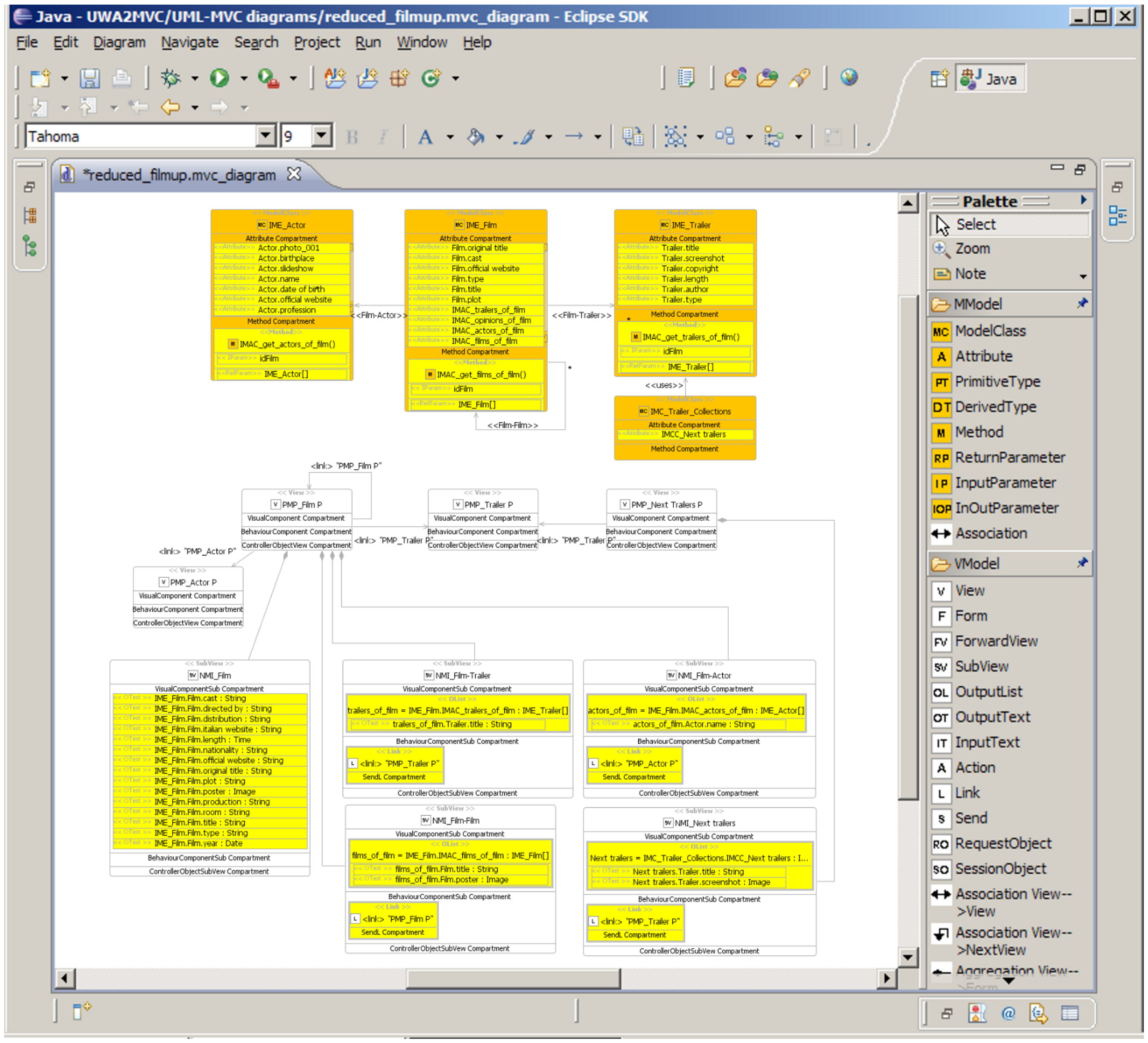

Fig. 2. Screenshot of UWA-MDD tool's graphical editor: generated MVC-JSF design models

and easier revised and modified to meet new requests, such as the migration towards new platforms and programming languages. The process is supported by a tool automatizing the process activities. It integrates the RE-UWA tool, to recover the UWA models, and the UWA-MDD tool, supporting the Model Driven phase [2][1]. The tool includes also graphical editors to visualize and modify the recovered and derived models.

