Developing Web Applications For Different Architectures: The MoWebA Approach

Magalí González[1,2], Luca Cernuzzi, Nathalie Aquino[1,2]
[1]Department of Electronics and Computer Science
Catholic University "Nuestra Señora de la Asunción"
Asunción, Paraguay
mgonzalez@uca.edu.py, lcernuzz@uca.edu.py, nathalie.aquino@uca.edu.py

Oscar Pastor
[2]Research Center on Software Production Methods
Universitat Politècnica de València
Valencia, Spain
opastor@pros.upv.es

***Abstract* — This study presents the Architecture Specific Model (ASM) defined by the MoWebA approach to improve the development of web applications for different architectures. MoWebA is a model-driven approach to web applications development. The article presents a general overview of MoWebA, including the methodological aspects related to its modeling and transformation processes, the process of defining the ASM, and an example of an ASM model. We finally present a preliminary validation experience and its main results. The experience was structured according to the proposal of Runeson et al. for case studies.**

***Keywords* — model-driven architecture; model-driven development; web application; web methodology; architecture specific model**

I. Introduction

Web applications are increasingly growing and covering very different domains. One of their main characteristics is its underlying evolution. Functionalities of web applications evolve to provide services that are more relevant to potential users. Furthermore, those services are offered over different architectures and platforms. Take into account the evolution of web environments (considering technologies, platforms, architectures, diverse access devices, among others) is very important for improving the development of current web applications. Consequently, web methodologies need to be flexible, in order to consider these new tendencies.

In the landscape of web engineering methodologies [6] [19], there is a current trend on following the Model Driven Development approach (MDD) [3]. In this sense, Model Driven Architecture (MDA, http://www.omg.org/mda/) proposes diverse models at different levels of abstraction to follow MDD. It also proposes the use of several standard languages to express models. One of the strategies that MDA promotes to facilitate changeability is the prescription of a Platform Independent Model (PIM) separated from a Platform Specific Model (PSM). However, in practice, current web methodologies tend to cope with evolution trends by extending their modeling notations directly at the PIM level (e.g., the Rich Internet Application, or RIA proposal for WebML [2]). This extension usually results in an enriched PIM that includes characteristics and constraints that are related to a certain specific architecture. As a result, several proposals do not have a first phase for modeling a PIM, a second phase for extending the PIM with architectural details, and a third phase to generate the final code. Instead, they just have one modeling phase in which the PIM modeling is performed at the same time than architectural details are provided, and then the final code is generated. Although it could be argued that after extending a PIM to support a specific architecture, it is no longer a PIM, but a PSM, in general, the literature does not analyze this distinction.

The enriched PIM loses part of its independence from the architecture/platform, and becomes increasingly complex to understand and manage. The consequence is a loss of portability and reusability of models. Therefore, a very important issue is to find solutions to assure the easy evolution of web applications as well as to preserve the independence of the PIM and the portability of models for different architectures and platforms.

To this end, some authors have already proposed the introduction of an Architecture Specific Model (ASM) (e.g. [10], [18]). MDA proposes the PSM as a view of a system from the platform specific viewpoint. In this sense, platform refers to a specific technological platform (e.g., a specific programming language, operating system, document file format or database). Nevertheless, one ASM model can result in different PSM models, since one same architecture can be implemented in different platforms (e.g., the RIA architecture can be implemented in different platforms such as Backbase, Dojo, GWT, jQuery, among others).

Among other proposals, MoWebA (Model Oriented Web Approach) [10] takes into account new technological tendencies related to web applications by means of prescribing, not just the PIM, but also the solution modeling space: the ASM. MoWebA is a navigational, role-centric, model-based approach for the development of web applications. It covers methodological aspects and offers a complete environment to support modeling activities, transformations, and automatic code generation. MoWebA separates the PIM, ASM and PSM models, in order to facilitate the evolution of applications. With this separation, a clear distinction is made between what would be the problem space, presenting a model that is completely independent of the target architecture and platform; and the

This work has been partially developed with the support of CONACYT under the project "Mejorando el proceso de desarrollo de software: propuesta basada en MDD" (14-INV-056).

solution space, through the ASM, PSM and the final code. Specially, MoWebA captures the requirements of specific architectures in the ASM model, by means of a semi-automated process.

With these characteristics, MoWebA facilitates the development of Web 2.0 applications. In fact, dynamic web sites, where users actively interact with the web applications, are usually developed using technologies and architectures like SOA Services and RIA, among others.

The goal of this study is to present the ASM proposed by MoWebA and show its relevance by means of a first validation experience. In doing so, we are paving the way for a more rigorous validation of the proposal.

The rest of the article is structured as follows. Section 2 discusses related works. Section 3 presents a general overview of the MoWebA proposal and, more specifically, of the ASM definition. Section 4 focuses on a preliminary validation of the capability of the ASM of MoWebA to support the evolution of web applications. Finally, Section 5 presents conclusions and future works.

II. Related works

As previously commented, MoWebA is a model-driven, web engineering methodology. Mendes [17] presents a systematic review of web engineering research. Wakil and Jawawi [24] present a systematic mapping study of model-driven web engineering. In previous work [10], taking as a basis the work of Schwinger and Koch [22], we have presented a list of concerns related to model-driven web engineering and we have explained how MoWebA differs from other approaches when dealing with these issues. As a summary, we can say that MoWebA relies on: i) using navigational oriented modeling, that focuses on a functional (behavioral) perspective, more than on a structural (data organization oriented) one; ii) adopting standards that facilitate the interoperability between models, methods, and transformation rules; and iii) taking into account the rapid evolution of web environments. As its main strategy in order to deal with this rapid evolution, MoWebA proposes to add an ASM besides the PIM and PSM.

As it has been already mentioned, some web engineering approaches have included architectural aspects at the conceptual modeling level, without making a clear distinction between the independent model and the architectural one. For example, in order to generate RIA, some current approaches have extended their notations with additional architecture-specific primitives or patterns (e.g., WebML RIA [2], UWE for RIA [11]). In MoWebA, the same PIM could be used as a base to generate models for different architectures (e.g., RIA, REST, client-server, SOA) because architectural aspects are not included in the PIM but in the ASM model. Therefore, MoWebA makes a clear separation between the conceptual space and architectural aspects, defining them on different modeling abstraction levels. In this way, our approach offers enough flexibility to evolve into different architectures starting from the same PIM model.

Since this work focuses on the ASM, the remainder of this section presents other model-driven approaches that have included such architectural perspective.

As early as 2004, Mikkonen et al. [18] realized that architectural styles do not have a clear place in MDA. According to them, architectural styles can be seen as recurring architectures of various systems, especially when designing product families. They analyzed that these architectural styles can reside in PIM, PSM, or be distributed between them. However, as there is no unique place for architecturally significant design decisions, following a prescribed architectural style is hardened. Therefore, they proposed to modify MDA adding a new layer called Architecture Specific Model. This new layer encapsulates architectural properties in it. At the same time, it makes architectural styles, design patterns, and other important design decisions explicit in the model.

