Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces

Damiano Distante[1], Paola Pedone[2], Gustavo Rossi[3], and Gerardo Canfora[4]

[1,4] Research Centre on Software Technology (RCOST), University of Sannio, Italy
{canfora,distante}@unisannio.it
[2] Faculty of Engineering, University of Salento, Italy
paola.pedone@unile.it
[3] LIFIA, Facultad de Informática, UNLP, La Plata, Argentina
gustavo@lifia.info.unlp.edu.ar

Abstract. This paper presents a model-driven approach to the development of web applications based on the Ubiquitous Web Application (UWA) design framework, the Model-View-Controller (MVC) architectural pattern and the JavaServer Faces technology. The approach combines a complete and robust methodology for the user-centered conceptual design of web applications with the MVC metaphor, which improves separation of business logic and data presentation. The proposed approach, by carrying the advantages of Model-Driven Development (MDD) and user-centered design, produces Web applications which are of high quality from the user's point of view and easier to maintain and evolve.

Keywords: Web Engineering, Model-Driven Development, Model Transformation, Model Driven Architecture, UWA, UML, MVC, JavaServer Faces.

1 Introduction

Web applications design methodologies available in the literature can be classified into those which focus on "what" the application is required to do (conceptual design in the problem domain) and those which focus on "how" the application can satisfy its requirements and implement the "what" (logical design in the domain of solutions). UWA [38], OOHDM [36], OOWS [32] and OO-H [9] pertain to the first category; Conallen's proposal [13] belongs to the second one; UWE [19][22] and WebML [10][11][12] can be considered hybrid methodologies, as they cover both conceptual and logical design.

Conceptual design methodologies abstract from implementation details and offer an overall view of the system from the view point of the users. Conceptual modeling is the right starting point to implement complex systems. However, the big distance between the conceptual model of a web application and its implementation makes the use of conceptual design methodologies insufficient for the development of a web application. If intermediate design levels for translating conceptual specification into implementation design are not provided, then the implementation activities of a web application may proceed independently from conceptual design, which will thus be under-exploited or even a waste of effort.

L. Baresi, P. Fraternali, and G.-J. Houben (Eds.): ICWE 2007, LNCS 4607, pp. 457–472, 2007.
© Springer-Verlag Berlin Heidelberg 2007

The trend of well-known design methodologies to evolve towards Model-Driven Development (MDD) shows the need for integrated approaches that support the whole web application lifecycle. Methodologies that have recently moved in this direction include OO-H [3], OOWS [3], UWE [18][20] and OOHDM [35], discussed more in detail in the related work section.

This paper presents a model-driven approach to developing web applications based on the Ubiquitous Web Application (UWA) conceptual design methodology and the Model-View-Controller architectural design pattern. Compared to others, the UWA design framework, with its methodology and models, is particularly suited for designing web applications which are intended to be ubiquitous (accessible by different user types in different usage contexts and with different goals) and user-centered. By combining the characteristics of UWA with the advantages of the MVC architecture and the MDD paradigm, the resulting approach is particularly suited for developing ubiquitous web applications in a user-centered perspective as well as for supporting their maintenance and evolution.

UWA models are made at the right abstraction level to be used with application stakeholders and, at same time, can be detailed enough (design in the small) to provide useful information to the application developers. The methodology is also supported by a number of tools for drawing up UWA models and generating application design documentation. Nevertheless, since UWA models are independent from implementation details, they do not specify aspects such as the architecture, the software components, nor the database that the developer should implement to realize the designed application. By creating an intermediate design model to represent the application being implemented and a set of heuristics to map the UWA conceptual models onto it, we lay the basis for a model-driven approach to developing UWA-based web applications.

The intermediate design model (referred to as the logical model in the following) is based on standard UML diagrams and adopts the Model-View-Controller (MVC) architectural design pattern [8].

The main contributions of the paper are:

1. The definition of a logical model that describes the application code to be developed and maps it to the application requirements;
2. The definition of a set of heuristics to translate UWA conceptual models to the new logical model;
3. The definition of an MDD approach based on UWA to support the entire web application lifecycle.

The remainder of the paper is organized as follows. The next section discusses the background and motivations of the work. Section 3 presents the proposed UWA-based MDD approach and an example of its application. Related work is presented in Section 4 while Section 5 concludes the paper and provides an overview of future work.

2 Background and Motivations

2.1 A General Framework for the Development of Web Applications

Despite the differences between the various methodologies for engineering web applications found in the literature, a set of common concerns and development steps

can be identified [37]. Fig. 1 graphically represents these common features. Basically any design methodology models web applications at three construction levels (content, navigation and presentation) and addresses a number of aspects spanning from structure to behavior. Aspects are orthogonal to all construction levels. Structure applies to content (how contents are structured) as well as to navigation (hypertext structures) and presentation (e.g., page organization). Similarly, behavior (e.g., business rules and user operations) can be associated with content, navigation and presentation.

