UNIVERSIDAD POLITECNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

A Navigational Role-Centric Model Oriented Web Approach MoWebA

Candidate:

Magalí González

Directors:

Oscar Pastor
Luca Cernuzzi

Tesis para optar por el Título de Doctor en Informática por la Universitat Politècnica de València.

– December 2021 –

Author's e-mail: mgonzalez@uc.edu.py

Author's address:

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia
España

Agradezco y ofrezco este trabajo Dios Todopoderoso, que ha estado siempre presente en mi vida a través de signos y rostros concretos, dándome la fuerza para seguir adelante en este gran desafío que la vida me ha dado.

Agradecimientos

Me gustaría agradecer primeramente a mis directores, el Dr. Oscar Pastor López y el Dr. Luca Cernuzzi, por el apoyo, rigor y orientación brindados. Agradezco la confianza que han depositado en mí y la oportunidad que he tenido de aprender de ellos.

Quisiera agradecer especialmente a Oscar por darme la posibilidad de vivir una experiencia inolvidable y enriquecedora en el grupo de investigación PROS, por sus valiosos aportes, así como su inmensa paciencia.

A Luca, que me ha acompañado en la tutoría de mi proyecto final de grado, y luego ha abierto las puertas para que pueda seguir con mis estudios de doctorado, aceptando, además, la dirección de este trabajo de Tesis. Agradezco su invaluable trabajo y apoyo desde el inicio de la Tesis Doctoral, ya que ha estado siempre disponible para guiarme en este largo camino recorrido.

Doy gracias, igualmente, al Departamento de Electrónica e Informática de la Universidad Católica “Nuestra Señora de la Asunción”, y al grupo de investigación que ha participado del proyecto denominado “Mejorando el proceso de desarrollo del software: una propuesta basada en MDD”, financiado por el CONACYT, ya que con los resultados de sus proyectos de grado y máster han dado valiosos aportes a la validación de este trabajo.

No quisiera dejar de hacer una mención especial a mis alumnos, que han adoptado esta propuesta en sus proyectos de grado.

Agradezco a mi padre, Andrés Eugenio (que ya está viendo este logro desde el Cielo), y a mi madre, Alba, quienes han realizado grandes sacrificios por mí y mis hermanos, y me han apoyado durante toda la carrera universitaria.

Finalmente, agradezco a mi esposo, Gustavo, a mis hijos María Emilia, Andrea Guadalupe e Ignacio Gabriel, que son el motor de mi vida y el signo más grande del amor de Dios hacia mi persona.

Abstract

Some of the major challenges facing Web applications today are those of portability, adaptability and evolution, not only in the environment in which they run, but also in the way in which they must be developed, often requiring different languages, frameworks, tools, environments, platforms, etc. MDD and MDA take into account these issues. However, to achieve portability, adaptability and evolution depends to some extent on the degree of independence that the models adopt.

This Thesis presents a method that take into account the problem of evolution and portability towards different environments. The approach is called MoWebA (Model Oriented Web Approach). Some key aspects of MoWebA that could have a positive impact in the portability and adaptability are:i) incorporation of an Architecture Specific Model (ASM) as a new modeling layer, in order to keep the portability of the Platform Independent Model (PIM) regarding the different architectures (e.g., RIA, SOA, Mobile); ii) clear separation of the presentation layer with regard to the navigation and behavior layers; iii) definition of the navigational structure according to a function-oriented approach, which prevents the modification of the navigation design caused by implementation changes; iv) and use of standards in order to facilitate the independence from the tools.

We justify MoWebA by highlighting a series of concerns for Web applications development. We present an overview of the method including the dimensions and the diagrams that we propose. Subsequently, we present each step in the modeling process, including the diagrams and notation, its definition (metamodels) and examples of use. Afterwards, we present the transformation process adopted by MoWebA, which includes model-to-model and model-to-code transformations.

We have devoted special attention to the validation of the approach. As a first validation, MoWebA has been used for modelling and generating different types of applications by both novice and experienced modellers and developers. These experiences were done in academic and industrial contexts. The experi-

ences have allowed to identify strengths and weaknesses of the PIM proposal, and to verify that the proposed notation covers the needs of different domains. Next, we present a preliminary validation of the ASM proposal, considering an experience of different ASM definitions made by a group of computer science students at the Catholic University “Nuestra Señora de la Asunción” (Paraguay). This preliminary validation has allowed us to determine how feasible is to adapt the proposal to other architectures. The analysis of the validation sought to answer the following questions: Can the same PIM model be used for different architectures?; Is it possible to specify clear limits between platform independent models (PIM) and architectural specific models (ASM)?; How does an architectural specific model facilitate the transformation rules definition?. Finally, we present a Case Study to validate the extensions of MoWebA to three different architectures. The experience was structured taking into account the framework of Runeson et al. [114]. The Case Study was done as part of a research project led by the Catholic University of Asunción called "Mejorando el Proceso de Desarrollo de Software: Una propuesta basada en MDD"[1], grant 14-INV-056 of CONACYT (Consejo Nacional de Ciencias y Tecnología), Paraguay. This experience have allowed to carry out three complete extensions. In such extensions we could analysed the grade of adaptability of MoWebA and of automation PIM-ASM, as well as the grade of independence of the PIM metamodel. We have also conducted some user's satisfaction experiences with modelers and developers.

[1]`https://www.dei.uc.edu.py/proyectos/mddplus/`

Resumen

Entre los grandes desafíos actuales de las aplicaciones Web podemos citar la portabilidad, adaptabilidad y evolución. Estos desafíos se dan tanto en el ambiente en que operan así como en su desarrollo, ya que a menudo requieren de diferentes lenguajes, frameworks, herramientas, entornos, plataformas, etc. El Desarrollo Dirigido por Modelos (MDD) y en particular, la Arquitectura Dirigida por Modelos (MDA) contemplan estos desafíos proponiendo distintos niveles de abstracción para las diferentes fases de modelado, partiendo de modelos más orientados al problema, que no deberían contemplar aspectos de implementación (CIM, PIM), hasta llegar a los modelos más orientados a la solución planteada (PSM, ISM). Sin embargo, encarar adecuadamente este tema puede depender del grado de independencia que adquieren los modelos.

Este trabajo de Tesis presenta un método que considera el problema de la evolución y portabilidad hacia diferentes entornos o arquitecturas en el diseño y desarrollo de aplicaciones Web. La propuesta se denomina MoWebA (*del inglés*, Model Oriented Web Approach). Durante el desarrollo de la tesis, hemos identificado una serie de aspectos que podrían tener un impacto positivo sobre los problemas de portabilidad y adaptabilidad que son: i) incorporación de un modelo específico de la arquitectura ASM (*del inglés*, Architecture Specific Model), que permita la portabilidad de los PIMs con respecto a la arquitectura (por ejemplo, RIA, SOA, Mobile); ii) clara separación de la capa de presentación con respecto a las capas navegacionales y de comportamiento; iii) definición de la estructura navegacional de acuerdo a un mecanismo orientado al comportamiento, que prevenga la modificación del diseño navegacional causada por cambios en la implementación; iv) uso de estándares para facilitar la independencia de las herramientas.

Se ha llevado a cabo una revisión del estado de la literatura teniendo en cuenta los fundamentos de la Ingeniería Dirigida por Modelos MDE (*del inglés*, Model Driven Engineering), las propuestas metodológicas para el desarrollo de

Aplicaciones Web, las tendencias actuales, analizando en forma especial de qué manera las propuestas Web las contemplan, y las evidencias empíricas tanto en la academia como en la Industria.

Presentamos así la propuesta MoWebA a partir de una serie de consideraciones sobre las aplicaciones Web que han sido identificadas durante el proceso de revisión del estado del arte. Posteriormente, detallamos cada una de las etapas del proceso de modelado, incluyendo los diagramas y notaciones propuestos, sus definiciones a través de sus metamodelos y ejemplos de uso. Seguidamente, presentamos el proceso de transformación adoptado por MoWebA, con los mecanismos de transformación Modelo-A-Modelo y Modelo-A-Código.

Hemos dedicado un importante esfuerzo en la validación de la propuesta. Se realizó una primera validación, adoptando los modelos de MoWebA y en algunos casos generando aplicaciones para diferentes dominios. Estas experiencias fueron realizadas por modeladores con poca experiencia, así como también modeladores y desarrolladores experimentados. Las experiencias fueron desarrolladas en ambientes tanto académicos como industriales. Con estas primeras experiencias de validación hemos podido identificar fortalezas y debilidades de la propuesta PIM de MoWebA, además de determinar en qué grado la misma cubre las necesidades de diferentes dominios. Como segunda experiencia, hemos realizado una validación preliminar con un grupo de estudiantes de último año de la carrera de Ingeniería Informática de la Universidad Católica "Nuestra Señora de la Asunción" (Paraguay), que consistió en la definición de ASM para tres arquitecturas diferentes. Con esta segunda experiencia de validación hemos logrado valorar qué tan factible es adaptar la propuesta a otras arquitecturas, guiados por las siguientes preguntas de investigación: i) ¿Puede un mismo modelo PIM ser utilizado como punto de partida para aplicaciones que adoptan diferentes arquitecturas?; ii) ¿Es posible especificar límites claros entre el PIM y el ASM?; iii) ¿De qué manera un ASM facilita la definición de reglas de transformación?. Finalmente, como tercera experiencia de validación, hemos llevado a cabo un Caso de Estudio que consistió en la definición de tres extensiones para MoWebA. La experiencia fue llevada a cabo siguiendo los lineamientos propuestos por Runeson et. al [114]. Dicha validación fue desarrollada como parte de un proyecto de investigación liderado por la Universidad Católica "Nuestra Señora de la Asunción",

denominado "Mejorando el Proceso de Desarrollo de Software: Una propuesta basada en MDD"[2], proyecto 14-INV-056 co-financiado por CONACYT (Consejo Nacional de Ciencias y Tecnología), Paraguay. La experiencia ha permitido analizar el grado de adaptabilidad de MoWebA y automatización en las transformacinoes PIM-ASM, así como el grado de independiencia del PIM. Durante esta validación, se han realizado además experiencias de satisfacción de usuarios con un grupo de modeladores y desarrolladores.

[2]https://www.dei.uc.edu.py/proyectos/mddplus/

Resum

Entre els grans desafiaments actuals de les aplicacions Web podem citar la portabilitat, l'adaptabilitat i l'evolució. Aquestos reptes es donen tant en l'àmbit en què operen així com en el seu desenvolupament, ja que sovint requereixen de diferents llenguatges, frameworks, eines, entorns, plataformes, etc. El Desenvolupament Dirigit per Models (MDD), i en particular l'Arquitectura Dirigida per Models (MDA), contempla aquests reptes proposant diferents nivells d'abstracció per a les diferents fases de modelatge. Aquestes estratègies s'inicien amb models més orientats cap al problema, que no consideren aspectes d'implementació (models anomenats CIM i PIM), que van després refinant-se i transformant-se fins a arribar als models més orientats cap al domini de la solució (models PSM i ISM). No obstant això, endreçar adequadament aquest problema depén del grau d'independència entre els models.

Aquest treball de Tesi presenta un mètode que considera el problema de l'evolució i la portabilitat, en diferents entorns o arquitectures, per al disseny i desenvolupament d'aplicacions Web. La proposta s'anomena MoWebA (*de l'anglés*, Model Oriented Web Approach). Durant el desenvolupament de la tesi hem identificat una sèrie d'aspectes que podrien tenir un impacte positiu sobre els problemes de portabilitat i adaptabilitat. Aquestos són: i) la incorporació d'un model específic d'arquitectura ASM (*de l'anglés*, Architecture Specific Model), que permet la portabilitat dels models PIM respecte a l'arquitectura (per exemple, RIA, SOA, Mobile); ii) la clara separació de la capa de presentació respecte a les capes navegacionals i de comportament; iii) la definició de l'estructura navegacional d'acord amb un mecanisme orientat al comportament, que restringisca la modificació del disseny navegacional causada per canvis en la implementació; i per últim, iv) l'ús d'estàndards per tal de facilitar la independència de les eines.

S'ha realitzat una revisió de l'estat de la literatura considerant els fonaments de l'Enginyeria Dirigida per Models MDE (*de l'anglés*, Model Driven Engineering), les propostes metodològiques per al desenvolupament d'Aplicacions Web, les tendències actuals (analitzant de forma especial la manera en que les propostes Web les consideren), i les evidències empíriques tant en l'acadèmia com en la Indústria.

Presentem així la proposta MoWebA a partir d'una sèrie de consideracions sobre les aplicacions Web que són identificades durant el procés de revisió de l'estat de l'art. Posteriorment, detallem cadascuna de les etapes del procés de modelatge, incloent els diagrames i les notacions proposades, les seues definicions (a través dels seus metamodels), i alguns exemples d'ús. Seguidament, presentem el procés de transformació adoptat per MoWebA, emprant mecanismes de transformació Model-a-Model i Model-a-Codi.

Hem dedicat un esforç considerable en la validació de la proposta. Es realitzà una primera validació adoptant els models de MoWebA, i en alguns casos es generaren aplicacions per a diferents dominis. Aquestes experiències van ser realitzades per modeladors amb poca experiència, així com també per modeladors i desenvolupadors experimentats. Les experiències van ser desenvolupades en ambients tant acadèmics com industrials. Amb aquestes primeres experiències de validació s'identificaren fortaleses i febleses de la proposta PIM de MoWebA. També permeteren determinar fins quin grau la proposta respon a les necessitats dels diferents dominis.

Com a segona experiència es realitzà una validació preliminar amb un grup d'estudiants d'últim curs de la carrera d'Enginyeria Informàtica de la Universitat Catòlica "Nuestra Señora de la Asunción" (Paraguai), que va consistir en la definició del model d'arquitectura ASM per a tres arquitectures diferents. Amb aquesta segona experiència de validació s'aconseguí valorar com de factible és d'adaptar la proposta a altres arquitectures. Per a aconseguir-ho, s'empraren les següents preguntes de recerca: i) pot un mateix model PIM ser emprat com a punt de partida per desenvolupar aplicacions que adopten diferents arquitectures?; ii) és possible especificar límits clars entre el PIM i el ASM?; iii) de quina manera un ASM facilita la definició de regles de transformació?

Finalment, com a tercera experiència de validació, es dugué a terme un Cas d'Estudi que va consistir en la definició de tres extensions per a MoWebA. L'experiència va ser duta a terme seguint les línies proposades per Runeson et. al [114]. Aquesta validació es desenvolupà en el marc d'un projecte de recerca liderat

per la Universitat Catòlica "Nuestra Señora de la Asunción", anomenat “Mejorando el Proceso de Desarrollo de Software: Una propuesta basada en MDD"[3], projecte 14-INV-056 co-finançat per CONACYT (Consell Nacional de Ciències i Tecnologia), Paraguai. L'experiència permeté analitzar el grau d'adaptabilitat i d'automatització en les transformacions PIM-ASM que s'obté amb MoWebA, així com el grau de independència del PIM amb respecte als altres models. Durant aquesta validació, s'han realitzat a més experiències de satisfacció d'usuaris amb un grup de modeladors i desenvolupadors.

[3]`https://www.dei.uc.edu.py/proyectos/mddplus/`

Contents

1 Introduction and Motivation 1

1.1 Problem Statement 2

1.2 Research Method 4

1.2.1 Research Goals and Questions 5

1.2.2 Problem Investigation 7

1.2.3 Treatment Design 7

1.2.4 Treatment implementation 8

1.2.5 Treatment Validation 9

1.2.6 Conclusion 9

1.3 Thesis Outline 10

2 Theoretical Framework and Related Works 13

2.1 Advances in Model Driven Engineering 13

2.2 Model Driven Web Engineering 17

2.2.1 Web Methods Approaches 20

2.2.2 Other Specific Proposals on Current Trends 31

2.3 MDE Evidences in Academia and Industry 34

2.4 Final Discussions 36

3 MoWebA: Model Oriented Web Approach 37

3.1 MoWebA in a nutshell 39

3.2 The Modeling process 42

3.2.1 Stage 1: Identify Potential Users and Functional Requirements 44

3.2.2 Stage 2: Specify Navigational Structure, User roles and Domain 44

3.2.3 Stage 3: Specify Navigational Behaviour 50

3.2.4 Stage 4: Specify Logic Behaviour and Presentation 55

3.2.5 Stage 5: Specify Personalisation 63

3.2.6 Stage 6: Detail Navigational, Logic, Adaptation and Presentation Services . . . 67
3.2.7 Stage 7: ASM and PSM definition . . . 69
3.3 MoWebA Transformation Process . . . 75
3.4 Summary of the chapter . . . 82

4 Validation Experiences of MoWebA 83
4.1 Adopting MoWebA: some experiences . . . 83
4.2 A Preliminary Validation Experience with the Architectural Specific Model . . . 87
4.2.1 Motivation and Goal . . . 87
4.2.2 Cases and Unit of Analysis . . . 88
4.2.3 Research questions . . . 90
4.2.4 Data Collection . . . 90
4.2.5 Data Analysis . . . 92
4.2.6 Analysis of Results . . . 95
4.3 Extending MoWebA to other Architectures: A Case Study . . . 96
4.3.1 Background . . . 97
4.3.2 Design of validation . . . 98
4.3.3 Data Collection . . . 151
4.3.4 Data Analysis . . . 154
4.3.5 Threat to Validity . . . 161
4.4 Discussion and Summary of the Chapter . . . 163

5 Conclusion and Future Works 165
5.1 Contributions . . . 165
5.2 Publications . . . 166
5.3 Research Collaborations . . . 168
5.4 Future Work . . . 169

Bibliography 171

List of Figures

1.1 The engineering cycle presented by Wieringa [148] 5

2.1 Comparison between Web Engineering Methods to Develop Modern Web Applications presented by Wakil and Jawawi [144] . . . 21

3.1 MoWebA dimensions 40
3.2 Diagrams in MoWebA 43
3.3 The MoWebA Modelling Process 45
3.4 The Use Case diagram for the Academic System 46
3.5 Navigational Tree for the Web-based Academic System 46
3.6 Metamodel of the navigational tree diagram 48
3.7 UML Profile of the navigational tree diagram 49
3.8 Example of zone diagram 50
3.9 Metamodel of the zone diagram 51
3.10 UML Profile of the zone diagram 52
3.11 Simplified entity diagram for the Web-based Academic System . . 53
3.12 Metamodel of the entity diagram 53
3.13 UML Profile of the entity diagram 54
3.14 Node diagram for the Authentication tree node 55
3.15 Metamodel of the node diagram 56
3.16 UML Profile of the node diagram 57
3.17 Logic Behaviour Diagram 58
3.18 Metamodel of the Logic Behaviour Diagram 58
3.19 UML Profile of the Logic Behaviour Diagram 59
3.20 Subject Management Presentation Page 60
3.21 Metamodel of Content Diagram 61
3.22 UML Profile of Content Diagram 62
3.23 Structure Diagram and example of a Content Diagram for the RightLayout 63

3.24 Metamodel of the Structure Diagram 64
3.25 UML Profile of the Structure Diagram 65
3.26 Source Information Diagram for Web-based Academic System . . 66
3.27 Rule example for language definition 66
3.28 Adaptation Metamodel 67
3.29 Adaptation UML Profile 68
3.30 Adaptation Service 69
3.31 Service Diagram Metamodel 70
3.32 Service Diagram UML Profile 71
3.33 Navigational Node applying the ASMRia model 72
3.34 Simplified example of ASMRia metamodel 74
3.35 Simplified example of ASMRia metamodel 74
3.36 MoWebA Transformation Process 75
3.37 QVT definition to obtain the PSMPostgres diagram 76
3.38 The Web-based Transformation Process 77
3.39 Acceleo modules definition for MoWebA 79
3.40 An example of a generated page 82

4.1 Cases and Unit of Analysis 89
4.2 MoWebA Mobile architecture scheme 102
4.3 Metamodel used for the definition of an ASM for RIA 104
4.4 ASM RIA Profile for MoWebA 106
4.5 Metamodel for the network communication ASM 107
4.6 UML Profile for the network communication ASM 108
4.7 MoWebA Mobile metamodel 110
4.8 MoWebA Mobile UML Profile 111
4.9 Mapping-RIA-Inheritance 114
4.10 Mapping-Mobile-Inheritance for *PersistentEntity* 114
4.11 Mapping-Mobile-Inheritance for *persistentEntityProperty* 114
4.12 Mapping-RIA-Asynchronous service 115
4.13 RIA development process 125
4.14 MoWebA Mobile development process 126
4.15 ASM Logic Diagram for "Employee Marking" application 127
4.16 ASM Content Diagram example for "Employee Marking" application127

4.17 Example of an application for offering products. Products 128
4.18 Example of an application for offering products. Customer, purchases diagram 129
4.19 E-market application data persistence ASM model 129
4.20 E-market application data provider ASM model 130
4.21 Example of an application for offering products 135
4.22 Example of an application for offering products. Load-image function 136
4.23 Comparison of development times 137

1 Introduction and Motivation

Web development has motivated the so-called "Web Engineering" [41] [101], which focuses on methodological Web proposals, in order to improve the quality of the Web development process and the final product. Current Web methods centre on developing techniques and/or models needed to define the design processes, and on providing tools to support them [82], following the MDD (Model Driven Development) approach in many cases [25]. Some methods have tool support for generating automatic prototypes (e.g. VisualWADE for OO-H [52]), but only a few, such as WebRatio for WebML, have automation tools tested in industrial settings [112]. There are various quantitative and qualitative studies that show how MDD practices contribute to increase the efficiency and effectiveness in software development [9] [96] [45], and others that proposes challenges that need to be addressed by the MDE community in the near future [89] [23] [112].

Some of the major challenges facing applications today are those of portability, adaptability and evolution, not only in the environment in which they run, but also in the way in which they must be developed, often requiring different languages, frameworks, tools, environments, platforms, etc. MDD and MDA take into account these issues, however, depending on the degree of independence that the models adopt, it may be more or less possible to achieve.

Considering the fact that there is still no an unified method for the development of current environment applications, in this work we first carried out an exhaustive study of the trends and technologies with the objective of identifying strengths and weaknesses of current tendencies in application development.

The result was the need of an approach that took into account the problem of evolution and portability towards different environments. This approach is called MoWebA (Model Oriented Web Approach). In the MoWebA approach, we have identified some aspects that could have a positive impact in the portability and adaptability problems. These aspects are: i) incorporation of an Architecture Specific Model (ASM) as a new modeling layer, in order to keep the portability of the Platform Independent Model (PIM) regarding the different architectures (e.g., RIA, SOA, Mobile); ii) clear separation of the presentation layer with regard to the navigation and behavior layers, would prevent additional complications to the presentation, since, it is the layer that presents more difficulties related to portability; iii) definition of the navigational structure according to a function-oriented approach, which prevents the modification of the navigation design caused by implementation changes; iv) and use of standards in order to facilitate the independence from the tools.

In this chapter we present the problem statement of this study, introducing a series of concerns about Web Development. Section 1.2 presents the research method adopted in this PhD Thesis and the activities done in each stage. Finally section 1.3 presents the structure of this book.

1.1 Problem Statement

The study of Web methods and the classification proposed by Schwinger and Koch [126], as well as our previous experiences and that of different authors [133] [127] [24] [77], reveal some concerns about Web development.

The first concern establishes that "Navigational oriented modelling could help simplify the models for Web Applications". Navigation has been identified as a critical and fundamental feature within Web Engineering [110] [133]. Nevertheless, navigational models are usually not the starting point of the modelling process. Most of the methodologies mentioned in the literature (UWE [65], WebML [36], OOWS [97], OO-H [53], OOHDM [111]) start the design of navigational models from the conceptual (i.e., structural) model. However, the way in which the information is arranged and structured in the organization, is not necessarily the way external users need to access it [134]. Thus, deriving the navigational model from the structural model may be useful in order to organize the information content, but it does not model users' interaction in all their dimensions. Modelling the Navigational perspective according to the way in which user wishes to explore the application (i.e. functional-oriented modelling) helps to obtain friendly and easy to access navigational paths.

A second concern is that the "adoption of standards will facilitate interoperability between models, methods, transformations rules, and tools". Methodologies such as UWE [65], WebML [36], W2000 [13], OOWS [97]; and tools such as Acceleo [1], AndroMDA [2], Olivanova [3], Optimal J [4], ArcStyler [5], among others, have partially adapted their models, processes and/or transformation languages to the Model Driven Architecture - MDA [112]; MDA proposes using several standard languages to follow MDD. Without adopting MDA approach in all its potential, the methodologies tend not to take advantage of the efficiency and effectiveness in Web engineering. Despite UWE being the only methodology whose models and processes completely follow the MDA approach, their code generation tools require additional adjustments for a complete transformation (e.g. UWE4JSF which works in the Eclipse environment and generates JSF applications requiring additional adjustments for some java classes, libraries, stylesheets, among others). For the semi-automatic generation of Web applications some other approaches were implemented and are currently under evaluation [6]. In any case, it is an open line of research how to take profit from the adoption of standards, transformation tools, and the thorough MDA potential in Web engineering.

Finally, the third concern is the belief that "taking into account evolution of Web environments is very important for improving the development of current Web applications". In fact, current Web applications evolve very fast (considering technologies, platforms, architectures, diversity access devices, among others) and methodologies need to be flexible in order to consider these Web tendencies. Normally, methodologies try to do this by extending their modelling notations (e.g. RIAs proposal for WebML [36]) at the level of Platform Independent Model (PIM). In doing so, the PIMs are not more technology/platform independent. The consequence is a loss of portability of the models. Therefore, the open issue is

[1]http://www.acceleo.org
[2]http://www.andromda.org
[3]http://www.sosyinc.com
[4]http://www.compuware.com
[5]http://www.markosweb.com/www/arcstyler.com
[6]http://uwe.pst.ifi.lmu.de/

to find alternative ways to assure the easy evolution of Web application as well as preserving the independence of the PIMs and the portability of models for different architectures or platforms.

Considering these concerns, we have defined MoWebA (Model Oriented Web Approach), a methodological proposal that intends to respond to the previous concerns and their related open issues. MoWebA adopts the MDA approach in every phase and the corresponding supporting tools trying to offer more efficiency and effectiveness in Web applications development; it offers an innovative proposal for the navigational perspective; and it considers the new technological tendencies in Web Applications.

The main contributions of MoWebA approach are: i) providing a view of navigation, more function-oriented (i.e. behavioural-oriented) than data-oriented, trying to better capturing the requirements of users interaction; ii) considering almost all the modelling process, starting from the navigational model instead of the conceptual/data model; iii) providing an architectural level of modelling definition titled ASM – Architectural Specific Model, in order to facilitate the evolution of applications, preserving the independence of the PIM.

Next, we will explain the research method adopted for this PhD thesis and its contributions.

1.2 Research Method

This PhD thesis has been developed following the Design Science Research methodology proposed by the literature (we considered particularly the methodologies proposed by Wieringa [148] and Vaishnavy and Kuechler [63]). Specifically, we follow the design science research that include set of analytical techniques and perspectives for performing research in Information Systems. The methodology involves the analysis of the use and performance of designed artifacts to understand, explain, and improve on the behavior of aspects of information systems. We applied the design science methodology for the purpose of defining, managing, and differentiating the practical and knowledge problems. In this research, knowledge problems identify existing knowledge about Web Methods under the model-driven paradigm, considering the trends and technologies of actual applications. The practical problem was the formulation of the MoWebA proposal.

Figure 1.1 shows the engineering cycle proposed by Wieringa that we adopted in this Phd Thesis.

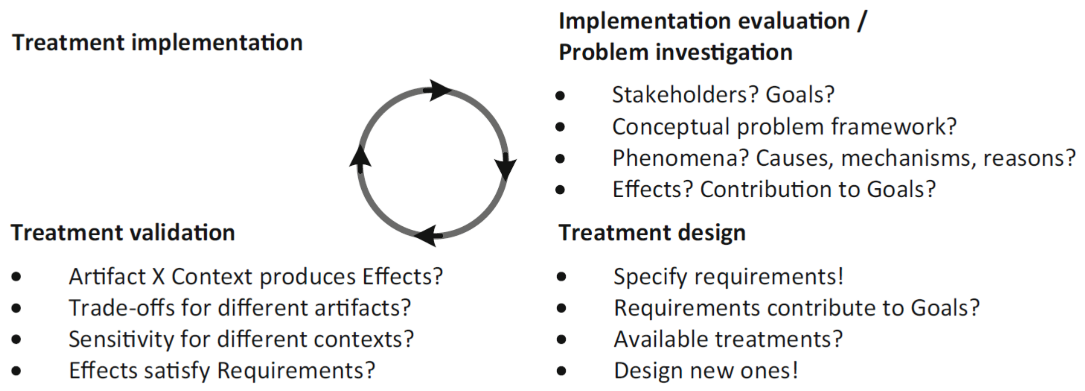

Treatment implementation
Implementation evaluation / Problem investigation
• Stakeholders? Goals?
• Conceptual problem framework?
• Phenomena? Causes, mechanisms, reasons?
• Effects? Contribution to Goals?
Treatment validation
• Artifact X Context produces Effects?
• Trade-offs for different artifacts?
• Sensitivity for different contexts?
• Effects satisfy Requirements?
Treatment design
• Specify requirements!
• Requirements contribute to Goals?
• Available treatments?
• Design new ones!

Figure 1.1: The engineering cycle presented by Wieringa [148]

Then, research activities of this PhD thesis include:

- Research goals and questions
- Problem investigation
- Treatment design
- Treatment implementation
- Treatment validation
- Conclusion

Next, we present the research goals and questions of this PhD thesis, and then the activities included in this research following the Design Science Research Methodology.

1.2.1 Research Goals and Questions

Before presenting the research goal of this thesis, we claim back the main aspects we are going to take into account:

- Web Applications evolves to new technologies, paradigms and platforms continuously.
- Web Methods do not progress at the same rate as current Web tendencies (technologies, paradigms, platforms).
- Model-driven development and Model Driven Architectures are tendencies that may improve software development.
- The relevance of navigational perspective as a central point for Web Engineering methods.

Then, the research goal of this PhD Thesis is the definition of a navigational role-centric oriented Web application development approach based on MDE standards for software development on multiple platforms and architectures to improve the portability and evolution of applications.