Afterwards, Manset et al. [15] defined an architecture-centric model-driven approach for the automatic generation of grid applications. In this approach, they merged model-driven development with architecture-centric [4] processes, which considers architecture as the main artifact in the software development process. Also in this direction, Marcos et al. [16] extended MIDAS, a methodological framework for the development of web information systems, by integrating architectural design aspects. MIDAS considers three different viewpoints of web information systems, namely content, hypertext and behavior, which are orthogonal to MDA abstraction levels (Computational Independent Model or CIM, PIM and PSM). Furthermore, the software architecture is conceived as a crosscutting perspective, which is in turn orthogonal to the mentioned three viewpoints. Therefore, both Platform-Independent Architecture and Platform-Specific Architecture models are defined. More recently, these efforts evolved into ArchiMeDes [21, 13], a model-driven framework for the specification of service-oriented architectures. At the PIM level, ArchiMeDes defines a domain-specific language that allows conceptual service architectures to be defined. At the PSM level, different domain-specific languages support the modelling of concrete execution platforms or implementation technologies.

ArchMDE [7, 8] is another architecture-centric model-driven engineering approach, and it focusses on the development and validation of embedded real time systems. In this case, the PIM is decomposed into two models: i) an Architectural-Style Independent Model, which defines the structure and behavior of the functional architecture of the real time system; and ii) an Architectural-Style Specific Model, which takes into account the functional, non-functional and architectural characteristics.

The proposals that have been presented in this section deal with different architectural aspects and propose corresponding models. They have served as an inspiration when defining the ASM in MoWebA. A difference with respect to our proposal is that, in MoWebA, the ASM has been defined with the goal of easing the evolution of web systems.

III. MoWebA in a nutshell

MoWebA defines methodological aspects (processes, stages, work products, dimensions) and complements them with an entire development environment, including modeling and transformation tools, automatic code generation, use of standards, and layered architecture, among others. For this reason, we refer to MoWebA as a "navigational, role-centric, model-based approach for the development of web applications" [10] [9].

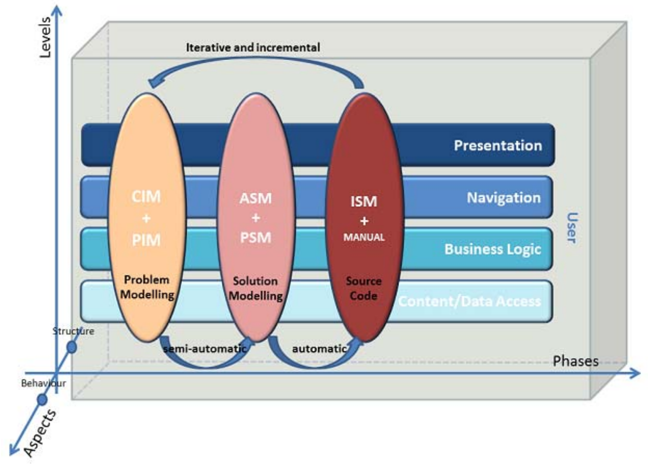


Fig. 1. MoWebA dimensions.

Fig. 1 presents the dimensions of MoWebA that cover its modeling and transformation processes. MoWebA adopts the MDA approach by identifying three different phases related to modeling and transformation activities (the *phases* dimension): i) the problem space, covered by CIM (Computational Independent Model) and PIM (Platform Independent Model); ii) the solution space, covered by ASM (Architectural Specific Model) and PSM (Platform Specific Model); iii) and the source code definition, covered by the ISM (Implementation Specific Model) and manual code. The *levels* dimension deals with complementary perspectives to be considered in every phase (content, business logic, navigation, presentation, users). Finally, the *aspects* dimension addresses structural and behavioral considerations for each perspective.

A. Modeling and transformation processes of MoWebA

MoWebA defines two main complementary processes: one related to modeling activities and the other one related to transformation activities. To formalize the modeling and transformation processes, it adopts the MOF language for the definition of the abstract syntax, and the UML profile extension for a precise definition of the modeling language (definition of the concrete syntax).

The *modeling process* (Fig. 2) encompasses seven stages and includes the activities that need to be performed in order to specify all diagrams that, together, represent the system-to-be (considering the problem space, architecture/s, and destination platform/s). This process considers the CIM, the PIM, the ASM and the PSM, with their corresponding modeling activities. The definition of the CIM covers late requirements identification, focusing on functional requirements specifications. The PIM specification is based on five models, offering a strong separation of concerns: Domain, Logic, Navigation, Presentation, and User. The ASM enriches the models with information for a specific architecture (e.g., RIA, SOA, REST, among others), and the PSM adds information related to the target platform (e.g., specific programming languages, frameworks, among others).

Stage 1 is related to requirements analysis. The artefact that is produced in this stage is a *Use Case* diagram that represents functional, navigational and usability requirements, as well as identifies potential users of the application. In stage 2, the navigational structure, roles and domain are specified. A *Navigational Tree* diagram is defined to organize the basic functionalities of the system in a hierarchical way. *Role and Zone* diagrams are created considering the potential users identified in stage 1. An *Entity* diagram defines the structure and the static relationships among classes identified in the domain of the problem. In stage 3, the navigational behavior of each node is defined through a *Node* diagram. Stage 4 uses the *Content* diagram to define which elements are going to be displayed on every presentation page. The structure of pages (position of headers, menus, footers, among others) is also defined in a *Structure* diagram. In addition, structural composition of business processes and transactional procedures are defined in a *Logic* diagram. In Stage 5, the main activity is to personalize the application through the *Adaptation* model. MoWebA proposes *Source and Rules* diagrams to model different kinds of adaptations. Stage 6 uses the Service diagram to provide a detailed definition of each service or action identified in the *Logic and Content* diagrams. Finally, stage 7 contemplates the architecture and platform aspects. It proposes an enrichment of existing models in order to consider aspects

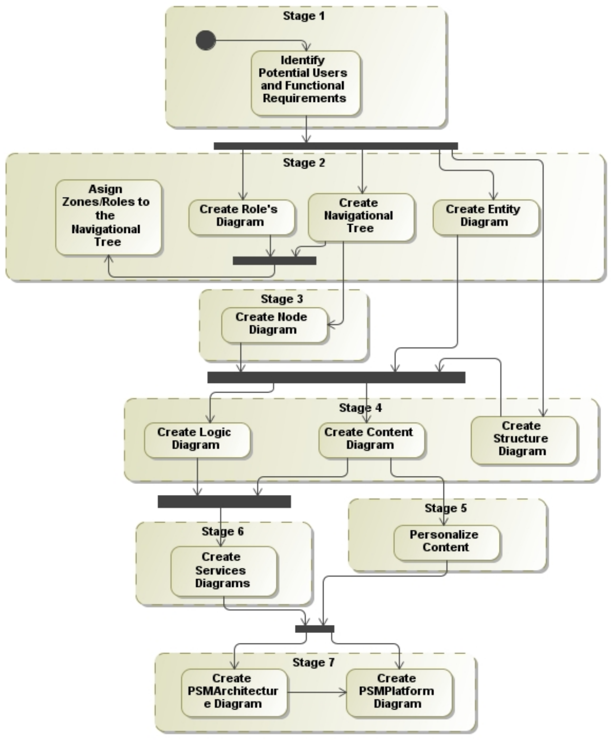


Fig. 2. MoWebA modeling process.