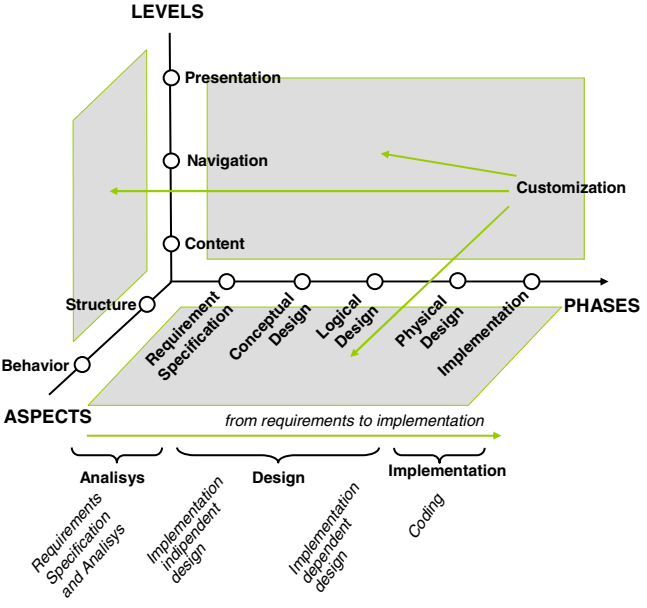


Fig. 1. A general development framework for web applications

As represented in Fig. 1, the development process starts with requirement specification and analysis and proceeds through a number of design steps to the implementation phase. Design steps include:

1. Conceptual design, focused on describing the problem domain and what the system is expected to do, independently of any technological detail.
2. Logical design, focused on the operation of the system while hiding implementation details specific to a particular platform.
3. Physical design, which adapts the logical model of the application to obtain detailed specifications for implementation with the chosen platform.

2.2 Model-Driven Development

Though the use of design methodologies is not yet a common practice in the field of web engineering, it is known that a model-based approach provides a better alternative to the ad-hoc development of web applications and its inherent problems

[37]. The need for systematic approaches to be adopted when developing complex systems and for design prior to implementation is now widely accepted.

Model-Driven Development (MDD) [5] sees software development as a process whereby a high-level abstract model is successively translated into increasingly more detailed models, in such a way that eventually one of the models can be directly executed by some platform [3]. The MDD approach not only advocates the use of models (such as those resulting from the design steps described in the previous section) for the development of software, but also emphasizes the need for transformations in all phases of development, from system specification to implementation and testing [23]. Transformations from one model to the next create a chain which enables the automated implementation of a system starting from requirements.

The possibly most well-known initiative of MDD is the Model Driven Architecture (MDA) proposal [25] defined by the Object Management Group (OMG). The central idea of MDA is to separate platform-independent design from platform-specific implementation of applications, delaying the dependence on specific technologies for as long as possible. Therefore, MDA advocates the construction of platform-independent models (PIM) and the support of model transformations. The model that is directly executed by a platform (PSM) which satisfies all requirements, including non-functional ones, is also called "code", and is usually the last model in the refinement chain.

The development of web applications is a specific domain in which MDD can be successfully applied, due to the web-specific separation of concerns: content, navigation, presentation and customization.

2.3 UWA

The Ubiquitous Web Applications design framework (UWA) [38] provides a methodology and a set of models and tools for user-centered conceptual design of ubiquitous web applications. Mapping UWA models onto the general web application development framework depicted in Fig. 1, produces the diagram shown in Fig. 2. UWA covers the phases of requirement specification and analysis and conceptual design. Requirement elicitation is the activity devoted by UWA for the identification of the application's stakeholders, their goals and, through a refinement process, the resulting requirements for the following design of the application. Requirements are classified into content, structure of content, access to content, presentation, and behavior (system and user operations).

Following the requirement elicitation phase, the conceptual user-centered design of the application is developed. The UWA Information Model, Navigation Model and Publishing Model cover the three levels of design represented on the vertical axis in Fig. 1, respectively. Each of these models comprises the structure aspect (e.g., the Information Model gives information on the application content, their structure, the structures for accessing content, etc.). The behavior aspect is addressed by two models: the Transaction Model and the Operation Model, the former addressing the business process design and the latter the design of more elementary user functionalities.

Finally, the UWA Customization Design activity, defines the customization rules that may apply to any of the UWA design models, thus providing the designed application with the ability to adapt to different usage contexts.

The UWA methodology enables separation of design concerns by devoting a design activity and a resulting model to each of the levels and aspects which characterize data- and operation-intensive web applications [34][14]. User-centered design ability (the perspective in developing the different UWA models is that of the different user types of the application) and context awareness for the resulting applications makes UWA one of the most valuable conceptual design methodologies for web applications. The availability of a MOF-Complaint metamodel for UWA [4] makes it possible to specify the semantics associated with each modeling concept, the valid configurations and the constraints that apply. At same time, being the UWA metamodel MOF-compliant it is possible to create modeling tools that generate models that can be easily exchanged, imported in different design tools, rendered into different formats, transformed, and used to generate application code [24].