Based on this objective, a series of research questions arise:

- RQ1: What current proposals exist that adopt the MDE standards?
- RQ2: How the existing proposals consider the development of applications on multiple architectures and platforms?
- RQ3: How do proposals addresses the problem of portability and evolution?
- RQ4: What aspects must be taken into account for a proposal to include portability and evolution aspects?
- RQ5: What are the effects on application development of adopting an approach that contemplates portability and evolution?

Theses request questions are related to the research methodology activities of this work in the following way:

- Problem Investigation: RQ1, RQ2, RQ3
- Treatment Design and implementation: RQ4
- Treatment Validation: RQ5

1.2.2 Problem Investigation

This stage included the study of the theoretical foundations of Model Driven Engineering and Model Driven Web Engineering. We also analyze the more traditional Web development methods and their adoption of MDE standard. Subsequently, taking into account a number of concerns indicated in the section 1.1, a comparative analysis of the traditional Web approaches is also made. We also analyse the current trends in Web applications and the way the Web methods approaches consider them.

As a second activity we did a study of MDD tools. We performed a comparative study of MDD tools, based on characteristics that have identified their strengths and weaknesses. In this regard, we have considered features aimed at ease of use, support, adaptability, models proposed and quality of the final products (generated code). We made further development experiences with different MDA tools, comparing them with the traditional approach in software development. The objective of this analysis was to determine the contributions in adopting the MDA Approach for Web development. Results of these experiences can be found at [42] [90] [74].

We finished this stage with an analysis of empirical evidences from academia and industry.

1.2.3 Treatment Design

Based on the results of previous stage, a new MDD proposal definition for Web Development was presented: MoWebA (Model Oriented Web Approach). MoWebA is a navigational role-centric proposal for Web Applications, which defines methodological aspects (process, stages, products, dimensions) and complements them with an entire environment, including modeling and transformation tools, automatic code generation, use of standards, and robust architecture, among others.

MoWebA adopts the MDA approach and its phases in the transformation process are: Platform Independent Model – PIM, Architecture Specific Model - ASM, Platform Specific Model – PSM, Implementation Specific Model – ISM, and manual adjustments. The PIM phase is based on a modeling process composed of five models considering a strong separation of concerns, where concerns are:

Domain, Logic, Navigation, Presentation, and User.

The ASM enriches the models with information for specific architecture (e.g. Rich Internet Applications, Service Oriented Applications, REST, among others). PSM phase keeps the focus on separation of concerns adding platform specific information to the model. In this phase, the new technological and platform tendencies of Web application are considered. MoWebA also defines the PSM metamodels and mapping rules for PIM-ASM-PSM.

ISM phase corresponds to the application code, generated in an automatic way. Since real experiences have shown that sometimes manual adjustments are necessary, mainly for tuning the user interaction dimension, we consider a "Manual Adjustment" phase, where additional code can be added to adapt the application.

The transformation process implies steps and activities for transformation specification in order to overpass through each MoWebA phase (PIM-ASM, ASM-PSM, PSM-ISM).

The proposal was reported in [57] [56]. Related works that helped to define the proposal are [10] [108] [55].

1.2.4 Treatment implementation

In this stage, we implemented the transformation process defined for MoWebA. The transformation process is based on metamodels (PIM-ASM-PSM transformation).

The PIM-ASM/PSM phase is done in a semi-automatic way; since sometimes the information to be added requires human intervention (e.g. in RIAs, the modeller needs to specify where services will be executed, on the client or on the server). The automation of this process is done using MDD standards such as QVT or ATL, along with a tool that supports these standards.

The ASM/PSM-ISM phase is done automatically by using open source tools(e.g. Acceleo, AndroMDA). We have selected two transformation tools (AndroMDA and Acceleo) and defined rules to generate code considering the following platforms: Zend for PHP, Ruby on Rails, HTML5, among others.

There are some results of transformation rules applied to different proof of concepts or case studies described in [55] [108] [10] [54].

1.2.5 Treatment Validation

For validation purposes, we did a series of experiences carried out in the academic and real contexts that allowed us to validate and improve the proposal for Web environments. Subsequently, we have considered to validate the contributions of this proposal that allows the methodology to be extended to other environments or architectures. Such extensions are possible in MoWebA through its architectural specific model definition phase (ASM). In this sense, we first did a preliminary validation experience of the architectural specific model (ASM) definition performed with computer engineering students. Subsequently, we carried out a Case Study to validate the extensions of MoWebA to three different architectures. The experience was structured taking into account a framework that Runeson et al. [114] have defined for case studies. In these extensions we could analysed the grade of adaptability of MoWebA and automation PIM-ASM, as well as the grade of independence of the PIM metamodel.

MoWebA has been used for modeling different types of applications: an academic system, a laboratory management, a budget execution system, a survey system, and a social network system. In these experiences, we used two types of validation instruments: interviews and questionnaires. We have also experimented with different types of users: modelers (novice and experienced) and developers. Results of these validation experiences can be found at [57] [56] [93] [120] [94].

1.2.6 Conclusion

In this stage we analysed the findings and contributions, points out limitations of the current work, and also outlines directions for future research. We also list the published results of this PhD thesis.

Finally, the main contributions of this PhD thesis are:

- Formal definition of MoWebA using standard languages and techniques.
- Rules transformation definition using MDD open source tools.
- Experiences of different proof of concepts and case studies.

- Validation of MoWebA with the evolution of the proposal to different architectures.

1.3 Thesis Outline

The thesis comprises three parts, according to Design Science Methodology: Part I (Problem Investigation), Part II (Solution Design) and Part III (Validation of the Solution). Therefore, this thesis has been structured as follows.

Part I: Problem Investigation

Chapter 1 - Introduction and Motivation. This chapter describes the problem statement, research goals, and research questions. Additionally, we describe the research methodology applied to the thesis, and the thesis context.

Chapter 2 - Theoretical Framework and Related Works. In this chapter we include a theoretical framework in order to establish a commitment about the terminology defined in this thesis. Subsequently, we make a comparative analysis of the traditional Web approaches taking into account a number of concerns indicated in the introduction. Next, we analyse the current trends in Web applications and the way the Web methods approaches consider them. The chapter ends with an analysis of empirical evidences from academia and industry.

Part II: Solution Design

Chapter 3 - MoWebA: Model Oriented Web Approach. In this chapter we describe the model-driven method proposed called MoWebA. We present an overview of the method including the dimensions and the diagrams that we propose. Next, we present the Modeling and Transformation processes. We finish the chapter with a summary of the chapter.

Part III: Validation of the proposal

Chapter 4 - Validation Experiences of MoWebA. This chapter presents a series of validation experiences of the MoWebA proposal. As a first step, we were interested in verifying the use of MoWebA within the environment in which it was conceived, the Web applications. Subsequently, section 4.2 resumes a preliminary validation experience of the architectural specific model (ASM) definition within computer engineering students. The chapter ends with a Case Study of the MoWebA ASM extension other architectures, carried out in the context of a MDD research project held in Paraguay.

Part IV: Final Part

Chapter 5 - Conclusion and Future Works. This chapter summarizes the contributions of this work and presents the conclusions of the thesis. The chapter lists the different publications in Journals and Conferences as a result of the thesis and the collaborations carried out with researchers from other universities within the framework of research projects. Additionally, we also outlines directions for future research.

2 Theoretical Framework and Related Works

In this chapter we presents the theoretical foundations of Model Driven Engineering and Model Driven Web Engineering. Subsequently, we make a comparative analysis of the traditional Web approaches taking into account a number of concerns indicated in the introduction. Next, we analyse the current trends in Web applications and the way the Web methods approaches consider them. The chapter ends with an analysis of empirical evidences from academia and industry and the final discussions of the chapter.

2.1 Advances in Model Driven Engineering

The nineties centuries were influenced by three main software development paradigms: Computer Aided Software Engineering - CASE, fourth generation languages - 4GLs, and object oriented paradigm. Object orientation, in turn, became the basis of the component technologies. In this sense, object oriented languages replaced, in large measure, previous generations of programming languages.

Although CASE tools and models generated much interest in the community, they, in the beginning, served mainly as a documentation source. Model Driven Engineering - MDE and Model Driven Development - MDD generated a major change in the use of models because the focus and key artifacts are models (not programs). In this sense, MDD aims at the automatic generation of programs based on models using modeling languages and implementation tools [25] [40] [99].

A model of a system is defined by a modeling language. Modeling languages are conceptual tools with the purpose of letting designers formalize their thoughts and conceptualize the reality in explicit form (textual or graphical) [25] [99]. A modeling language is formalized by a metamodel and corresponds to a set of all possible models that are conformant with its respective metamodel. Therefore, it is a set of all possible models that are conformant with the modeling language's abstract syntax (metamodel), represented by one or more concrete syntaxes (textual or graphical notation) and that satisfy a given semantics [40] [25] [99]. Moreover, Embley et. al [44] indicate that conceptual models, with which modelers program, must be: i) complete and holistic; and ii) conceptual but precise.

MDD considers models as first-class citizens in software engineering. For this reason, model transformation plays an extremely important role in MDD. Transformation, in fact, is a fundamental issue in computer science and software engineering. The differences are seen more in the communities, the objects to be processed, and the set of requirements to be considered [39]. Transformation facilitates the process of automatic code generation, and that is why in recent years much effort has been directed to define and propose new techniques and tools that serve as support for this field.

Czarnecki and Helsen proposed a diagram defining a hierarchy of common characteristics and variables describing the concept of "model transformation" [39]. On the other hand, Stahl and Volter stated that in order to determine the transformation model used by a particular approach should be taken into account the following aspects: specification, transformation rules, control of rule application, addressability, traceability, among others [129]. In the same study the authors also propose a categorization of transformation approaches: i) Model-to-Text and ii) Model-to-Model. Mens and Van Gorp propose a model transformation taxonomy based on a series of criteria for allowing categorization of tools, techniques and formalisms based on common qualities [81].

Model Driven Architecture - MDA [1] is presented as an approach that combines several standard languages to follow MDD [79] [99]. It does not define methods or steps required for the development of software, but provides the conceptual and technological infrastructure with which to build the MDD methods. The outstanding feature of this proposal is the transformation models for generating

[1]http://www.omg.org/mda

intermediate models or platform dependent during the transformation process. There are several tools (Acceleo [2], AndroMDA [3], Xpand [4], among others) and methodological proposals that adopts the MDA approach (UWE [66], WebML [36], W2000 [13], OOWS [97], among others).

MDA proposal manages three main ideas: 1) Separate the system functionality specification from its implementation on a specific technological platform, 2) Control the software evolution from abstract models to implementations in order to increase the degree of automation and 3) standardize the MDD process.

As one of the objectives of the MDA proposal is the MDD standardization, the base is made up of standards defined by the OMG for working with models as essential components in the software development process [99]. These standards are: UML (Unified Modeling Language)[5]; OCL (Object Constraint Language)[6]; XMI (XML Metadata Interchange)[7]; MOF (Meta Object Facility)[8]; CWM (Common Warehouse Metamodel)[9]; and QVT (Query/Views/Transformations)[10].

Burgeño et al proposed a List of topics for the MBEBOK (Model-Based Software Engineering Body of Knowledge), which is actually under development [32]:

1. Model Foundations: basic modeling concepts and practices.
2. Model Quality: quality aspects of models, including completeness, consistency, correctness, comprehensibility, confinement and changeability.
3. Analysis: structural model analysis, behavioral model analysis and model transformation analysis.
4. Modeling Languages: language definition, types of modeling languages and multiview modeling.
5. Model Representation which covers concrete syntax.

[2]http://www.acceleo.org
[3]http://www.andromda.org
[4]http://wiki.eclipse.org/Xpand
[5]http://www.omg.org/uml
[6]http://www.omg.org/spec/OCL/
[7]http://www.omg.org/spec/XMI/
[8]http://www.omg.org/mof/
[9]ttp://www.omg.org/spec/CWM/
[10]http://www.omg.org/spec/QVT/

6. Model Maintenance and Evolution concerned with model operations (diff, merge, refactoring), model versioning, and model migration.

7. Model Execution: model simulation and co-simulation, execution strategies, and model debugging and testing.

8. Model Transformations: model transformation languages, types, and applications.

9. Use of MBE in application domains, advanced topics and some engineering best practices.

Recently, the research community held three events: the Grand Challenges in MDE workshop [11], and the Winter Modeling Meeting [12] and the Second Winter Modeling Meeting (WMM2020)[13]. Experts from industry, academia and the open-source community attended these meetings and presented their views that reflected on the research roadmaps over the last 10 years, what challenges remain, and the challenges facing the community in the future [30] [31]. In [30] they considered that challenges from 2007 through present day are related to the following topics: language engineering, language workbenches, model management, model analysis, models at runtime, modeling repositories, and the scalability across different dimensions.

They mentioned technical challenges related to foundation, domain, and tool challenges. In the context of foundation, agile and lean software development is increasingly adopted in the software industry and the use of AI techniques to automate and make more powerful all the maintenance solutions. One important aspect related to this context is the evolution. In this sense, considering the runtime phase of systems, and the adaptive nature of most of the complex systems developed in the last years, the authors mentioned that software changes are ubiquitous and unavoidable. To manage them, it is necessary to go toward a theory of software agility in MDE able to consider different kinds of maintenance, including repair and improvement, adaptation to a new platform, extension with

[11]http://www.edusymp.org/Grand2017/en
[12]http://eventmall.info/AMM2018/
[13]http://eventmall.info/WMM2020/

new functionality, reuse in different contexts, refactoring to make the above kinds of maintenance more accessible [30].

In WMM2020 it was mentioned that the role of models in improving productivity in SE is a recurring theme. Professional software engineers from industry, open source project contributors and researchers from academia, indicate the successful shapes of modeling as: model-based systems engineering (MBSE), low-code software development, and informal software modeling [31]. Some open challenges mentioned in this meeting include: AI-Based MDE techniques (AI extensions for automation and bringing quantifiable advantages [34]), multi-paradigm modeling (modeling everything), model for modeling (support organizations on their way toward applying modeling successfully), model management (heterogeneous collections of related models). Empirical studies have shown that modeling positively affects both engineers' productivity and the products' resulting quality, thanks to consortia and standardization bodies [31].

Although domain-specific languages have gained great relevance, the use of standards, such as UML, it is still a current trend. In a recent industrial survey on the state-of-practice, where quantitative data were collected from 113 subjects, mostly professionals working with MBE, the majority uses UML and/or SysML for modeling [71].

2.2 Model Driven Web Engineering

Web development is not an easy task, because of rapid evolution and complexities of Web applications. Mendes et al. [80] differentiate basically three types of Web applications considering the complexity and scope of this type of systems: Web hypermedia application, Web software applications, and Web applications. These applications by default use communications technology and have multi-platform accessibility. In addition, since they employ a hypermedia paradigm, they are non-sequential by nature, using hyperlinks to interrelate Web pages and other documents. This is why navigation and pluralistic design become important aspects to take into account. In order to deal with these considerations, the discipline Web engineering surged in the literature. Muruguesan and Deshpande [88] describe Web Engineering as "the use of scientific, engineering, and management principles and systematic approaches with the aim of successfully developing, deploying and maintaining high quality Web-based systems and applications".

Web applications are increasingly growing and covering very different domains. The evolution of Web 1.0 into the Web 4.0, has resulted in the introduction of several improvements and new challenges in development of Web Applications. Clusters of webs 3.0 and 4.0 represent the modern web applications that possess great extent of complexities, encompassing Ubiquitous Web Applications (UWAs), Rich Internet Applications (RIAs), Semantic Web Applications (SWAs), and Intelligent Web Applications (IWAs) [142], often resulting in the necessity to contemplate specific architectures in order to deal these trends. Functionalities of Web applications also evolve to provide services that are more relevant to potential users. Furthermore, those services are offered over different architectures and platforms. Therefore, taking into account the evolution of Web environments (considering technologies, platforms, architectures, diverse access devices, among others) is very important to 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 [41], there is a trend of following the Model Driven Development approach (MDD) [25] [99] and in some cases to adopt the standards proposed by MDA. 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). As new platforms emerge and changes in technologies occur continuously in this area, MDA mainly permits successful highlighting of interoperability, model evolution and adaptation issues of Web systems [143]. 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 [47]). 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 have one modeling phase in which the PIM modeling is performed at the same time that architectural details are provided, and then the final code is generated.

There are some proposals that deal with architecture considerations. As early as 2004, Mikkonen et al. [83] realized that architectural styles do not have a clear place in MDA. They analyzed that architectural styles can reside in PIM, PSM, or be distributed between them. Therefore, they proposed to modify MDA adding a new layer called Architecture Specific Model to encapsulate architectural properties in it. Afterwards, Marcos et al. [76] 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). In this proposal, 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 [122] [123], 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. The foregoing proposals deal with different architectural aspects and propose corresponding models. We think that considering architecture aspects as an intermediate phase between PIM and PSM is a way to ease the evolution of Web systems.

During the last 30 years researchers built support for different Web technologies, such as RIA, Semantic Web, Mobile Web and also improved application life cycle coverage by addressing requirement specification, testing and maintenance [112]. Several methodologies have emerged in this period, and the most prominent of which continue to be studied are: UWE [65], WebML [36], OOHDM [111], OO-H [98], OOWS [97], HERA [138], RUX [132], among others. In addition, IFML [28] is an important milestone, but there is still a lot of work to be done in order to assure that modern Web applications are developed faster, safer, and with less errors and higher quality [112]. The aforementioned methodological proposals follow the MDD approach and are oriented to develop Web applications. Therefore, they could be classified as part of the Model Driven Web Engineering paradigm (MDWE) [85].

Some recent studies show that the actual challenges of MDWE are: adoption, better support for non-functional requirements, agility, support for end-users, better coverage of Web engineering technologies, improvement of architectural and implementation issues, evaluation of MDWE approaches [112]. A Study made by Wakil et al. [143] indicates that there is a wide variety of Web development methodologies, using a multiplicity of different notations, models and techniques, but no single Web development approach provides coverage for the whole life cycle.

Comparative recent studies about Web methodologies show the capability of the methods to support modern and multi Web applications [144] [107]. The methodologies analysed were: WebML, UWE, Rux Method, OOHDM, OOWS, HERA, WebRE, IFML, OO-H, among others.

More specifically, in the next section we will compare the most relevant methodologies for Web Development with regards the main concerns presented in section 1.1.

2.2.1 Web Methods Approaches

In the last 25 years, many methodological proposals have been defined for the development of Web applications. Over the years, some of them have remained relevant and prominent, as they have defined adaptations to meet different needs.

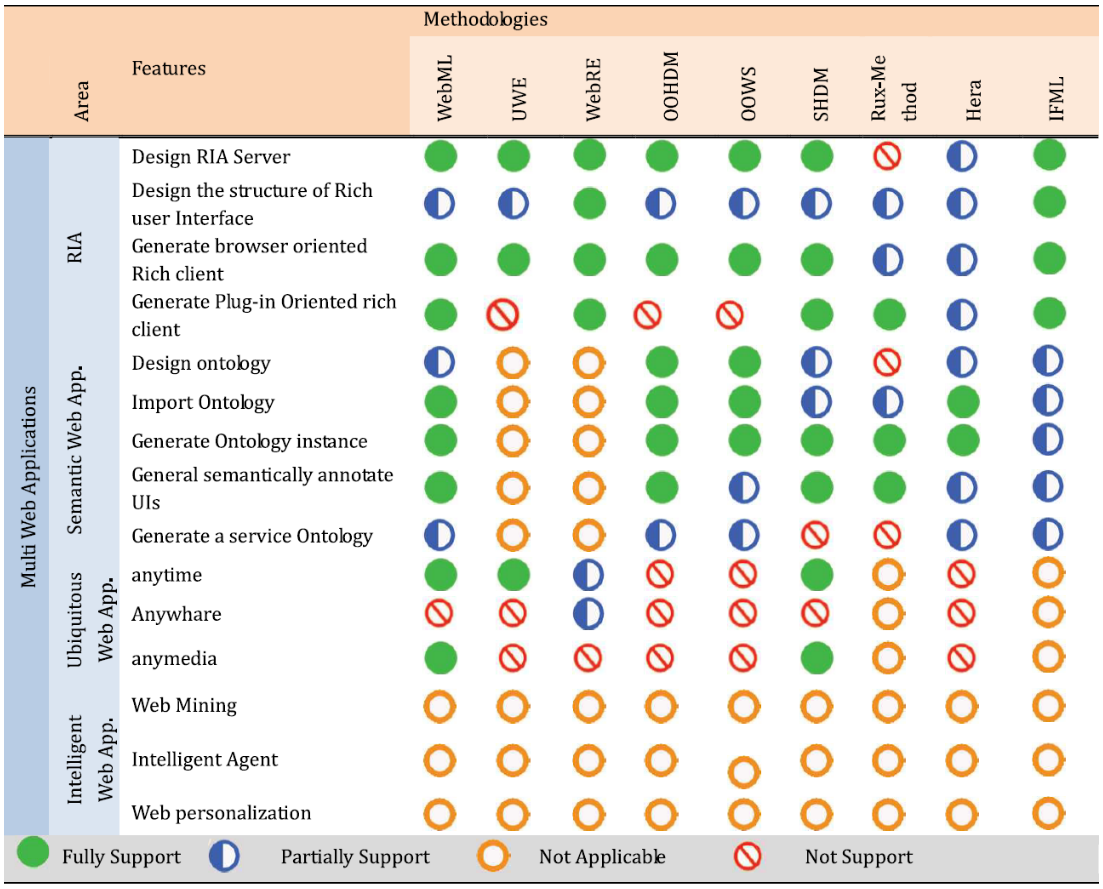
The methodological proposals most frequently mentioned in the literature are as follows: WebML, UWE, Rux Method, OOHDM, OOWS, SHDM, HERA, WebRE, IFML, OO-H, RMM, SOHDM, among others. It is worth noting that some proposals have remained in academic environments (such as UWE, OO-H, OOHDM, etc.), and others, although they have emerged in academy, have subsequently transcended into the industrial world, such as WebML, supported by the commercial tool Webratio [14], and today has great relevance in the industry, with very encouraging results [136] [27].

Over the years, it has become clear that, for a Web method to have continuity, it is essential to consider techniques that easily adapt the methodology to the new

[14]https://www.webratio.com

trends since they arise at a very fast rate. Then, providing techniques and tools for an easy adaptation is an aspect that has gained importance in recent years.

A recent comparative study for Web methodologies reveals the fact that evolution and/or adaptability is an issue not fully contemplated by the proposals. Although it is possible to find studies that present adaptations of traditional methods to cover new trends, Wakil and Jawawi [144] point out that the speed with which methodologies define their adaptations does not correspond to the speed at which environments and technologies evolve. For this reason, according to Figure 2.1, presented by them, there are still limited studies that cover the current needs of modern Web applications, such as UWA and IWA.



			Methodologies								
	Area	Features	WebML	UWE	WebRE	OOHDM	OOWS	SHDM	Rux-Method	Hera	IFML
Multi Web Applications	RIA	Design RIA Server	●	●	●	●	●	●	⊘	◐	●
		Design the structure of Rich user Interface	◐	◐	●	◐	◐	◐	◐	◐	●
		Generate browser oriented Rich client	●	●	●	●	●	●	◐	◐	●
		Generate Plug-in Oriented rich client	●	⊘	●	⊘	⊘	●	●	◐	●
	Semantic Web App.	Design ontology	◐	○	○	●	●	◐	⊘	◐	◐
		Import Ontology	●	○	○	●	●	◐	◐	●	◐
		Generate Ontology instance	●	○	○	●	●	●	●	●	◐
		General semantically annotate UIs	●	○	○	●	◐	●	●	◐	◐
		Generate a service Ontology	◐	○	○	◐	◐	⊘	⊘	◐	◐
	Ubiquitous Web App.	anytime	●	●	◐	⊘	⊘	●	○	⊘	○
		Anywhare	⊘	⊘	◐	⊘	⊘	⊘	○	⊘	○
		anymedia	●	⊘	⊘	⊘	⊘	●	○	⊘	○
	Intelligent Web App.	Web Mining	○	○	○	○	○	○	○	○	○
		Intelligent Agent	○	○	○	○	○	○	○	○	○
		Web personalization	○	○	○	○	○	○	○	○	○

● Fully Support ◐ Partially Support ○ Not Applicable ⊘ Not Support

Figure 2.1: Comparison between Web Engineering Methods to Develop Modern Web Applications presented by Wakil and Jawawi [144]

As well as the necessity for method evolution already mentioned, in section 1.1, we have identified two other concerns that affect the adoption of methodologies for the development of Web applications, which are: the mechanism used to define navigation, and the notation proposed by the methodology. Consequently, in order to perform a more exhaustive analysis of these concerns, we present the table 2.1 that contemplates the Web methodological proposals most mentioned in the literature and an analysis of the mechanisms in which they consider these three factors. For this analysis, we have considered the documentation about the different methodological proposals, the adaptations made to consider new trends and the comparative study between Web Engineering Methods to develop modern Web applications carried out by Wakil and Jawawi [144].

Table 2.1: Comparison between Web Engineering Methods considering Navigation, Notation and Evolution

Web Method	**Navigation**	**Notation**	**Evolution**
UWE	The navigational structures are derived from the content model (structure) and are extended with process classes that represent entry and exit points of the business processes.	UML, UML Profiles	Extensions have been proposed for RIA [66] and Web security [33]. The first one through patterns that are incorporated to UWE, and the second one enriches the UWE models with stereotypes and OCL. There is another work extending the navigation options by adding new constructors to the model [146].

Web Method	Navigation	Notation	Evolution
	Navigation model definition: From the data model	**Use of standards:** Totally	**Evolution to Modern Web Application:** RIA: Partially; SWA: N/A; UWA: Partially; IWA: N/A; and Cloud: N/A
OOHDM	An application is seen as a navigational view over the conceptual model, which recognizes that the objects the user navigates are not the conceptual objects, but other kinds of objects that are "built" from one or more conceptual objects, to suit the users and tasks that must be supported.	Requirements and Conceptual Model: UML. UID (User Interaction Diagrams). Navigation: Navigational Context Diagram. Presentation: Abstract Widget Instance and concrete interface.	xOOHDM for executable models [51], OOHDM for RIA with ADV extensions [113]
	Navigation model definition: From the data model	**Use of standards:** Partially	**Evolution to Modern Web Application:** RIA: Partially; SWA: Totally; UWA: none; IWA: N/A; and Cloud: N/A

Web Method	**Navigation**	**Notation**	**Evolution**
WebML	Hypertext design produces site view schemas on top of the data schema. Site views express the composition of the content and services within hypertext pages, as well as the navigation and the interconnection of components. **Navigation model definition:** From the data model	Data model: UML (Class diagrams). Logic and Navigation: DSL. Presentación: IFML. **Use of standards:** Partially	Extensions for RIA [24]. Extensions for Mobile with IFML [8]. Extensions for cloud [26] **Evolution to Modern Web Application:** RIA: Totally; SWA: Totally; UWA: Totally; IWA: N/A; and Cloud: Totally

Web Method	**Navigation**	**Notation**	**Evolution**
OOWS	Views are defined over the class diagram (structure), in terms of the visibility of class attributes, operations and relationships. Navigation specifications are captured in two steps: the “Authoring-in-the-large” (global view) and the “Authoring-in the-small” (detailed view). **Navigation model definition:** From the data model	Conceptual Model: UML (class diagrams). Navigational maps: UML stereotypes. Navigational Content: UML stereotypes and additional specific elements. **Use of standards:** Partially	Extensions for RIA [137]. Inclusion of new concepts using Patterns. Extensions for semantics web services [131]. **Evolution to Modern Web Application:** RIA: Partially; SWA: Totally; UWA: none; IWA: N/A; and Cloud: N/A
OO-H	The domain information is the main input for the design navigation activity, where the navigational paths are defined to fulfil the different functional requirements and the organization of that information in abstract pages.	UML and UML Stereotypes for all diagrams.	OOH4RIA [78].

Web Method	**Navigation**	**Notation**	**Evolution**
	Navigation model definition: From the data model	**Use of standards:** Totally	**Evolution to Modern Web Application:** RIA: Totally; SWA: N/A; UWA: N/A; IWA: N/A; and Cloud: N/A
HERA	Based on the domain definition, application modeling results in the application model (AM) that specifies the navigational behavior of the Web application. The AM enables designers to specify how the (navigational) access to the data (dynamically retrieved from the domain) is structured by describing which data are shown to the user and what Web pages the user can navigate to.	DSL supported by different tools.	Aspect oriented extension for adaptation [35].

Web Method	**Navigation**	**Notation**	**Evolution**
	Navigation model definition: From the data model	**Use of standards:** None	**Evolution to Modern Web Application:** RIA: Partially; SWA: Partially; UWA: none; IWA: N/A; and Cloud: N/A
UWA	The navigational model is obtained from the UWA Requirement Elicitation.	It adopts UML for almost all diagrams but in some cases complements them with its own notation, as in the case of navigation.	RE-UWA for reengineering and evolution [18]. UWA for RIA [19].
	Navigation model definition: From the Requirement model	**Use of standards:** Partially	**Evolution to Modern Web Application:** RIA: N/A; SWA: Totally; UWA: Partially; IWA: N/A; and Cloud: N/A

Web Method	**Navigation**	**Notation**	**Evolution**
WSDM	A navigation track is created for each audience class. The internal structure of an audience track is derived from the task models made for this audience class. In addition, navigational requirements formulated during audience modeling are also taken into account.	The method define a specific DSL	WSDMLite to take current development practice into consideration [118].
	Navigation model definition: From the audience classes.	**Use of standards:** None	**Evolution to Modern Web Application:** RIA: N/A; SWA: Totally; UWA: N/A; IWA: N/A; and Cloud: N/A

From table 2.1, we can conduct the following analysis taking into account the three concerns mentioned above.

C1: How do the different methodologies consider the navigation?

From table 2.1, we can notice that most of the methodological proposals define their navigational models from the data model, or consider it as a fundamental factor to create their structures and/or navigational content (e.g. OOHDM, WebML, OOWS, OO-H, Hera). We think that although deriving the navigational model from the structural model may be useful in order to organise the information content, it does not model users' interaction in all their dimensions.