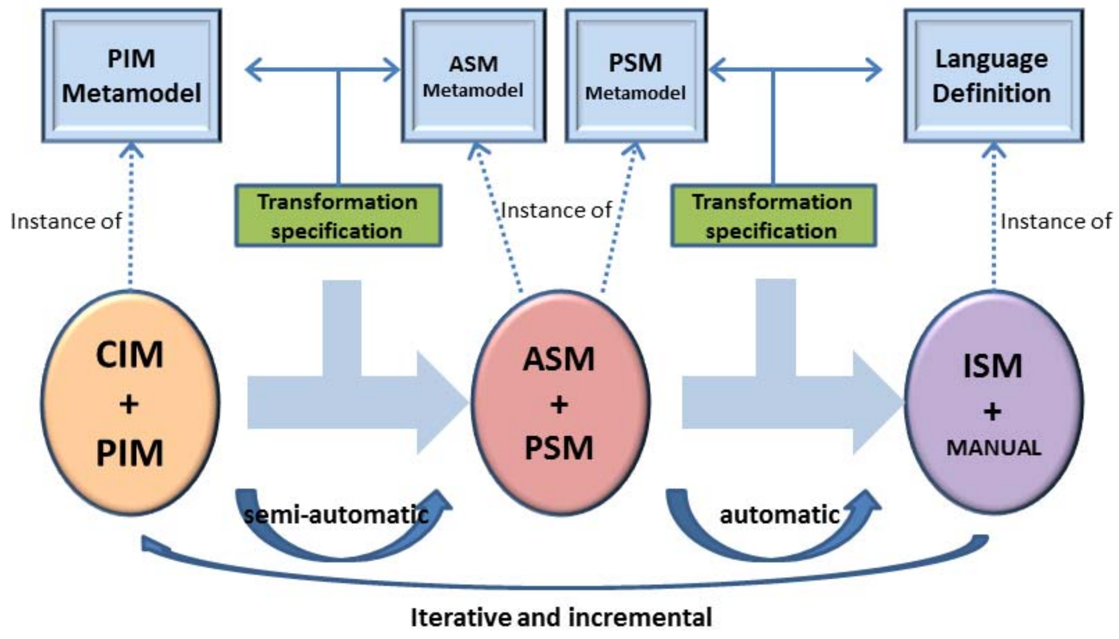


Fig. 3. MoWebA transformation process.

related to the final architecture of the system (e.g., RIA, SOA, REST), specifying the ASM diagram. The next step proposes to add platform specific information (e.g., Ruby on Rails, Python, PHP, Java), specifying the PSM diagrams.

The *transformation process*, on the other hand, is related to the steps, techniques, and tools that allow M2M (i.e., model-to-model) and/or M2T (i.e., model-to-code) transformations to be performed (Fig. 3). This process is based on the MDA approach and implies steps and activities that allow specifications to be performed, in order to go through each MoWebA phase (i.e., from CIM/PIM to ASM/PSM and from ASM/PSM to ISM/manual adjustments). The transformation that goes from CIM/PIM to ASM/PSM is performed in a semi-automatic way (i.e., introducing some manual adjustments), by defining metamodels for specific architectures or platforms, and the corresponding mapping rules for PIM to ASM/PSM transformations. The transformation that goes from ASM/PSM to ISM is an automatic transformation that goes from models to the application code. Since real experiences have shown that manual adjustments are necessary sometimes, we consider a phase of "manual adjustments", where additional code can be added to adapt the application. Finally, it is worth to mention that the transformation process can be performed iteratively, allowing an incremental application development.

B. *The ASM model*

The ASM model is obtained in the stage 7 of the MoWebA modeling process. This model is generated in a semi-automatic way, from diagrams defined during previous stages. ASM enriches previous models with additional information related to the system architecture (e.g., RIA, REST, among others). Complementary, the PSM is oriented to refine models adding information related to the platform and language that were selected for the final system (e.g., Java, .NET, PostgreSQL, among others). At this stage, software engineers are moving from the conceptual definition (CIM/PIM models) to the solution definition (ASM/PSM models).

In order to use an ASM Model, it must be defined first. This definition encompasses the specification of the corresponding metamodel, among other steps. Brambilla et al. have recommended a process for defining an abstract syntax [3]. MoWebA suggests this process to be used, in order to define an ASM metamodel. Furthermore, MoWebA complements the suggested process with additional steps that go from the definition of the concrete syntax till the generation of the final code of an application. The steps of this process are synthetized below:

1) *Define the metamodel using MOF.*
2) *Define the corresponding UML Profile.*
3) *Define tranformation rules for elements that can be obtained in an automatic way.*
4) *Apply tranformation rules in order to obtain the first version of the ASM model.*
5) *Make necessary manual adjustments to complete the ASM model.*
6) *Generate the PSM models, for selected platforms and/or the final code applying transformation rules.*
7) *Include manual adjustments, if necessary.*