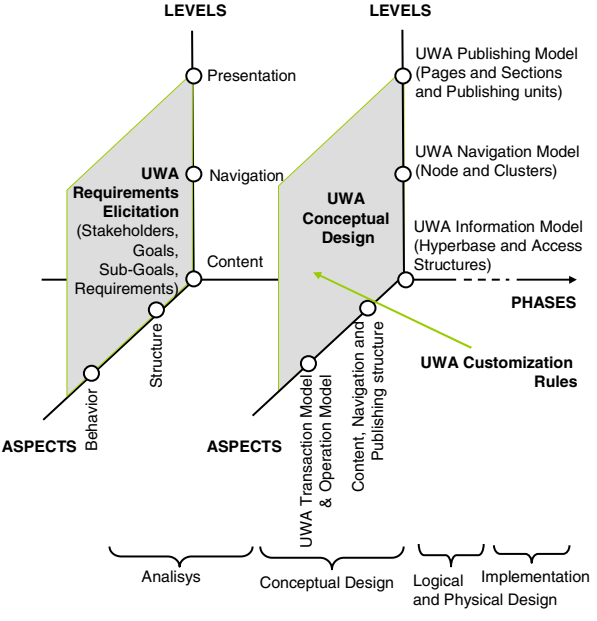


Fig. 2. UWA Requirement Elicitation and Conceptual Modeling overview

3 A UWA-Based MDD Approach

3.1 Process Overview

The proposed UWA-based MDD approach to the development of web applications uses a series of models and successive model transformations to progressively refine

and enrich the application requirements so as to obtain the application source code. The different models are also the basis for updated project documentation, at different levels of abstraction.

Fig. 3 depicts the overall development process and the generated models. The process begins with UWA Requirements Elicitation, to identify the stakeholders of the application and their goals, from which the application requirements are derived. Stakeholders, goals, sub-goals and requirements are represented by means of stereotyped UML use-case diagrams.

The conceptual design is carried out using the UWA methodology and the conceptual model of the application produced. This model includes the Information Model, the Transaction Model, the Navigation Model, the Operation Model, the Publishing Model and Customization Model.

The conceptual design phase is followed by logical design. The UWA conceptual models are transformed, by means of suitable translation rules, into a logical model that is closer to the specifics of implementation but still platform-independent. The adoption of standard UML diagrams and of the MVC architecture for the resulting application gives the name "UML-MVC" to our proposed logical model.

The final phase of the MDD process consists of specializing the UML-MVC logical model to the specific platform chosen for implementation, obtaining a

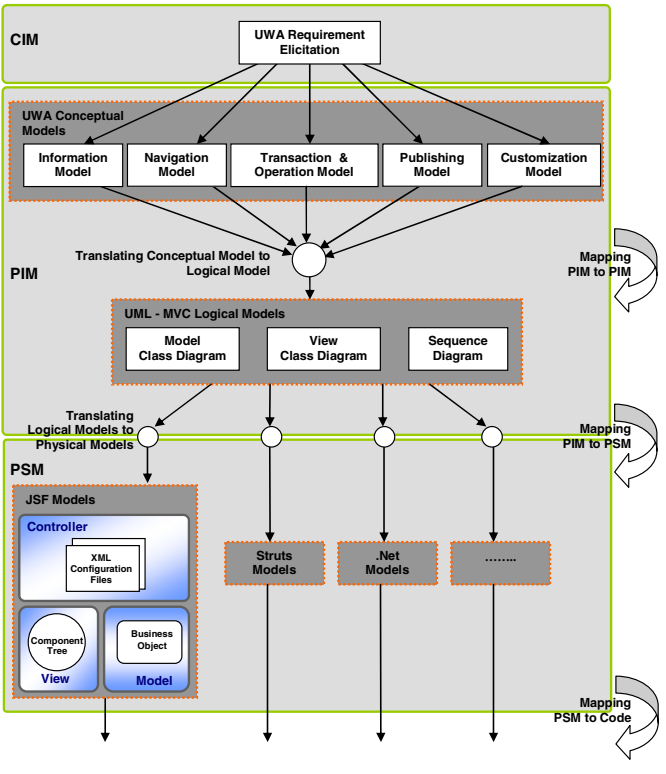


Fig. 3. An overview of the UWA-based MDD process

platform-specific model. Any implementation of the MVC architectural design pattern is suitable as the destination platform for the UML-MVC logical model defined in the previous step. In particular, we detail the case of the JavaServer Faces (JSF) technology [15][17] which is the technology we chose for experimenting our approach with an e-commerce web application.

As emphasized by Fig. 3, the proposed process, in addition to be model-driven, adheres to MDA. The UWA Requirement Elicitation model can, indeed, be considered a CIM as it focuses on functionalities required for the application. The UWA conceptual model and the UML-MVC logical model are PIMs, as they do not imply any specific technology to be used for the implementation. Finally the JSF model is a PSM as it is the specialization of the UML-MVC PIM for the JavaServer Faces technology.

