Using MDA in Web Software Architectures

SANTIAGO MELIA, CRISTINA CACHERO AND JAIME GOMEZ[1]
Universidad de Alicante, Spain

The new challenges posed by the Internet market have increased the need for Web Applications that require more development efforts and guarantee a higher quality level. In order to contribute to this goal, in this position paper we present a new proposal called WebSA which proposes the inclusion of a software architecture models to complement the specification of Web Applications. This strategy, together with the definition of the different models following the MDA standard provides the proposal with the necessary mechanisms to (1) improve the pace at which Web Applications are developed, (2) ease its integration with other systems and (3) cutting off the web application development cost.

1. Introduction

During the last years, in the context of the development of Web Applications many different methodologies, most of them partly or fully based on UML (see e.g. OO-H [Cachero 2003], UWE [Koch and Kraus 2003], OHDM [Schwabe et all. 1999]) have been proposed for the high-level and platform-independent specification of the different software components that build up a typical Web application, covering the presentation tier, the business tier, and the data tier. Common to all the design proposals is the goal of expressing in a formal yet intuitive way "how" a Web application works, so to achieve higher design quality, more pervasive code generation, better documentation, and easier maintenance.

However, less attention has been paid to the problem of software architecture for Web applications, defined as that of identifying and formalizing "what" subsystems, components and connectors (software or hardware) the application should have. In particular, the process of integrating non-functional requirements into the design of a web application that conform to them is still largely unsupported. The methodology gap between software architectures and web application design diminishes the benefits of pervasive Web methodologies, increases the time to market of Web applications, and opens the way to misalignments between non functional requirements and design schemes.

In this context, this paper describes a new proposal called WebSA (Web Software Architecture) which proposes the inclusion of a software architecture model to complement other proposals in the specification of Web Applications, and the use of the MDA standard [OMG 2001] to formalize and describe such models. Furthermore, the use of MDA to define the web software architecture provides WebSA with three important features: to (1) improve the pace at which Web Applications are developed, (2) ease the integration of web-aware interfaces with pre-existing modules, and (3) cutting off the web application development cost.

The remaining of the article is structured as follows: section 2 presents how a web architecture can be formalized with MDA describing the different views that make up a software architecture. In particular, we describe the logical architectural view that structures the subsystems, modules and connectors a web application should have. Section 3 proposes a set of extensions in the architecture for MDA models to describe software architectural issues. Finally section 5 sketches the conclusion and further work.

2. Web Architecture expressed with MDA

As we have stated above, WebSA is based on the specification of a software architecture for Web Applications, where by software architecture we mean: "*the description of the subsystems and components of a software system and the relationships between them, typically represented in different views to show the relevant functional and non functional properties* **"**[Buschman et all. 1996]. This definition introduces both the main architecture elements (subsystems, components, connectors), how to represent them (by means of a set of different views) and what they actually reflect (both functional and non functional requirements).

[1] Authors addresses: Santiago Meliá, Cristina Cachero, and Jaime Gomez, Universidad de Alicante, España, http://www.ua.es, {santi, ccachero, jgomez}@dlsi.ua.es.

In this way, WebSA relies on the separation of the web software architecture description in several concurrent views. Such views are defined in WebSA based on the MDA standard. Although, in the traditional use of MDA, software architecture has not been a main concern, it is also true that, as it is mentioned in its specification [OMG 2001] when referring to quality attributes, "*it is desirable to support and integrate such features in the modeled applications*". Such integration can be achieved, as stated in [Bass et all. 2000], by specifying the system software architecture.

There are previous works that have successfully tackled the specification of the application architecture in terms of a set of different views that make it up, among which we could cite [Kruchten 1995] , who defines 4+1 concurrent views in order to define a system and [Hofmeister et all. 1999], who splits the software architecture into 4 views. Unlike WebSA, both approaches are exclusively based on UML, and are defined to express any type of architecture. On the contrary, WebSA uses MDA and centers on the web domain, which permits a higher level of detail in the specification process.

Next we will present the main views considered in WebSA in order to get the software architecture of a web application. These views are formalized by means of a UML metamodel.