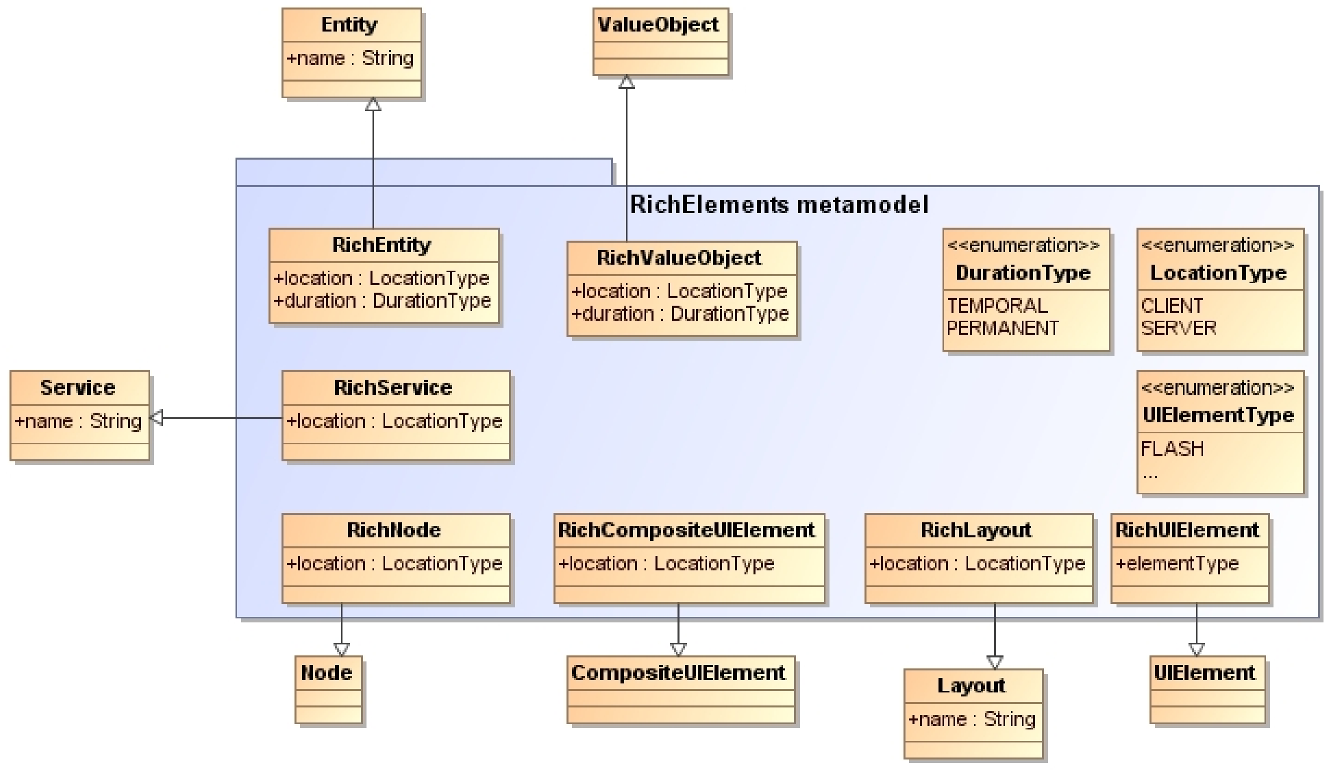


Fig. 4. Simplified example of *ASMRia* metamodel.

As it can be seen in the steps of the previous process, as a counterpart of offering a greater reusability of the PIM and facilitating the architectural evolution of web applications, the MoWebA approach requires some additional effort, including the need for the specification of ASM metamodels and the definition of the corresponding transformation rules, in order to achieve automatic transformations on the proposed architecture and platform. In any case, it should be noticed that steps 1, 2 and 3 of the process will only be executed once, when targeting to a new architecture for the first time.

To illustrate the relevance of the ASM model, let us show an example that instantiates the PIM for the RIA architecture, obtaining an *ASMRia* model. This example is part of an experience carried out as a preliminary validation of the approach. More details of this experience will be presented in section IV.

RIA are web applications in which data can be processed by both, the server and the client. The data exchange takes place in an asynchronous way, so that clients stay responsive while continuously recalculating or updating parts of the user

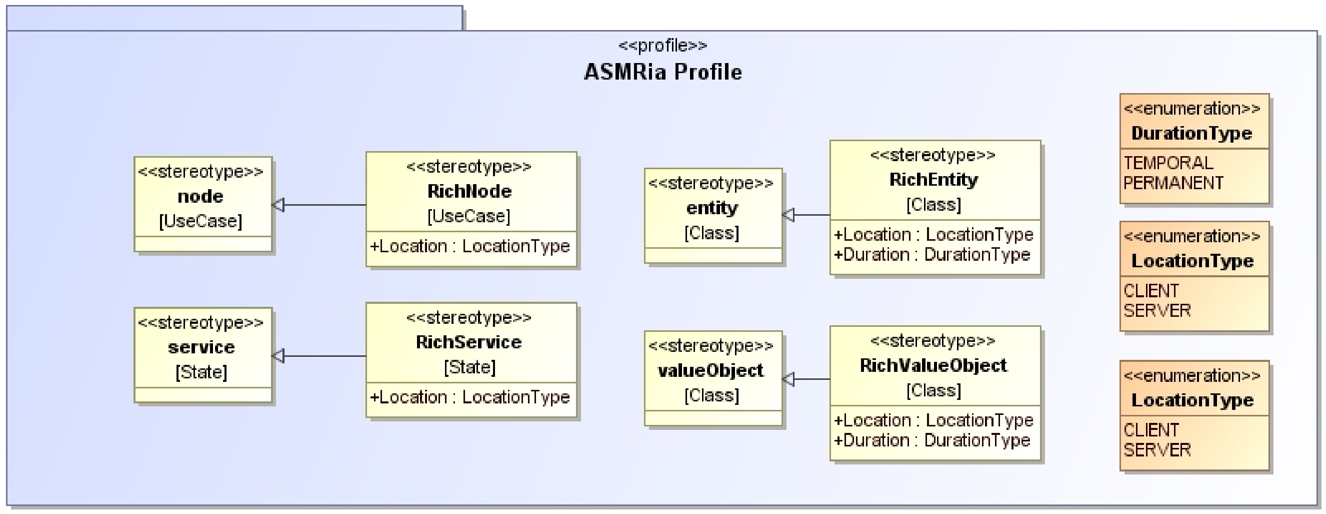


Fig. 5. Simplified example of *ASMRia* profile definition.