In UWA, however, navigation is obtained from the requirements model. In this sense, this approach is an alternative way to model the navigational perspective better fitting the requirements of users' interaction and making user navigation more adherent to its mental model.

The WSDM proposal regarding the navigational structure is quite similar to the UWA approach because is function oriented, but a navigation track is created for each audience class, containing all and only the information and functionality needed by the members of the associated audience class.

OOHDM and OOWS, discriminates between intra-contextual and inter-contextual navigations. This simplifies the overall understanding of the application structure and makes a distinction between the different levels of navigation.

As a result of this analysis, we can determine that most of the methodological proposals consider appropriate to create their navigational structures from the data model, despite the fact that many of the required interactions arise from user needs, and not precisely from the way in which the data is organized or structured.

C2: Which notation do the different methodologies contemplate?

All methodologies mentioned in table 2.1 adopt the MDD approach and most of them follow the object-oriented paradigm in every phase (e.g. UWE, OO-H, and OOWS). OOHDM introduces some specific elements with a specific DSL for navigation and presentation layers. In the case of WebML, it uses the object-oriented language for the data layer, specific extensions for navigation, and the IFML standard for presentation. OOWS also adopts UML as standard notation, but introduces some additional concepts to describe particular aspects in the navigation and presentation layers. OO-H and UWE are fully based on UML to define their different dimensions, also adopting MOF as the metamodeling language.

Hera and WSDM make use of a proprietary notation (DSL) supported by a series of tools. The UWA proposal uses UML for most of the diagrams but in some cases complements them with its own notation as in the case of navigation.

In our best knowledge, UWE is the only methodology whose models and processes completely follows the MDA approach. UWE code generation process is done in a semi-automatic way, since the generated code requires additional adjustments for obtaining the final application (e.g. UWE4JSF which works in the Eclipse environment and generates JSF applications requiring additional adjustments for some java classes, libraries, stylesheets, among others).

C3: Do the methodologies consider the possibility of evolving to other architectures, technologies, platforms?

From table 2.1, it is possible to notice that the analyzed methodological proposals have contemplated extensions to consider emerging technologies and architectures, which in most cases focuses on RIA architectures.

In this sense, one aspect to highlight is that extensions are generally contemplated with new constructors that are introduced in the proposal's own notation, so that the original proposal becomes an enriched one for the extension under consideration. In other words, the adaptation mechanism is based on the inclusion of notational elements in the models, thus reducing their independence from the platform or architecture. This is because the extensions are not defined as a separate modeling layer. The result of this extension mechanism is that the platform independent modeling (PIM) stage contemplates elements of a specific platform or architecture, combining both concepts in a single final model. Other proposals decided to add architectural specific information at the PSM level. In this case, architecture and platform details are included at the same modeling level, loosing portability at the architectural level with respect to different technologies where it can be implemented (e.g., UWA for RIA [20]).

Methodologies such as WebML, UWE, OOWS, OOHDM, OO-H propose extensions for RIAs (or other final platforms) introducing these new extensions by UML stereotypes, or specific DSL included in the original notation form modeling. Considering the methodologies of Table 2.1, no one captures the requirements for specific architectures in a different level of abstraction. As a counterpart, to offer a greater reusability of the PIM facilitating the architectural evolution of the Web applications, this mechanism of extension requires some additional effort, including the need for metamodels and the definition of the corresponding transformation rules, to achieve automatic transformations on the proposed architecture or platform. But 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 or platform; and the solution space, through the specific extension oriented to architecture, platform and the final code. Such proposals tend to facilitate the support of Web development for current Web systems.

Moreover, the evolution concern is not only one for architectural/technological issues, since the functional requirements of Web applications also evolve fast. In this sense, the methodologies that adhere to the MDD approach, follows an incremental process, facilitating such types of functional changes that are defined at the model level which will then be transformed into code by using automated tools.

2.2.2 Other Specific Proposals on Current Trends

In this sub section we will mention other recent proposals that show the current trends for MDD methods in Web environments. Some of the common characteristics of these proposals are the use of standards (such as UML and extensions with UML profile), and the need to take into account the ability to adapt to these new trends.

One recent method is UEWDM that applies UML Profiles as the graphical notation in the modeling stage. This method copes with conceptual design by dividing it into two-sub design called informational class and dynamic process design [86] [87].

A study made by Wakil and Jawawi [145] analyzed IFML models in the process development lifecycle to show capability of the method used in the process development. Results of this study showed that IFML is a good method with best practice but cannot fully support the web development lifecycle. The same authors showed as a result of another study that some of the current web engineering methods were extended for new concerns of web applications but with some limitations, meaning these methods have a lack of adaptability to support features from modern web applications. In an attempt to solve this gap, the authors of this study defined a new adaptive model for the web engineering methods that can support the new features of modern web applications [141]. In this study, the adaptive model and an example of its use are presented, but it is not entirely clear how this new model would be introduced into each methodological proposal to consider elements of modern applications.

Another interesting proposal is the Agile and Collaborative Model- Driven Development method forWeb applications (WebAC-MDD), that was conceptualized to transform agile models into Web application source-codes, using a Unified

Modeling Language (UML) profile named Web Agile Modeling Language (Web-AML). This method intends to represent a proposed solution for existing and inherent problems, regarding productivity in the development of Web applications and efforts for modeling and documentation, which do not add any value to clients [109].

The XIS-Web is a model-driven approach focused on the development of responsive web applications. This approach includes two main parts: the XIS-Web modeling language, implemented as a UML profile; and the XISWeb framework, which is a set of integrated software tools. XIS-Web stands out in four key aspects: supports the modeling of web applications around six viewpoints, which ultimately promotes the separation of concerns that is key to managing complexity; generates user-interface models from extended use-case models, relieving this cumbersome and time consuming task from the user; employs latest generation web technologies (such as HTML5, JavaScript, CSS) that allow the required flexibility of developing responsive web applications; and allows the creation of platform-independent models without requiring a significant learning curve [128].

In the world of mobile application development there are several proposals presented in the literature. A study we have published in the Journal of System and Software [94], presents a detailed analysis of different proposals for the development of mobile applications. In total, we analyzed and compared 23 MDD solutions for mobile application development. In particular, determined to review concepts and implementations of the data layer, the adoption of MDA, output platforms, the native code generation, and the modeling aspects considered in respective proposals.

Some of the results of this review are listed below:

1. The majority of the proposals present comparatives with at least two output platforms, but in some cases, the approaches were described in only one platform [15] [14] [48] [37] [17] [139]. Android and iOS were the first and second most selected option as output platforms for case studies and evaluations. Windows Phone appears in third place.

2. Most of the applications generated are native and data-oriented. Between those approaches that do not generate code, some consider only a modeling language [50] [119], or present a UML meta-model proposal [84] [64], or

need additional libraries to interpret the modeling for different platforms (i.e., MobDSL [67]). Other proposals generate hybrid code for PhoneGap framework [28] [21] [46] [7] [8]. The remaining works generate native code.

3. Just a few proposals contemplate only one aspect when modeling the mobile applications, in particular, modeling only the Graphical User Interface (GUI) of the applications [117] [50] [46] [37]. The rest of the proposals in the majority, contemplate GUI, Data, and even behavior modeling of the mobile application.

4. Most of the projects generate the application taking into account the structure of the project according to the corresponding Integrated Development Environment (IDE) of the chosen platforms. Through this, a compilation can be carried out with their respective SDK and thus obtain the final application. Nevertheless, a different approach was found in MD^2 [60] [59] [75] and the MAML framework [105] [106]. MD^2 allows compiling and generating the final application without modifying the generated code, which includes the project's files and settings, even static libraries. MAML is closely related to MD^2 because it uses MD^2 for the generation of the final code [106, p. 7433].

Finally, we have also made a Mapping Study on Development of Mobile Applications with Functions in the Cloud (MobileApps-FC) through the Model Driven Approach that was published in CLEI Electronic Journal [120]. In this study we found that there are some model driven proposals for the development of MobileApps-FC that address the portability problem which include: WebRatio [29], MD^2 [58], SIMON [38], MobiCloud [102], and the proposals of Steiner et al. [130], and Ruokonen et al. [115]. The analyzed proposals agree in pointing some possible positive aspects about the use of MDD regarding the portability issues in the context of the development of MobileApps-FC. More details of this study can be found at [120].

2.3 MDE Evidences in Academia and Industry

Although the introduction of MDD in the industry has not been easy in its beginnings, we can say that there is currently optimism in its adoption. According to Gartner, by 2024, MDD platforms will be responsible for over 65% of the application development activity, and three-quarters of large enterprises will be using at least four MDD platforms, as such platforms enable enterprises to develop applications quicker using more capabilities and fewer conventional developers [140]. Gartner [43] and Forrester [116] categorized MDD platforms, including low-code/no-code platforms and business process management systems into the following two sets: Business Process Management Suite (BPMS) and intelligent Business Process Management Suite (iBPMS).

Below, we present some recent empirical work from both academia and industry that demonstrate the interest and advantages of adopting this paradigm in development environments.

Bordeleau et. al [23] discuss the introduction of Model Based Engineering (MBE) tools in industry, which is often hindered by outdated assumptions on the process and a slow return on investment. With respect to tool usage in industry, the study indicates that are several key features currently missing that effectively prevent the use of MBE tools. In this sense, mirroring the industrial use of MBE tools, several similar challenges can be seen in modeling education, which suggests that the topic of MBE education should not only be studied in isolation. To tackle existing challenges, they outline directions for future work in the area of MBE tooling. They found it important to improve model diff/merge capabilities and the introduction of novel MBE strategies. A recommendation they suggest for industry and communities is to improve technology transfer and initiatives to start and foster open source communities around MBE. Some directions for academic policy include an increased focus on tool creation or improvements in tenure procedures.

Panach et. al [96] present a family of experiments replicating a baseline experiment to analyze the quality of systems developed using MDD against traditional software development methods. The results of the replications indicated that MDD yields better values for accuracy than traditional development methods, and that the differences between MDD and the control are bigger when the problems to be solved are more complex since the effect size with complex problems is moderate even with low statistical power. The authors of this study consider that to exploit the strengths of MDD, it should be applied to complex problems.

Luna et. al [73] conduct a survey among hundreds of engineers from different companies around the world and, by statistical analysis, they present the current problems of MDD approaches in scale. In this study, a set of guidelines where provided to improve Model-Driven Web Engineering approaches in order to make them viable industry solutions. One important aspect mentioned in this article is related to the necessity of an adaptive architecture that allows the development team to abstract high-level concepts through graphs that contemplate the importance of understanding and controlling the architecture of an application.

Farshidi et. al [45] presented a study of four industry case studies from different domains for Model-Driven Development platform selection. In this study, twenty-six domain experts from different software-producing organizations have participated to answer the research questions.

Some of the discussions made from this study are:

1. Software products may be more successful in some regions. In this sense, not every MDD platform is equally represented in different regions of the world.

2. The total cost of ownership of MDD platforms plays an inevitable factor in the decision-making process. However, participants believed that functional suitability, maturity, and popularity of potential solutions should be prioritized higher.

3. The decision support system assigns higher scores to the general-purpose platforms, such as Mendix and Appian, as they offer a vast set of services and functions.

4. The experts asserted that MDD platforms should not be employed in three use cases: (1) Complex applications with rich functionality, such as software products, since requires continuous development and maintenance to integrate a significant number of services and components from third parties; (2) applications for enterprises that employ the generated applications

to perform their core businesses; (3) businesses that rely principally on freemium end-users, as MDD platforms may charge their customers based on the number of their end-users.

From all the studies analyzed in this section, it is possible to notice that although there is still a need for further empirical studies, the advances in both industry and academia are increasing. Finally, the aforementioned predictions indicate that in a few years they could become a trend in all sectors.

2.4 Final Discussions

Having analyzed the MDD theoretical framework, related works, current trends and challenges in this chapter, it is possible to notice that the concerns mentioned in Section 1.1 on model-driven development in Web applications are still requiring research effort. Despite we see that although domain-specific languages are gaining relevance, the use of standards is still a common practice, particularly those related to modeling languages (such as UML). With respect to deal with the evolution of the applications, it is still a necessity that also becomes even more relevant with the increase in technologies, frameworks, platforms, architectures, etc.

Furthermore, we see that although traditional methodologies have evolved, the adaptation mechanisms have been carried out with certain limitations, not achieving yet the grade of adaptability according to the evolution of technology. Therefore, we see the need for the proposals to also adapt to new trends, in such a way that it is possible to evolve both horizontally (e.g., moving from one architecture to another, for example from web to mobile) and vertically (where new elements are necessary to contemplate the requirements and functionalities, such as including services and the cloud).

Finally, although this PhD Thesis will consider concerns 1 and 2 for the development of a new proposal, it will mainly emphasize the 3rd concern, considering specific mechanisms to adapt to new architectures to adequately evolve. Analysis and validations of this PhD Thesis will be oriented to this goal.

3 MoWebA: Model Oriented Web Approach

Web development has motivated the so-called "Web Engineering" [41] [101], which focuses on methodological Web proposals, in order to improve the quality of the Web development process and the final product. Current Web methods centre on developing techniques and/or models needed to define the design processes, and on providing tools to support them [82], following the MDD (Model Driven Development) approach in many cases [25]. Some methods have tool support for generating automatic prototypes (e.g. VisualWADE for OO-H [52]), but only a few, such as WebRatio for WebML, have automation tools tested in industrial settings. There are various quantitative and qualitative studies that show how MDD practices contribute to increase the efficiency and effectiveness in software development [9] [96] [45].

The study of Web methods and the classification proposed by Schwinger and Koch [126], as well as our previous experiences and that of different authors [134] [24] [77], reveal some concerns. Below we list those more important from our point of view.

The first concern establishes that "Navigational oriented modelling could help simplify the models for Web Applications". Navigation has been identified as a critical and fundamental feature within Web Engineering [110] [133]. Nevertheless, navigational models are usually not the starting point of the modelling process. In some situations, navigational models do not provide an appropriate syntax to model common behaviours of current Web Systems, such as the dynamic navigation behaviour observed during users' interaction, or inter-intra contextual navigation. Most of the methodologies mentioned in the literature (UWE [65], WebML [36], OOWS [97], OO-H [53], OOHDM [134] start the design of navigational models from the conceptual (i.e., structural) model. Thus, deriving the navigational model from the structural model may be useful in order to organise the information content, but this does not model users' interaction in all their dimensions. Modelling the Navigational perspective according to the way in which user wishes to explore the application (i.e. functional-oriented modelling) helps to obtain friendly and easy to access navigational paths. Therefore, the open issue is to find alternative ways to model the navigational perspective better fitting the requirements of users' interaction and making user navigation more adherent to its mental model.

A second concern is that the "adoption of standards will facilitate interoperability between models, methods, transformations rules, and tools". In recent years, methodologies such as UWE [65], WebML [36], W2000 [13], OOWS [97]; and tools such as Acceleo [1], AndroMDA [2], Integranova [3], Optimal J [4], ArcStyler [5], among others, have partially adapted their models, processes and/or transformation languages to the Model Driven Architecture - MDA [6]; MDA propose using several standard languages to follow MDD. Without adopting MDA approach in all its potential, the methodologies tend not to take advantage of the efficiency and effectiveness in Web engineering. Despite UWE being the only methodology whose models and processes completely follow the MDA approach, their code generation tools require additional adjustments for a complete transformation (e.g. UWE4JSF which works in the Eclipse environment and generates JSF applications requiring additional adjustments for some java classes, libraries, stylesheets, among others). For the semi-automatic generation of Web applications some other approaches were implemented and are currently under evaluation (http://uwe.pst.ifi.lmu.de/). In any case, it is an open line of research how to take profit from the adoption of standards, transformation tools, and the thorough MDA potential in Web engineering.

Finally, the third concern is the belief that "taking into account evolution of

[1]http://www.acceleo.org

[2]http://www.andromda.org

[3]http://www.integranova.com

[4]http://www.compuware.com

[5]http://www.markosweb.com/www/arcstyler.com

[6]MDA Guide Version 1.0.1," 2003. [Online]. Available: http://www.omg.org/docs/omg/03-06-01.pdf.

Web environments is very important for improving the development of current Web applications". In fact, current Web applications evolve very fast (considering technologies, platforms, architectures, diversity access devices, among others) and methodologies need to be flexible in order to consider these Web tendencies. Normally, methodologies try to do this by extending their modelling notations (e.g. RIAs proposal for WebML [47]) at the level of Platform Independent Model (PIM). In doing so, the platform independent models (PIMs) are not technology/platform independent anymore, and they are becoming increasingly complex to understand and manage. The consequence is a loss of portability of the models. Therefore, the open issue is to find alternative ways to assure the easy evolution of Web application as well as preserving the independence of the PIM and the portability of models for different platforms.

MoWebA (Model Oriented Web Approach) try to respond to the previous concerns and their related open issues. It adopts the MDD approach in every phase and the corresponding supporting tools trying to offer more efficiency and effectiveness in Web applications development; it offers an innovative proposal for the navigational perspective; and it considers the new technological tendencies in Web Applications.

The main contributions of MoWebA are: i) providing a view of navigation, more function-oriented (i.e. behavioural-oriented) than data-oriented, trying to better capturing the requirements of users interaction; ii) considering almost all the modelling process, starting from the navigational model instead of the conceptual/data model; iii) providing an architectural level of modelling definition titled ASM – Architectural Specific Model, in order to facilitate the evolution of applications.

3.1 MoWebA in a nutshell

MoWebA defines methodological aspects (processes, stages, work products, dimensions) and complements these aspects with an entire environment, including modelling 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 to Web Application Devel-

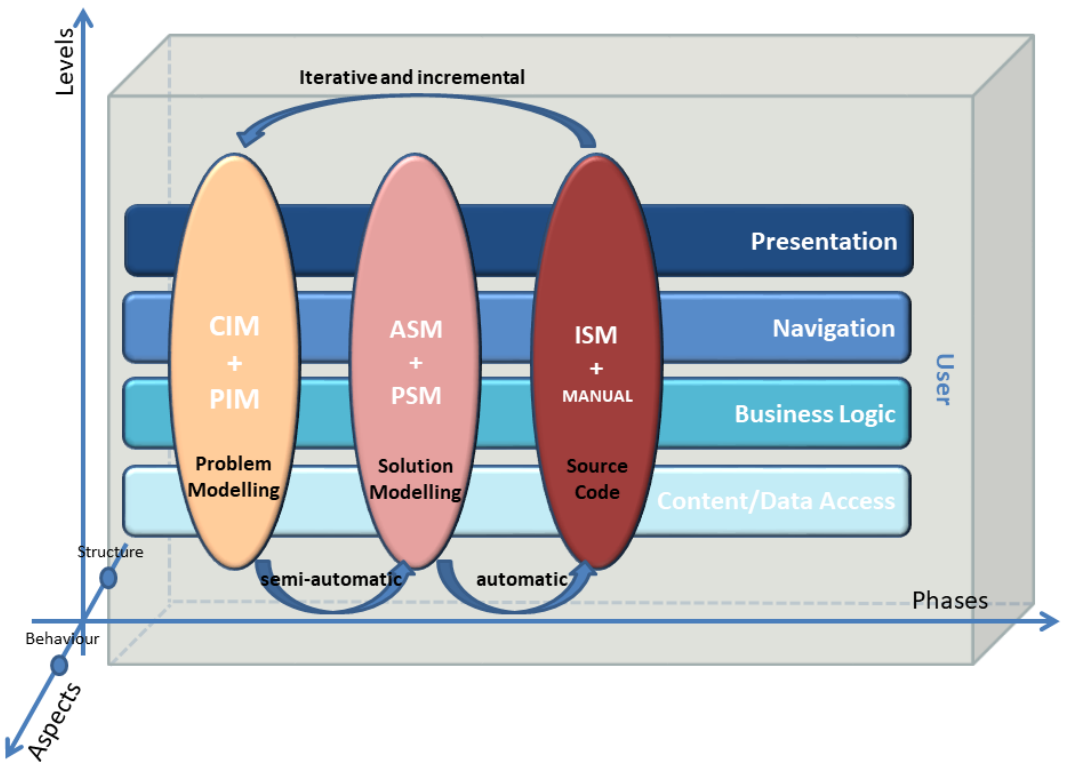


Figure 3.1: MoWebA dimensions

opment". Figure 3.1 shows the MoWebA dimensions: phases, levels and aspects.

The phases dimension covers the modelling and transformation processes. MoWebA adopts the MDA approach by identifying three different abstractions for modelling: the problem space, covered by CIM (Computational Independent Model) and PIM (Platform Independent Model) models; the solution modelling space, covered by ASM (Architectural Specific Modelling) and PSM (Platform Specific Modelling) models; and the source code definition, covered by 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 the structure and behaviour considerations for each perspective.

MoWebA defines two main complementary processes: one related to the modelling activities and the other to the transformation activities. As shown in Figure 3.1, the horizontal axis represents the MoWebA transformation process. To formalize the modelling and transformation processes, it adopts the MOF language for abstract syntax definition, and the UML profile extension for a precise definition of the modelling language.

The modelling process includes the necessary activities to get all the diagrams for the complete specification of 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 modelling activities. CIM definition covers the late requirements identification, focusing on functional requirements specifications. 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. Rich Internet Applications, Service Oriented Applications, REST, among others) and the PSM contemplates information for a target platform (e.g. a specific language, or a framework).

The transformation process, on the other hand, is related to the steps, techniques, and tools, which allow M2M (i.e., model-to-model) and/or M2T (i.e., model-to-code) transformations. This process is based on the MDA approach, and implies steps and activities for transforming specification in order to go through each MoWebA phase (i.e., CIM/PIM-ASM/PSM, ASM/PSM-ISM/Manual adjustments). The CIM/PIM-ASM/PSM transformation is done in a semi-automatic way (i.e., introducing some manual adjustments), by defining the metamodels for specific architecture or platform, and the corresponding mapping rules for PIM-ASM/PSM transformations. The ASM/PSM-ISM transformation corresponds to the automatic transformation from the models to the application code. Since real experiences have shown that sometimes manual adjustments are necessary, we consider a "Manual adjustment" phase, where additional code can be added to adapt the application. Finally, the transformation process is done iteratively, allowing an incremental application development. This proposal was published at International Journal of Web Engineering and Technology in 2016 [57].

Some of the main features of MoWebA are:

1. well-defined layered structure (achieves a clear separation of concepts);

2. modeling centered in a hierarchical function-oriented navigation (allows a more appropriate design of a user interaction based navigation); and,

3. enrichment of existing models considering aspects related to the final architecture of the system (e.g., RIA, SOA, REST, among others).

The next sections detail the modelling and transformation processes of MoWebA.

3.2 The Modeling process

This section starts by presenting a general overview of the stages and activities, and then going into details for each stage, considering diagrams, notations and tasks involved. To clarify the proposal, we use as an example a Web-based Academic System. The system supports teachers, students, staff and the general public, and covers a range of basic functions such as: student registrations processing, courses monitoring, and school, department and career management. Teachers have sufficient privileges to manage the courses they are in charge of and provide students with information regarding their current status. Students have the required privileges to track the courses they are enrolled in and also access their current academic status. Finally, the system should provide the facility to perform administrative tasks such as faculty, course, department, and subject management.

The modelling process includes the CIM, PIM, ASM and PSM specification and systematized in seven stages (see Figure 3.3). Stages 1 through 6 are oriented to CIM and PIM definitions, based on the dependency relationships between the different models, the level of granularity of the modelling task, and the type of modelling to be done; these stages are done manually. MoWebA adopts the Use Case model for CIM definition, focusing on modelling the functional requirements of the system-to-be. For PIM definition, MoWebA proposes the following models: i) Entity Model; ii) Navigational Model; iii) Behavioural Model; iv) Presentation Model; and v) User Model. Each model is composed of one or more diagrams. Figure 3.2 presents the dependency relationships between the different models.

Stage 1 is related to the requirements analysis. The artefact produced in this stage is a Use Case diagram representing the functional, navigational and usability requirements, as well as potential users of the application. Stage 2 corresponds

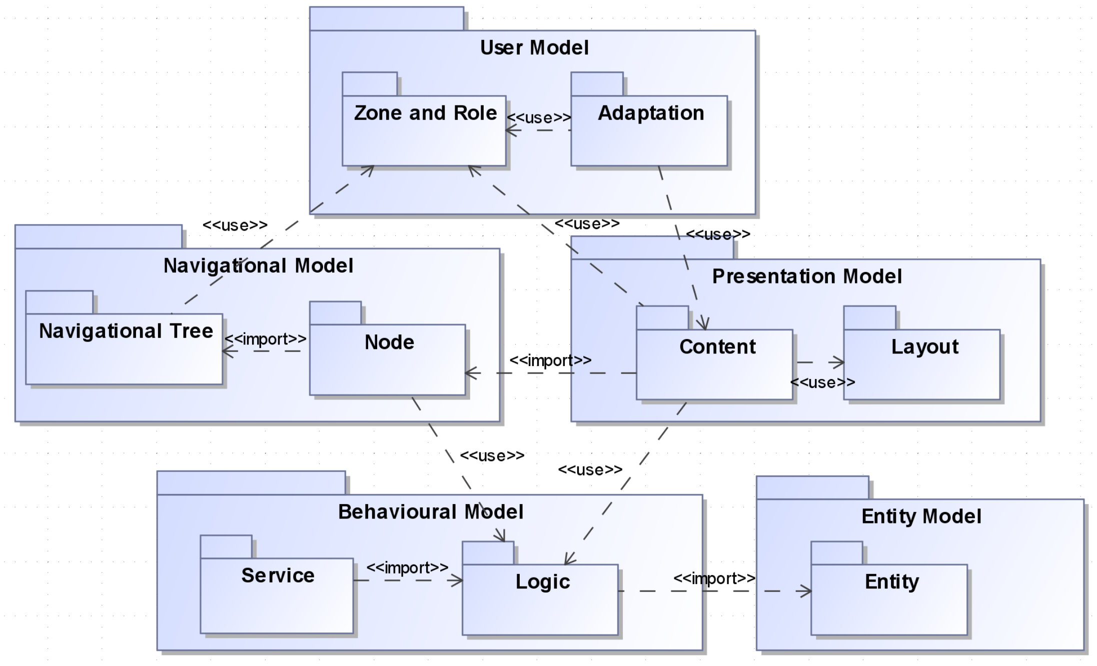


Figure 3.2: Diagrams in MoWebA

to the navigational structure, role and domain definition. In this stage a Navigational Tree Diagram is defined to organise the system basic functionalities in a hierarchical way. The Role and Zone diagrams are created considering the potential users identified at stage 1. An Entity diagram defines the structure and the static relationships between classes identified in the problem domain. Stage 3 defines the navigational behaviour for each node through the Node diagram. Stage 4 defines which elements are going to be displayed on every presentation page using the Content diagram. The pages structure (positions of headers, menus, footers, among others) is also defined through the Structure diagram. In addition, structural composition of business process and transactional procedures are defined with the Logic diagram. In Stage 5 the main activity is to personalise the models through the Adaptation model. MoWebA proposes Source and Rules diagrams to model different kinds of adaptations (i.e. adaptive). Stage 6 proposes a detailed definition of each service or action identified at Logic and Content diagrams using the Service diagram. Stage 7 contemplates the architectural and platform aspects. This stage is done in a semi-automatic way. It proposes an enrichment of existing models in order to consider aspects related to the final architecture of the system (e.g. RIAs, SOAs, 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 modelling process is an iterative and incremental process, allowing for diagram refinement. Next sub-sections describe the different stages of the modelling process.

3.2.1 Stage 1: Identify Potential Users and Functional Requirements

As a first stage, we need to specify the main goal of the system. In the example, the main goal could be stated as follow: "To develop a Web-Based Application for academic management of a University in order to process student registrations, course monitoring, and school, department and career management; oriented to students, professors and administrators".

Early requirements are out of the scope of MoWebA. However, we assume that the designer may use specification scenario based techniques that already exist in order to get a good understanding of the problem domain [95]. MoWebA covers the Use Case Diagram with the identification of the different actors and a list of functions associated to the actors (see Figure 3.4). In this classification, there are some similar or common functions that should be re-organised or re-grouped. In the next stage, we will refine the potential users, identify the domain model and define a navigational structure based on the functionalities defined in this stage.

3.2.2 Stage 2: Specify Navigational Structure, User roles and Domain

This stage defines the following artefacts: Navigational Tree, Role-Zone and Entity.