In the following of this section we describe the UML-MVC logical model, the transformation rules to obtain this model from the UWA conceptual models, and the guidelines that can be used to map the UML-MVC model onto the JavaServer Faces implementation framework.

3.2 The UML-MVC Logical Model

The logical model we elaborated consists of stereotyped standard UML diagrams and is intended to model the software components to be developed for each of the three layers of the MVC architecture [8] as well as the relationships among them. Specifically, the model is structured into the following UML diagrams:

1. Model Class Diagram (MCD): an UML class diagram modeling the classes that participate in the Model layer and implement the application business logic and data persistency.
2. View Class Diagram (VCD): an UML class diagram representing the client and server pages in the View layer. These pages have the responsibility of presenting data and content to the user and enabling user interaction with the system. This diagram also models:
    a. Classes making part of the Controller layer;
    b. Associations between user interactions (e.g., the “Submit” of a form of a View page) and methods of classes of the Model that are in charge to serve them;
    c. Associations between attributes of classes of the View (which correspond to data provided/requested by the application) to methods of classes of the Model that manage them.
    d. Navigation links between pages of the View.
3. One or more UML Sequence Diagram describing the interactions between the various components of the system and their state transitions during the execution of complex user activities and web transactions. In addition, this diagram is used to describe the navigation steps between pages of the View that are associated with the different return values of the execution of the methods of Model classes.

The resulting overall model is independent of the specific technologies chosen for the implementation of the application, thus it is a PIM in the MDA architecture, but it is already sufficiently detailed to guide the application's development team. The

model can be used to create the application with any implementation technology based on the MVC pattern. On an experimental basis, we used the Java ServerFaces framework [15].

3.3 Mapping UWA Conceptual Models onto UML-MVC Logical Model

The transformation of UWA conceptual models into UML-MVC logical models is accomplished by means of a set of mapping heuristics which create appropriate correspondences between the UWA models and modeling concepts, with the components of the MVC architecture and the elements of the logical model. Table 1 summarizes these heuristics: the first column lists the UWA conceptual models; the second lists the main modeling concepts included in each of the UWA models; the third column reports the layers of the MVC architecture onto which the considered UWA modeling concept is mapped; finally, the fourth column reports the specific elements of the logical model that originate from the UWA concept. Elements of the UML-MVC logical model we defined include: (*i*) Classes, Class Attributes and Class Methods of the MCD; (*ii*) Client and Server Page Classes of the VCD; (*iii*) Controller Classes mapping the user interactions with pages of the VCD onto methods of classes on the MCD; (*iv*) Associations between Classes of the MCD, (*v*) Associations

Table 1. Summary of UWA to UML-MVC transformation rules

UWA Conceptual Model	UWA modeling Concept	MVC Component	UML-MVC Modeling Element
Information Model	**Entity Type**	Model	Class into MCD
	Entity Component	Model	Class into MCD aggregated to the Class created for the Entity Type
	Slot	Model	Private Attribute and associated set and get methods into MCD
	Semantic Association	Model	Association between Classes into MCD
	Association Center	Model	Association Class into MCD
	Collection Type	Model	- Class into MCD associated to the Collection Center - Method added to the Class involved in the collection
Transaction Model	**Complex Activity**	Model	Class into MCD with a method originated from the Activity PropertySet
	Suspendable Transaction	Model	Class into MCD with attributes *TransactionHistory*, *CurrentActivity*, *ExecutionState*, *Suspension* and *DeviceType*
	Suspendable Complex Activity	Model	Class into MCD with attributes *TransactionHistory*, *CurrentActivity*, *ExecutionState*, *Suspension* and *DeviceType*
	Elementary Activity	Model	Method added to the involved Class of the MCD to implement the activity
	Execution Flow	Controller	Sequence Diagram representing the interaction between elements of the logical model and the navigation rules associated to the different return value from the execution of methods of Classes of the MCD
Navigation Model	**Navigation Node**	View	Server Page Class or Client Page Class into VCD
	Navigation Cluster	View	Navigation Links between Pages into VCD
	Activity Node	View	- Server Page Class into VCD with input/output attributes - Associations between input/output attributes of the Server Page Class into VCD with methods of MCD classes managing the data - Association between action elements of the Server Page Class (e.g. Submit button of a Form) and Controller classes to serve them
		Controller	- Controller Class into VCD with an attribute Result. - Association between attribute Result and method of the MCD class to be invoked in correspondence of the user interaction with the page
	Activity Cluster	View	- Navigation Links between Pages into VCD
Publishing Model	**Publishing Unit**	View	Client Page or Server Page Class into VCD
	Section	View	Client Page or Server Page Class into VCD aggregating classes originated from Publishing Units
	Page	View	Client Page or Server Page Class into VCD aggregating classes originated from Publishing Sections
	Page Template	View	Frameset Class into VCD
	Link	View	Navigation Links between Pages of the VCD

between attributes of Classes of the VCD with methods of MCD classes; (*vi*) Navigation Links between Pages of the VCD; (*vii*) Sequence diagrams representing the workflow of a transaction and navigation rules to be implemented by the Controller.