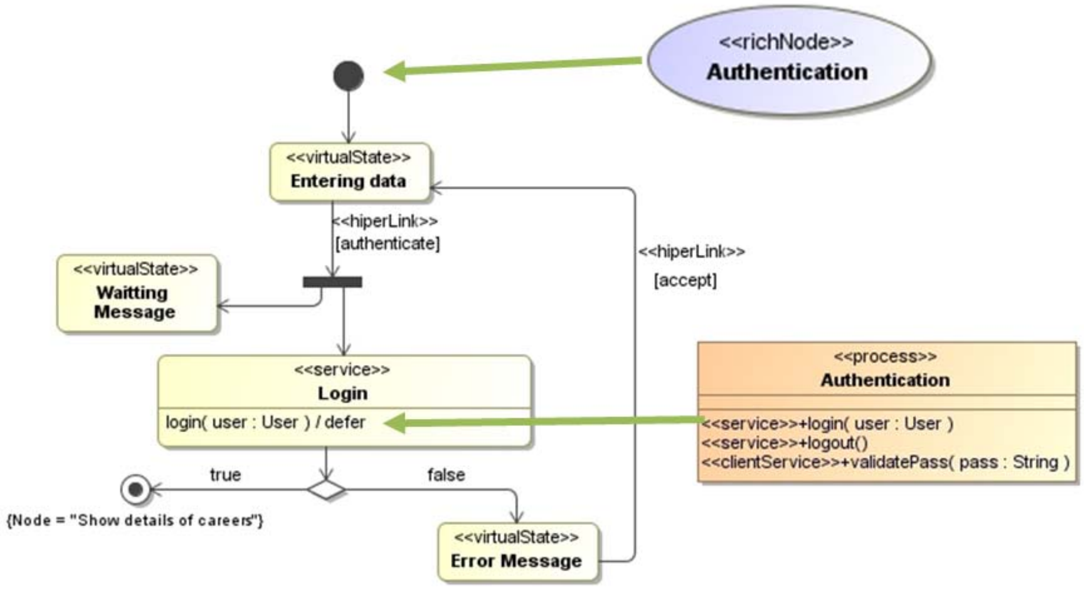


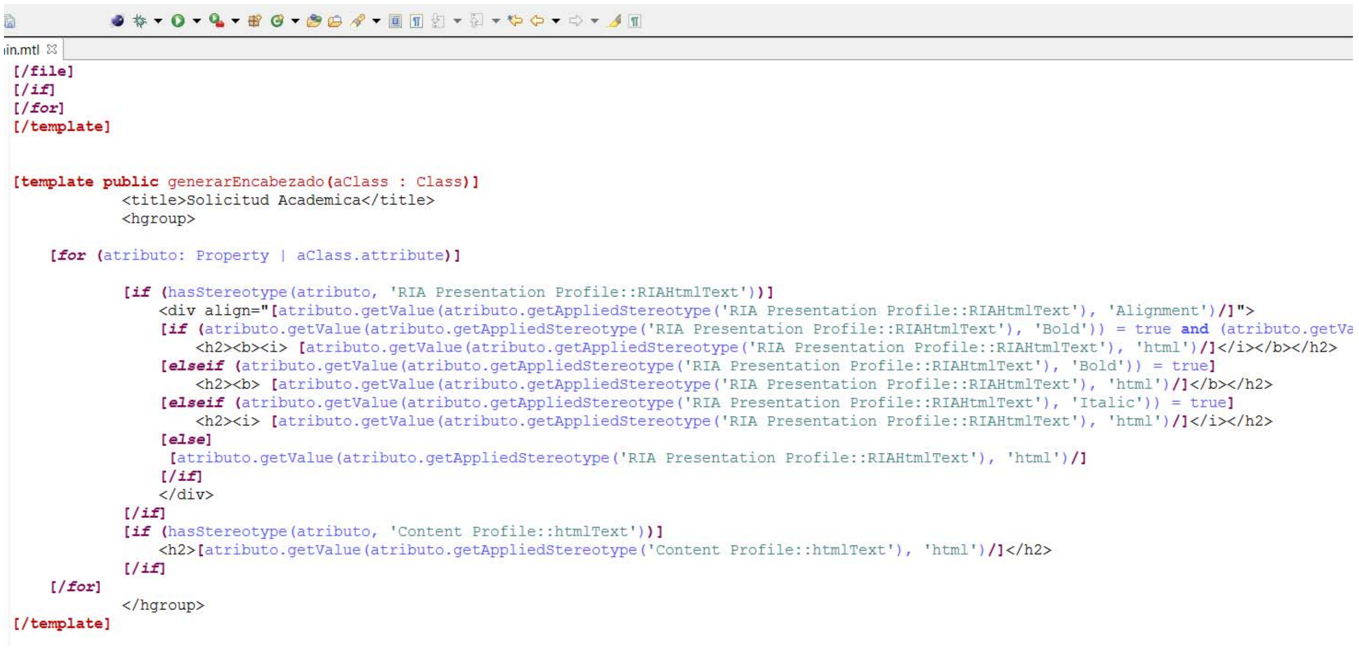
Fig. 6. Node diagram with the *ASMRia* extension.

interface. The main characteristics of RIA are: distributed computation of data and pages, asynchronous communication between client and server, and enhanced user interface behavior [5] [2].

Fig. 4 shows a simplified version of the *ASMRia* metamodel. It shows the extensions made on different elements with regard to distribution (client/server) and duration of persistent data and services.

In order to model these characteristics in an ASM model, MoWebA defines a series of *stereotypes* and *tagged values* in a *UML Profile*. Fig. 5 shows the profile that corresponds to the metamodel presented in Fig. 4.

As an example of an *ASMRia* model, Fig. 6 presents the navigational node diagram for the "Authentication" node. The navigational node "Authentication" is stereotyped with <<richNode>>, meaning that everything inside this node will be executed mostly on the client side. Asynchronous communication is achieved, for example, by transitions modeled after the "Entering data" virtual state, since user validation is processed on the server. An example of a client



```
[/file]
[/if]
[/for]
[/template]


[template public generarEncabezado(aClass : Class)]
            <title>Solicitud Academica</title>
            <hgroup>

    [for (atributo: Property | aClass.attribute)]

            [if (hasStereotype(atributo, 'RIA Presentation Profile::RIAHtmlText'))]
                <div align="[atributo.getValue(atributo.getAppliedStereotype('RIA Presentation Profile::RIAHtmlText'), 'Alignment')/]">
                [if (atributo.getValue(atributo.getAppliedStereotype('RIA Presentation Profile::RIAHtmlText'), 'Bold')) = true and (atributo.getVa
                    <h2><b><i> [atributo.getValue(atributo.getAppliedStereotype('RIA Presentation Profile::RIAHtmlText'), 'html')/]</i></b></h2>
                [elseif (atributo.getValue(atributo.getAppliedStereotype('RIA Presentation Profile::RIAHtmlText'), 'Bold')) = true]
                    <h2><b> [atributo.getValue(atributo.getAppliedStereotype('RIA Presentation Profile::RIAHtmlText'), 'html')/]</b></h2>
                [elseif (atributo.getValue(atributo.getAppliedStereotype('RIA Presentation Profile::RIAHtmlText'), 'Italic')) = true]
                    <h2><i> [atributo.getValue(atributo.getAppliedStereotype('RIA Presentation Profile::RIAHtmlText'), 'html')/]</i></h2>
                [else]
                 [atributo.getValue(atributo.getAppliedStereotype('RIA Presentation Profile::RIAHtmlText'), 'html')/]
                [/if]
                </div>
            [/if]
            [if (hasStereotype(atributo, 'Content Profile::htmlText'))]
                <h2>[atributo.getValue(atributo.getAppliedStereotype('Content Profile::htmlText'), 'html')/]</h2>
            [/if]
    [/for]
            </hgroup>
[/template]
```

Fig. 7. Example of transformation rules in Acceleo.

side service could be "validatePass", stereotyped with <<clientService>>. This service should be invoked at the presentation layer when the user sets a password in order to validate security levels. Finally, Fig. 7 presents an extract of transformations rules for the generation of final code. Transformation rules were implemented using Acceleo.

IV. ASM EXPERIENCES WITH MOWEBA: A PRELIMINARY VALIDATION

In this section, we present a preliminary validation of the ASM proposal, considering an experience of ASM definitions made by a group of computer science students at the Catholic University "Nuestra Señora de la Asunción" (Paraguay). The experience was structured taking into account a framework that Runeson et al. [20] have defined for case studies.