2.1 The view model

The view model shows the links among the different views (regarded as a set of artifacts created during the software development process) that make up a web application software architecture. In WebSA, the web application model is made up of 8 views, further grouped in viewpoints. A viewpoint is a set of views that share concerns. Fig. 1 we can observe a UML diagram that depicts the set of viewpoints and views in WebSA, as well as the relationships among them.

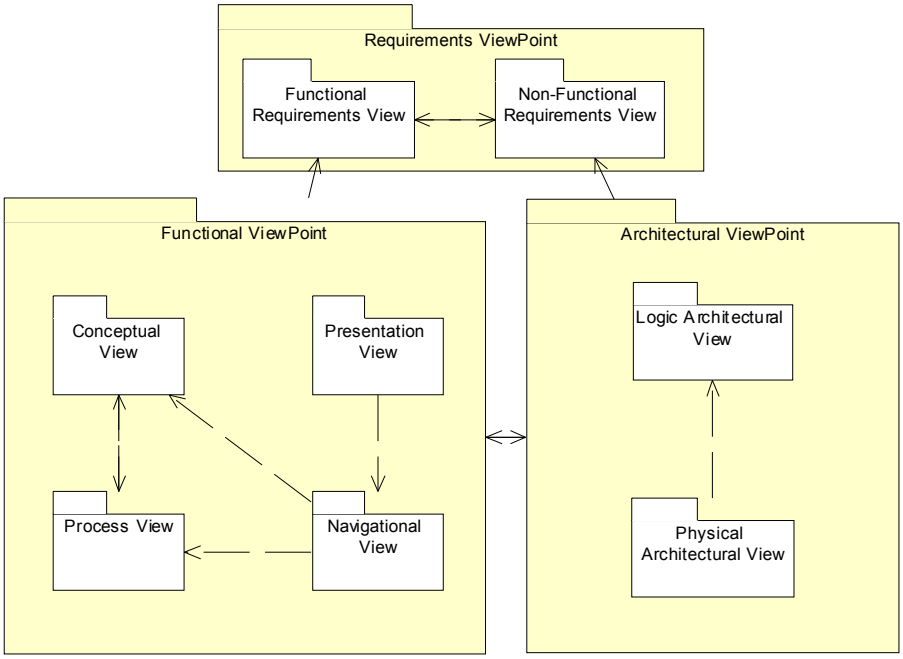


Figure 1.The View Model in WebSA.

The requirements viewpoint gathers the information needed to specify the system: in particular the set of use cases scenarios (functional requirements view) and quality scenarios (non-functional requirements view) are captured.

Departing from the requirement engineering phase, the functionality of a web system is defined by means of a Functional Viewpoint. That functionality is captured by means of the corresponding views defined by the web engineering community for web applications. In particular, the conceptual view captures the structure of the information system that lies behind the application. The navigation view

specifies the interactions that the user may perform in order to step through the different application scenarios. The presentation view is concerned with the general appearance of the application and the functionality associated with this appearance. Finally, the process view gathers process activities and flows.

Also, WebSA includes an architectural viewpoint to explicitly address the architectural issues. Departing from non-functional requirements a set of architectural patterns can be inferred to gather the logical and physical architecture views. The first one gathers the set of logical components (subsystems, modules and/or software components) and the relationships among them. The last one describes which are the physical components that integrates the final representation (clients, servers, networks, etc…).

Note that the interdependency between the functional and the architectural viewpoints (expressed with the double arrow in Fig. 1 implies that on some occasions it will be advisable to take architectural decisions based on functional features and vice versa.

The existing relationships among the different WebSA views are formalized in a common metamodel, that permits the establishment of a traceability between the elements in the different views. WebSA, aiming at being compliant with MDA, defines a conservative extension of the UML metamodel in the context of a UML profile.

Next, we will center on the architectural viewpoint defined in WebSA, and we will incorporate the necessary concepts to the model architecture defined in MDA. Due to space constraints, inside this viewpoint we will only focus on the logical architectural view.