Broadly speaking, the UWA Information and Transaction Models merge into the MCD, while the UWA Navigation and Publishing Models merge into the VCD. Associations between attributes and user interaction elements of the View pages with methods of Model classes originate from the Navigation Model and, indirectly, from the Transaction Model. In addition, from the UWA Transaction Model the UML Sequence Diagrams are also created.

3.4 Mapping UML-MVC Logical Model onto JavaServer Faces Platform Specific Model

The JSF technology is a Java implementation of the MVC architectural design pattern which simplifies the building of user interfaces for web applications by assembling reusable UI components in a page, connecting these components to an application data source; and wiring client-generated events to server-side event handlers [17].

As synthesized by bottom part of Fig. 3, the JSF architecture is a specialization of the MVC architecture in which the Model component is realized by means of Java business objects, the View component is made up of JavaServer Pages (JSP) in which custom tag libraries are used for expressing the JSF user interface components, and the Controller is implemented by a Servlet named FacesServlet.

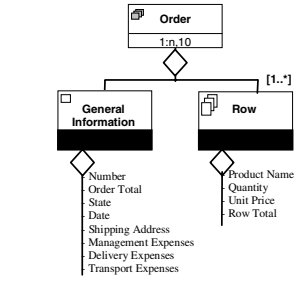
To map the UML-MVC logical models onto the JSF software components the following guidelines can be used:

1. Classes of the MCD are mapped onto Java Business Objects, such as JavaBeans (JB) or Enterprise JavaBeans (EJB).
2. Classes of the VCD are implemented by means of Java Server Pages including the JSF user interface components specified by means of the JSF tags.
3. Associations between pages of the VCD with methods of classes in the MCD are used to define associations between presentation JSF components included in the Java Server Pages and attributes and methods of the Java Business Objects.
4. Associations between Controller classes and classes of the MCD in the VCD, together with information derived from the Sequence Diagrams on navigation rules associated with the different return values of the execution of a method of a Model class are used to define the “faces-config.xml” configuration file for the FacesServlet Controller.

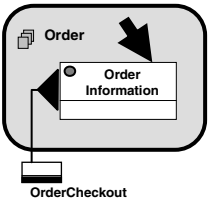
3.5 An Example Application

The proposed UWA-based MDD approach was applied to the development of an e-commerce website, of which we report here the portion concerning the order checkout and visualization. The process involved the analysis of the requirements and the conceptual design of the application by means of the UWA methodology. The transformations rules summarized in Table 1 were then used to obtain the logical UML-MVC model of the application, which was finally specialized for

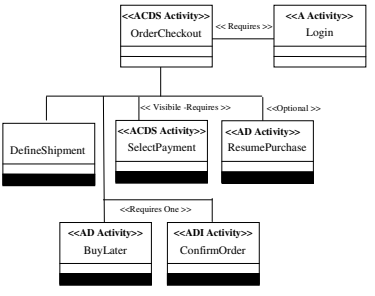
Table 2. An excerpt of the conceptual and logical models generated by the UWA-based MDD process for developing an e-commerce web application



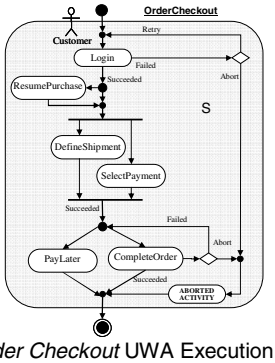
(A) *Order* UWA Entity Type Diagram



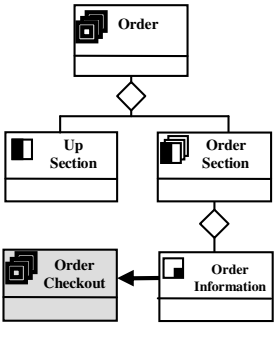
(B) *Order* UWA Navigation Cluster



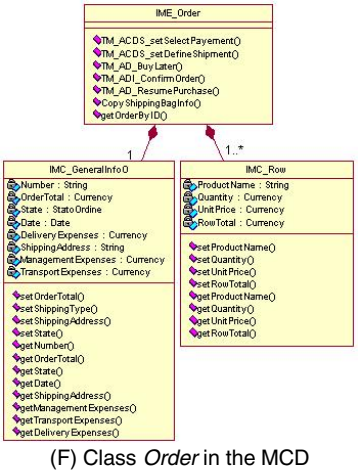
(C) *Order Checkout* UWA Organization Model