Navigation in MoWebA covers both structural and behavioural aspects. The structural aspects are modelled in this stage in terms of "navigable nodes" and their relationships. A "Navigable Node" is a functional unit of the system, and

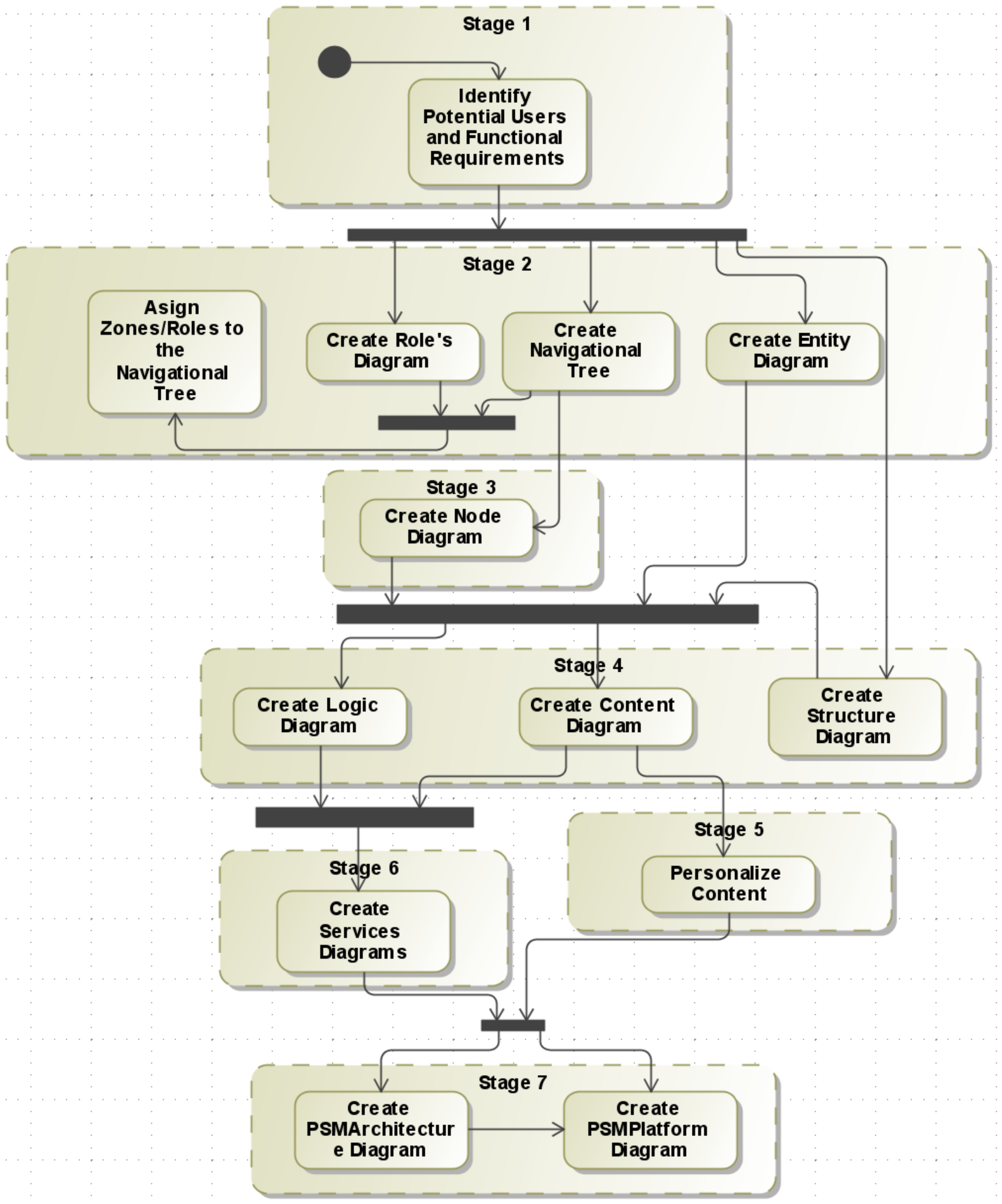


Figure 3.3: The MoWebA Modelling Process

the navigation is “the change from one navigational node to another as a result of an invocation from the user or an external agent”. Therefore, navigation occurs

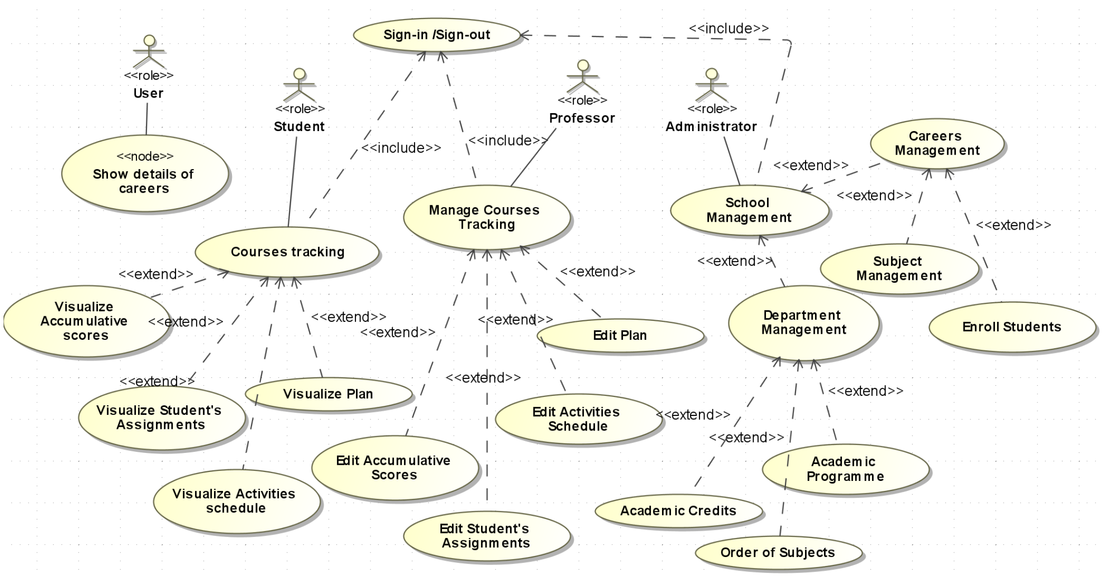


Figure 3.4: The Use Case diagram for the Academic System

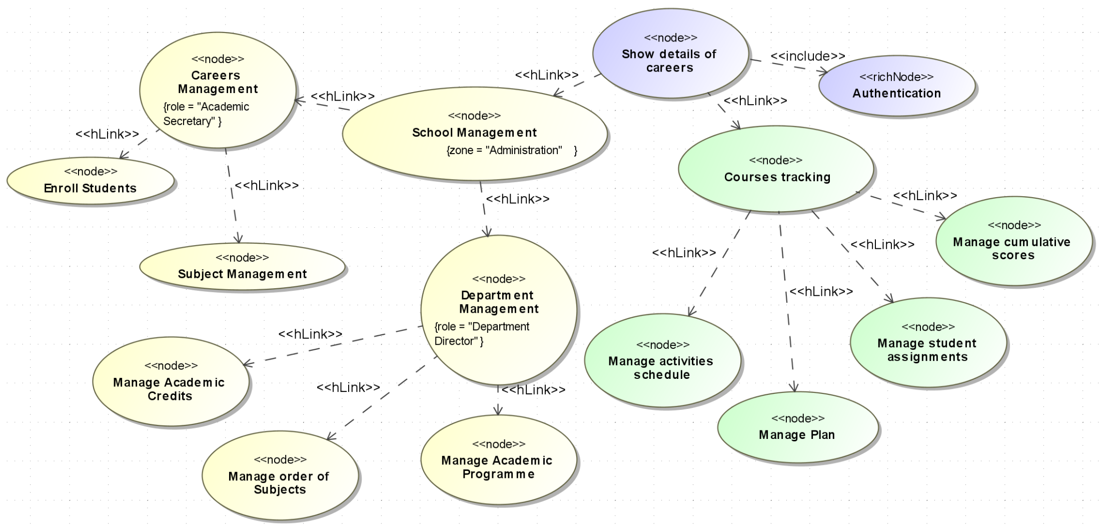


Figure 3.5: Navigational Tree for the Web-based Academic System

when an external agent interacts through the invocation of a "Navigational Node".

The Navigational Tree diagram represents the application's navigational space and it is composed of zero or more navigational elements. These elements may be nodes or links. A navigational node connects to other nodes by means of relationships, called hard links, which denote a hierarchy in the Navigational Tree. The Navigational Tree is defined following four activities: i) analyse the use cases defined at stage 1; ii) analyse the actors diagram for a functional unit hierarchy definition; iii) define an initial point for the hierarchical structure; and iv) create a structure considering the relationships between Use Cases and actors. Figure 3.5 shows an example of a Navigational Tree.

The Navigational Tree has remarkable differences with other approaches in the fundamental concept of the "navigable node". The most mentioned methodologies in the literature create the navigational structure from the conceptual model. This has two important implications: i) the level of granularity of navigational elements are directly related to structural elements (e.g. classes); and, ii) navigation is obtained considering the way information is structured (e.g. classes relationships), not the way it is accessed. In the case of MoWebA, navigation structure is defined considering the functional units as the granularity level, and navigation paths are defined considering hard links between the units, defining though the navigation from the way users interact with the system. With this approach it is possible to model a functional-oriented navigational structure, and to generate several exploration levels, which represent menus and sub-menus, keeping the user located by using "breadcrumbs" and "history of navigation".

However, hard links are not sufficient to specify the navigational structure of an application, because there are situations in which navigation through a different context will be necessary (e.g. once authenticated, the user must specify the destination node). To meet this need, we define the softLink, which will be specified in the Node diagram (next section).

To formalize the modelling and transformation processes, we used the MOF language for the abstract syntax definition, and UML profile extension for the concrete syntax of the modelling language. The MOF definition specifies MoWebA in terms of a metamodelling language, allowing the definition of concepts in a more rigorous way. Figures 3.6 and 3.7 shows the navigational tree metamodel and the corresponding UML profile. In this case, only two stereotypes («node» and «hLink») are necessary. The Role diagram represents the hierarchy of user roles, that is, groups of users that can access the same functionalities. For this diagram MoWebA proposes the use of the UML actors stereotyped with «role».

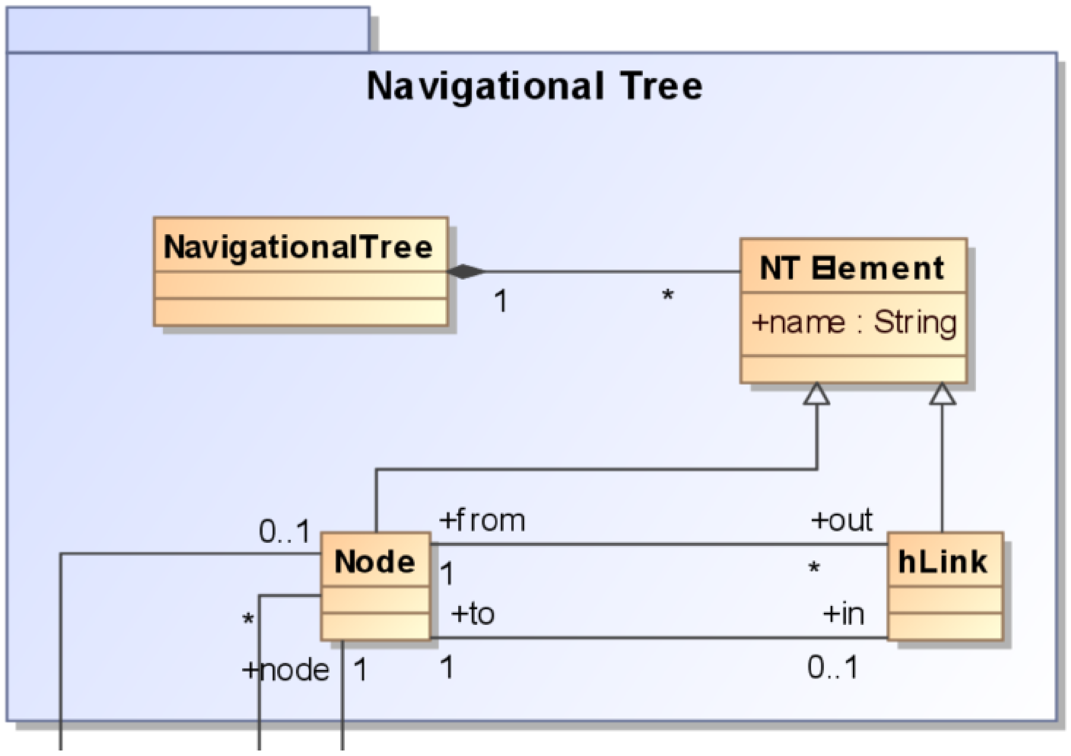


Figure 3.6: Metamodel of the navigational tree diagram

The Zone diagram represents contexts containing certain behavioural profiles in relation to each other. The zones provide system designers the possibility to explicitly define different contexts with multiple roles assumed by users. There may be several zones defined in a system, each one accessed by several roles, and, in turn, users could have more than one role. For example we define a zone in which both students and teachers can access (e.g. subjects or career) and, a different zone for managers (e.g. department). Moreover, the zone could be relative, that is, dependent on a domain class indicating that for a user to assume a certain role, additional information is needed (e.g. at the "Academic" zone, which is accessed by Professor and Student roles, each user would take at most one of these roles for each subject; see Figure 3.8).

To complete the Role and Zone modelling task, it is necessary to define roles/-zones access privileges on the elements of the system by establishing a dependency relationship between a «role» or a «zone» and elements of another diagram (i.e.

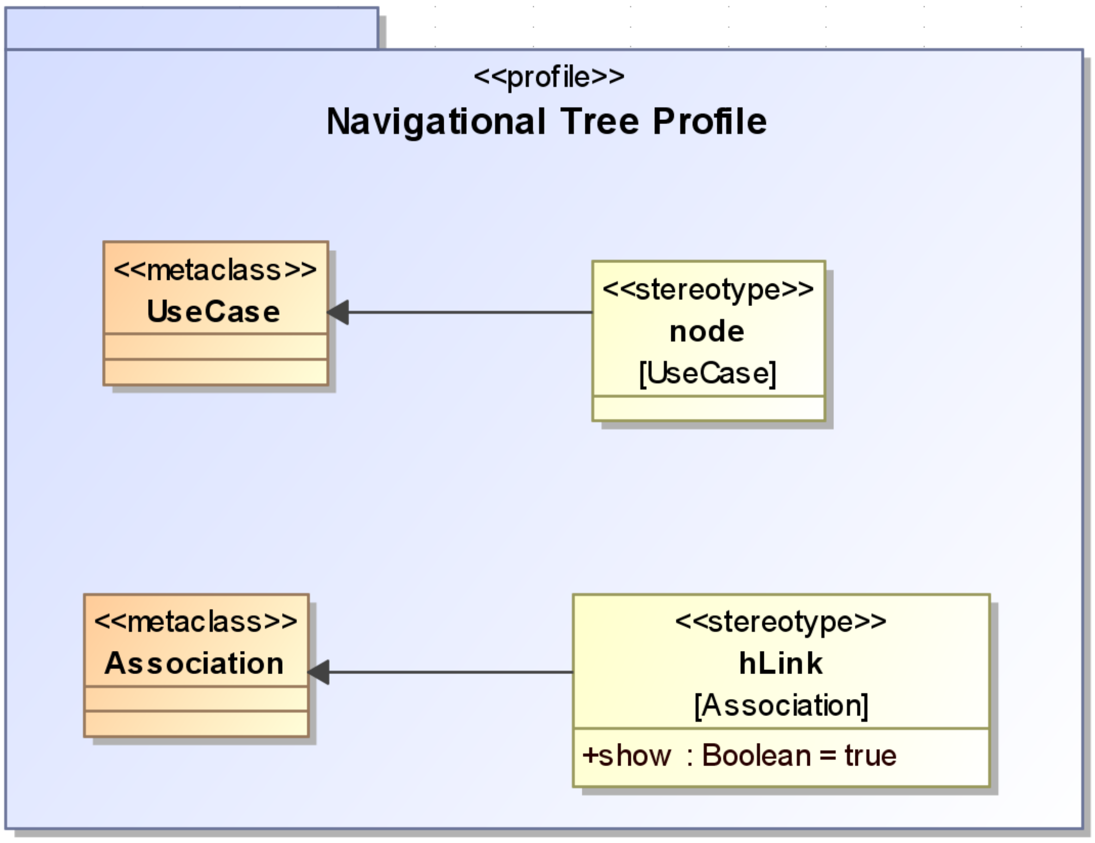


Figure 3.7: UML Profile of the navigational tree diagram

nodes access privileges in Navigational Tree diagram). A relationship implies that the elements are available for the specified role/zones assigned. Such relationships would be refined in the next stages of other diagrams (logic, presentation, among others). In Figure 3.5 the node "Course tracking" has privileged access to the "Academic" zone, indicating that both students and teachers have access to that node. The same privileges are inherited by the nodes below in the hierarchy, maintaining access restricted to students and teachers.

Figures 3.9 and 3.10 present the zone and role metamodel and UML profile. A role diagram is composed of one or more RD elements, which could be specialized in “User”, “Role” and “Zone”. Each zone can be composed in one or more roles which could have attributeRoles. The zones could be aggregated by other zones, and roles can be defined in a hierarchy.

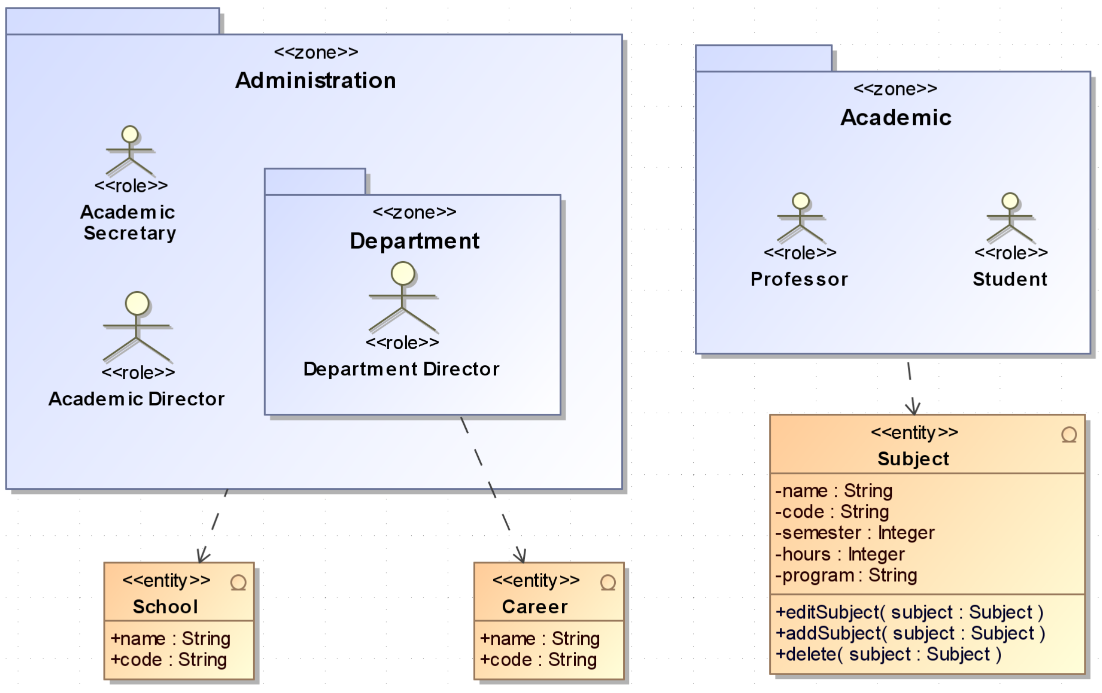


Figure 3.8: Example of zone diagram

For the Entity diagram definition, MoWebA adopts the UML class diagram, where each class is stereotyped with «entity». Entities, attributes and relationships are identified by the functionalities description of stage 1. A simple example of an Entity diagram is shown in Figure 3.11.

Figures 3.12 and 3.13 presents Entity metamodel and UML Profile that includes a new stereotype («entity»).

3.2.3 Stage 3: Specify Navigational Behaviour

Each node in the Navigational tree must have an associated Navigational Node diagram representing its navigational behaviour. The Node diagram is defined using the UML State diagram.

There are three categories of states: flow states, virtual states and final states. Flow states are transient and as such, they are visited only momentarily to create linkages with other elements of the diagram. Flow states can be further classified into four types: 1) initial states, 2) pseudo states, 3) junctions, 4) and service

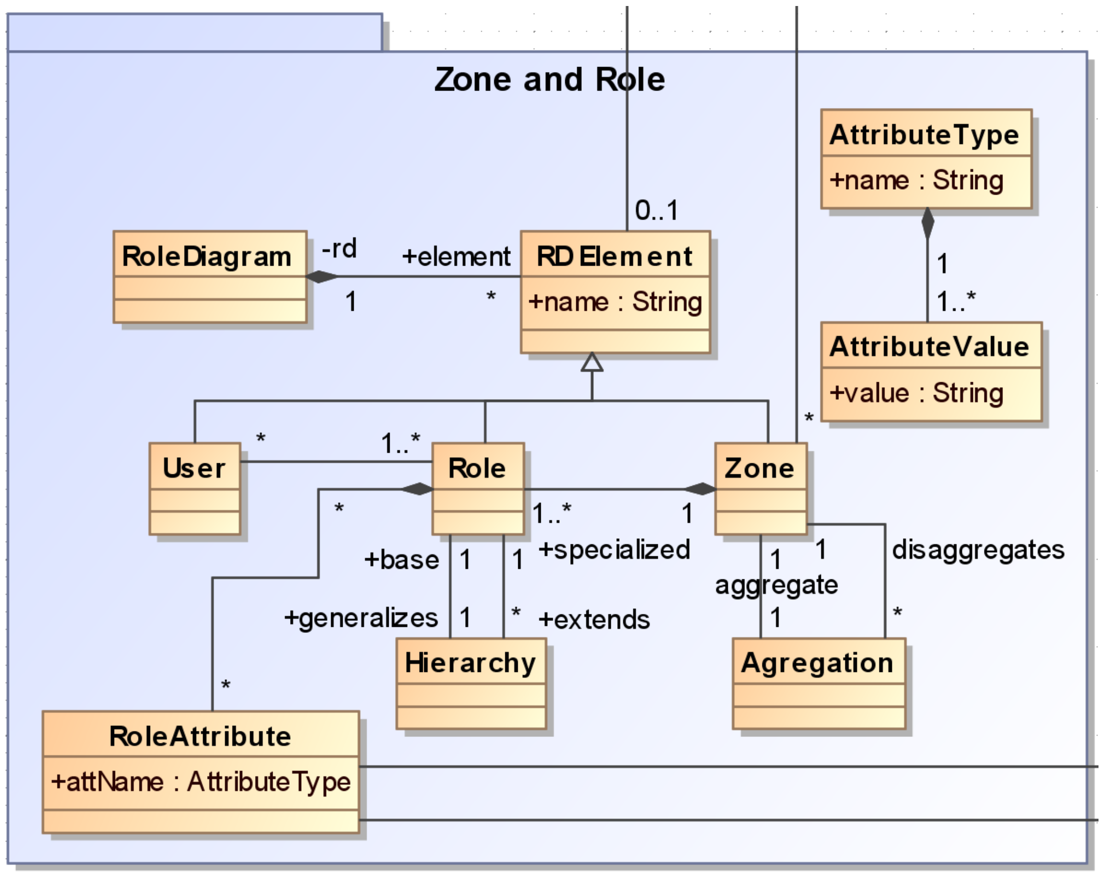


Figure 3.9: Metamodel of the zone diagram

states, which model the services provided by the node. Virtual states represent stationary states indicating the fact that the navigation flow remains in a "virtual point" within a node, waiting for an interaction from an external agent. In stage 4, each virtual state will be linked to a presentation page.

The transitions between two states (o state nodes) are specialized in two subtypes: the control flow transitions and the hyperlinks. The control flow models the natural control transfer that occurs between two states, without requiring an external user interaction. The hyperlink models a transition between two states resulting from an invocation of an internal link, which leads to an interaction between the user and the system. A control flow transition can only have a flow state as source, and any type of state as target (e.g. the transition between the service "Login" and "Error Message"). The hyperlink transition can only have

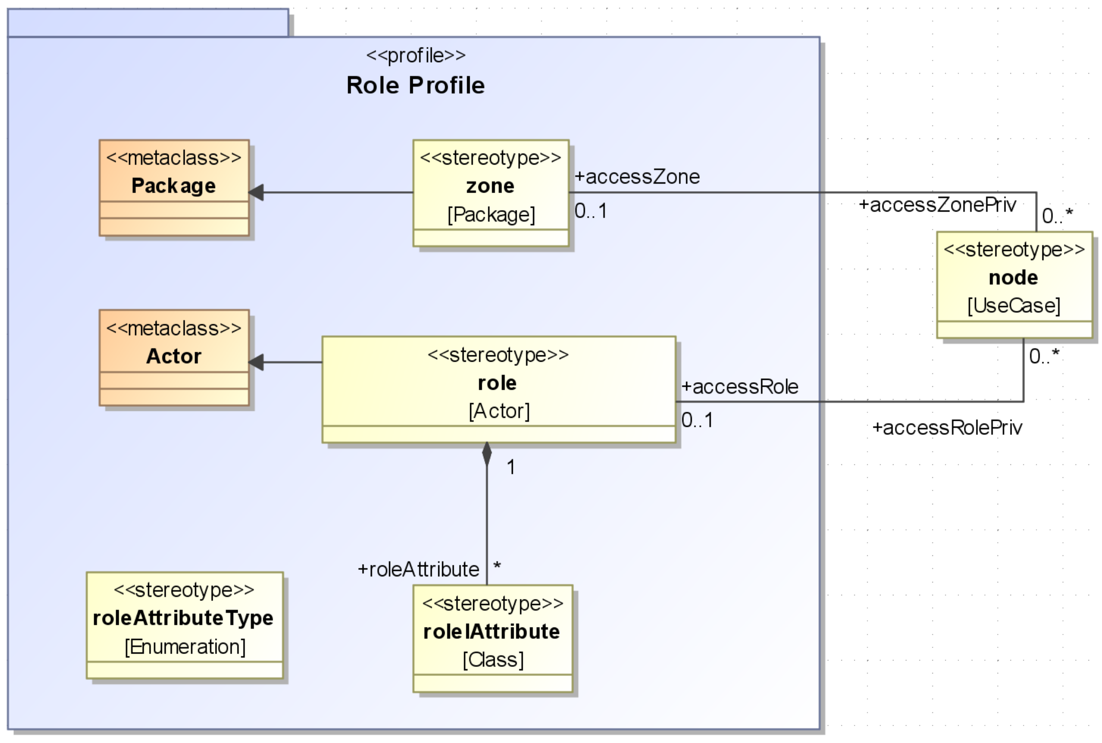


Figure 3.10: UML Profile of the zone diagram

a virtual state as source, and any state as target (e.g. the transition between "Entering data" and service "Login"). Hyperlinks defined in the node diagram correspond to possible internal navigations, triggered by user interactions. The final state can be connected to another node in the navigational tree; if there is such linkage, it defines a soft link (sLink). This will allow navigation to a unit not directly linked to the functional node of the navigational tree structure.

Figure 3.14 shows an example for the authentication process in which the user has to type a user name and a password ("Entering data" virtualState), then a login service is executed to validate data, and finally, depending on the results, an error message will appear ("Error Message" virtualState) or a soft link will take the user to the root node of the system ("sLinkNode=Show details of careers").

The Node diagram allows modelling navigational behaviour aspects obtained from dynamic interactions with the user.

As shown in Figure 3.15, a node diagram is composed of ND Elements (node

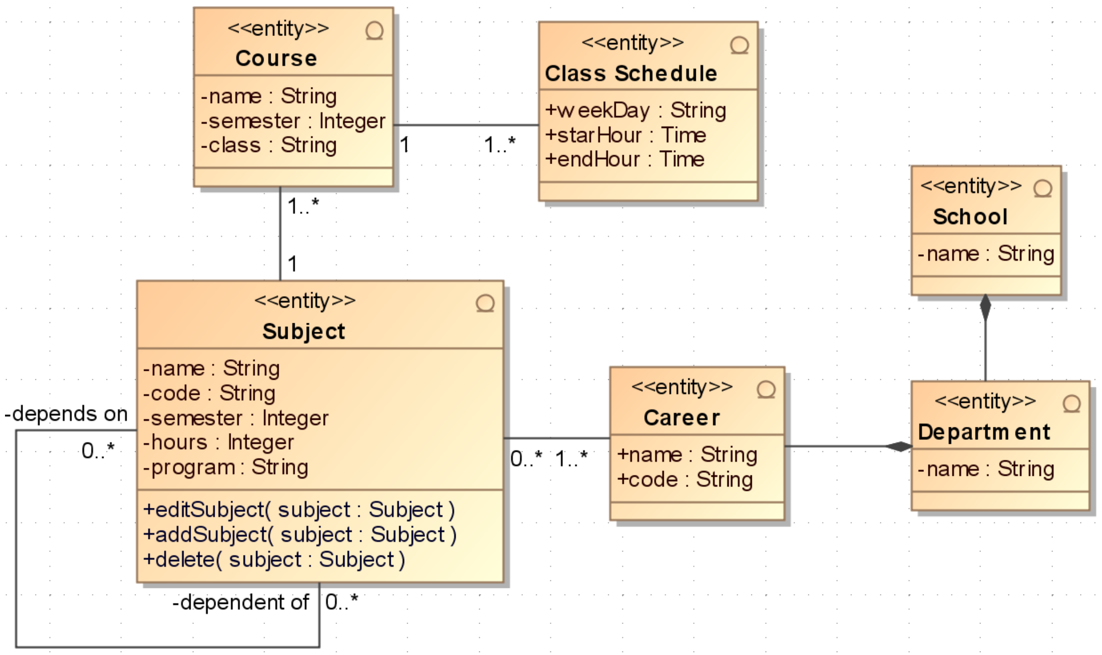


Figure 3.11: Simplified entity diagram for the Web-based Academic System

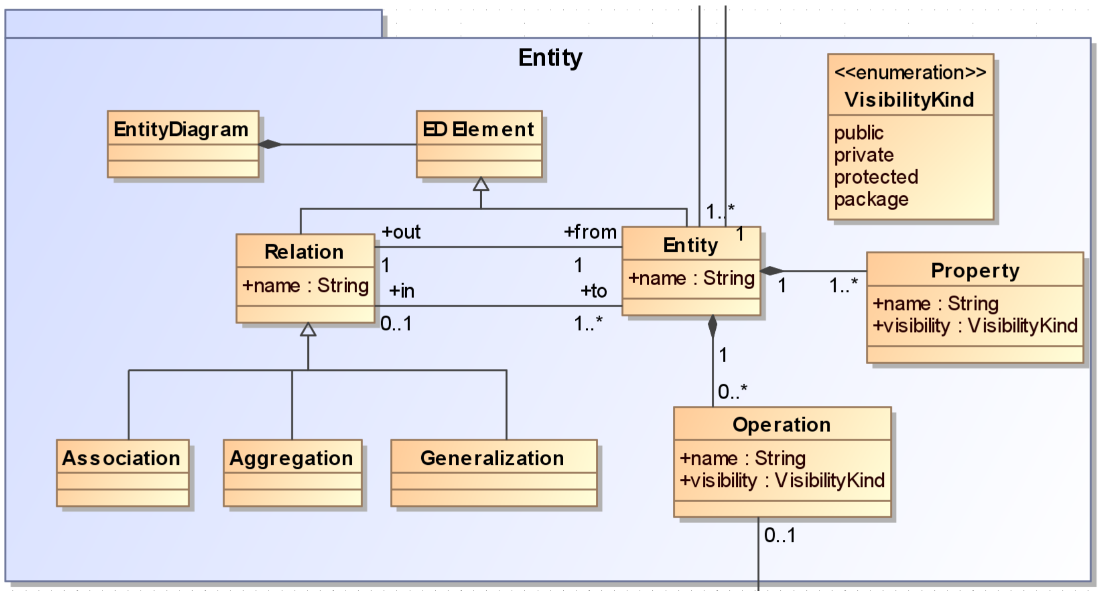


Figure 3.12: Metamodel of the entity diagram

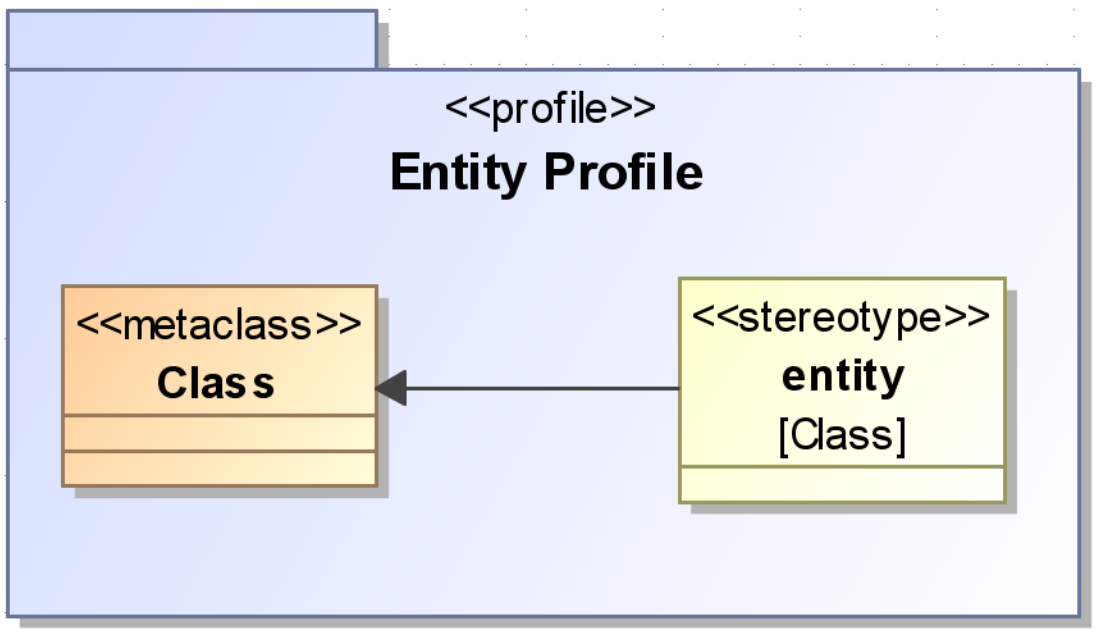


Figure 3.13: UML Profile of the entity diagram

diagram elements). The ND Elements are classified into State and Transition. States in turn are specialized into FlowState, FinalState and VirtualState. On the other hand, Transitions can be classified as sLink, HyperLink, or ControlFlow. Finally, a number of relationships between the elements have been defined indicating associations that must be considered in order to comply with the different proposed constraints.