A. The UWA Reverse Engineering phase

The existing WA is analyzed to recover the UWA Information Model, UWA Navigation Model, and the UWA Publishing Model. In the following the main actions to perform the recovering of these UWA models are summarized (details are in [1][5]).

1) UWA Information Model Recovering: This model is recovered by analyzing the client pages of the application to abstract UWA Entities, Semantic Association and Collections. The identification of UWA Entities is carried out by searching for groups of related attributes (i.e. *keywords*) in the client-side HTML pages (both static and dynamically generated) of the WA. A group of keywords involved in the same user input or output operation or included in the same HTML form or output report is considered as a possible group of Slots characterizing a UWA Entity. Keywords are identified also analysing sets of cloned client pages (i.e. a group of client pages characterized by the same HTML control structure but different content) 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.. Each identified keyword is candidate to be a UWA Slot and the keywords in a group are candidate to be an Entity Component. A validation phase is manually carried over the automatically identified groups of keywords to finally obtain validated sets of Entities.

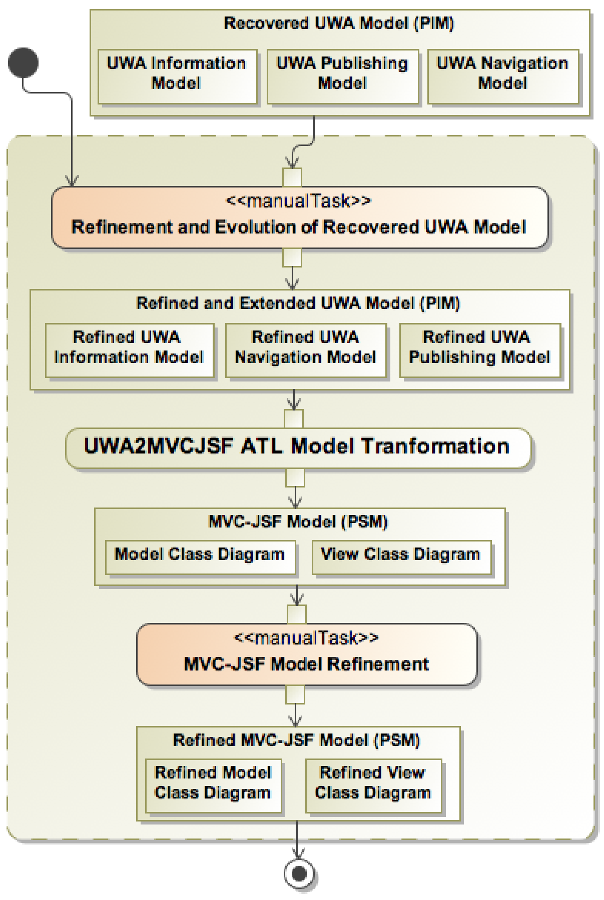
Recovered UWA Model (PIM)
UWA Information Model
UWA Publishing Model
UWA Navigation Model
<<manualTask>>
Refinement and Evolution of Recovered UWA Model
Refined and Extended UWA Model (PIM)
Refined UWA Information Model
Refined UWA Navigation Model
Refined UWA Publishing Model
UWA2MVCJSF ATL Model Tranformation
MVC-JSF Model (PSM)
Model Class Diagram
View Class Diagram
<<manualTask>>
MVC-JSF Model Refinement
Refined MVC-JSF Model (PSM)
Refined Model Class Diagram
Refined View Class Diagram

Fig. 3. The UWA-MDD process activity diagram

A candidate Semantic Association is assumed to exist between pairs of Entities having some Slots in common as well as when different Entities are shown in the same HTML page. Semantic Associations are also derived from hyperlinks connecting pages showing different Entities. Similarly to candidate Entities, candidate Associations automatically found in this step will 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, such as: (*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.