2.2 Logical architecture view in MDA

As commented above, the logical architecture can be initially inferred from the non-functional requirements, gathered during the requirement phase. The matching between non functional requirements and the logical architecture may be undertaken by the application of a set of architectural patterns. Such patterns are defined at different levels of abstraction in each model, and suggest which are the relevant set of requirements to be taken into account for each architectural level. On the other hand, the functional requirements determine the connection between architectural and functional components, and will be defined according to the relationships established among the views in the WebSA metamodel.

There are several techniques and languages devoted to the specification of the logical architecture of a web application based on UML notation [Conallen 2002][Hassan and Holt 2002]. However, none of them use a notation according to the MDA standard. Such situation imposes a set of restrictions that have been taken into account when defining WebSA.

In our approach, the logical architecture is divided in three models, each one corresponding to each of the three phases in the architectural design. Each phase is located in a different level of abstraction, and reflects different architectural features, possibly influenced by a different set of user requirements. These three models are:

1. **Subsystem Model**: also known as structural design, determines which are the subsystems that make up our application. It makes use of the set of architectural patterns defined in [Renzel and Keller 1997], which determine which is the best distribution in layers of our system. Such distribution patterns define not only reusable elements but also permit the matching of non-functional requirements with the Web Application logical architecture.

2. **Web Component Configuration Model**: consists on the refinement of each subsystem by its decomposition in a set of abstract components that are particular to the web domain. They are defined by means of a web component ontology, where each component type has established its relationships at metamodel level with the functional view, which in turns makes possible the mapping between the architectural and the functional features. In this case the reuse elements are the architectural patterns defined by authors like [Buschman et all. 1996] and [Conallen 2002] for the web.

3. **Web Component Integration Model**: also known as integration model, due to the fact that it connects the functional and architectural views under a common set of concrete components and modules, which will eventually make up the Web Application. This model can be initially

inferred from the relationships that exist between the abstract component types defined in the configuration model. Such default mapping must be further refined in order to adjust the system needs. The reuse elements in this level are again the design patterns [Gamma et all. 1995] and [Beck and Johnson 1994].

Fig. 2 depicts such logical architecture refinement process.

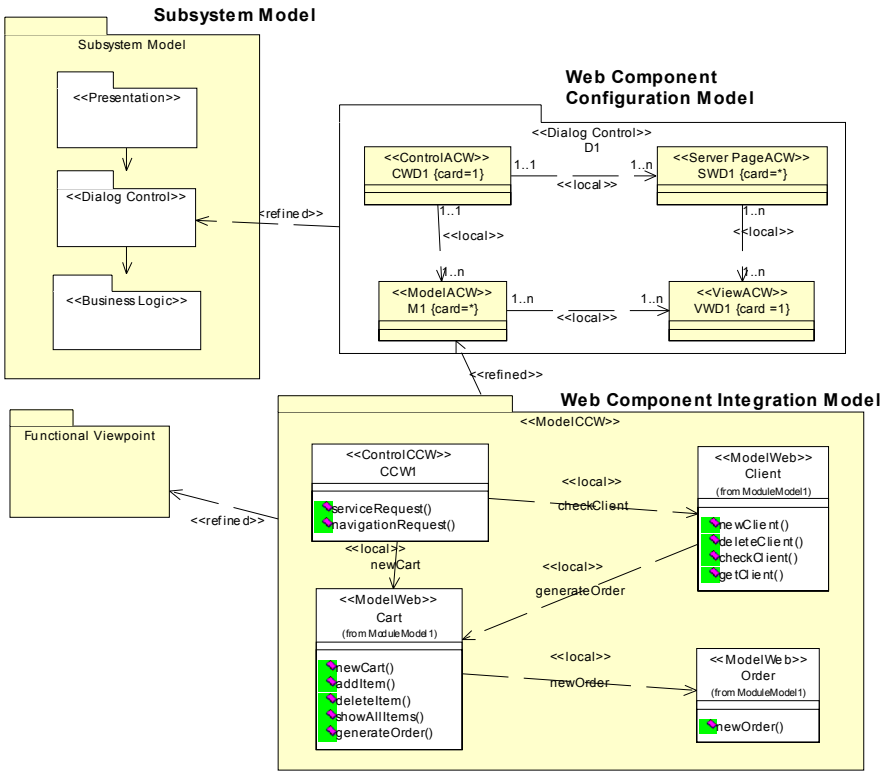


Figure 2. Logical Architecture refinement process

Note that, WebSA define the logical architecture by means of a Top-down process that goes from the subsystem model (higher level of abstraction) to the concrete components that make up the integration model and that determine the final application. In each phase WebSA defines a set of reuse mechanisms and provide the mechanisms to reflect different sets of requirements.