In the corresponding UML Profile definition (figure 3.16), it is possible to notice that the «state» stereotype is an extension of the State UML metaclass. The «transition» stereotype is an extension of the Transition UML metaclass, and the «sLink» stereotype is an extension of the FinalState UML metaclass. This figure also shows that «virtualState» and «service» are specializations of «state», and «transition» is specialized in «hyperlink» and «controlFlow». Finally, the association between «virtualState» and «presentationPage» establishes that for each «virtualState» of the Node Diagram there should be a «presentationPage». The association between «sLink» and «node» allows modellers to link a destination node to a final state in the Node diagram.

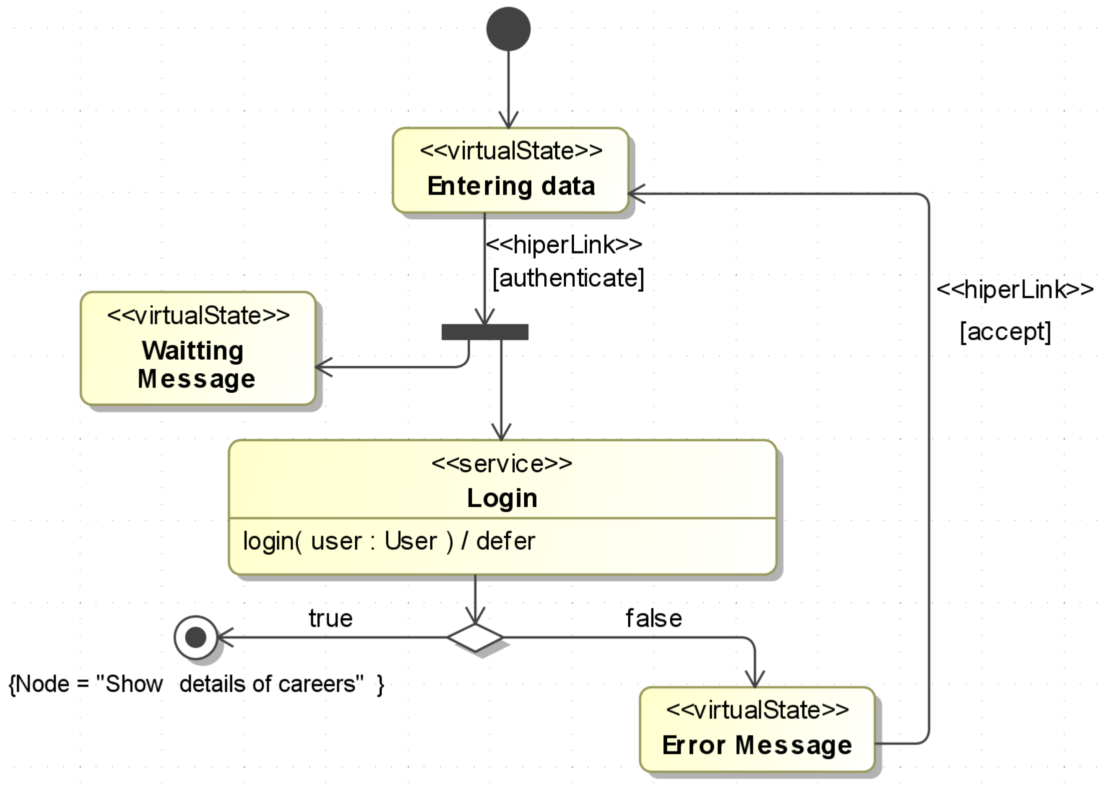


Figure 3.14: Node diagram for the Authentication tree node

3.2.4 Stage 4: Specify Logic Behaviour and Presentation

To consider the behavioural modelling, MoWebA defines two diagrams: Logic Behaviour and Service diagrams. The Logic Behaviour diagram encapsulates and structures all the behaviour actions (business processes and transactional procedures) that affect the system. This is done by defining classes stereotyped with «process» and «valueObjects». The "process" class encapsulates business processes that represent complex transactions and are associated through a dependent relationship with one or more classes of the Entity diagram. These dependency relationships imply that the partners are accessed by the operations defined in the process. On the other hand, the "valueObjects" class encapsulates data, and depends on one or more entities, containing a subset of attributes defined in the dependent classes. Every service identified in other diagrams, should also be included into the Logic Behaviour diagram as a service for some process.

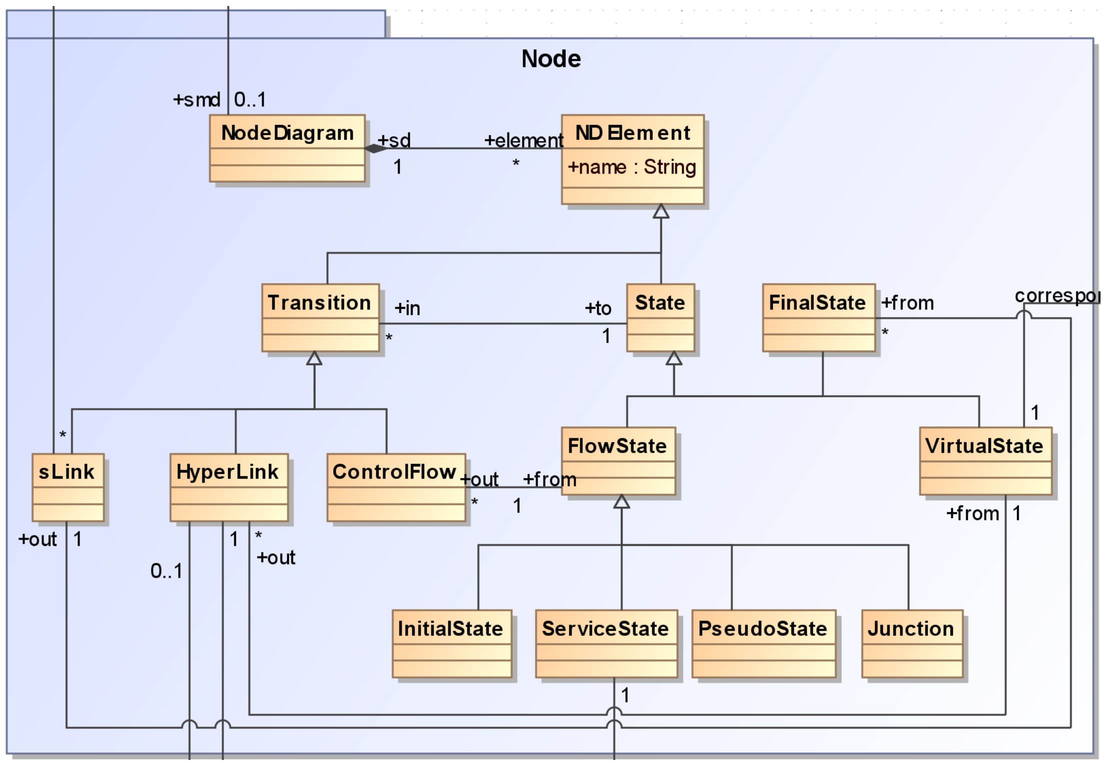


Figure 3.15: Metamodel of the node diagram

Furthermore, value objects provide domain visibility to the presentation layer. This means that access to the domain has to be done by appropriate value objects defined at the logic behaviour layer. The other behavioural diagram, called Service diagram, will be explained in stage 6.

A simplified example of Logic Behaviour diagram is shown in Figure 3.17 representing a logic process called "Authentication" which is conformed of two services (login, logout). It is important to notice that the "login" service has been already defined at the Navigational Node diagram "Authentication" (see Figure 3.14). In Figure 3.17 , we define two «valueObject» elements, SubjectVO and CareerVO. Notice the dependency between entities and value objects (e.g. SubjectVO and the Subject entity).

The LD Elements of the Logic Behaviour metamodel (see Figure 3.18) are classified into ValueObjects and TProcess. The ValueObjects are composed of Attributes, and the TProcess of Services which can be defined in other diagrams

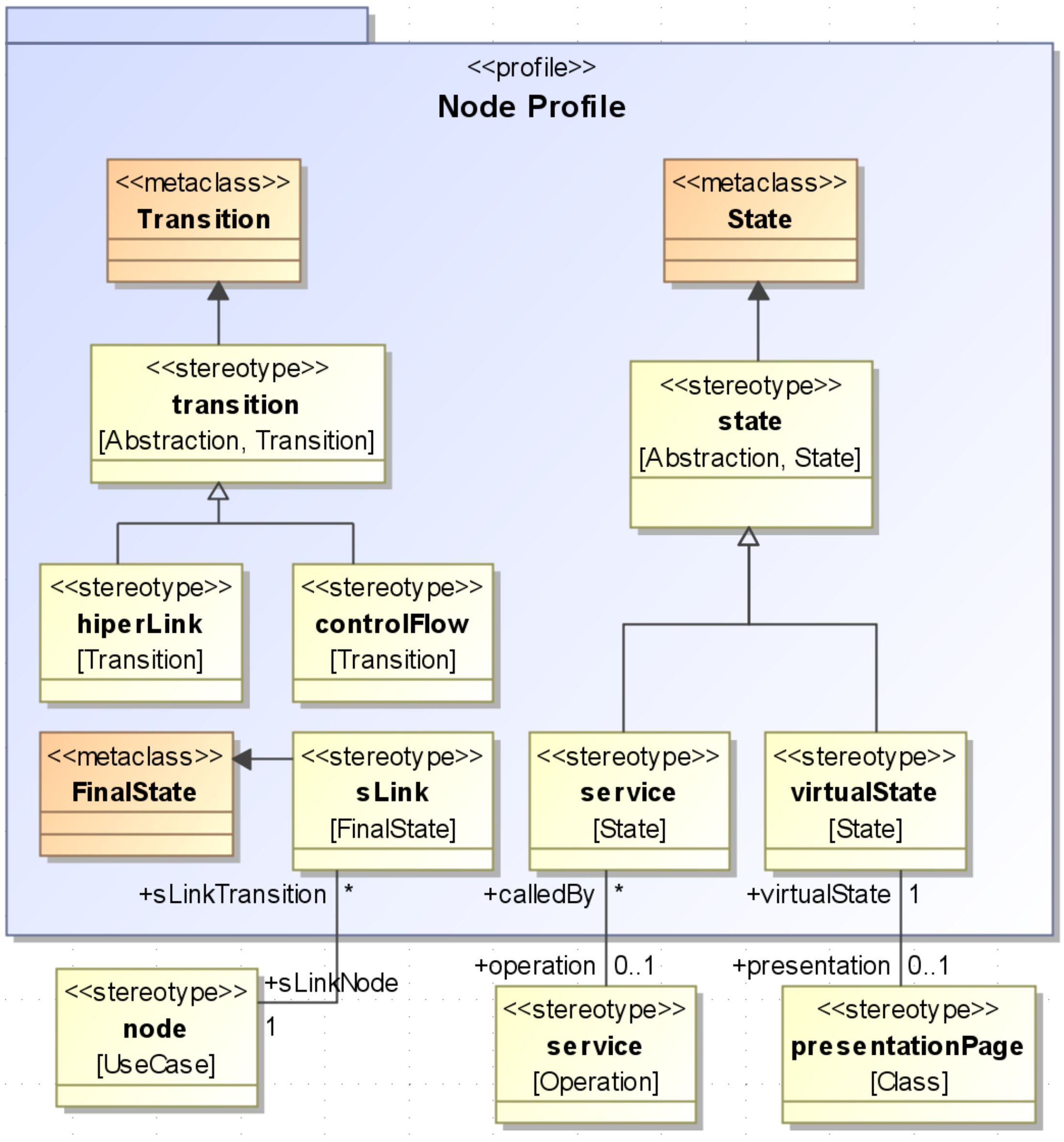


Figure 3.16: UML Profile of the node diagram

(e.g. services defined in the node diagram).

The presentation is mainly aimed to facilitate the interaction with the outside world and to provide the necessary elements for users to successfully perform tasks, such as entering data, enabling processes and browsing. For the Presentation Model, MoWebA considers the following aspects: the presentation content;

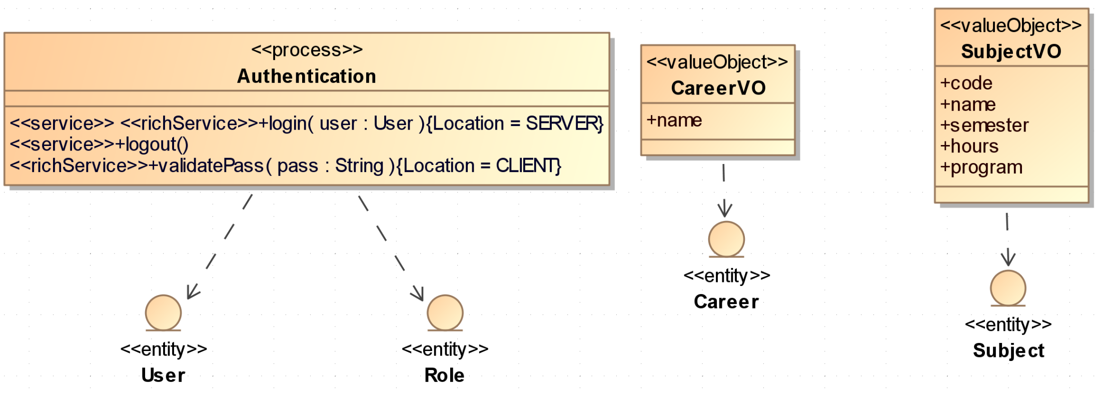


Figure 3.17: Logic Behaviour Diagram

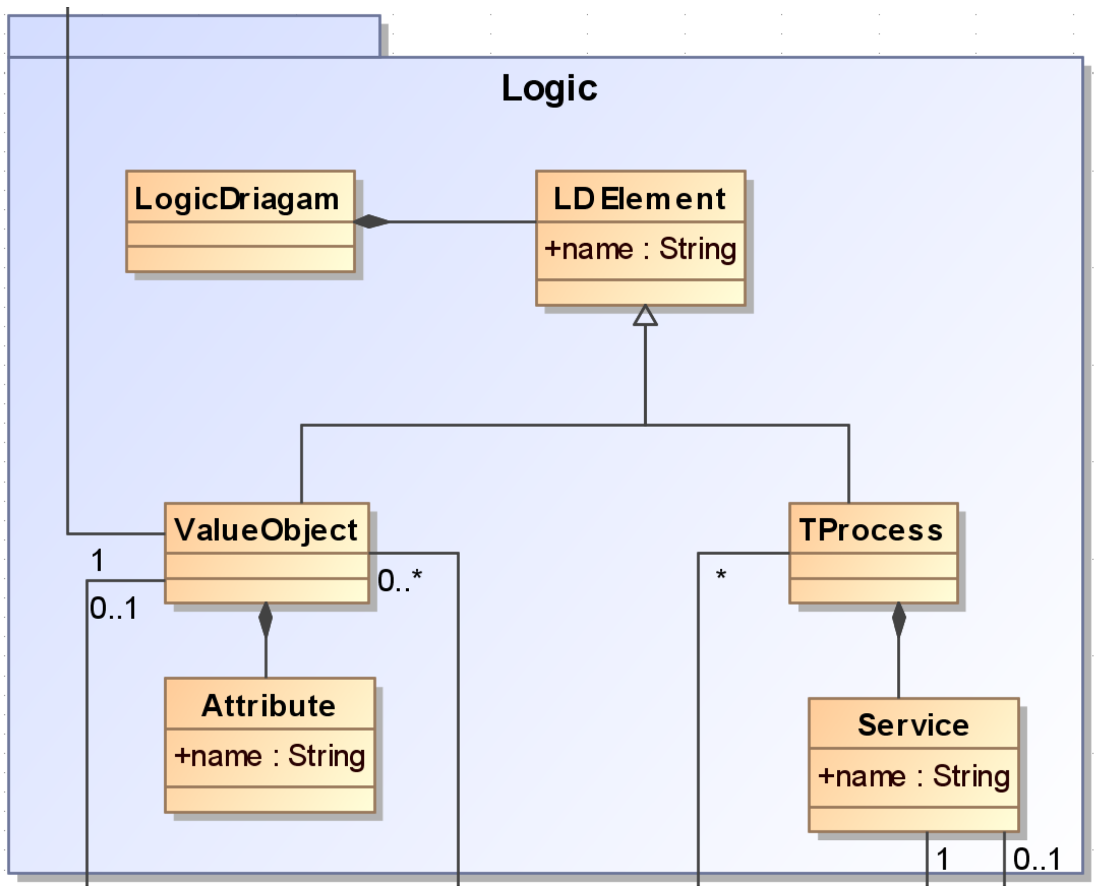


Figure 3.18: Metamodel of the Logic Behaviour Diagram

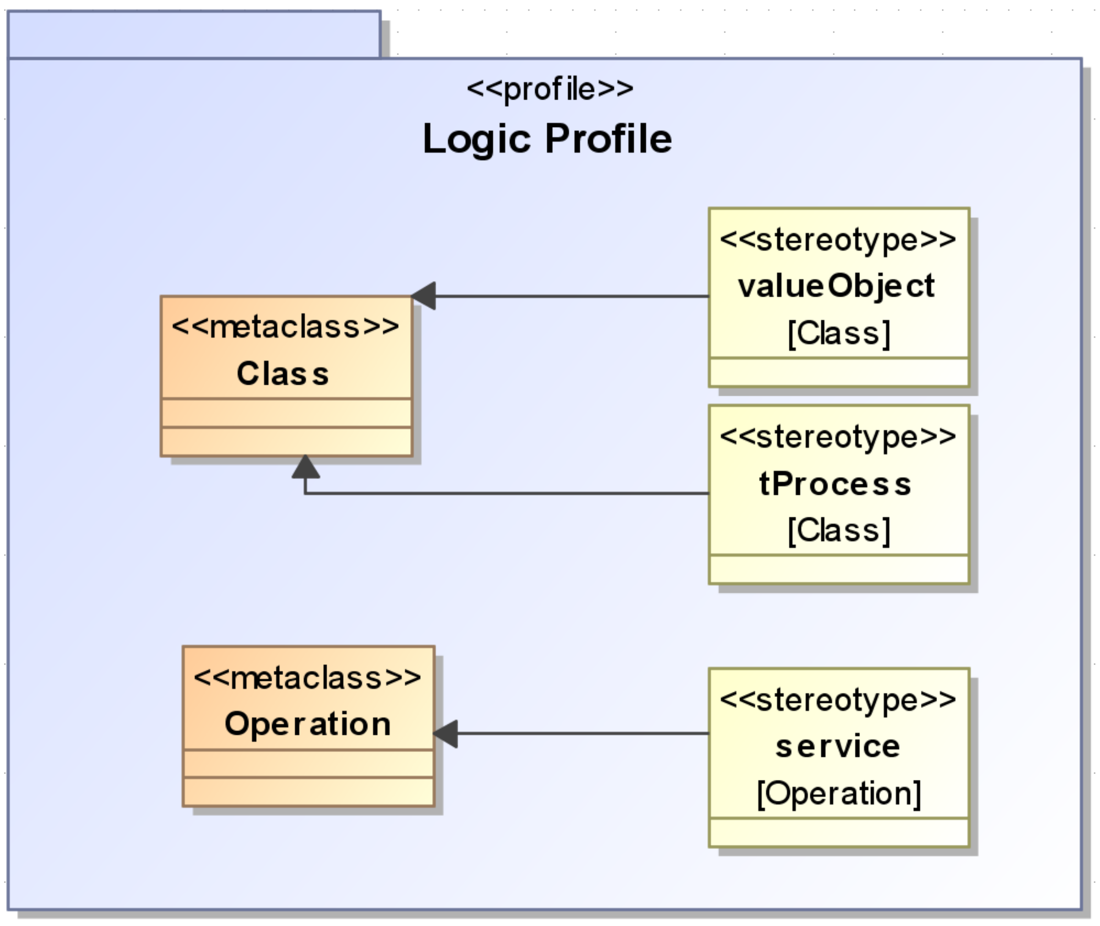


Figure 3.19: UML Profile of the Logic Behaviour Diagram

the presentation structure; the format of elements within each region; and the elements' style. Thus, MoWebA defines two presentation diagrams: Content and Structure diagrams.

The Content diagram allows modellers to specify the different elements that will be presented to final users in each page. The diagram consists of a set of presentation pages, each one related to a «virtualState» of the Node diagrams, which contain one or more «compositeUIElements». Each «compositeUIElements» class can have attributes classified as follows: static attributes, which represent static information not related to any other element of the different diagrams (e.g. the title of the web page or static text information); and binding attributes, which

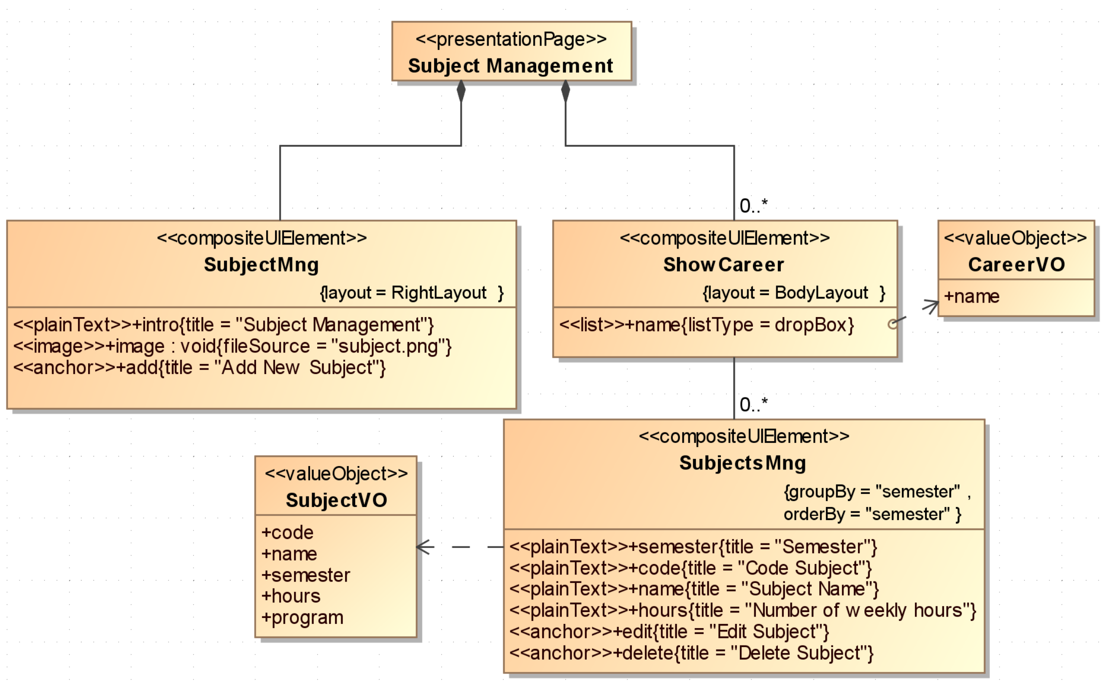

<<presentationPage>>
Subject Management
<<compositeUIElement>>
SubjectMng
{layout = RightLayout }
<<plainText>>+intro{title = "Subject Management"}
<<image>>+image : void{fileSource = "subject.png"}
<<anchor>>+add{title = "Add New Subject"}
0..*
<<compositeUIElement>>
ShowCareer
{layout = BodyLayout }
<<list>>+name{listType = dropBox}
<<valueObject>>
CareerVO
+name
0..*
<<valueObject>>
SubjectVO
+code
+name
+semester
+hours
+program
<<compositeUIElement>>
SubjectsMng
{groupBy = "semester" ,
orderBy = "semester" }
<<plainText>>+semester{title = "Semester"}
<<plainText>>+code{title = "Code Subject"}
<<plainText>>+name{title = "Subject Name"}
<<plainText>>+hours{title = "Number of weekly hours"}
<<anchor>>+edit{title = "Edit Subject"}
<<anchor>>+delete{title = "Delete Subject"}

Figure 3.20: Subject Management Presentation Page

allows the transition from one state to another (e.g. a submit button). The presentation classes can also display information from a «valueObject» by establishing a dependency relationship between the class and a "valueObject" defined in the Logical Layer diagram. Figure 3.20 shows the presentation page "Subject Management" which is made up of two «compositeUIElements»: SubjectMng and ShowCareers. The composite element ShowCareers, contains a DropBox attribute to display all the available careers, and an association with the «compositeUIElement» SubjectsMng, to display all available subjects of a specific career. It is worth noting that the data that will be shown in the name attribute of ShowCareers, is defined by the dependency relationship between ShowCareers and CareersVO (this is also true for SubjectMng and SubjectVO). Finally, groupBy and orderBy tagged values defined for SubjectMng allows grouping and ordering subjects by semester.

Figure 3.21 shows the Presentation Diagram composed of one or more PresentationPages, which aggregate different PD Elements. The PD Elements are classified into UIElements and CompositeUIElements. UIElements in turn are

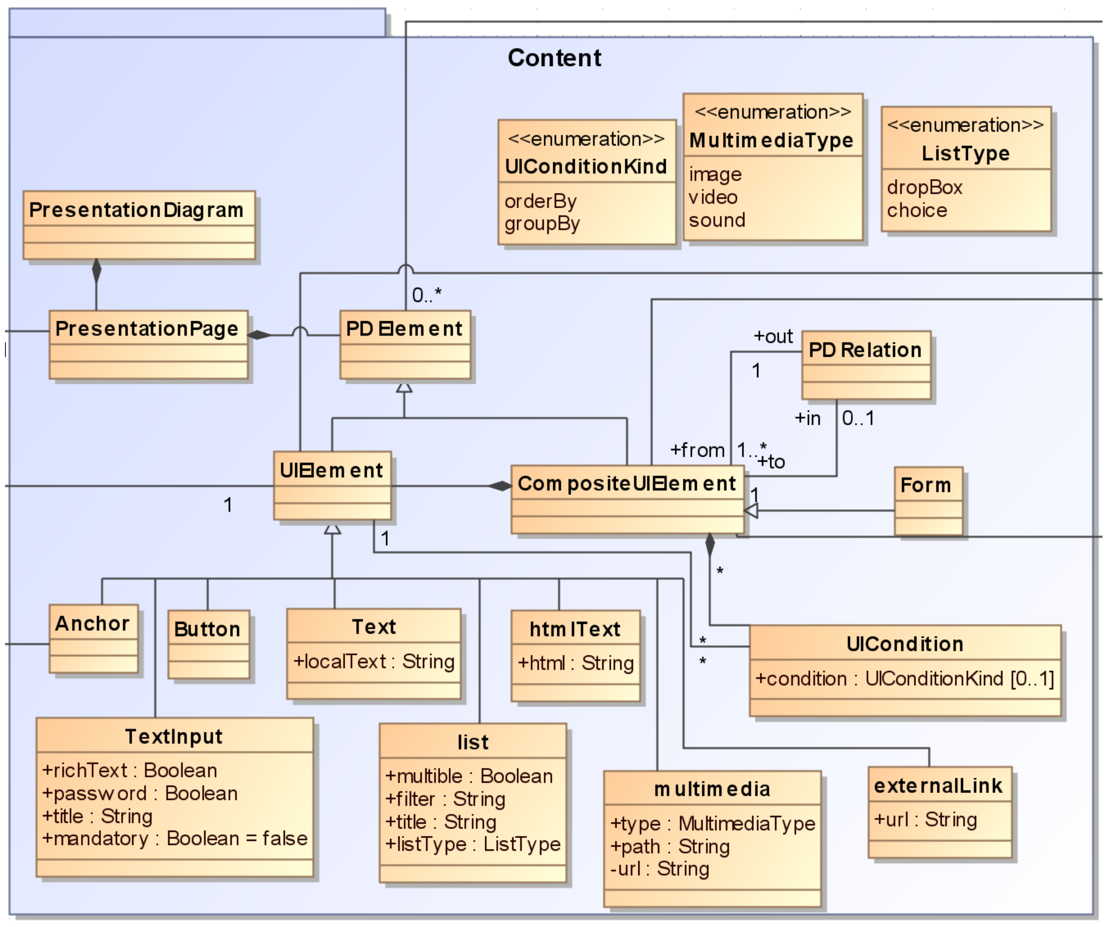


Figure 3.21: Metamodel of Content Diagram

specialized into Anchor, TextInput, Button, Text, List, htmlText, Multimedia and ExternalLinks. Each element has properties in order to model additional aspects related to constraints, limitations, possible values, among others.