(D) *Order Checkout* UWA Execution Model



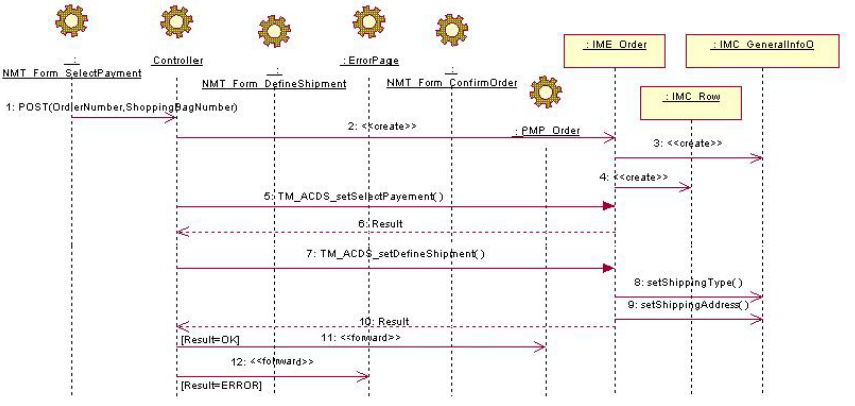
(E) *Order* UWA Publishing Page



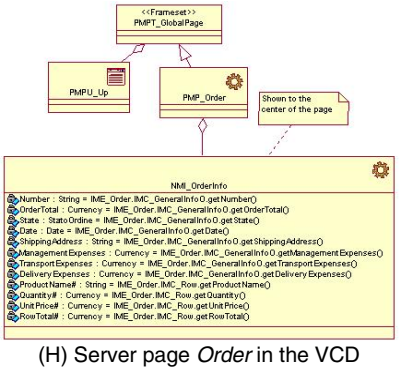
(F) Class *Order* in the MCD

implementation with the Java ServerFaces technology. Table 2 shows an excerpt of the different models that were generated during the process and a screenshot of the final application. More in detail, the table reports the Entity Type diagram (A) and the

Table 2. *(Continued)*



(G) UML-MVC Sequence Diagram for the checkout transaction



(H) Server page *Order* in the VCD

(I) A screenshot of the *Order* View web page

Navigation Cluster (B) associated to an *Order* made by a customer. The first diagram models the information characterizing the *Order* and is part of the UWA Information Model. The latter models the navigation associated with the Order and the actions that can be invoked by the user, such as *Order Checkout*. The Organization Model (C) and the Execution Model (D) describe the activities in which the *Order CheckOut* transaction is organized and represent a portion of the UWA Transaction Model of the application. Diagram (E) of the table shows the UWA Publishing Model for the *Order Page*. By applying the transformation rules, the Entity Type *Order* of the UWA conceptual model was transformed into the class *Order* in the MCD (F) of the logical model, while the Navigation Cluster and the Publishing Page *Order* originated the server page *Order* in the VCD (H). The UML sequence diagram (G) was derived from the *Order Checkout* UWA Execution Model.

Attributes of the *Order* class were derived from Slots of the *Order* Entity Components in the conceptual model. Get and Set methods were automatically associated to each of the attributes. The set of activities included in the Organization

Model of the *OrderCheckout* transaction were transformed into methods for the class *Order*. These methods include: *setSelectPayement*, *setDefineShipment*, *BuyLater* and *ConfirmOrder*. The class diagram modeling the server page *Order* in the View Class Diagram (H) also reports the associations between attributes of the page (i.e., JSF user interface components in the implementation) and methods of the class *Order* in the Model Class Diagram. To support traceability, class names and methods of the UML-MVC logical model are prefixed with the acronyms of the UWA model and modeling concept from which they originate.

The structure of the Navigation and Publishing Models impacted onto the VCD, in which they appear also the dependency with the respective classes of the MCD apt to the management of the information. A screenshot of the web page resulting from the implementation of the prototype of the application with the JSF technology is shown by figure (I).

3.6 Costs/Benefits of the Approach

The introduction of an additional design phase in the development process of a web application cause the process to lengthen and complicate, and more effort to be required if supporting tools towards MDD are not provided.

On the other hand, no matter of if tools for the automatic transformation of conceptual models into logical models are available, providing developers with models which are closer to the implementation simplifies implementation choices, reduces coding time and helps in producing higher quality software. In fact, having a model which from one hand is directly linked to the conceptual model and from the other is very close to the implementation details helps the development of applications which exhibit:

- greater internal consistency, as they fully satisfy the requirements of conceptual design;
- greater usability, as they are more able to satisfy the expectations of the users;
- greater maintainability, as design and requirements, since the impact on the code of any changes to the model (or indeed the impact on the model of any changes to the code) can be traced.

The proposed approach, by defining the UML-MVC logical model, makes it possible to establish a correspondence between the UWA conceptual design of a web application and its implementation. Maintenance and evolution operations become easier, requirements traceability is made possible as well as alignment between software and documentation during the entire application lifecycle.