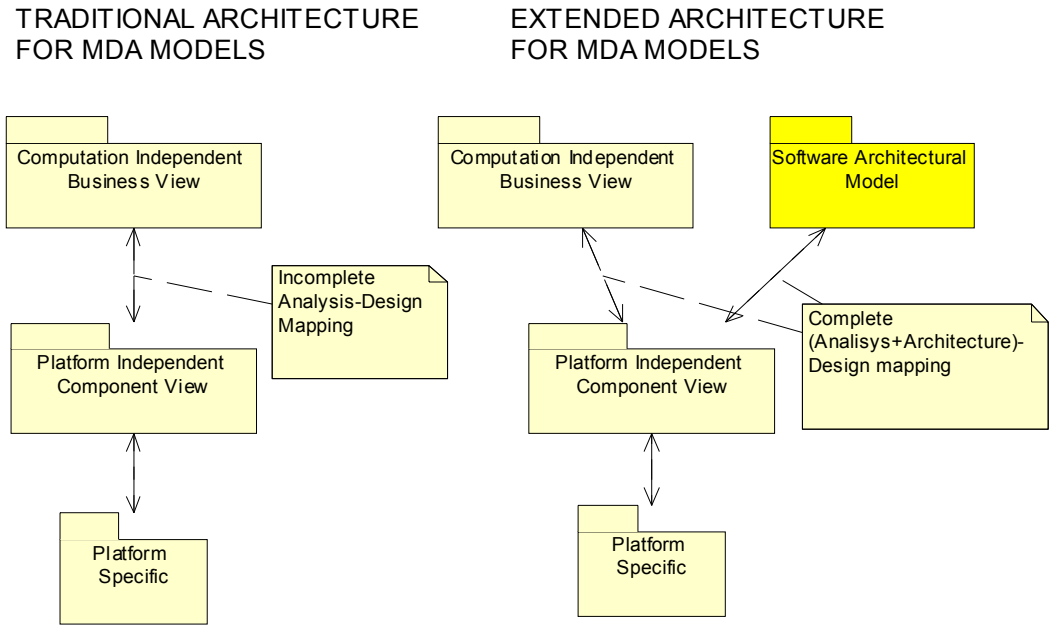
It is important to stress that the logical architecture centers on design aspects, and does not say anything about implementation (just components, their interfaces and their relations). In this way the architectural models are independent from the target platform. This fact makes possible to classify the three architectural models as PIMs (Platform Independent Model) in the context of MDA. A PIM to PIM mapping between the three models would leave us just a step away from platform considerations. Such lowest abstraction level is in WebSA the Component Integration Model. This model is defined at a level of abstraction similar to that of the EDOC profile[OMG 2002]. Due to the fact that the components defined for Web Applications in the business logic layer can be mapped to this profile, WebSA is capable of automatically provide different mappings to PSM models such as EJB, CORBA or .NET.

However, the fact that WebSA considers architectural and functional features in parallel implies that traditional model architecture defined for MDA must be extended with a new view that will group the different conceptual architecture PIMs models. In the next section, we will present that extension.

3. Extension of the architecture for MDA Models

In WebSA, the subsystem model and the abstract component configuration model cannot be integrated in the traditional views of MDA, due to the fact that they are independent from the problem domain, and are exclusively influenced by the non functional requirements. This fact implies that such models may be defined concurrently with those defined to gather the functional features of the Web Application.