The Structure diagram is used for the definition of page areas (e.g. header, footer, or menu areas). UML packages stereotyped with «layout» represent regions. Each region can be composed of other sub-regions, and it is possible to define different layout structures for the same application (e.g. one structure diagram for each different target platform). It is also possible to define a basic content diagram for each region, which can then be complemented with the diagrams defined for each «virtualState». An example of the latter is shown in Figure

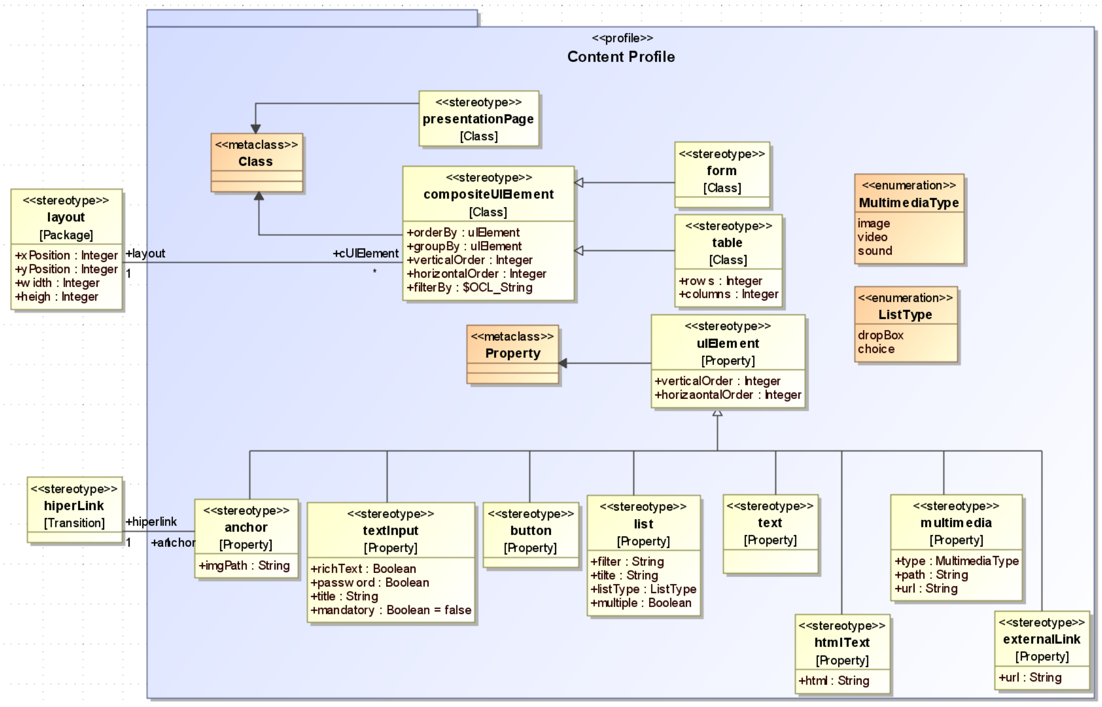

<<profile>>
Content Profile
<<stereotype>> presentationPage [Class]
<<metaclass>> Class
<<stereotype>> compositeUIElement [Class]
+orderBy : uIElement
+groupBy : uIElement
+verticalOrder : Integer
+horizontalOrder : Integer
+filterBy : $OCL_String
<<stereotype>> form [Class]
<<stereotype>> table [Class]
+rows : Integer
+columns : Integer
<<enumeration>> MultimediaType
image
video
sound
<<enumeration>> ListType
dropBox
choice
<<stereotype>> layout [Package]
+xPosition : Integer
+yPosition : Integer
+width : Integer
+heigh : Integer
+layout
1
+cUIElement
*
<<metaclass>> Property
<<stereotype>> uIElement [Property]
+verticalOrder : Integer
+horizaontalOrder : Integer
<<stereotype>> hiperLink [Transition]
+hiperlink
1
+anchor
<<stereotype>> anchor [Property]
+imgPath : String
<<stereotype>> textInput [Property]
+richText : Boolean
+password : Boolean
+title : String
+mandatory : Boolean = false
<<stereotype>> button [Property]
<<stereotype>> list [Property]
+filter : String
+tilte : String
+listType : ListType
+multiple : Boolean
<<stereotype>> text [Property]
<<stereotype>> multimedia [Property]
+type : MultimediaType
+path : String
+url : String
<<stereotype>> htmlText [Property]
+html : String
<<stereotype>> externalLink [Property]
+url : String

Figure 3.22: UML Profile of Content Diagram

3.20 and Figure 3.23. Figure 3.23 shows the basic content of the rightLayout region that will show the latest news available (ShowNews class), and some basic page information (RightElements class). On the other hand, Figure 3.20 shows the Content diagram for the “Subject Management” «virtualState». This diagram indicates that the elements of the “SubjectMng” class will be placed in “Rigth-Layout” of the Structure diagram, extending the basic content (news and basic information) of the region with the specific content of this page (SubjectMng elements). ShowCareers class, on the other side, will be placed in a different region of the Structure diagram ("BodyLayout"). Finally, to indicate the order in which presentation elements will be shown, a pair number property is defined, where the first number sets the vertical order and the second number the horizontal order.

With respect to the presentation style, even though it is considered a relevant aspect for the presentation layer, in our vision it is more reasonable to deal with style specifications in the ISM phase. Reasons for this decision are the style being very changing and normally taken into account in the final stages of

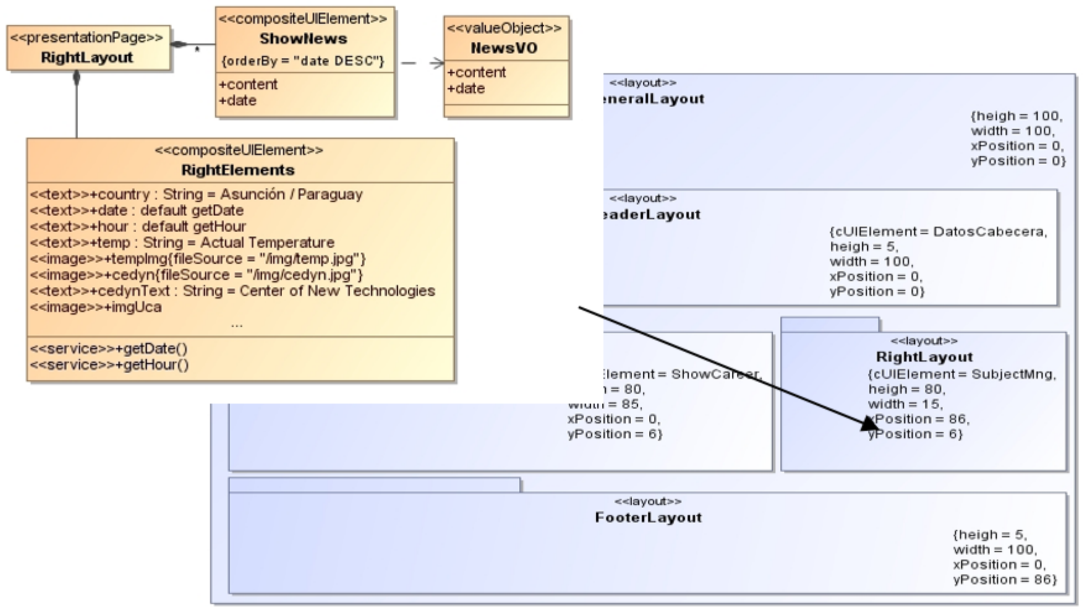

<<presentationPage>>
RightLayout
<<compositeUIElement>>
ShowNews
{orderBy = "date DESC"}
+content
+date
<<valueObject>>
NewsVO
+content
+date
<<compositeUIElement>>
RightElements
<<text>>+country : String = Asunción / Paraguay
<<text>>+date : default getDate
<<text>>+hour : default getHour
<<text>>+temp : String = Actual Temperature
<<image>>+tempImg{fileSource = "/img/temp.jpg"}
<<image>>+cedyn{fileSource = "/img/cedyn.jpg"}
<<text>>+cedynText : String = Center of New Technologies
<<image>>+imgUca
<<service>>+getDate()
<<service>>+getHour()
<<layout>>
GeneralLayout
{heigh = 100, width = 100, xPosition = 0, yPosition = 0}
<<layout>>
HeaderLayout
{cUIElement = DatosCabecera, heigh = 5, width = 100, xPosition = 0, yPosition = 0}
Element = ShowCareer, = 80, width = 85, xPosition = 0, yPosition = 6}
<<layout>>
RightLayout
{cUIElement = SubjectMng, heigh = 80, width = 15, xPosition = 86, yPosition = 6}
<<layout>>
FooterLayout
{heigh = 5, width = 100, xPosition = 0, yPosition = 86}

Figure 3.23: Structure Diagram and example of a Content Diagram for the Right-Layout

development; the lack of a standard language at the modelling phase to specify this aspect and; the possibility to separately differentiate style from other aspects, allowing modifications of the application without changing any code (e.g. with CSS templates we could change the style at any time, affecting the appearance of the application).

Figure 3.24 depicts the Structure Diagram metamodel, which is mainly composed of LD Elements. The LD Elements are classified into Layout, which can be composed of other layouts. The layouts define dimensions and positions properties.

3.2.5 Stage 5: Specify Personalisation

According to Weibelzahl, personalisation refers to both adaptability and adaptivity [147]. Adaptability requires user interaction in order to conceive personalisation (e.g. change colors, or types). On the other hand, adaptivity allows personalisation considering other factors without a direct user intervention (e.g.

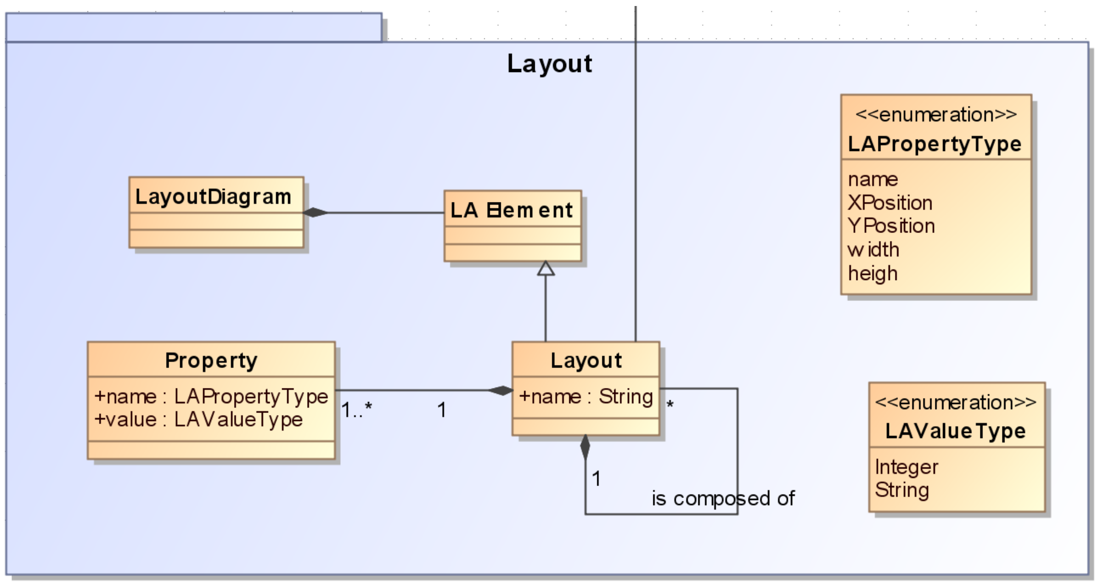


Figure 3.24: Metamodel of the Structure Diagram

suggest list of books based on previous purchases). In order to consider these concepts, MoWebA defines two diagrams: Information Source and Rule diagrams.

The Information Source diagram models user information needs for adaptation. The information sources refer to the system domain factors to be considered for rule conditioning, (e.g. in the example, an information source could be the level of knowledge for specific users). The next step is to define associations between sources and users considering the roles that they should play in the system. Therefore, we define a set of information sources and associate them with a given role; these are stereotyped with «roleAttribute». The «roleAttribute» stereotype is used to establish relationships between sources and roles, and it is possible to set default values to these attributes. Figure 3.26 shows an example, we have defined two sources (Preference and Knowledge), assigned roleAttributes to the Student, and assigned default values to these attributes (language=English and level=beginner). Such default values could be changed at any time in the future.

The Rules diagram allows the definition of "Condition-Action" rules that establish under which conditions a rule must be triggered in order to perform a specific action. The final result will be a dynamic adaptation of the system. An example of an adaptivity personalisation is a rule defined to filter exercise exam-

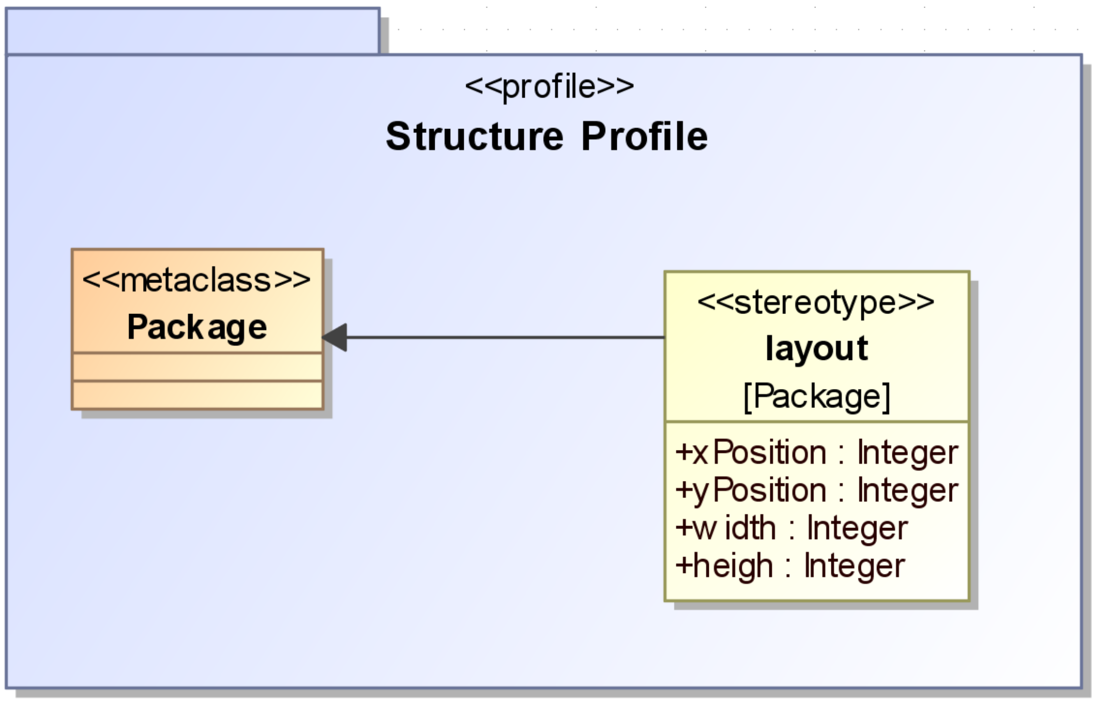


Figure 3.25: UML Profile of the Structure Diagram

ples, the filtering could be done based on types of exercise that the student has already solved.

There are two types of rules: i) general rules (e.g. if language is set to “English”, whenever a «text» element appears, it should be in English); and, ii) specific rules applied to specific elements (e.g. even though the font type is set to “normal”, a specific title of a page should be “large”).

Rules are specified using an OCL Expression as the tagged value of the class. For example, in Figure 3.27, the general rule called “LanguageRule” has been defined for «compositeUIElements» of the content diagrams, belonging to Academic Zone (i.e., the zone associated to the student and professor roles). The OCL expression defines a condition related to the language attribute, triggering the selectContentLanguage action if the default language is “English”. The behaviour of the selectContentLangage action must be specified in some way. In order to do this we define a process in the logic layer diagram called AdaptationService, and add the action selectContentLanguage as a «service» operation.

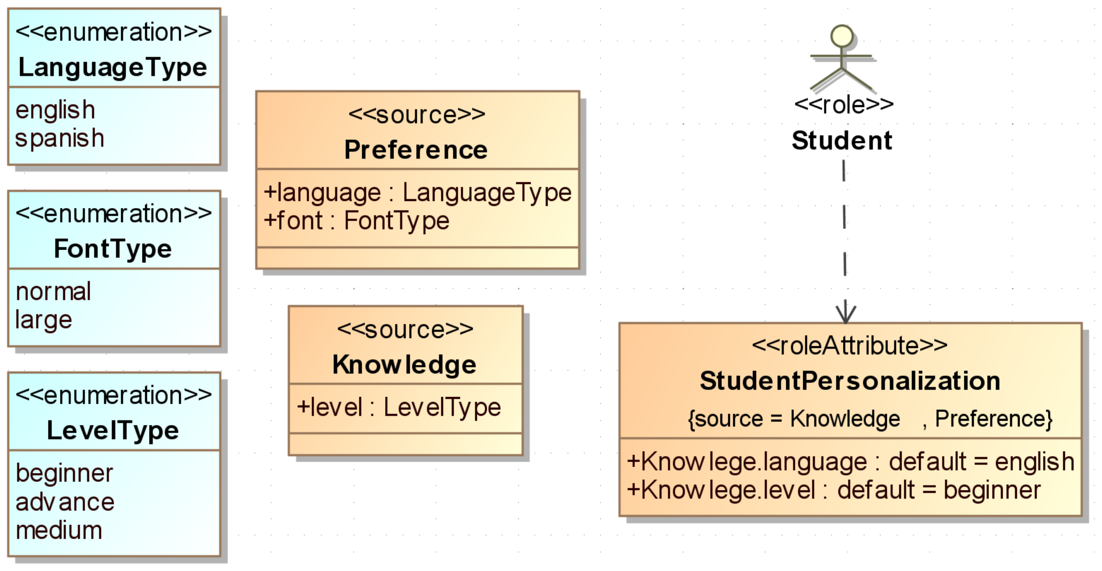
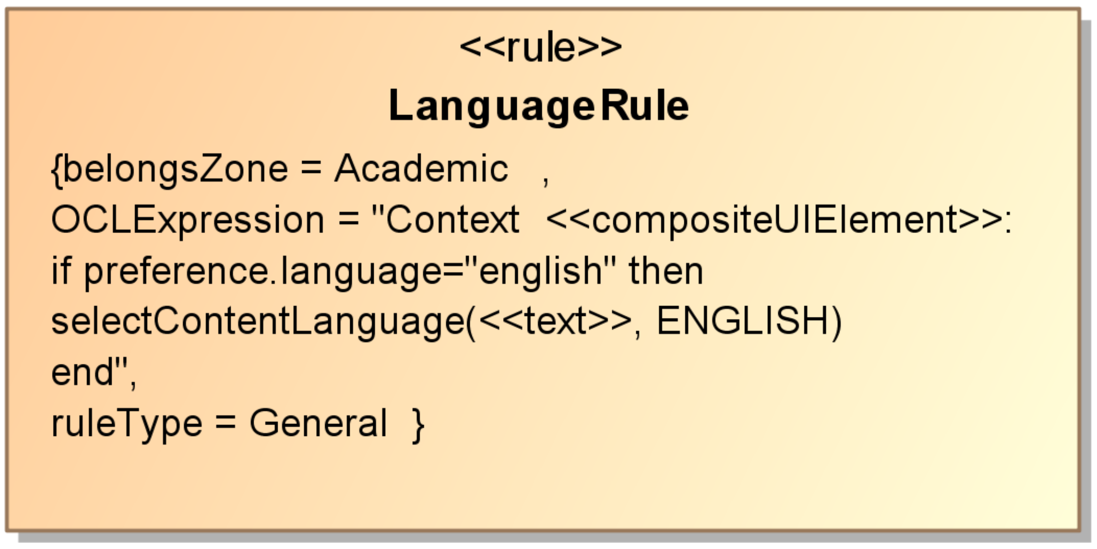


Figure 3.26: Source Information Diagram for Web-based Academic System



<<rule>>
LanguageRule
{belongsZone = Academic ,
OCLExpression = "Context <<compositeUIElement>>:
if preference.language="english" then
selectContentLanguage(<<text>>, ENGLISH)
end",
ruleType = General }

Figure 3.27: Rule example for language definition

The detailed behaviour of the selectContentLanguage «service» is then modelled in the service diagram, which will we be explained in the next section.

An Adaptation Diagram is composed of rules and sources (see Figure 3.28).

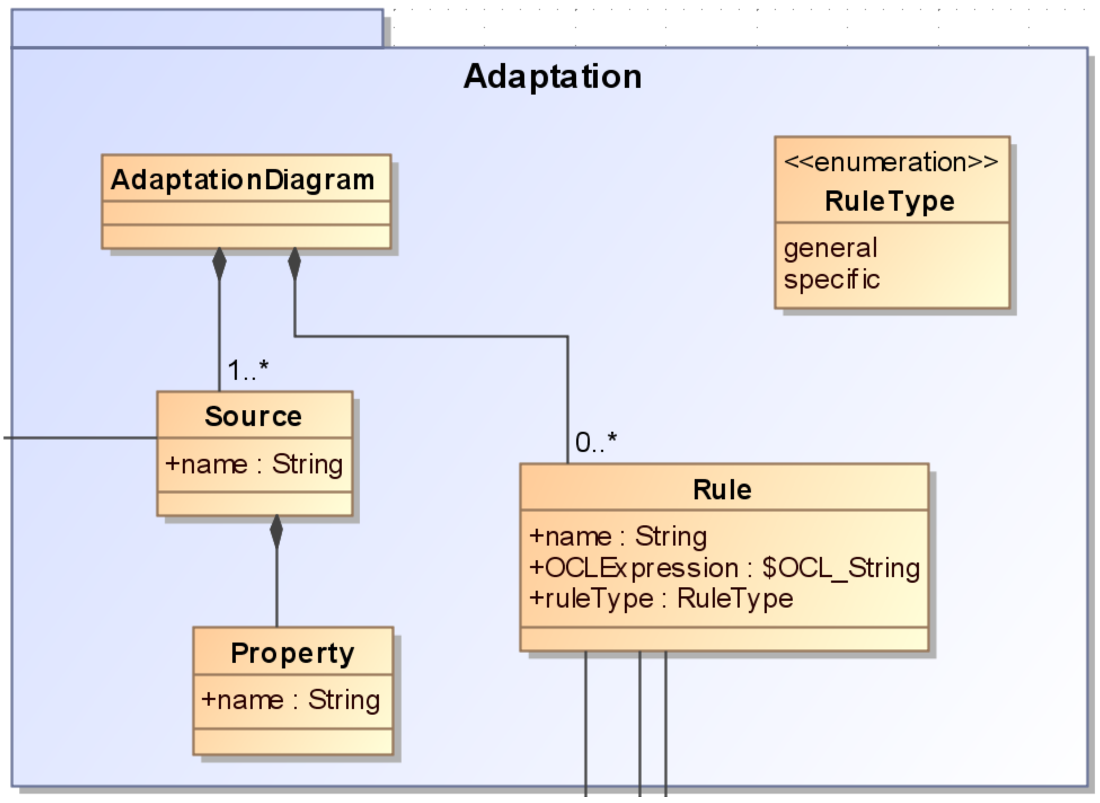


Figure 3.28: Adaptation Metamodel

For each rule we can specify a series of properties (name, OCLExpression and rule type). The rules can be associated to one or more roleAttributes of the role diagram, as well as one or more compositeULElement of the content diagram.

3.2.6 Stage 6: Detail Navigational, Logic, Adaptation and Presentation Services

Behavioural actions for each service specified at the navigational, logic, adaptation, and presentation diagrams can be modelled through the MoWebA Services diagrams. The Service diagrams use UML Activity diagrams enriched with OCL and Action Semantics. For each service/action defined in the other diagrams, it is possible to create a Service diagram that encapsulates the associated service behaviour. Services are defined in the logic layer diagram and could be invoked

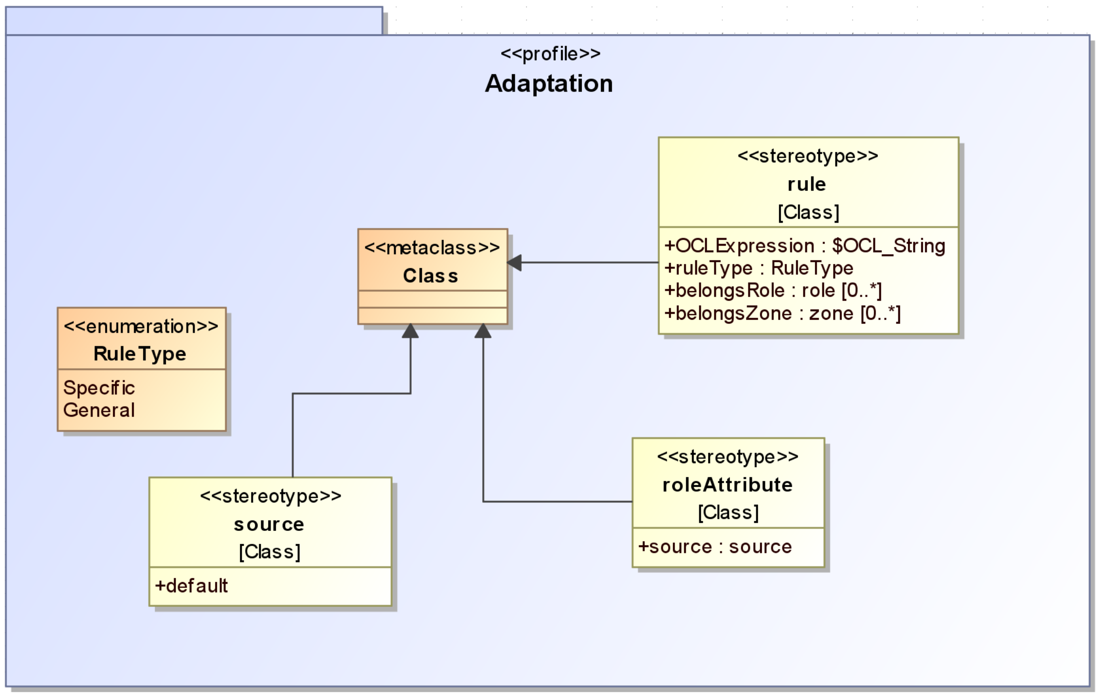


Figure 3.29: Adaptation UML Profile

by entities, rules, node or content diagrams elements.

To specify behavioural actions we use a set of basic and fundamental constructors. The basic constructors represent actions, transitions and pseudo-states. Fundamentals constructors consist of action specializations classified into: CallBehaviorAction, representing a type of action that can invoke other behaviour; DomainAccessAction, representing access to the Entity model to perform an operation on it; and VariableAction, representing a special type of action whose implementation performs various operations on variables. Figure 3.30 shows the Service diagram for the selectContentLanguage action, invoked by the rule "languageRule" (see figure 3.27).

Services allow the definition of behaviour actions at the modelling phase. In some situations a Service diagram can be very complicated, because of the complex logic that it represents. In this case the Service diagram definition could be avoided leaving the task of definition for the ISM phase.

The main idea of the service metamodel (see Figure 3.31) is to define spe-

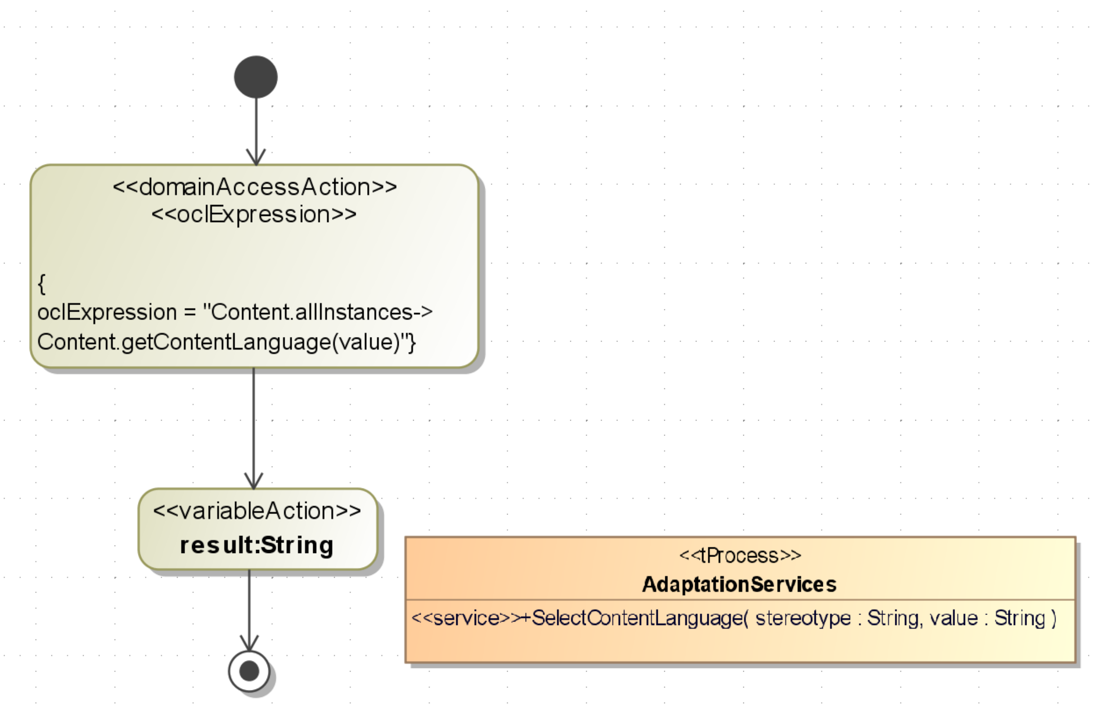

<<domainAccessAction>>
<<oclExpression>>
{
oclExpression = "Content.allInstances->
Content.getContentLanguage(value)"}
<<variableAction>>
result:String
<<tProcess>>
AdaptationServices
<<service>>+SelectContentLanguage(stereotype : String, value : String)

Figure 3.30: Adaptation Service

cializations of Action, which will enable to define more complex behaviours in the metamodel. The metaclass CallBehaviorAction represents a special kind of action that can invoke other behaviours represented by an activity diagram, or a behaviour that will come built into the final platform destination. In the figure, there are listed others specialization of Action (variableAction, domainAccessAction and writePage), and their relationships with other classes. The corresponding UML Profile for the Service metamodel is presented in Figure 3.32.