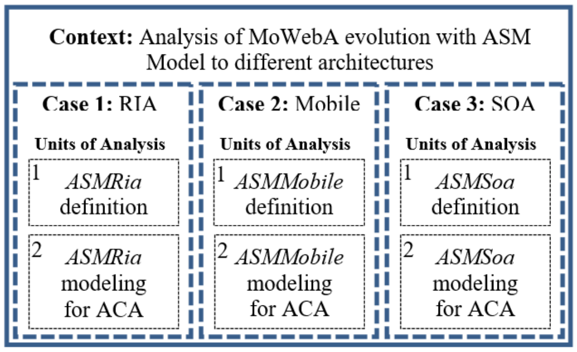
A. Motivation and goal

As previously presented in the related works section, we have identified a number of concerns related to the development of web applications. MoWebA intends to deal with these concerns. The experience described in this paper is focused on conducting a preliminary validation of one of these concerns, related to the evolution of web applications.

For this reason, the main goal of this experience is defined as "Investigate how the ASM model defined in MoWebA can help to easily evolve the development of web applications".

B. Cases and units of analysis

According to Runeson et al. [20], a case may be anything that is a contemporary software engineering phenomenon in its real-life settings. A case can be composed of one or more units of analysis. Furthermore, Yin distinguishes between holistic and embedded cases [25]. In this experience, we used a "multiple-embedded case study" approach (Fig. 8). Yin suggests this kind of study to be used when the case is inherently complex and there is a need to collect, analyze and report on many details. As the figure illustrates, an interesting way to analyze the evolution of MoWebA is to consider an experience of ASM definition for three different architectures (RIA, SOA, and mobile) and analyze the PIM, ASM and code by means of an example. The example used for the modeling process is called "Academic Credits Application (ACA)". ACA is intended to automate a process by which students request the recognition of extracurricular activities to gain academic credits.



Context: Analysis of MoWebA evolution with ASM Model to different architectures
Case 1: RIA
Units of Analysis
1 ASMRia definition
2 ASMRia modeling for ACA
Case 2: Mobile
Units of Analysis
1 ASMMobile definition
2 ASMMobile modeling for ACA
Case 3: SOA
Units of Analysis
1 ASMSoa definition
2 ASMSoa modeling for ACA

Fig. 8. Cases and units of analisys

Groups of three participants have carried out each case. Participants were students of the last year of study, with moderate expertise in programming languages, and with moderate experience in modeling using the MoWebA approach. The experience was performed in a period of four months, divided in six stages:

- *Stage 1:* overall presentation of the experience. In this stage, researchers presented the experience design, its goals, activities, tools, the example to be used (the ACA application), expected results, documentation to be elaborated. Furthermore, deadlines for activities were defined.
- *Stage 2:* PIM modeling and research about architectural context. In this stage, all groups worked together in order to obtain a unified PIM model for the ACA problem. Each group also worked on their architectural context (RIA, mobile, SOA) in order to understand the scope of the problem as well as general considerations that were necessary to define an ASM.
- *Stage 3:* definition of metamodel and UML profile. Each group defined a metamodel for its architecture (i.e., RIA, mobile, and SOA) and a first version of a corresponding ASM profile.
- *Stage 4:* ASM modeling of the ACA example, taking as a base the PIM model elaborated in stage 2.
- *Stage 5:* definition of transformation rules, from ASM to code, and generation of code.
- *Stage 6:* final presentations, in which students presented and explained their works.

Students used the following tools during the experience:

- *Magic Draw*: for the PIM modeling with MoWebA, and for the definition of metamodels (using MOF) and UML profiles.
- *Acceleo*: for the definition of transformation rules.

During the development of the experience, students decided the final platform for each architecture: the group working with RIA developed transformation rules targeting HTML5; the group working with the mobile architecture adopted Android as its platform; and the group working with the SOA architecture decided to use SOAP (Simple Object Access Protocol) and REST.

C. Research questions

Research questions are statements about the knowledge that is being sought, or is expected to be discovered, during the experience [1]. Considering the goal of this experience, we have defined the following research questions:

- *RQ1:* Can the same PIM model be used for different architectures?

- *RQ2:* Is it possible to specify clear limits between platform independent models (PIM) and architectural specific models (ASM)?
- *RQ3:* How does an architectural specific model facilitate the transformation rules definition?

D. Data collection

According to a classification of Lethbridge et al. [12], in this project, data collection can be classified as of first degree. We decided for first-degree data collection because we were able to use direct methods, since we were in direct contact with participants and we collected data in real time using different methods. Archival data was the primary source of data for the experience, complemented with open semi-structured interviews, observations and focus groups.

Interviews were approximately 30 minutes in length, with one or two researchers interviewing each group. Some notes were taken during interviews, but the main sources of data were results/outputs expected for each stage. Focus groups and interviews were done in a same meeting (at the end of stages 1 and 6). For stages 2 through 5, observations were done considering a category 3, according to the classification of Runeson et al. [20] (i.e. there was a low degree of interaction with researchers, but high awareness from groups being observed). In these stages, the main source of data was the outputs of each stage (models, research documents, metamodels, mapping rules, transformation rules, and source code). TABLE I. presents a summary of data collected during the experience.

E. Threats to validity

Since documentation was generated in the context of an experience developed in an academic environment, there are threats to validity that need to be considered. The following are some sensitive issues, with comments about decisions that were taken to intent to preserve the reliability of results.

One threat to validity could be the lack of sufficient knowledge about the MoWebA approach. However, it is important to mention that participants had moderate expertise and knowledge before the experience, including modeling of a complete application, and its subsequent implementation. We also considered a unified PIM model for every group. MDD experts with solid background on the MoWebA approach have analyzed this model.

Another aspect that could affect the results was the level of knowledge of the adopted architecture (i.e., SOA, RIA, and mobile). As it can be seen in the description of the experience, stage 2 has been focused on deepening knowledge on the architectural problem (section IV.B). Each group have presented a document with the theoretical framework and state of the art for each architecture. This document was analyzed and evaluated.

The ACA application was selected considering that it corresponds to a process that is well-known for every participant of the experience. We also consider that ACA has a reasonable degree of complexity (21 use cases, 23 entities, 7 different user profiles) to pass through PIM, ASM and code generation phases for analysis purposes.

Despite the fact that participants had moderate knowledge to work with MoWebA, before the beginning of the experience they did not have much familiarity with metamodeling and the MOF language. To address this point, theoretical and practical classes on the subject were lectured by MDD experts, in parallel with the advance of the experience. Remarks and tutoring that could help a better understanding of theoretical concepts were also made.

While participants did not have a deep knowledge related to model-to-code transformation rules, they hold a moderate knowledge in programing languages. In effect, besides being students in their last year of studies in the computer science field, the majority was already working in the field.

Limited time was another important factor, which mainly affected the development of model-to-code transformation rules. One of the groups was unable to complete the definition of model-to-code transformation rules (mobile group). This is reflected in the TABLE II.