2) UWA Navigation Model Recovering: The recovery of the UWA Navigation Model is carried out by identifying Nodes and Clusters for the analyzed application. 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. Each identified Node and each Cluster is assigned a unique name derived from the elements of the Information Model they are associated to.

3) UWA Publishing Model Recovering: 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 (each template grouping a set of cloned client pages) obtained during the phase of Information Model recovery.

Figure 1 shows the UWA Editor reporting an excerpt of UWA models recovered from a WA implementing a movie information portal, while Figure 2 shows the editor visualizing the MVC-JSF model derived from the recovered UWA models.

At the top of the figure 1, three UWA Entities (Film, Actor and Trailer) are shown along with their internal components and slots. On the left, just below the Entities, a UWA Semantic Association (between the Entity Film and the Entity Trailer) is shown, along with the Association Center. The Trailers Collection models the list of incoming trailers. At the center two Clusters are shown: on the left is the Association Cluster related to the the Semantic Association between Film and Trailers, while the one on the right shows the collection cluster for the incoming trailers associated to a UWA Collection. In the bottom of the figure the structure of a client page is depicted, by the sections and publishing units composing it.

B. The UWA Model Driven phase

The activity diagram shown in Figure 3 depicts the main actions of this phase.

The recovered UWA models are the inputs of the phase that, through an automatic transformation, has in charge to produce a low level design model (i.e. a MVC-JSF Model) that the developer can use to re-engineer the application adopting a Model-View-Controller (MVC) architectural design pattern and the JavaServer Faces (JSF) technology. Prior to proceed with the transformation step, the developer may decide to refine and extend, by using the UWA editor, the recovered models in order to satisfy new requirements or to improve the design in response to user feedback. The generated model is an instance of the MVC-JSF MOF metamodel defined to represent a generic WA adopting the MVC pattern at the architectural level and the JavaServer Faces technology at the implementation level.

An excerpt of the MVC-JSF model derived from the UWA models recovered in the reverse engineering phase is reported in Figure 2. The model is based on two diagrams: a Model Class Diagram (MCD) and a View Class Diagram (VCD), each purposely defined to represent the Model and View/-Controller components of the MVC design pattern. These diagrams have enough details to start an adequate evolution of a WA. 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 [2].

IV. Conclusions

A general process integrating reverse engineering and model driven engineering techniques for the evolution of WAs has been described. Conceptual and design models, according to a web engineering methodology, are recovered from a WA by analyzing the source code of the existing application. The recovered models are then used to identify weaknesses and propose solutions to overcome them, as well as to meet new requirements using a model driven web engineering method. The general process has been used to define and develop a process specifically tailored for the user-centered design methodology Ubiquitous Web Applications (UWA), to evolve existing WAs. The main advantages by using the model driven evolution process are: a reduction of the evolution effort, a higher quality of the resulting design, a higher re-usability of the evolved WA models and components. Of course, any kind of WA model can be selected to tailor the general evolutionary process. Future work will be devoted to extend the tailoring of the general process (and of supporting tools) both to other types of models that can be recovered (not only UWA models, but also the ones defined by other web engineering methods), and to other target design models (not only MVC) and target platforms (e.g., mobile platforms).

References

[1] M. Bernardi, M. Cimitile, and D. Distante. Web applications design recovery and evolution with RE-UWA. *Journal of Software: Evolution and Process*, 25(8), 2013.

[2] M. Bernardi, M. Cimitile, D. Distante, and F. Mazzone. Web applications design evolution with uwa. In *Proceedings of the 12th IEEE International Symposium on Web Systems Evolution (WSE 2010)*, pages 3–12. IEEE Computer Society, 2010.

[3] M. Bernardi, G. Di Lucca, and D. Distante. A model-driven approach for the fast prototyping of web applications. In *Proceedings of the 13th IEEE International Symposium on Web Systems Evolution (WSE 2011)*, pages 65–74. IEEE Computer Society, 2011.

[4] M. L. Bernardi, M. Cimitile, G. A. Di Lucca, and F. M. Maggi. M3d: a tool for the model driven development of web applications. In *Proceedings of the twelfth international workshop on Web information and data management*, WIDM '12, pages 73–80, New York, NY, USA, 2012. ACM.