3.2.7 Stage 7: ASM and PSM definition

Stage 7 is composed of two different models, which can be generated in a semi-automatic way from the diagrams defined during the previous stages: the Architectural Specific Model (ASM) and the Platform Specific Model (PSM). ASM enriches the previous models with additional information related to the system architecture (e.g. RIAs, REST, among others). PSM is oriented to refine the models by adding information related to the platform and language selected for

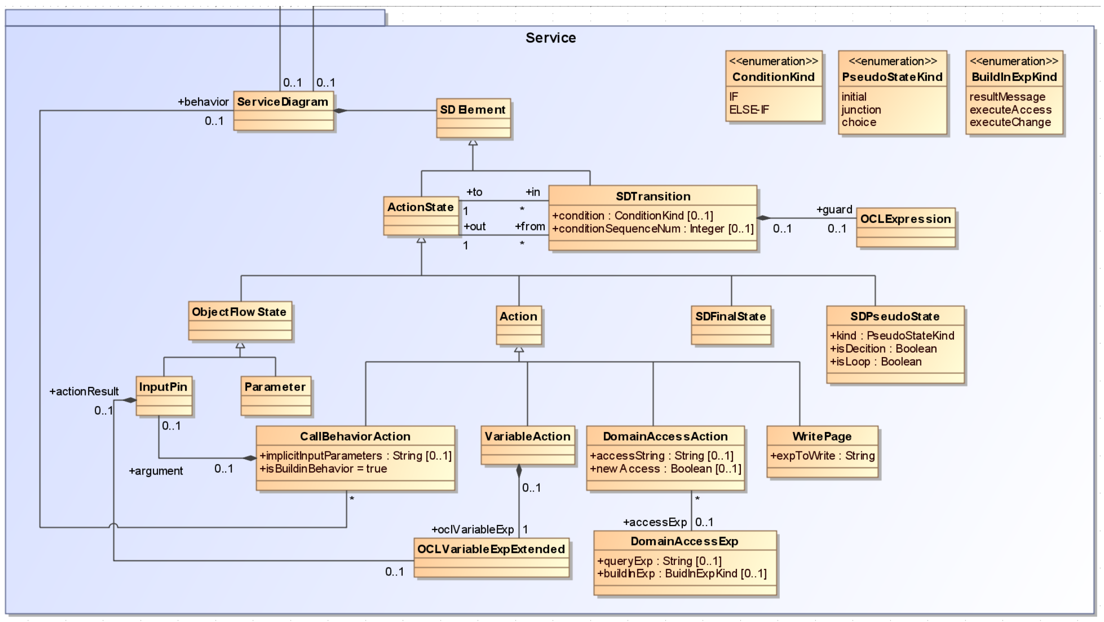


Figure 3.31: Service Diagram Metamodel

the final system (e.g. Java, .NET, PostgreSQL, among others). At this stage, we are moving from the conceptual definition (CIM/PIM models) to the solution definition (ASM/PSM models).

It is important to mention that other approaches generally include architectural aspects at the conceptual modelling level, without making a clear distinction between the independent model and the architectural one. For example, in order to generate Rich Internet Applications - RIAs, current approaches extend their notations with additional primitives or patterns considered at the conceptual modelling phase (e.g. WebML RIA [47], UWE for RIA [66]). In MoWebA, the PIM could be used for different architectures (e.g. RIAs, REST, client-server, SOAs) since architectural aspects are not contemplated in this model. Therefore, MoWebA makes a clear separation between the conceptual space and architectural aspects, defining them on different modelling abstraction levels. In this way, our approach offers enough flexibility to evolve into different architectures starting from the same PIM model.

It is necessary to define the ASM model before using it. This definition encompasses the specification of the corresponding metamodel, among other steps.

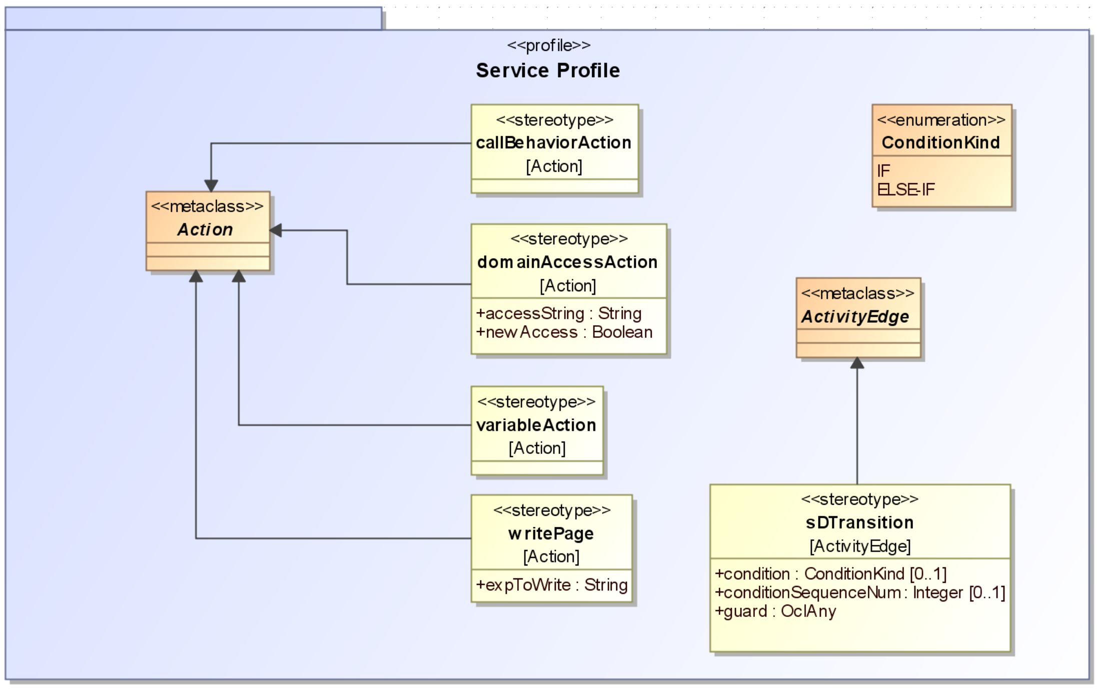

<<profile>>
Service Profile
<<stereotype>> callBehaviorAction [Action]
<<enumeration>> ConditionKind
IF
ELSE-IF
<<metaclass>> Action
<<stereotype>> domainAccessAction [Action]
+accessString : String
+newAccess : Boolean
<<metaclass>> ActivityEdge
<<stereotype>> variableAction [Action]
<<stereotype>> writePage [Action]
+expToWrite : String
<<stereotype>> sDTransition [ActivityEdge]
+condition : ConditionKind [0..1]
+conditionSequenceNum : Integer [0..1]
+guard : OclAny

Figure 3.32: Service Diagram UML Profile

Brambilla et al. have recommended a process for defining an abstract syntax [25] and MoWebA follows it 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 [56].

The steps of this process are synthesized below:

1. Define the ASM metamodel using MOF.

2. Define the corresponding UML Profile.

3. Specify the mapping rules from PIM elements to ASM elements.

4. Define transformation rules from PIM to ASM using standard transformation languages (e.g., ATL or QVT).

5. Define transformation rules from ASM to PSM and PSM to code, or from

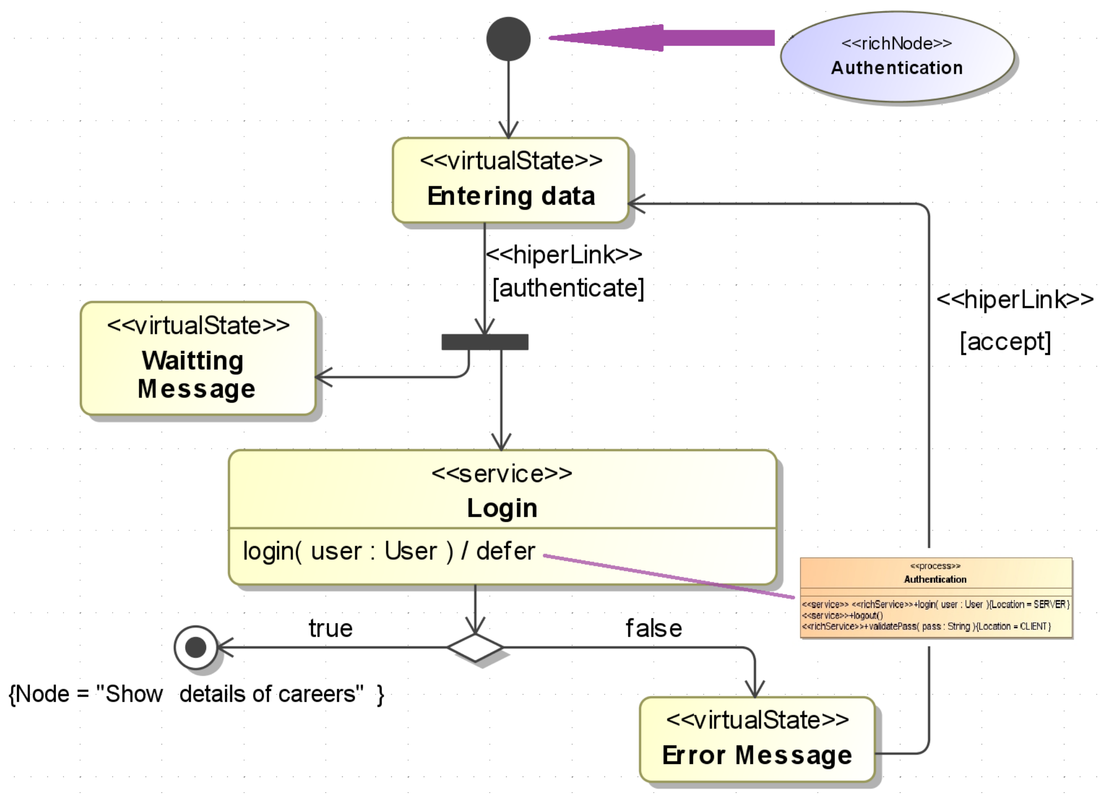


Figure 3.33: Navigational Node applying the ASMRia model

ASM to code, using M2M and M2T (e.g., Acceleo) transformation languages, respectively.

As it can be seen in the steps of the previous process, the MoWebA approach requires some additional effort as a counterpart of improving the portability of the PIM and facilitating the architectural evolution of applications. This includes 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. In any case, it should be noticed that all previous steps will only be performed once, when targeting to a new architecture for the first time.

Once the ASM for a specific architecture is defined, it can be used to develop an application for the selected architecture following these steps:

1. Define the MoWebA CIM/PIM diagrams following the modeling process

(see previous stages).

2. Apply transformation rules in order to obtain the first version of the ASM model.

3. Make manual adjustments (if necessary) to complete the ASM model.

4. Generate the PSM models and/or the final code, applying transformation rules.

5. Include manual adjustments, if necessary.

As an example, we show a simplified ASM model for the RIA architecture, called ASMRia. RIAs are web applications, which use data that can be processed both by the server and the client. The data exchange takes place in an asynchronous way, so that the client stays responsive while continuously recalculating or updating parts of the user interface. RIAs main characteristics are: data and page computation distribution, asynchronous communication between client and server, and enhanced user interface behaviour [66] [24]. In order to model these characteristics in an ASM Model, MoWebA defines a series of stereotypes and tagged values. As an example of an ASMRia model for the academic system, Figure 3.33 shows 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 modelled after the "Entering data" virtual state, since user validation is processed on the server. An example of a client 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.

Figure 3.34 shows the ASMRia metamodel for this example and Figure 3.35 its corresponding UML Profile. In this metamodel, we show the extensions made on different elements related to distribution (client/server) and duration of persistent data and services. Later on we will present a more complete version of the ASMRia metamodel considering presentation patterns, synchronization, among other and the process applied for the ASM definition.

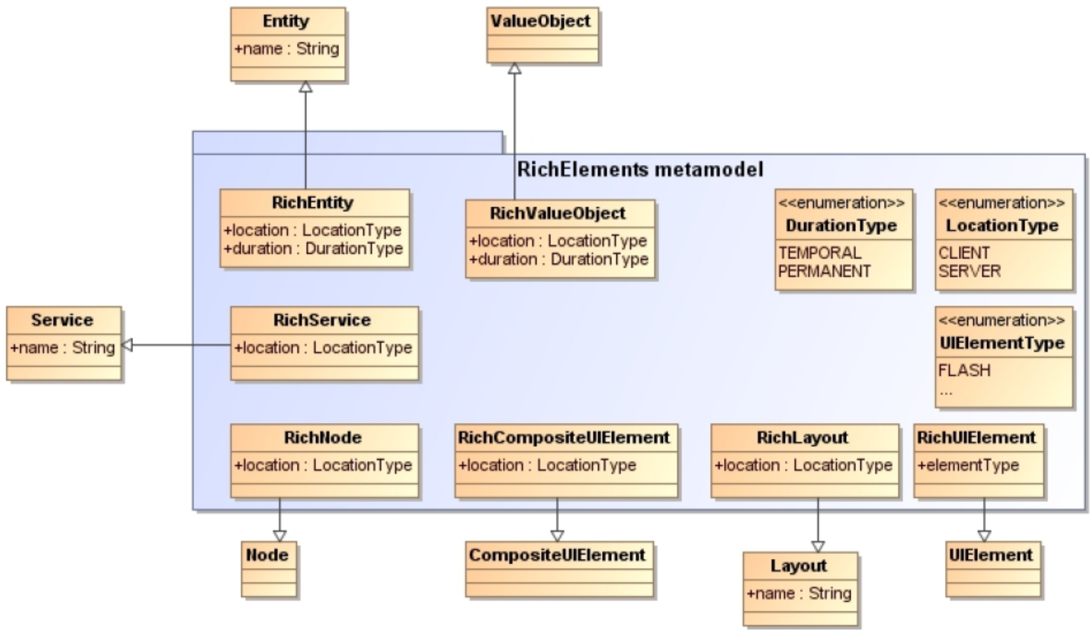


Figure 3.34: Simplified example of ASMRia metamodel

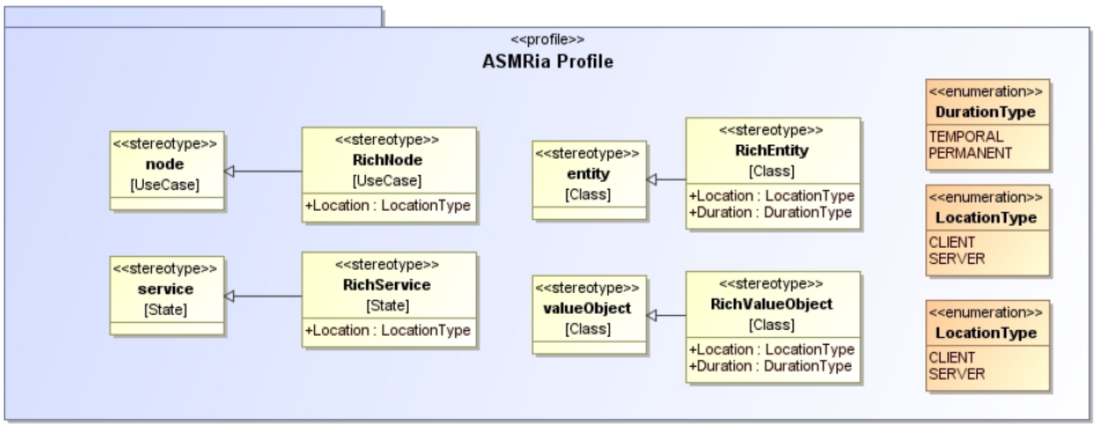

<<profile>> ASMRia Profile
<<stereotype>> node [UseCase]
<<stereotype>> RichNode [UseCase]
+Location : LocationType
<<stereotype>> entity [Class]
<<stereotype>> RichEntity [Class]
+Location : LocationType
+Duration : DurationType
<<enumeration>> DurationType
TEMPORAL
PERMANENT
<<enumeration>> LocationType
CLIENT
SERVER
<<stereotype>> service [State]
<<stereotype>> RichService [State]
+Location : LocationType
<<stereotype>> valueObject [Class]
<<stereotype>> RichValueObject [Class]
+Location : LocationType
+Duration : DurationType
<<enumeration>> LocationType
CLIENT
SERVER

Figure 3.35: Simplified example of ASMRia metamodel

The PSM model enriches the models with specific platform information as the MDA approach suggests. In this sense, we can have one or more PSM models depending on the target platform selected for the application.

The ASM and the PSM can be defined and included into the model as plug-in

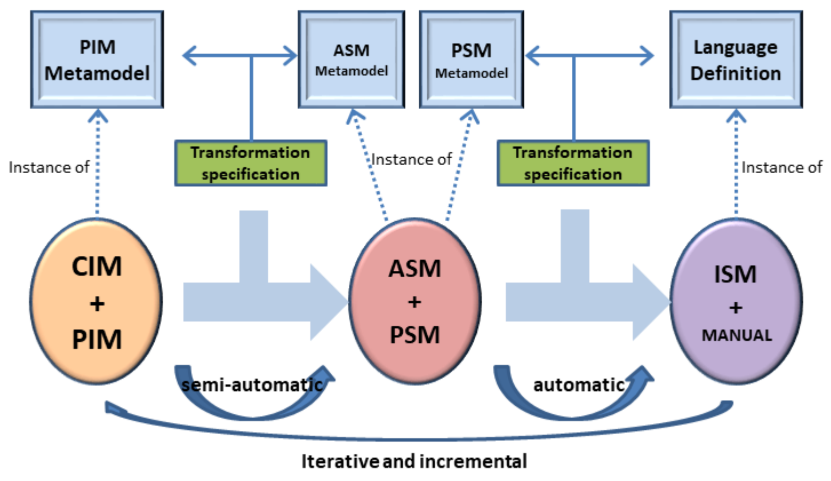


Figure 3.36: MoWebA Transformation Process

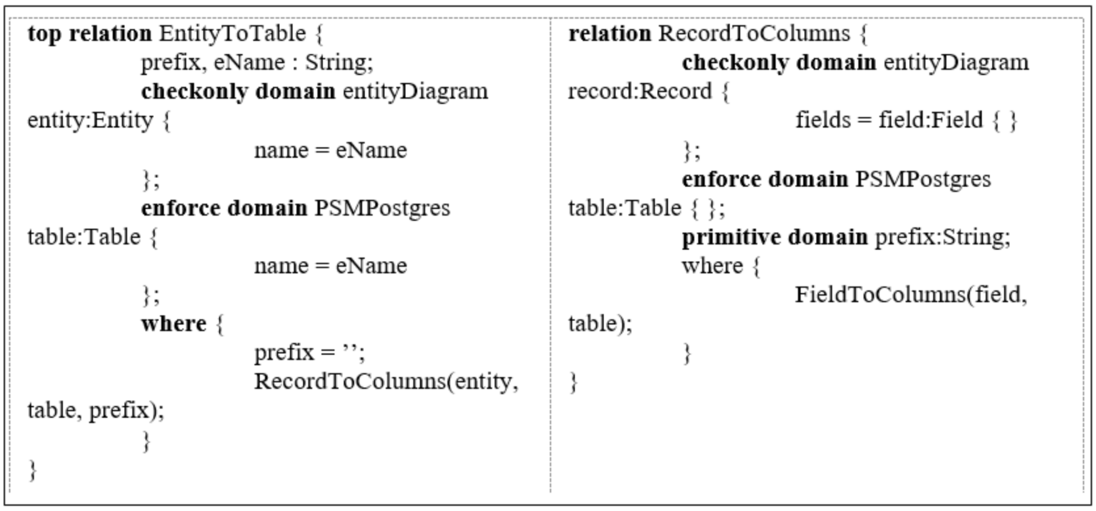
extensions. Indeed, to consider emerging Web Technologies, MoWebA proposes to define a new ASM and/or PSM metamodel.

In chapter 4 we will present the whole process with different experiences we made in order to validate the proposal.

3.3 MoWebA Transformation Process

The transformation process implies steps and activities for transformation specification in order to go through each MoWebA phase (CIM/PIM-ASM/PSM, ASM/PSM-ISM/Manual). This process aims to define intermediate specific models before the final implementation (see Figure 3.36).

The transformation process is based on metamodels (PIM-ASM-PSM trans-



```
top relation EntityToTable {
            prefix, eName : String;
            checkonly domain entityDiagram
entity:Entity {
                        name = eName
            };
            enforce domain PSMPostgres
table:Table {
                        name = eName
            };
            where {
                        prefix = '';
                        RecordToColumns(entity,
table, prefix);
            }
}
```

```
relation RecordToColumns {
            checkonly domain entityDiagram
record:Record {
                        fields = field:Field { }
            };
            enforce domain PSMPostgres
table:Table { };
            primitive domain prefix:String;
            where {
                        FieldToColumns(field,
table);
            }
}
```

Figure 3.37: QVT definition to obtain the PSMPostgres diagram

formation). The PIM-ASM/PSM phase is done in a semi-automatic way; since sometimes the information to be added requires human intervention (e.g. in RIAs, the modeller needs to specify where services will be executed, on the client or on the server). The automation of this process is done using MDD standards such as QVT or ATL, along with a tool that supports these standards. An example of the QVT transformation rule is shown in Figure 3.37. In this figure, the QVT transformation rule is defined by using the Relation language, in order to transform the MoWebA Entity Diagram (which corresponds to the input model) in a PSMPostgres (which corresponds to the output model) diagram. Input and output diagrams vary according to each specific QVT transformation rule.

The ASM/PSM-ISM phase is done automatically by using open source tools (e.g. Acceleo, AndroMDA). The input models of this phase are the PSMs obtained at the previous phase, and the output will be see source code.

We refer to the final implementation of the System as ISM. The ISM will contain code for every platform selected and the bridges between them, in order to get a functional system ready to be deployed. We have experienced two types of ISM obtained by defining transformation rules with two different tools: AndroMDA and Acceleo.

In order to implement the MoWebA transformation rules, we defined a series

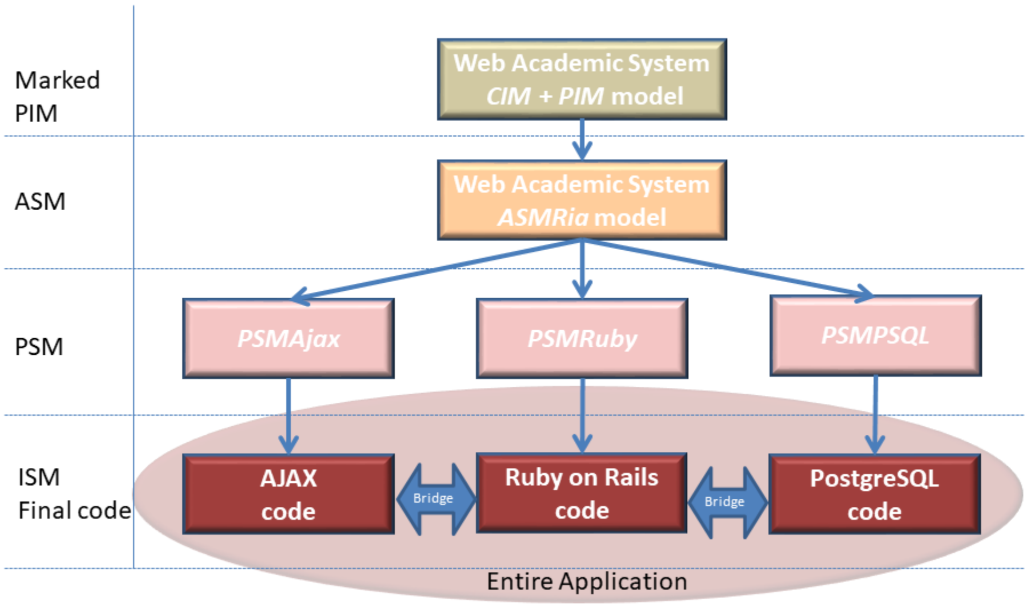


Figure 3.38: The Web-based Transformation Process

of modules (shown in Figure 3.39). For reasons of space, we will only explain in detail the Source and Rule models, defined for the Adaptation code generation phase.

The transformation process for our Web Academic System example is shown in Figure 3.38.

The Academic System was generated using the Acceleo Tool. Acceleo is considered a template-based M2T (model to text) transformation open source MDD tool, which adopts the MTL (Model to Text Language) standard for transformation rules definition[7]. This tool was created in 2006 as a part of the Eclipse Modelling Project (EMP)[8]. The Acceleo code generation process considers the following steps:

[7]http://www.omg.org/spec/MOFM2T/1.0/PDF
[8]http://www.eclipse.org/modeling/

1. Code generator project creation
2. Input models inclusion (XMI files)
3. Modules definition and templates creation
4. Associated services creation
5. Code Generation
6. Project depuration
7. Generators modules exportation

Modules are considered as partial or full implementations of transformation rules for a specific platform. They can be executed as plug-ins of Eclipse to generate an application in the target platform. Modules are composed of templates, services and queries written in the Java programming language. Templates use a specific syntax composed of tags. Queries are used to extract information from the model, which can return values or collections. Java services are used to define complex or common operations that can be accessed by the different templates defined within the module.

The Adaptation transformation rules are composed of the Source and Rule modules. The Source module contains templates defined for information source generation and the Rule module corresponds to the adaptation rules processing.

The Source module is composed of the following templates:

- generateTableSource: creates the database tables with the parameters defined in the information source model.
- loadSources: generates a file with SQL sentences to insert possible values defined in enumerations.
- generateTableSourceType: generates Ruby files for modules in order to manipulate the database tables.
- generateSourcesForRoleAttribute: associates a user with a specific role, and information sources with default values defined in the model.

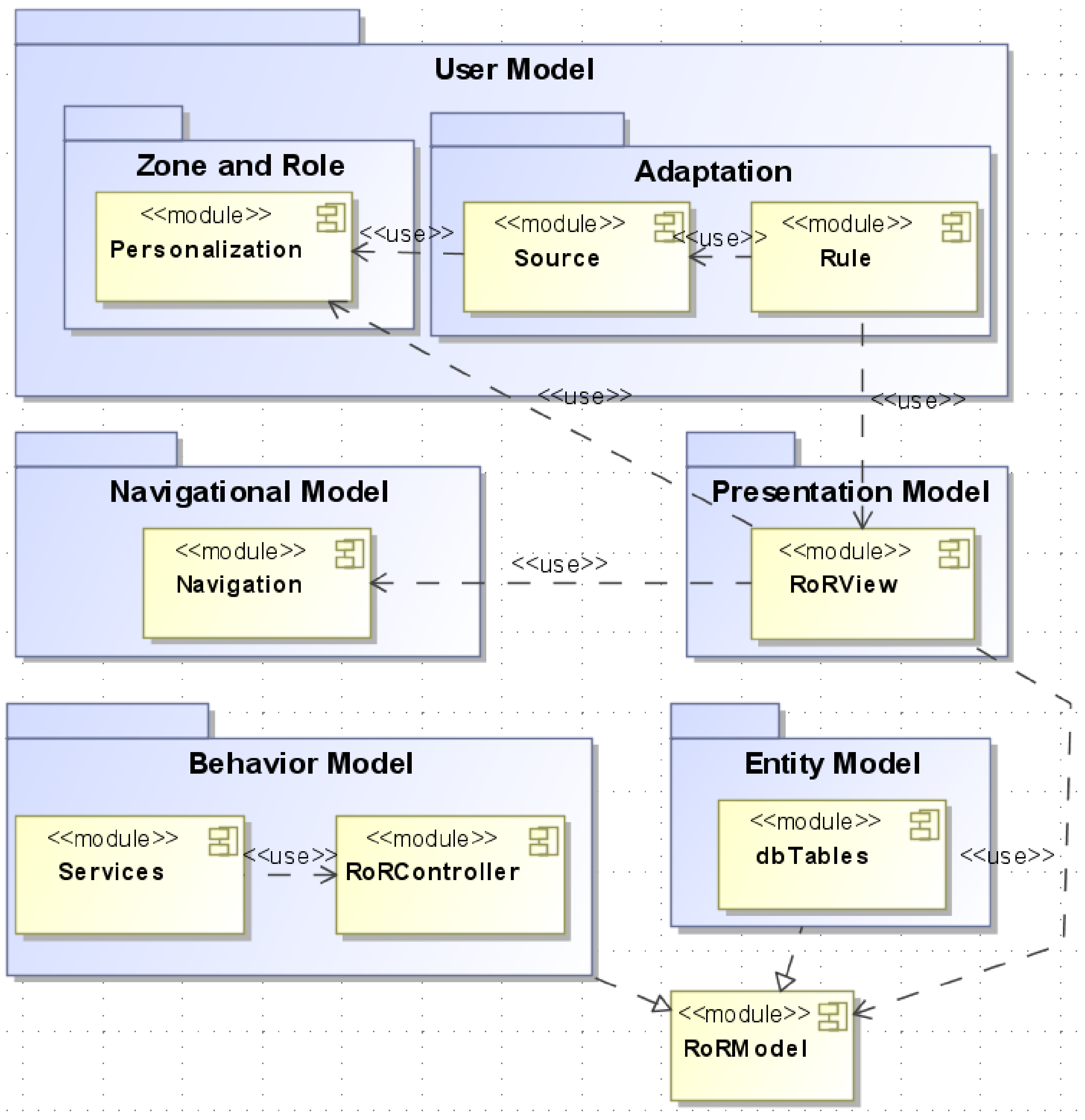


Figure 3.39: Acceleo modules definition for MoWebA

Listing 3.1 shows the generateTableSourceType template.