Figure 3 shows the proposal to extend the architecture for MDA models [OMG 2001]. It provides a more rigorous mapping between the analysis phase (represented by the Computation Independent Business model) and the design phase (represented by Platform Independent Component view model). This mapping is driven by the Architecture View that makes a progressive refinement of architectural models and that finishes in a Web Component Integration Model. It is in this last model where the integration of the architectural model with the business domain model is achieved. The greater effort that must be made in the abstract modeling phase decreases the amount of work necessary during the design and implementation phases. Unlike approaches where architecture is always considered independently until reaching the implementation, the abstract architecture model defined in WebSA permits the unification of functional and non functional requirements earlier in the process.



Summarizing, the advantages of extending the architecture for MDA models can be named as follows:

1. Possibility of capturing the non functional requirements in order to improve the quality of the resulting Web Applications.

2. The reuse of the architecture models for different systems.

3. A more rigorous mapping between the domain model and the different component view models.

4. A better quality of the generated code, because it permits the definition of a generation mechanism where the functional and architectural parts are combined.

5. An easier connection with actual approaches that make use of PIMs to PSMs mappings.

This ideas are being applied in the context of the design and implementation of a CASE Tool called VisualWADE for the Web [VisualWADE 2002]. The goal is to improve the productivity in the development of web applications. Starting from WebSA, we are working on the definition of a set of MDA mappings to provide automated code-generation for that kind of applications. Such mappings are based on a domain-specific strategy (for web applications) as recommended in [Bettin 2002].

4. Conclusions and future work

In this paper we have presented a new proposal called WebSA whose main objective is improving the quality and the pace of development of Web Applications. This proposal includes, as its main features:

- An extension in the model architecture of MDA with a requirement viewpoint that contains functional and non functional requirements views.
- A software architecture viewpoint that fills the gap between analysis and design phases.
- A set of new analysis domain specific models for Web applications.
- A set of mapping rules from PIM to PIM and from PIM to PSM that permit to obtain complete Web Application models that take into account architectural concerns.

This work is far from completed. We are on the process of carrying out a complete definition of the UML profile of WebSA, which we expect to be incorporating in the VisualWADE CASE tool in the near future.

5. References

[Bass et all. 2000] Bass, L., Klein, M., Bachmann, F. "Quality Attribute Design Primitives" CMU/SEI-2000-TN-017, Carnegie Mellon, Pittsburgh, December, 2000

[Beck and Johnson 1994] Beck K., Johnson R. Patterns generate architectures. Proceedings of the 8th European Conference on Object-Oriented Programming. Bologna, Italy. 1994. pp 139-149.

[Bettin 2002] Jorn Bettin. Measuring the potential of domain-specific modelling techniques. DSVL (2002).

[Buschman et all. 1996] Frank Buschman, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: Pattern-Oriented Software Architecture – A System of Patterns; John Wiley & Sons Ltd. Chichester, England, 1996

[Conallen 2002] Jim Conallen. Building Web Applications with UML Second Edition. Adisón Wesley Longman. September 2002.

[Gamma et all. 1995] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design patterns : elements of reusable object-oriented software. Reading, Mass.: Addison-Wesley.

[Hassan and Holt 2002] Admed E Hassan, R.C. Holt. Architecture Recovery of Web Applications. ICSE'02, May 2002.

[Hofmeister et all. 1999] C. Hofmeister, R. L. Nord, D. Soni. *Siemens Corporate Research, Princeton, New Jersey, USA.. 1999*

[Koch and Kraus 2003] Koch N., Kraus A. (2003), The expressive power of UML-based engineering, Proceedings of the IWWOST'02, CYTED, pp. 105-119

[Kruchten 1995] Kruchten, P. (1995) The 4+1 View Model of Architecture, *IEEE Software*.

[OMG 2001] Architecture Board ORMSC. Model driven architecture. Technical report, OMG, July 9 2001.Document Number ormsc/2001-07-01.

[OMG 2002] EDOC. UML Profile for Enterprise Distributed Object Computing Specification. Febrary 2002.

[Renzel and Keller 1997] Renzel K., Keller W. Client/Server Architectures for Business Information Systems. A Pattern Language. PLoP'97 Conference.

[Schwabe et all. 1999] Schwabe, D., Almeida, R., & Moura, I. 1999a. Leveraging Template-based Website Implementations Using Design Methods. In: 8th International World Wide Web Conference.

[VisualWADE 2002] http://www.visualwade.com