4 Related Work

The Model-Driven Development paradigm is applied successfully by a number of web engineering methods, such as UWE, OO-H, OOHDMDA, 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 [23] process to developing web systems follows the MDA principles and uses the OMG standards [24-29]. The process makes use of model transformations defined at metamodel level and specified in general purpose transformation languages, such as QVT [26] and graph transformations. Currently, many of the transformations have already been automated, thanks to the OpenUWE [31] tool suite. One of the main characteristics of this suite is its open architecture based on established standards. These standards are supported by both open-source and commercial tools. The common data exchange language within this architecture is based on the extensible UWE meta-model [37].

OO-H [9] supports the transformation-based construction of a presentation model based on modelling elements of the navigation model, and code generation based on the conceptual, navigation and presentation models [18]. OO-H transformation rules are a proprietary part of a CASE tool called VisualWADE [39]. This tool supports modelling and automatic generation of applications based on XML, ASP, JSP, and PHP.

OOHDM [36] may be considered as a platform-independent domain-specific language for web applications that provides an object model, in contrast to other web application modeling languages. OOHDMDA [35] generates servlet-based web applications from OOHDM models. The OOHDMDA approach follows MDA principles by employing the OOHDM conceptual and navigational scheme of a web application as the basic PIM for the MDA process, using any UML-based design tool, such as Rational Rose [33], which produces an XMI-file as output. The basic PIM is transformed into the intermediate PIM, by adding to it the behavioural semantics of the OOHDM core features and business processes. This transformation is achieved by modifying the XMI-file of the basic PIM. The PIM is then transformed into a servlet-based PSM.

WebML [10-12] is a model-driven method for the development of data intensive web applications, with an associated supporting CASE tool called WebRatio [40]. 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 [11]. The web application generated by WebRatio is deployed in a runtime framework based on a set of Java components, which can be configured by use of XML files. The runtime architecture is based on the MVC design pattern and is suited for the Apache Struts open-source web application development framework [1] and the JSP tag libraries [30].

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 not explicitly adopt the MVC architecture pattern in their PIMs. As above shortly 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.

A final as obvious as notable difference between our approach and others proposed in the literature lies in the conceptual design methodology on which the approach is

based. Our UWA-based MDD approach enables the development of web applications which faithfully implement their UWA conceptual model, thus benefiting from the UWA peculiarities and the quality design characteristics as summarized in Section 2.3. This difference is the main motivation for the present work.

5 Conclusion

Conceptual design methodologies enable the analyst to abstract from implementation and technological details and focus on application requirements in the problem domain. This is referred to as "what" the application has to do.
The UWA design methodology and associated models are particularly suited for the user-centered design of complex ubiquitous web applications. UWA provides different related models to design the different levels and aspects characterizing a web application (content, navigation, presentation, structure, behavior).

Despite the benefits deriving from the use of conceptual design methodologies, there is a wide gap between the models they generate and the implementation of the application. A further design step and related models are required to specify "how" to implement the final application adhering to the conceptual models.

This paper proposed an intermediate model, named UML-MVC logical model, to be used between UWA conceptual design phase and the implementation phase, and a set of transforming heuristics.

The UML-MVC logical model is a platform independent model, based on standard UML diagrams and incorporating the MVC architecture design pattern. The set of heuristics permits the automatic transformation of the UWA models into the logical ones. The model and the mapping heuristics are the basis of a Model-Driven Development approach based on the UWA methodology, for web applications. In addition, the rules to transform the logical model to a platform specific model for Java ServerFaces applications has been defined and experimented in an example case study.

Overall, the resulting approach combines the advantages of MDD, such as requirement traceability and maintenance and evolution better support, with the ability of UWA to design application in a user-centered perspective and ubiquitous. The logical model creates a link between the application implementation and the UWA design models and application requirements.

We are currently working on the specification of our model transformations rules using the QVT language and on the development of appropriate tools to support the whole UWA-based MDD approach by extending the design tools provided with the UWA methodology.

References

1. Apache STRUTS open-source framework: http://struts.apache.org/
2. ArgoUML: http://www.argouml.org
3. Arraes Nunes, D., Schwabe, D.: Rapid Prototyping of Web Applications combining Domain Specific Languages and Model Driven Design. In: ICWE'06. Proceedings of the 6th International Conference on Web Engineering July 11-14, 2006, Palo Alto, California, USA (2006)

