An MDA Approach for the Development of Web Applications

Santiago Meliá Beigbeder and Cristina Cachero Castro

Universidad de Alicante, España
{santi,ccachero}@dlsi.ua.es

Abstract. The continuous advances in Web technologies are posing new challenges to Web Engineering proposals, which now require the inclusion Software Architecture techniques in order to integrate the explicit consideration of non-functional features in the Web application design process. In this article we propose a new approach called WebSA, based on the the MDA (Model Driven Architecture) paradigm. WebSA specifies a model driven process that adds to the traditional Web-related functional viewpoint a new software architectural viewpoint that permits, by means of successive model transformations, to establish the desired target application structure.

1 Introduction

The high pace at which advances in Web technologies are taking place has changed the idiosyncrasy of Web applications, that now imply not only more complex functional requirements but also stricter constraints posed on features such as distributability, scalability, platform-independence or extensibility. In order to tackle such new requirements, several authors [1] propose the use of Software Architecture techniques. Following this trend, in this article we present WebSA (Web Software Architecture). WebSA is a Web model-driven approach that is based on the standard MDA (Model Driven Architecture) [5]. The MDA framework provides WebSA not only with the possibility to specify and formalize a set of Web-specific models by means of a UML profile, but also to specify each process step from the models to implementation by means of a set of transformation rules.
The remaining of the article is structured as follows: in section 2 we present briefly the WebSA Development process and its main views and models. From these views, the architectural viewpoint and, more specifically, its logical architectural view is discussed in section 3. Last, section 4 outlines the conclusions and further lines of research.

2 WebSA: Model Driven Architecture of Web Applications

As we stated above, WebSA is a proposal whose main target is to cover all the phases of the Web application development and to contribute to cover the gap existing between traditional Web design models and the application implementation. In order

to achieve this aim, it defines an instance of the *MDA Development Process* for the Web application domain. Also, it proposes the formalization of the models by means of a MOF-compliant repository (metamodel) and a set of OCL constraints (both part of the OMG proposed standards) that together specify (1) which are the semantics associated with each element in the models, (2) which are the valid configurations and (3) which constraints apply. This formalization is being performed, as other Web methodologies [14] have done before, by the definition of a UML [8] profile that gathers the main constructs of the models involved in the specification of applications in the Web domain.
In order to define a Web application System proposes a Web application view model that is made up of 8 *views*, grouped into three *viewpoints:* requirements, functional and architectural viewpoints. From them, the architectural viewpoint is a main contribution of WebSA. This viewpoint includes a logical architectural view that gathers the set of logical components (subsystems, modules and/or software components) and the relationships among them. Also, it includes a physical architecture view that describes the physical components that integrate the lower level specification of the application (clients, servers, networks, etc.). As we have stated above, and in order to shift from one view to the other, WebSA defines a process that is explained next.

2.1 The WebSA Development Process

The WebSA Development Process [3] is based on the MDA development process. In order to fulfill this goal, it establishes a correspondence between its Web-related artifacts and the MDA artifacts. Also, and as a main contribution, it defines a transformation policy partly driven by the architectural model that can be seen in Fig. 1. In this figure we observe how in the analysis phase the Web application specification is divided horizontally into two viewpoints. The functional-perspective models reflect the functional analysis, while the architectural models define the system architecture. Both of these models are PIMs in the context of an MDA framework. In this phase, the architectural models are based on the concept of *Conceptual Architecture* [4], and are made up of conceptual elements, obtained by abstraction of the elements found in the Web application domain. These models fix the application structure orthogonally to its functionality, therefore allowing their reuse in different Web applications.

The *PIM-to-PIM* transformation (see T1 in Fig. 1) of these models into platform independent design models (PIMs) provides a set of artifacts in which the conceptual elements of the analysis phase are mapped to concrete elements where the information about functionality and architecture is integrated. It is important to note how these models, being still platform independent, are the basis on which several new transformations, one for each target platform (see e.g. T2, T2' and T3 in Fig. 1), can be defined. The output of these *PIM-to-PSM* transformations is the specification of our Web application for a given platform. At this level of abstraction, the models can still endure a final *PSM-to-code* transformation, usually implemented in WebSA by means of templates. In this way, the WebSA process guarantees the traceability from analysis to code. In order to complete the specification of this process, WebSA formalizes the three transformations (PIM-to-PIM, PIM-to-PSM y PSM-to-code) by means of QVT (Query View Transformation) [6]. QVT defines a transformation language that is based on an extension of the MOF 2.0 metamodel and which allows to link models situated in different views and map them through the different life cycle phases. Also, QVT extends OCL for the specification of queries and filters over the models. The inclusion of an architectural view in this process has a preeminent role for the completion of the specification of the final Web application, and drives the refinement process from analysis to implementation. In the next section we will center on this *architectural viewpoint* and how it influences the refinement process.