[5] M. L. Bernardi, G. A. D. Lucca, and D. Distante. The RE-UWA approach to recover user centered conceptual models from web applications. *International Journal on Software Tools for Technology Transfer*, 11(6):485–501, 2009.

[6] J. Bezivin. Model driven engineering: An emerging technical space. In R. Lmmel, J. Saraiva, and J. Visser, editors, *Generative and Transformational Techniques in Software Engineering*, volume 4143 of *Lecture Notes in Computer Science*, pages 36–64. Springer Berlin / Heidelberg, 2006. 10.1007/11877028_2.

[7] D. Distante, P. Pedone, G. Rossi, and G. Canfora. Model-driven development of web applications with uwa, mvc and javaserver faces. In *Proceedings of the 7th International Conference on Web Engineering (ICWE2007)*, pages 457–472, Como, Italy, 2007. Springer Berlin / Heidelberg.

[8] D. Distante, G. Rossi, G. Canfora, and S. Tilley. A comprehensive design model for integrating business processes in web applications. *International Journal on Web Engineering and Technology*, 3(1):43–72, 2007.

[9] F. Estivenart, A. Francois, J. Henrard, and J.-L. Hainaut. A tool-supported method to extract data and schema from web sites. In *WSE '03: Proceedings of the IEEE International Workshop on Web Site Evolution (WSE)*, page 3, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[10] C. D. Francescomarino, A. Marchetto, and P. Tonella. Reverse engineering of business processes exposed as web applications. In *European Conference on Software Maintenance and Reengineering*, volume 0, pages 139–148, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[11] G. Kappel, B. Pryyll, S. Reich, and W. Retschitzegger. *Web Engineering: The Discipline of Systematic Development of Web Applications*. John Wiley and Sons, 2006.

[12] N. Koch, A. Knapp, and S. Kozuruba. Assessment of effort reduction due to model-to-model transformations in the web domain. In *Proceedings of the 12th International Conference on Web Engineering (ICWE 2012)*, volume 7387 of *Lecture Notes in Computer Science*, pages 215–222. Springer, 2012.

[13] C. Kroiss, N. Koch, and A. Knapp. Uwe4jsf: A model-driven generation approach for web applications. In *ICWE '9: Proceedings of the 9th International Conference on Web Engineering*, pages 493–496, Berlin, Heidelberg, 2009. Springer-Verlag.

[14] G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, and U. D. Carlini. Comprehending web applications by a clustering based approach. In *IWPC '02: Proceedings of the 10th International Workshop on Program Comprehension*, page 261, Washington, DC, USA, 2002. IEEE Computer Society.

[15] G. A. Di Lucca, A. R. Fasolino, and P. Tramontana. Reverse engineering web applications: the WARE approach. In *Journal of Software Maintenance and Evolution*, volume 16, pages 71–101, New York, NY, USA, 2004. John Wiley & Sons, Inc.

[16] OMG. Object management group (mof,mda,xmi,qvt,uml) - http://www.omg.org/.

[17] F. Ricca and P. Tonella. Understanding and restructuring web sites with reweb. In *IEEE MultiMedia*, volume 8, pages 40–51, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[18] D. C. Schmidt. Model-driven engineering. *Computer*, 39:25–31, 2006.

[19] UWA Project Consortium. Ubiquitous web applications. In *eBusiness and eWork Conference 2002*, 2002.

[20] J. Vanderdonckt, L. Bouillon, and N. Souchon. Flexible reverse engineering of web pages with vaquista. In *WCRE '01: Proceedings of the Eighth Working Conference on Reverse Engineering (WCRE'01)*, page 241, Washington, DC, USA, 2001. IEEE Computer Society.

[21] S. Weijun, L. Shixian, and L. Xianming. An approach for reverse engineering of web applications. In *Proceedings of the International Symposium on Information Science and Engineering (ISISE '08)*, volume 2, pages 98–102, Los Alamitos, CA, USA, 2008. IEEE Computer Society.