TABLE I. SUMMARY OF DATA COLLECTED DURING THE EXPERIENCE

Data Collection Method	***Materials***	***Outputs***
1. Overall presentation: researchers conduct the presentation and leave for open questions, define three groups, each with three participants, and assign architecture for each group.		
- Focus group - Open and semi-structured interviews	- Slide presentation	- Groups were defined. - Architectures were selected. - Final platforms were selected.
2. PIM modeling: groups elaborate PIM models for the ACA example. Research about selected architectures in order to understand the problem.		
- Observation (category 3) - Archival data	- Magic Draw - MoWebA specification - Internet	- ACA PIM model - Theoretical framework and state of the art related to each architecture
3. Metamodel and UML profile: definition of architecture metamodels, considering the Brambilla et al. framework [3]. Definition of UML profiles.		
- Observation (category 3) -Archival data	- Magic Draw - Brambilla specification - UML profile specification	- MOF definition for each architecture - UML profile definition for each architecture - Documentation
4. ASM modeling: definition of mapping rules between the PIM metamodel and the ASM metamodel. Definition of the ASM model for the ACA example.		
- Observation (category 3) - Archival data	- Magic Draw	- ASM model for the ACA example for each architecture - Mapping rules definition
5. Transformations: definition of transformation rules from the ASM model to the final code. Generation of source code.		
- Observation (category 3) - Archival data	- Acceleo	- Transformation rules in Acceleo - Code generated
6. Final presentation: presentation of the whole experience. One researcher interviewed groups of three people. Everybody was allowed to ask questions.		
- Focus group -Open and semi-structured interviews	- Presentation slides - Acceleo	-Documentation

F. *Data analysis*

The variety of types of data collected during the experience implies that several approaches were used to organize data and to analyze it.

Analysis of data was mainly done in an iterative way. The results of each stage were analyzed in order to give feedback to each group before they started the next stage, giving them the opportunity to make improvements in their documents (outputs of the stage). The four mainly steps followed for the analysis were: identification of criteria, analysis of data, identification of metrics, and conclusions.

Researchers with solid knowledge in software engineering and model-driven engineering performed data analysis. For each criterion, we used one or more data collected in a previous stage. For example, we evaluated the understanding of architecture of each group using two sources: the theoretical framework and state of the art presented as results from the first stage, and the interviews made with each group. TABLE II. summarizes the results of data analysis.

TABLE II. SUMMARY OF DATA ANALYSIS

Data Analysis				
Criteria		***RIA***	***SOA***	***Mobile***
1	Understanding of architecture	90%	100%	80%
2	Quality of MoWebA PIM models[a]	95%	95%	95%
3	Number of elements defined in the metamodel	19	15	18
4	What percentage of the defined concepts are specific to the architecture?	80%	98%	95%
5	Are the PIM-ASM mappings clear?	Yes	Yes	Yes
6	Was it necessary to extend the PIM to represent concepts not considered in the metamodel?	No	No	No
7	Quality of metamodels	98%	100%	80%
8	Quality of ASM profiles	100%	100%	80%
9	Quality of ASM models	100%	100%	70%
10	Possible degree of PIM-ASM automation	92%	93%	50%
11	Quality of transformation rules	90%	100%	30%
12	Number of final platforms	1	2	1
13	LOC of transformation rules	301	109-44	92
14	Quality of generated code	90%	100%	30%
15	LOC of generated code	396	142-106	666
16	Degree of coverage of the code generated regarding the architectural specifications	95%	98%	50%

[a.] All groups used the same PIM model

From TABLE II. and the data collected it is possible to answer the research questions in the following way:

RQ1: Can the same PIM model be used for different architectures?

- The same PIM model was used for three different architectures without modifications. For this reason, this experience shows that the MoWebA proposal has the required constructors for architecture/platform independent modeling in the PIM phase, considering the web environment (see point 6 in TABLE II.).
- The ASM metamodel has reflected the specific concepts of architecture (see points 4 and 6 in TABLE II.).

RQ2: Is it possible to specify clear limits between platform independent models (PIM) and architectural specific models (ASM)?

- Metamodels and ASM profiles were good enough for mapping purposes and ASM modeling (see points 5, 7 and 9 in TABLE II.).
- A considerable good number of concepts of ASM models can be generated in a semi-automated way, from the PIM model (see points 5 and 10 in TABLE II.).

RQ3: How does an architectural specific model facilitate the transformation rules definition?

- Although this experience has been focused on analyzing the evolving capacities of web applications through the ASM model proposed by MoWebA, we have included activities related to the generation of final code for a specific platform. These activities allowed us to verify the degree to which proposed constructors defined in the ASM facilitated code generation for these architectures (see points 11, 12, 13, 14 and 15 in TABLE II.).

Some other considerations, more related to threats to validity are presented next:

- Each group achieved a reasonable good knowledge of the studied architecture (see point 1 in TABLE II.). This fact favored a good definition of the corresponding architectural metamodel.
- While it is important to notice that participants have already started the experience with previous moderate knowledge of MoWebA, the resulting models were good, so it was found a good understanding of the modeling process (see point 2 in TABLE II.).

G. *Final considerations*

In the experience carried out, regardless of the chosen architecture, there was no need to make changes to the PIM. However, in the case of the mobile architecture, the semi-automatic definition of the ASM was limited, because some elements between the PIM and ASM have been difficult to map. Despite this, it is important to emphasize that the separation between the two models (PIM and ASM) was maintained.

Some difficulties have arisen when defining transformation rules with *Acceleo*. These difficulties were mainly focused in the configuration and operation of the tool. Documentation and forums were not of much help, as they are not fully updated.

The percentage of ASM elements that were automatically obtained from the PIM is quite significant (see item 10, TABLE II.). However, in certain cases, human intervention was necessary in order to define additional properties that were exclusive to the selected architecture (e.g., human decision was needed to map the layout to an *accordion* or *tab*).

Some additional comments made by participants and/or researchers are listed below:

- *Regarding the RIA architecture*: i) the definition of the ASM has helped to better understand the architecture; ii) it is relatively simple (but not trivial) to introduce new or particular characteristics of an architecture in the ASM metamodel, which allows independence from PIM; iii) the metamodel development process proposed by Brambilla et al. [3] has facilitated the definition of the ASM metamodel.

- *Regarding the SOA architecture*: i) models were clear, the metamodel was complete, and mapping rules were well established; ii) transformation rules were well organized and structured; iii) it was possible to generate code for SOAP and REST (with languages php and java).

- *Regarding the mobile architecture*: i) not all concepts were covered by the metamodel and some definitions were unnecessary; ii) the separation of concerns was slightly lost, since value objects were not covered; iii) transformation rules were limited to generate classes, attributes and operations in Java.

Considering the global considerations about this first experience, we are positive about the usefulness of the ASM in the way prescribed by MoWebA. However, more structured and formal experiments should offer a better insight about the proposal.

V. CONCLUSION AND FUTURE WORK

This study has presented the Architectural Specific Model (ASM) proposed by MoWebA, an approach for the development of web applications. In MoWebA, the ASM establishes an architectural level of modelling separated from the PIM, to facilitate the evolution of web applications, considering new tendencies. Based on the results of a preliminary experience, this study has analyzed how the MoWebA ASM model can help to evolve more easily the development of web applications.