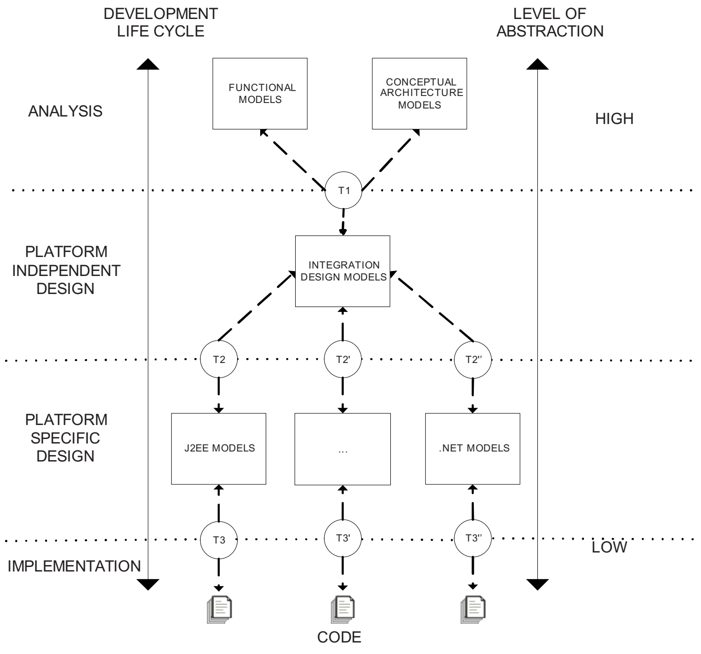

Fig. 1. WebSA Web Development Process.

3 Logical Architectural View for Web Applications

The logical architectural view is responsible for the definition of the logical components (subsystem, modules and/or software components) that collaborate in the system, as well as the relationship among them. In WebSA this view is made up of three models, namely (1) the *Subsystem Model* (SM), which determines the conceptual subsystems that make up our application, (2) the *Web Component Configuration Model* (WCCM), which decomposes each subsystem in a set of abstract components related by abstract connectors and (3) the *Web Component Integration Model* (WCIM) which, as its name may suggest, performs an integration of views.

3.1 Subsystem Model

Also known as structural design, the Subsystem Model determines which are the subsystems that make up our application.
This model is made up of two main constructs:

- **Subsystem**: element of coarsest granularity in any Web application architectural design. It defines a group of software components developed to support the functionality assigned to a given logical layer. Subsystems are depicted in WebSA by means of the UML package symbol.
- **Dependency Relationship**: link element that reflects the use dependencies between subsystems. In WebSA it is depicted as a UML dependency arrow.

This model provides the most abstract perspective of the logical architecture view, and the subsystems obtained during this phase will be later identified with each logic layer in the application. This separation on layers is essential to reduce the complexity of the system, as it is justified in the "layering approach" pattern presented in [2]. In this model, and in order to ease its construction, WebSA proposes the use of any of the five *distribution patterns* defined in [7].
The different layers identified for a given system are reflected in the Subsystem Model by means of a stereotype associated to the subsystem package symbol. WebSA defines nine subsystem stereotypes, namely: «user interface», «server», «business logic», «presentation», «dialog control», «process control», «business object», «data access» and «physical data». Additionally, WebSA provides a set of restrictions that apply to the possible relationships between subsystems.
Once the different subsystems have been identified, we must specify the contents of each subsystem, that is, the set of abstract components that configure them. Such contents are specified in the Web Component Configuration Model, which is introduced next.