4. Baresi, L., Garzotto, F., Maritati, M.: W2000 as a MOF Metamodel. In: Proceedings of World Multiconferemce On Systemics, vol. 1 (2002)
5. Bézivin, J.: In Search of a Basic Principle for Model Driven Engineering. UPGRADE V(2), Novótica (April 2004)
6. Brambilla, M.: Extending hypertext conceptual models with process-oriented primitives. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 246–262. Springer, Heidelberg (2003)
7. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. ACM Transactions on Software Engineering and Methodology (TOSEM) (in print, 2006)
8. Buschmann, F., Meunier, R., Rohnert, H., Sornmerlad, P., Stal, M.: Patter-Oriented Software Architecture - A system of pattern, vol. 1. John Wiley & Sons Ltd., West Sussex, England (2000)
9. Cachero, C., Gómez, J., Pastor, O.: Object-Oriented Conceptual Modeling of Web Application Interfaces: the OO-H Method Abstract Presentation Model (2000)
10. Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. IEEE Internet Computing 6(4) (2002)
11. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Application. Morgan Kaufmann Publishers, Elsevier Science (USA) (2003)
12. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. Computer Networks 33(1-6), 137–157 (2000)
13. Conallen, J.: Building Web application with UML, 2nd edn. Addison Wesley, Redwood City, CA, USA (2002)
14. Distante, D., Rossi, G., Canfora, G., Tilley, S.: A Comprehensive Design Model for Integrating Business Processes in Web Applications. International Journal of Web Engineering and Technology (IJWET) 2(1), pp. 43–72 (2007)
15. Dudney, B., Lehr, J., Willis, B., Mattingly, L.: Mastering JavaServer™ Faces. Wiley, New York (2004)
16. Gomez, J., Cachero, C., Pastor, O.: Extending a Conceptual Modeling Approach to Web Application Design. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 5–9. Springer, Heidelberg (2000)
17. JavaServer Faces Technology: http://java.sun.com/javaee/javaserverfaces/
18. Koch, N.: Transformation Techniques in the Model-Driven Development Process of UWE. In: MDWE 06. Proceedings of the 2nd Model-Driven Web Engineering Workshop, Palo Alto, CA, July 11, 2006, ACM Press, New York (2006)
19. Koch, N., Kraus, A.: The expressive Power of UML based Web Engineering. In: Proceedings of the 2nd International Workshop on Web Oriented Software Technology (IWWOST02) June 10, 2002, Málaga, Spain (2002)
20. Koch, N., Zhang, G., Escalona, M., j.: Model Transformations from Requirements to Web System Design. In: ICWE'06, Palo Alto, California, USA, July 11-14, ACM Press, New York (2006)
21. Langham, M., Ziegeler, C.: Cocoon: Building XML Applications, Sams Publishing (2002)
22. Meliá, S., Gómez, J., Koch, N.: Improving Web Design Methods with Architecture Modeling. In: 6th International Conference on E-Commerce and Web Technologies (EC-Web 2005) August 22-26, 2005, Copenhagen, Denmark (2005)
23. Moreno, N., Fraternali, P., Vallecillo, A.: A UML 2.0 Profile for WebML Modeling. In: MDWE 06. Proceedings of the 2nd Model-Driven Web Engineering Workshop, Palo Alto, CA, July 11, 2006, ACM Press, New York (2006)

24. Object Management Group (OMG) Meta Object Facility Specification (MOF): http://www.omg.org/mof/
25. Object Management Group (OMG). Model Driven Architecture (MDA): www.omg.org/mda/
26. Object Management Group (OMG). Query/Views/Transformations (QVT): www.omg.org/
27. Object Management Group (OMG). UML 2 Object Constraint Language (OCL): www.omg.org/docs/ptc/03-10-14.pdf
28. Object Management Group (OMG). Unified Modeling Language (UML): Superstructure, version 2.0 www.uml.org/
29. Object Management Group (OMG): XML Metadata Interchange (XMI) www.omg.org/
30. Open Source JSP Tag Library: http://www.java-source.net/open-source/jsp-tag-libraries
31. OpenUWE: http://www.pst.ifi.lmu.de/projekte/uwe
32. Pastor, O., Abrahão, S., Fons, J.: An Object-Oriented Approach to Automate Web Applications Development. In: Bauknecht, K., Madria, S.K., Pernul, G. (eds.) EC-Web 2001. LNCS, vol. 2115, pp. 16–28. Springer, Heidelberg (2001)
33. ROSE: IBM Rational Software. Online at www.ibm.com/rational
34. Rossi, G., Gordillo, S., Distante, D.: Improving Web Applications Evolution by Separating Design Concerns. In: STEP 2005. IEEE Software Technology and Engineering Practice 2005, September 24-25, 2005, Budapest, Hungary, Workshop on Evolution of Software Systems in a Business Context (2005)
35. Schmid, H.A., Donnerhak, O.: OOHDMDA - An MDA Approach for OOHDM. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 569–574. Springer, Heidelberg (2005)
36. Schwabe, D., Rossi, G.: An Object-Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems (TAPOS) 4, 207–225 (1998)
37. Schwinger, W., Koch, N.: Modeling Web Applications, in Web Engineering - Systematic Development of Web-Applications. In: Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (eds.) John Wiley & Sons Ltd., West Sussex, England (2006)
38. UWA Consortium, Ubiquitous Web Applications. In: Proceedings of the eBusiness and eWork Conference 2002, (e2002: October 16-18 2002, Prague, Czech Republic) (2002)
39. VisualWADE. http://www.visualwade.com
40. WebRatio. http://www.webratio.com