Results were quite encouraging. They have stimulated new on-going experiences, case studies and more rigorous experiments. These include the definition of ASM for other architectures; the application of MoWebA and other methodologies (UWE, OOHDM, OO-H, WebML) to the same real applications; among others.

With these preliminary results, we believe that MoWebA has sufficient flexibility to support innovative technologies, such as those typical of Web 2.0 (e.g., *ASMRia* extension). For a more rigorous validation of these considerations, proofs of concepts and case studies are being planned. These studies will be focused on building applications with current technologies (e.g. RIA, REST, cloud computing) that facilitate the development of Web 2.0 applications. Furthermore, it will be very interesting to compare MoWebA against competing approaches as well as against approaches which are not based on models and automatic transformations. Comparisons could include metrics for development time and others. Maintenance activities should be included, in order to analyze evolution capabilities.

Finally, we should also analyze how ideas from approaches such as SANTA (Solution Architecture for N-Tier Applications) or MAAG (Microsoft Application Architecture Guide) could be integrated in MoWebA. These approaches define meta-architectures, structured in several layers [14] [23]. They serve as reference models for developing applications for a specific architecture (e.g., service-oriented cloud-based web and mobile applications). Assuming the integration is made; resulting models would present architectural non-functional qualities such as maintainability, adaptability, understandability, among others. This would be a significant enhancement in MoWebA, since it would go from a stage in which only functional requirements are being taken into account, to a new stage, in which important non-functional requirements are also considered.

REFERENCES

[1] Carina Andersson and Per Runeson. A spiral process model for case studies on software quality monitoring - method and metrics. *Software Process: Improvement and Practice*, 12(2):125–140, 2007.

[2] A Bozzon, S Comai, P Fraternali, and G Toffetti. Capturing ria concepts in a web modeling language. New York, USA, 2006.

[3] M Brambilla, J Cabot, and M Wimmer. *Model-Driven Software Engineering in Practice*. Morgan&Claypool, USA, 2012.

[4] Manfred Broy. Architecture driven modeling in software development. In *9th International Conference on Engineering of Complex Computer Systems (ICECCS 2004), 14-16 April 2004, Florence, Italy*, pages 3–12. IEEE Computer Society, 2004.

[5] M Busch and N Koch. Rich internet application. state of the art. Technical report, Munchen, Germany, 2009.

[6] Y Deshpande, S Murugesan, A Ginige, S Hansen, D Schwabe, M Gaedke, and B White. Web engineering. *Journal of Web Engineering*, 1 (1):3–17, 2002.

[7] Nourchène Elleuch, Adel Khalfallah, and Samir Ben Ahmed. Archmde approach for the development of embedded real time systems. In Nabil Abdennadher and Fabrice Kordon, editors, *Reliable Software Technologies - Ada Europe 2007, 12th Ada-Europe International Conference on Reliable Software Technologies, Geneva, Switzerland, June 25-29, 2007, Proceedings*, volume 4498 of *Lecture Notes in Computer Science*, pages 142–154. Springer, 2007.

[8] Nourchène Elleuch, Adel Khalfallah, and Samir Ben Ahmed. Archmde approach for the formal verification of real time systems. In *11th IEEE International Conference on Computer and Information Technology, CIT 2011, Pafos, Cyprus, 31 August-2 September 2011*, pages 533–538. IEEE Computer Society, 2011.

[9] M González, J Casariego, JJ Bareiro, L Cernuzzi, and O Pastor. A mda approach for navigational and user perspectives. *Clei Electronic Journal*, 14:1, 2011.

[10] M González, L Cernuzzi, and O Pastor. A navigational role-centric model oriented web approach - moweba. *International Journal of Web Engineering and Technology*, 2016 (to be published).

[11] N Koch, M Pigerl, G Zhang, and T Morozova. Patterns for the model-based development of rias. San Sebastian, Spain, 2009.

[12] Timothy C. Lethbridge, Susan Elliott Sim, and Janice Singer. Studying software engineers: Data collection techniques for software field studies. *Empirical Software Engineering*, 10(3):311–341, 2005.

[13] Marcos López-Sanz and Esperanza Marcos. *Modeling Platform-Independent and Platform-Specific Service Architectures with UML and the ArchiMeDeS Framework*, chapter 11, pages 254–277. IGI Global, 2014.

[14] Leszek A. Maciaszek, Tomasz Skalniak, and Grzegorz Biziel. *Business Modeling and Software Design: 4th International Symposium, BMSD 2014, Luxembourg, Luxembourg, June 24-26, 2014, Revised Selected Papers*, chapter Architectural Principles for Service Cloud Applications, pages 1–21. Springer International Publishing, Cham, 2015.

[15] David Manset, Hervé Verjus, Richard McClatchey, and Flávio Oquendo. A formal architecture-centric model-driven approach for the automatic generation of grid applications. In Yannis Manolopoulos, Joaquim Filipe, Panos Constantopoulos, and José Cordeiro, editors, *ICEIS 2006 - Proceedings of the Eighth International Conference on Enterprise Information Systems: Databases and Information Systems Integration, Paphos, Cyprus, May 23-27, 2006*, pages 322–330, 2006.

[16] Esperanza Marcos, César J. Acuña, and Carlos E. Cuesta. Integrating software architecture into a MDA framework. In Volker Gruhn and Flávio Oquendo, editors, *Software Architecture, Third European Workshop, EWSA 2006, Nantes, France, September 4-5, 2006, Revised Selected Papers*, volume 4344 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2006.

[17] Emilia Mendes. A systematic review of web engineering research. In *2005 International Symposium on Empirical Software Engineering (ISESE 2005), 17-18 November 2005, Noosa Heads, Australia*, pages 498–507. IEEE Computer Society, 2005.

[18] Tommi Mikkonen, Risto Pitkänen, and Mika Pussinen. On the role of architectural style in model driven development. In Flávio Oquendo, Brian Warboys, and Ronald Morrison, editors, *Software Architecture, First European Workshop, EWSA 2004, St Andrews, UK, May 21-22, 2004, Proceedings*, volume 3047 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 2004.

[19] R Pressman and D Lowe. *Web Engineering: A Practitioner's approach*. McGraw-Hill, New York, 2009.

[20] Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell. *Case Study Research in Software Engineering - Guidelines and Examples*. Wiley, 2012.

[21] Marcos López Sanz and Esperanza Marcos. Archimedes: A model-driven framework for the specification of service-oriented architectures. *Inf. Syst.*, 37(3):257–268, 2012.

[22] Wieland Schwinger and Nora Koch. *Modeling Web Applications*, chapter 3, pages 39–64. Wiley, 2006.

[23] T. C. Shan and W. W. Hua. Solution architecture for n-tier applications. In *2006 IEEE International Conference on Services Computing (SCC'06)*, pages 349–356, Sept 2006.

[24] Karzan Wakil and Dayang N. A. Jawawi. Model driven web engineering: A systematic mapping study. *e-Informatica*, 9(1):87–122, 2015.

[25] R Yin. *Case Study Research: Design and Methods, 3rd Edition*. SAGE Publications, 2003.