3.2 Web Component Configuration Model (WCCM)

The second model of the logic Architecture view is the Web Component Configuration Model, which consists of a set of abstract components and connectors, particular to the Web domain. These abstract elements, also based on the Conceptual Architecture, are the result of a refinement process performed on each subsystem identified in the previous model. The main constructs of the Web Component Configuration Model are *abstract components* and *abstract connectors.*

- An *abstract component* represents an abstraction of one or more software components with a shared functionality in the context of a Web application. An example of it is a Client Page, which represents any artifact that contains information and/or interaction code relevant for the user. Note how this kind of component does not necessarily map to a single physical page but reflects a general task that must be performed by the application, such as showing certain information to the user. Abstract components are depicted with a UML class symbol.

- The *abstract connector* represents a dependency relationship between two abstract components in the system, and is depicted with a stereotyped UML dependency relationship. This relationship may affect either the interface or the whole component.

In order to construct a WCCM, the designer may use any of the architectural and design patterns. Such patterns provide powerful configurations that, applied to abstract components, not only provide a reuse mechanism but also contribute to a more efficient development process.

3.3 Web Component Integration Model (WCIM)

The last model defined as part of the logical architectural view is the Web Component Integration Model (WCIM). This view is also known as *integration model*, as it connects the functional and architectural views under a common set of concrete components, modules and connectors which will eventually make up the Web application. This model is defined during the WebSA Platform Independent Design phase, and still centers on design aspects (components, their interfaces and their relations).
The WCIM includes three main constructs: concrete components, modules and concrete connectors. *Concrete Components* (CC) are the smallest unit in the context of the integration model. It represents a software component in a given application domain, and it is obtained as an instance of an abstract component. A module (M) is a container of one or more concrete elements in the context of a given Web application. Such elements can be either other modules, concrete components or concrete connectors. *Modules* condense the functionality of a set of elements, reducing in this way the complexity of the models, and are depicted by means of a UML package metaclass. Last, *Concrete Connectors* (CN) express a relationship between two concrete components or modules of the system. It can be regarded as an instance of a dependency relationship defined in the WCCM. The number of instances (concrete connectors) generated for each abstract connector depends both on the cardinality of such abstract connector and on the existing relationships between those abstract components and the functional side of the application. Concrete connectors belong, just as modules and concrete components, to a given application domain.

4 Conclusions

In this paper we have presented a Web development approach called WebSA. WebSA fosters the use of the MDA philosophy to define a set of suitable models and a complete refinement process to cover the Web application domain. In this way, WebSA adds a new architectural viewpoint to explicitly address the architectural issues. This view is made up of three models, and its construction follows a top-down process that goes from the Subsystem Model, where the layers of the application are defined, to the Web Component Integration Model, where the designer determines the low level platform-independent components that make up the final application. In each phase WebSA promotes the use of a set of reuse practices and provides the mechanisms to reflect different sets of requirements. Currently, we are working on a formal definition of the UML 2.0 profile and metamodel for WebSA, and also on the set of QVT transformation models that support the WebSA refinement process, which we expect to be incorporating in the VisualWADE Development Enviroment [9] in the near future.

References

1. Bass, L., Klein, M., Bachmann, F. "Quality Attribute Design Primitives" CMU/SEI-2000-TN-017, Carnegie Mellon, Pittsburgh, December, 2000.
2. Buschman F., Meunier R., Rohnert H., Sommerlad P., Stal. M.: "Pattern-Oriented Software Architecture – A System of Patterns"; John Wiley & Sons Ltd. Chichester, England, 1996.
3. Santiago Meliá, Cristina Cachero y Jaime Gomez. Using MDA in Web Software Architectures. 2nd OOPSLA Workshop in "Generative Techniques in the context of MDA". http://www.softmetaware.com/oopsla2003/mda-workshop.html. October, 2003.
4. Nowack, P.: "Structures and Interactions – Characterizing Object-Oriented Software Architecture" PhD thesis. The Maersk Mc-Kinney Moeller Institute for Production Technology, Univertity of Southern Denmark".
5. Model-Driven Architecture (MDA) Home Page: http://www.omg.org/mda/index.htm.
6. Object Management Group. Request for Proposal: MOF 2.0 Query / Views / Transformations RFP,2002. ad/2002-04-10.
7. Renzel K., Keller W. Client/Server Architectures for Business Information Systems. A Pattern Language. PLoP'97 Conference.
8. UML 2.0 Standard, OMG (2003). http://www.omg.org.
9. VisualWADE. http://www.visualwade.com