Listing 3.1: generateTableSourceType.mtl template in Acceleo.

```
[module generateTableSourceType('http://www.eclipse.org/uml2/5.0.0/UML')]

[template public generateTableSourceType(c: Class)]

```

```
[comment @main/]
[if(c.hasStereotype('source')) ]
[file ('create_'+c.name.toLower()+'.rb',false,'UTF-8')]

class Create[c.name/] <ActiveRecord::Migration
    def self.up
        create_table :[c.name.toLower()/] do |t|
        [for( a : Property | c.attribute)]
            t.integer :[a.type.name.toLower()/]_id
        [/for]
        end
    end

    def self.down
        drop_table :[c.name.toLower()/]
    end
end
[/file]
[/if]
[/template]
```

The Rule model, on the other side is composed of:

- generalRuleTransformation: is applied to the rule classes stereotyped with «rule» and isGeneral=True. This template is composed of auxiliary templates: getOclExpression, to retrieve the OCL expression; getSource, to identify the source referencing; and sourceType, to identify the source type.
- applyGeneralRule.mtl: is defined to apply the general rule to the presentation elements.
- specificRuleTransformation.mtl: analyses the specific rules, retrieving the OCL expressions, sources and actions.
- applySpecificRule.mtl: applies the specific rule to the presentation elements associated to it.

The Listing 3.2 shows the generalRuleTransoformation.mtl template.

Listing 3.2: generalRuleTransformation.mtl template in Acceleo.

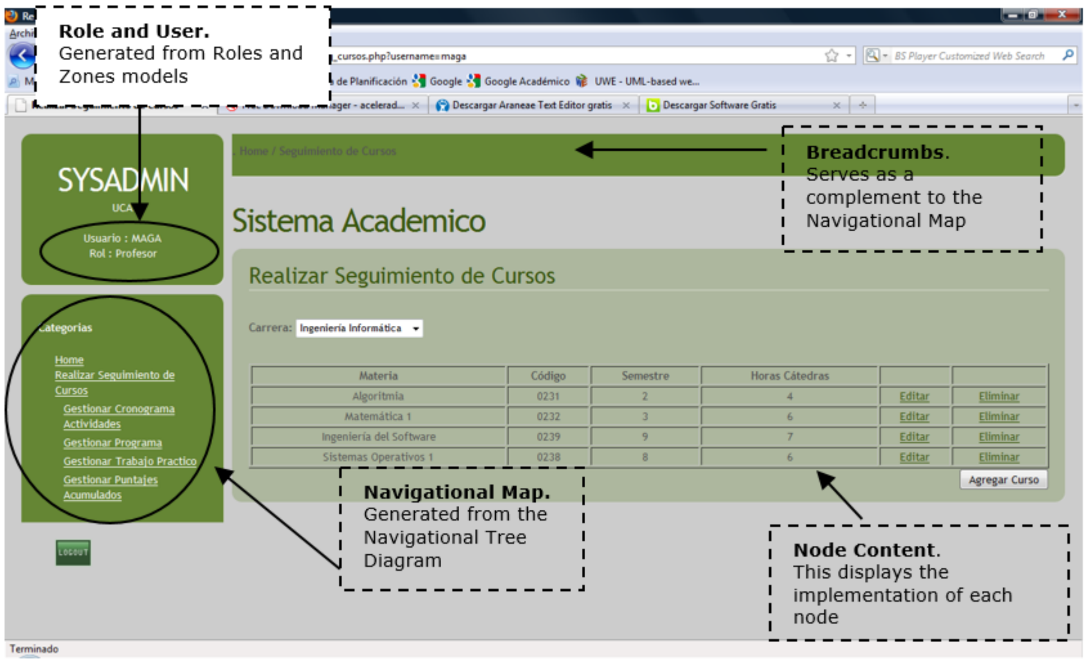
```
[comment encoding = UTF-8 /]
[module
    generalRuleTransformation('http://www.eclipse.org/uml2/5.0.0/UML')]
```

```

[template public generalRuleTransformation(c: Class) post (trim())]

[comment @main/]
[if(c.hasStereotype('rule')) ]
[for (a : Stereotype | c.getAppliedStereotypes())]
    [if (c.isGeneral(c.getValue(a, 'ruleType').ToString()))]
        [file (c.name, false, 'UTF-8')]
def self.get_[c.getSource(a)/]_[c.sourceType(a)/]([c.getSource(a)/]_id)
    if [c.getSource(a)/]_id
        @[c.sourceType(a)/] = [c.sourceType(a).toString().toUpperFirst()/].
        find_by_id([c.getSource(a).toString().toUpperFirst()/].
        find_by_id([c.getSource(a)/]_id).[c.sourceType(a)/]_id)
    end
    return @[c.sourceType(a)/]
end
        [/file]
    [/if]
[/for]
[/if]
[/template]
```

Figure 3.40 shows an example of a page of the Web Academic System resulting from the transformation process. In this figure we can visualize some parts generated from the MoWebA models (e.g. from the navigational tree, node content, and roles and zones diagrams).


cursos.php?username=maga
BS Player Customized Web Search
de Planificación
Google
Google Académico
UWE - UML-based we...
Descargar Araneae Text Editor gratis
Descargar Software Gratis
SYSADMIN
UCA
Usuario : MAGA
Rol : Profesor
Categorias
Home
Realizar Seguimiento de Cursos
Gestionar Cronograma Actividades
Gestionar Programa
Gestionar Trabajo Practico
Gestionar Puntajes Acumulados
LOGOUT
Home / Seguimiento de Cursos
Carrera: Ingeniería Informática
Agregar Curso
Terminado

Role and User. Generated from Roles and Zones models

Breadcrumbs. Serves as a complement to the Navigational Map

Sistema Academico

Realizar Seguimiento de Cursos

Materia	Código	Semestre	Horas Cátedras		
Algoritmia	0231	2	4	Editar	Eliminar
Matemática 1	0232	3	6	Editar	Eliminar
Ingeniería del Software	0239	9	7	Editar	Eliminar
Sistemas Operativos 1	0238	8	6	Editar	Eliminar

Navigational Map. Generated from the Navigational Tree Diagram

Node Content. This displays the implementation of each node

Figure 3.40: An example of a generated page

3.4 Summary of the chapter

In this chapter we describe the model-driven method proposed called MoWebA. We begin the chapter by highlighting some of the points already mentioned in Chapter 1, such as the problem statement and concerns for Web applications. Then, we present an overview of the method including the dimensions and the diagrams that we propose. Subsequently, we present each step in the modeling process, including the diagrams and notation, its definition (metamodels) and examples of use. Afterwards, we present the transformation process adopted by the MoWebA, which includes model-to-model and model-to-code transformations. The section ends with an example of use.

4 Validation Experiences of MoWebA

This chapter presents a series of validation experiences of the MoWebA proposal. As a first step, we were interested in verifying the use of MoWebA within the environment in which it was conceived, the Web applications. In this sense, section 4.1 synthesizes a series of experiences carried out in the academic and real contexts that allowed us to validate and improve the proposal for Web environments.

Subsequently, we have considered to validate the contributions of this proposal that allows the methodology to be extended to other environments or architectures. Such extensions are possible in MoWebA through its architectural specific model definition phase (ASM). In this sense, section 4.2 resumes a preliminary validation experience of the architectural specific model (ASM) definition performed with computer engineering students. Finally, section 4.3 presents three full experiences of the MoWebA ASM extension for RIA and Mobile applications, carried out in the context of a MDD research project held in Paraguay[1].

4.1 Adopting MoWebA: some experiences

MoWebA has been used for modelling and generating different types of applications by novice and experienced modellers and developers. The experienced modellers were already familiar the UML notation and Web methodologies (e.g. UWE, WebML, OOWS, or OOHDM), while developers were familiar with different programming languages.

[1]Mejorando el proceso de desarrollo de software: propuesta basada en MDD, web site http://www.dei.uc.edu.py/proyectos/mddplus/

These experiences, which are summarized in Table 4.1, are proofs of concepts in academic and real settings. They have offered insights for improving specific aspects of the processes and of different models of MoWebA. In addition, they have paved the way for more rigorous validation experiences to be presented in next sub-sections. The experiences summarized next relied on two types of validation instruments (i.e. interviews and questionnaires) in order to identify strengths and weaknesses. Results of these experiences were presented at International Journal of Web Engineering and Technology [57].

Table 4.1: Aspects of MoWebA adoption in the different experiences

Application	**Type**	**Team**[a]	**Profiles**	**Project Context**	**Analysis**
on-line course	e-learning	2 EM, 1 ED	professionals, advance students	academic	interview
University Administration	admin	2 NM	students	academic	interview
Aquatic Birds Portal	management	4 EM	professionals	real project	interview
Academic System	e-learning	2 EM, 2 ED	professionals	academic	interview
laboratory management	management	3 EM, 3 D, 12 NM	2 advance students, student	academic	Questionnaire
Budget execution	admin	4 NM, 4 MD	students, advance students	academic	interview
Surveys	interactive	3 NM	students	real project	interview
Social Network	community	12 MM, 12 MD	advance students	academic	interview

[a] Team: Level E (Experienced), N (New), M (Medium); Type M (Modeler), D (Developer)

Table 4.2: Summary of first experiences with MoWebA

	Modelling aspects analyzed					**Dev. aspects considered**		
Application	**# UC**	**# Nodes**	**# Classes**	**# Pres. Pages**	**# Services**	**Dev. time (months)**	**Target Platform**	**Tool adopted**
On-line course	32	28	23	59	48			
University Admin.	98	92	72	247	248			
Aquatic Birds Portal	95	109	25	266	83			
Academic System	20	35	22	105	93	6	Ruby on Rails	Acceleo
laboratory management	15	17	13	28	19	4	PHP	AndroMDA
Budget execution	27	19	16	79	26	6	PHP-Zend	Acceleo
Surveys	12	21	14	35	25	6	PHP-Zend	Acceleo
Social Network	17	38	12	40	26	4	Ruby on Rails	Acceleo

For a more objective analysis, Table 4.2 summarizes the diverse characteristics of these applications. Some characteristics are related to the complexity of applications and modelling elements, and others to the development process. A summary of the most important considerations arising from these experiences are presented below:

- A first positive aspect is that Navigational structures considered were easy to model, and easy to understand by subjects. For example, the Academic System is composed of 35 navigational nodes, with a mean of 3 virtual states per node, where each virtual state represents a page. Having a global hierarchical view of the system with 35 elements is more manageable than 105 pages.

- The node diagrams were helpful to identify behavioural and presentation elements more easily. We could note that for each navigational node there were identified, in average, 2 to 3 services and 3 to 4 virtual pages. Thus, it is possible to decompose the overall navigational structure into smaller parts, taking into account the specific behavioural navigation for each functional element.

- The CIM/PIM phase was standardized, and could be modelled with any tool that supports UML 2.0 (e.g. Magic Draw and Papyrus). The generated models were exported to the XMI format in order to integrate them with Acceleo and AndroMDA. Even though it was possible to work with different tools, some details had to be considered, especially specially when defining tagged values.

- The automation was performed using two different tools: AndroMDA and Acceleo. On average, the automatic generation percentages for each layer were the following: data layer, 100%; logic layer, 61%; navigational layer, 100%; and presentation layer, 73%. The reason for logic layer not being totally generated is that some services were difficult to model because of their behavioural complexity; therefore they had to be added manually. With respect to presentation, there are some aspects related to style (e.g. fonts, colours, among others) that can only be defined manually.

- MoWebA allows the modelling of diverse types of Web Applications. Even though, special characteristics e.g., such as RIAs or REST, need further specification. For this reason, in order to add RIA characteristics to our Web Academic System example, we had to define the ASMRia model.

- One of the limitations we encountered was that services were sometimes difficult to model, but despite services not being totally defined, the PIM could be defined almost completely. We noticed that for service definition it was necessary to have knowledge in Action Semantics and OCL, but most of the modellers were not as experienced with these, as they are with UML.

However, considering all the services defined in models for the different applications we saw that only 8.6% of the services were complex, while most of them were medium (30.5%) or simple (60.9%) services.

- The transformation rules defined using AndroMDA and Acceleo, made it possible to generate code for three different target platforms: PHP, Python and Ruby on Rails.

The experiences previously presented have been very useful to determine the user-friendliness of the MoWebA proposal and the ability to define transformation rules using different tools and target platforms. We have also determined strengths and weaknesses that have been used to improve the methodological processes and transformation mechanisms.

In the following section we will present a preliminary validation experience, with emphasis on the analysis of the MoWebA proposal for adaptability. That is, we will analyze in particular the use of the ASM model.

4.2 A Preliminary Validation Experience with the Architectural Specific Model

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. [114] have defined for case studies. This work was presented at RCIS Conference in 2016 [56].

4.2.1 Motivation and Goal

As previously presented in the section 1.1, 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 section is focused on conducting a preliminary validation of one of these concerns, related to the adaptability and 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”.

4.2.2 Cases and Unit of Analysis

According to Runeson et al. [114], 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 [150]. In this experience, we used a “multiple-embedded case study” approach (Fig. 4.1). 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.

Groups of three participants have carried out each case. Participants were students of the last year of career, 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 metamodel.
- Stage 3: definition of metamodel and UML profile. Each group defined

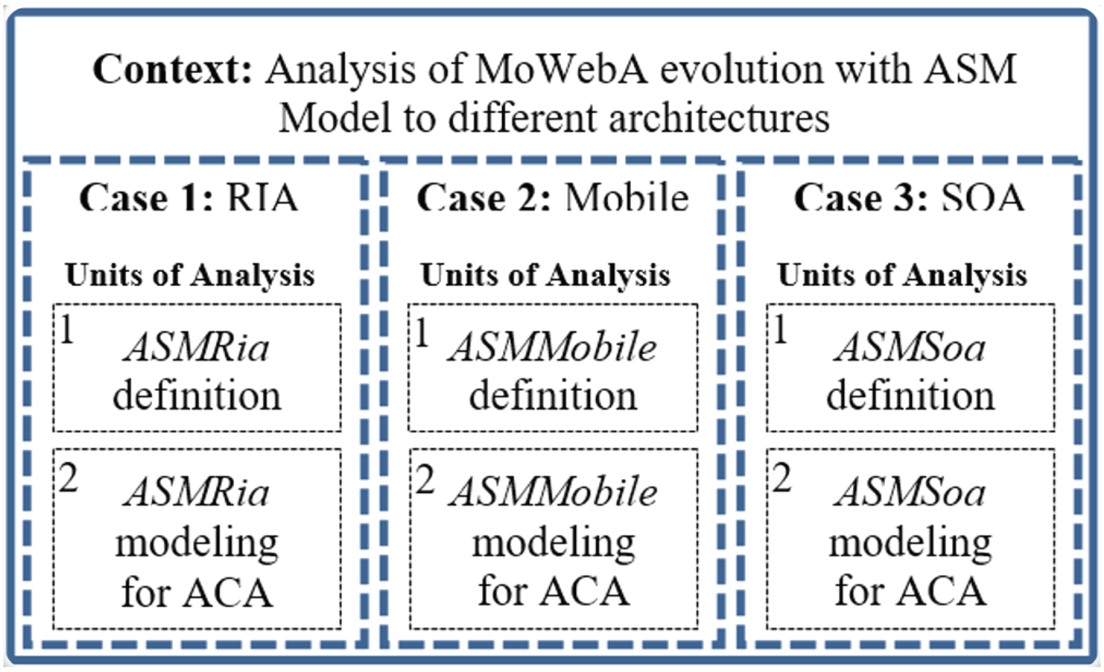

Context: Analysis of MoWebA evolution with ASM Model to different architectures
Case 1: RIA
Units of Analysis
1 ASMRia definition
2 ASMRia modeling for ACA
Case 2: Mobile
Units of Analysis
1 ASMMobile definition
2 ASMMobile modeling for ACA
Case 3: SOA
Units of Analysis
1 ASMSoa definition
2 ASMSoa modeling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

Figure 4.1: Cases and Unit of Analysis

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.

4.2.3 Research questions

Research questions are statements about the knowledge that is being sought, or is expected to be discovered, during the experience [11]. 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?

4.2.4 Data Collection

According to a classification of Lethbridge et al. [68], 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. [114] (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 4.3 presents a summary of data collected during the experience.

Table 4.3: 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 [25]. 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

Table 4.3: Summary of data collected during the experience.

Data Collection method	**Materials**	**Outputs**
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

4.2.5 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 4.4 summarizes the results of data analysis.

From Table 4.4 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 (point 6).
- The ASM metamodel has reflected the specific concepts of architecture (points 4 and 6).

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 (points 5, 7 and 9).
- A considerable good number of concepts of ASM models can be generated in a semi-automated way, from the PIM model (points 5 and 10).

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 (points 11, 12, 13, 14 and 15).

Table 4.4: Summary of data analysis

Data Analysis				
	Criteria	**Ria**	**SOA**	**Mobile**
1	Understanding of architecture	90%	100%	80%
2	Quality of MoWebA PIM models	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%

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

- Each group achieved a reasonable good knowledge of the studied architecture (point 1). 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 (point 2).

4.2.6 Analysis of Results

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 on 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 4.4). 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. [25] 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 experience, we are positive about the usefulness of the ASM in the way prescribed by MoWebA. However, more structured and formal experiences should offer a better insight about the proposal.

In the next sub-section we present more rigorous validation experiences about this issue.

4.3 Extending MoWebA to other Architectures: A Case Study

In this section we present a Case Study to validate the extensions of MoWebA to three different architectures. The experience was structured taking into account a framework that Runeson et al. [114] have defined for case studies. The Case Study was done as part of a research project from the Catholic University of Asuncion called "Mejorando el Proceso de Desarrollo de Software: Una propuesta basada en MDD"[2], grant 14-INV-056 of CONACYT (Consejo Nacional de Ciencias y Tecnología), Paraguay. I have led the technical aspects of the extensions of MoWebA and their validations. The results of the experience permitted an analysis of the adoption of ASM to facilitate adaptability and evolution of MoWebA. The results have also allowed validations to the different extensions and the undergraduate students to obtain their bachelor's degrees, as well as a series of publications in indexed journals and international conferences.

In the next sections we will present the context of the experience, the design of the validation, the data collection, the data analysis and the threat to validity.

[2]`https://www.dei.uc.edu.py/proyectos/mddplus/`

4.3.1 Background

4.3.1.1 Problem Statement

The case study is focused on the ASM phase, both for modelling and transformation processes. The extensions made to MoWebA are: MoWebA4RIA (extension of MoWebA for RIA functionalities), MoWebAMobile4FC (extension of MoWebA for mobile applications for functions in the cloud), and MoWebAMobile4Persistence (Extension of MoWebA in mobile applications for the persistence layer). The extensions have follow the ASM definition process presented in section 3.2.7 which are the following:

1. Define the ASM metamodel using MOF and the corresponding UML Profile.
2. Specify the mapping rules from PIM elements to ASM elements.
3. Define transformation rules from PIM to ASM using standard transformation languages (e.g., ATL or QVT).
4. Define transformation rules from ASM to PSM and PSM to code, or from ASM to code, using M2M and M2T (e.g., Acceleo) transformation languages, respectively.

The design of the validation experience has considered the steps proposed by MoWebA for the ASM phase, and other activities to answer the research questions posed for this experience.

4.3.1.2 Research Objective and Research Questions

Using the template defined in GQM [16] for the definition of goals in experimentation processes, the main purpose of this validation is described as follows:

Analyze the MoWebA method **for the purpose of** determining the grade of adaptability and evolution of the method **with respect to** the architecture **from the point of** view of the researcher **in the context of** a research project with students, professionals, trainee researchers and MDD experts.

Based on this general objective, we propose the following research questions:

- **RQ1:** To what extent the evolution and adaptability of the extension mechanism proposed by MoWebA to incorporate new architectures are achieved?

- **RQ2:** How independent is the MoWebA PIM for use in the modeling stage prior to ASM?
- **RQ3:** To what extent the automation can be obtained with MoWebA's model-to-model and model-to-code transformation rules?
- **RQ4:** To what extent the user's satisfaction is achieved with the use of the MoWebA proposal?

4.3.1.3 Participants

In this study have participated a research team composed of undergraduate, master and PhD students, and MDD experts. The total number of active participants where 11 (five undergraduate students, one master student, two PhD students and three MDD PhD experts). The final year undergraduate students involved in the project had knowledge of MDE and MDD, in addition to previous knowledge and experience with the MoWebA proposal (in a Software Engineering class). The PhD experts that participated where from Polytechnic University of Valencia (Spain), La Plata University (Argentina) and Catholic University of Asuncion (Paraguay). Moreover, the validation experiences have counted with the participation of students of the Computer Science career, and professionals with experience on development of mobile applications.

4.3.2 Design of validation

The study was attended by a number of final year undergraduate students, PhD students and expert researchers in the area of MDD. Four students participated in the definition of the extensions. One of them was in charge of performing the PIM-ASM mappings and developing the M2M transformation rules from PIM to ASM and the others have worked independently in the definition of the metamodels and the code generation rules for each extension. All the members of the project have collaborated during the validations of each of the extensions. Moreover, other students and professionals with experience in mobile development have participated in the validations experiences. I have led the work carried out in the extensions and validations with the advice of the MDD experts who acted as tutors.

The follow-up of the development of the extensions was accompanied by weekly meetings of approximately 2 hours of duration. The weekly meetings were held for one year. The meetings were oriented to follow the experiences done in the period, complemented with open interviews, observations and focus groups.

The extension proposals have been divided into stages, considering the process proposed by MoWebA for ASM extension and other additional activities that were carried out to achieve the objectives. The activities carried out for the development of the extensions are listed below:

1. Definition of the scope to be considered for extensions.
2. Revision and adjustments to the PIM taking into account the architecture established for the extension.
3. ASM metamodeling using MOF.
4. PIM-ASM mapping.
5. PIM-ASM transformation rules definition with ATL.
6. Definition of code generation rules.
7. Proof of Concept development.
8. Validation of the extension.

For each activity, we identified sources of data to be considered during the experience analysis stage. In the following sub-sections we present each of this activities.

4.3.2.1 Scope of the MoWebA Extensions

As mentioned before, for the validation experience we have selected three different environment: RIA's Web applications (called MoWebA4RIA), mobile applications with functions in the cloud (called MoWebAMobile4FC) and the persistence layer of mobile applications (called MoWebAMobile4Persistence). In turn, for each architecture, we considered a series of features that will be explained below.

4.3.2.1.1 MoWebA4RIA

Many Web applications offer the possibility of distributing their data and their business logic between the client and the server, also allowing an asynchronous communication between them. These features, originally associated with the arrival of Rich Internet Applications (RIA), remain particularly relevant and desirable. In the area of RIA, there are some proposals that simultaneously consider these features, adopt Model-Driven Development (MDD), and use implementation technologies based on scripting. For this reason, we have decided to dedicate efforts to define an extension of MoWebA for this environment. A first version of MoWebA4RIA have been presented at CIbSE 2016 [72] considering the widgets and the logic of the client-side application. Subsequently, we have decided to extend other RIA features as well. Specifically, in this extension we incorporate the features of client data, client business logic and asynchronous communication between client and server. The results of this new extension have been published at the CLEI EJ journal [91].

4.3.2.1.2 MoWebAMobileApps4FC

Although MoWebA has been conceived as a methodological proposal for Web environments, we believe that it could be worthwhile to verify extensions to other environments in addition to Web, such as mobile. In this sense, in order to decide which aspect of mobile environment to extend with the ASM phase of the MoWebA approach, we did a systematic mapping study that analyzes different proposals that apply MDD to the development of MobileApps-FC (mobile applications for functions in the cloud) and at the same time, consider the improvement of the portability of such applications [120]. We have focused on the network communication aspect because its implementation necessarily implies working with different platforms and technologies (iOS, Android and the cloud service providers' platforms) of the MobileApps-FC. In this sense, the platform abstraction which proposes a MDD approach is highlighted on the network communication aspect.

Starting from the official documentations of Android[3] and iOS[4] (currently, the

[3]Android, `https://goo.gl/LwMLXn`
[4]iOS, `https://goo.gl/3FlZqW`

most popular platforms)[5] we focused on four types of network communication functions. There are more variants of such functions, but we have focused on such four ones to cover some of the most common cases.

Following, we describe the types of such functions: i) *light-data*, where the data exchange does not include files (e.g., images, documents, video or audio). In this sense, the data to exchange is *light*; ii) *load-image*, to get and to load images in memory for displaying them. Commonly, this function is used for image and video previews; iii) *download-files*, to download files in background; and iv) *upload-files*, to upload files in background.

The mentioned functions include implementations in both sides, the mobile and the cloud, respectively. Such implementation are based on the REST architecture [104].

The results of this extension were presented at CIbSE 2018 Conference [121].

4.3.2.1.3 MoWebAMobile4Persistence

The data layer access design is a critical task for mobile applications which need constant access to remote data, making them available offline in case of network connectivity problems. Moreover, the variety of mobile operating systems and platforms (fragmentation phenomenon) that handles data storage differently affects the portability of mobile applications. Therefore, we decided to extend MoWebA for the development of native mobile applications focusing on the data layer. This extension covers data persistence concepts to achieve offline applications in case of network connectivity problems.

Based on the MAAG's guidelines for designing the data layer [100], in Figure 4.2, we represent which elements from these guidelines were adopted to establish the architecture for our approach.

The data layer provides access to data hosted within system boundaries and to data exposed by other networked systems. Furthermore, this layer, aside from the data persistence handling, defines data providers for the mobile application. Regarding data persistence, the data persistence mechanisms we considered are databases, files, and key-value pairs. These mechanisms are implemented in different ways on each platform. Currently, there are different mobile databases,

[5]Popularity of Android and iOS, `https://goo.gl/ZAu8Ho`

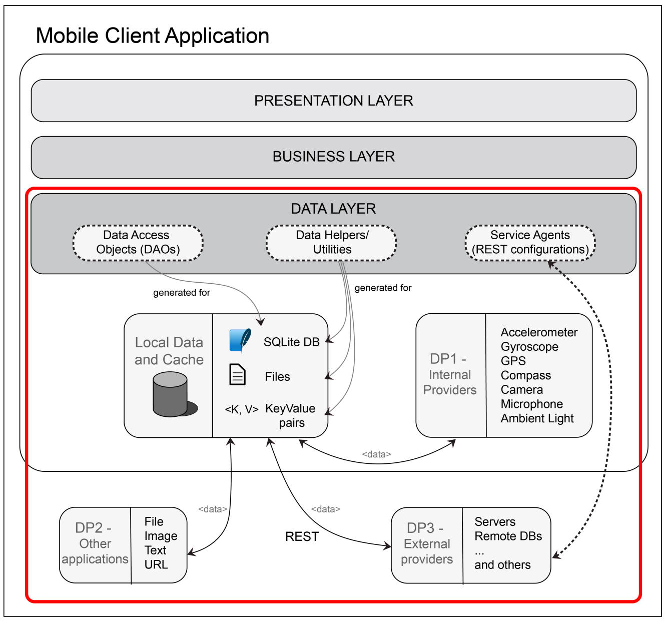


Figure 4.2: MoWebA Mobile architecture scheme based on MAAG's guidelines [100]. The scope of this work is framed in the data layer (indicated by the red rectangle).

but for reasons of this work, we opt for SQLite as a database due to is likely the most widely deployed and used database engine, and used extensively in major mobile platforms [4]. About data providers, and according to XIS-Mobile [103] profile specifications, we could identify that mobile applications can receive data from three types of sources.

1. External data providers, such as servers and remote databases. In this context, we chose REST, as a uniform and portable communication interface, adopted in extension in mobile application network communication approaches [121].

2. Internal data providers, such as sensors like the gyroscope and accelerometer, and device-specific hardware resources, such as camera and microphone.

3. Through interoperability with other applications, to exchange different data types like File, Image, and more.

The results of this work were published at Journal of Systems and Software

[94].

4.3.2.2 ASM Metamodel and UML Profile Definition

In this section, we present the metamodels and UML profile definitions for the three environment.

4.3.2.2.1 Metamodel and Profiles in MoWebA4RIA

The MoWebA4Ria metamodel is shown in Figure 4.3. This metamodel presents the incorporated functionalities to model the features of client data, client business logic and asynchronous communication between client and server. Next, we present the elements defined in the ASM metamodel.

To save data in a web client, specifically in a web browser, there were created two concepts, a *ClientValueObject*, which extends the value object of the original MoWebA logic diagram, and a *ClientStaticObject*, which extends a static object, a new element introduced in the logic diagram to represent sets of statically defined values as properties of the class.

These new concepts differ from the original ones at the implementation level, since both the *ClientValueObject* and the *ClientStaticObject*, are mapped to variables stored in the browser. In addition, these new concepts allow a level of *persistence* to be specified, which can be *permanent* or *temporary*, allowing the variable to persist or not at the end of a session or browser close.

In terms of processes executed on the client, there were created three elements considered in the presentation page, the *RichForm*, the *RichTextInput* and the *RichTable*. The *RichForm* is a specialization of the *Form* element of the original MoWebA content diagram. It contains an autocomplete attribute based on history called *historyAutocomplete*, which allows the presentation page to suggest values to complete the form's entries, based on previously entered values. The *RichTextInput* extends the *TextInput* of the original MoWebA content diagram. This element introduces the attribute *tagAutocomplete*, which differs from the previous attribute *historyAutocomplete* in that the values to be suggested are obtained from an association with objects of value or static objects and their specializations. The *RichTable* specializes the element *Table* of the original content diagram of MoWebA. This element allows to specify the service of the node

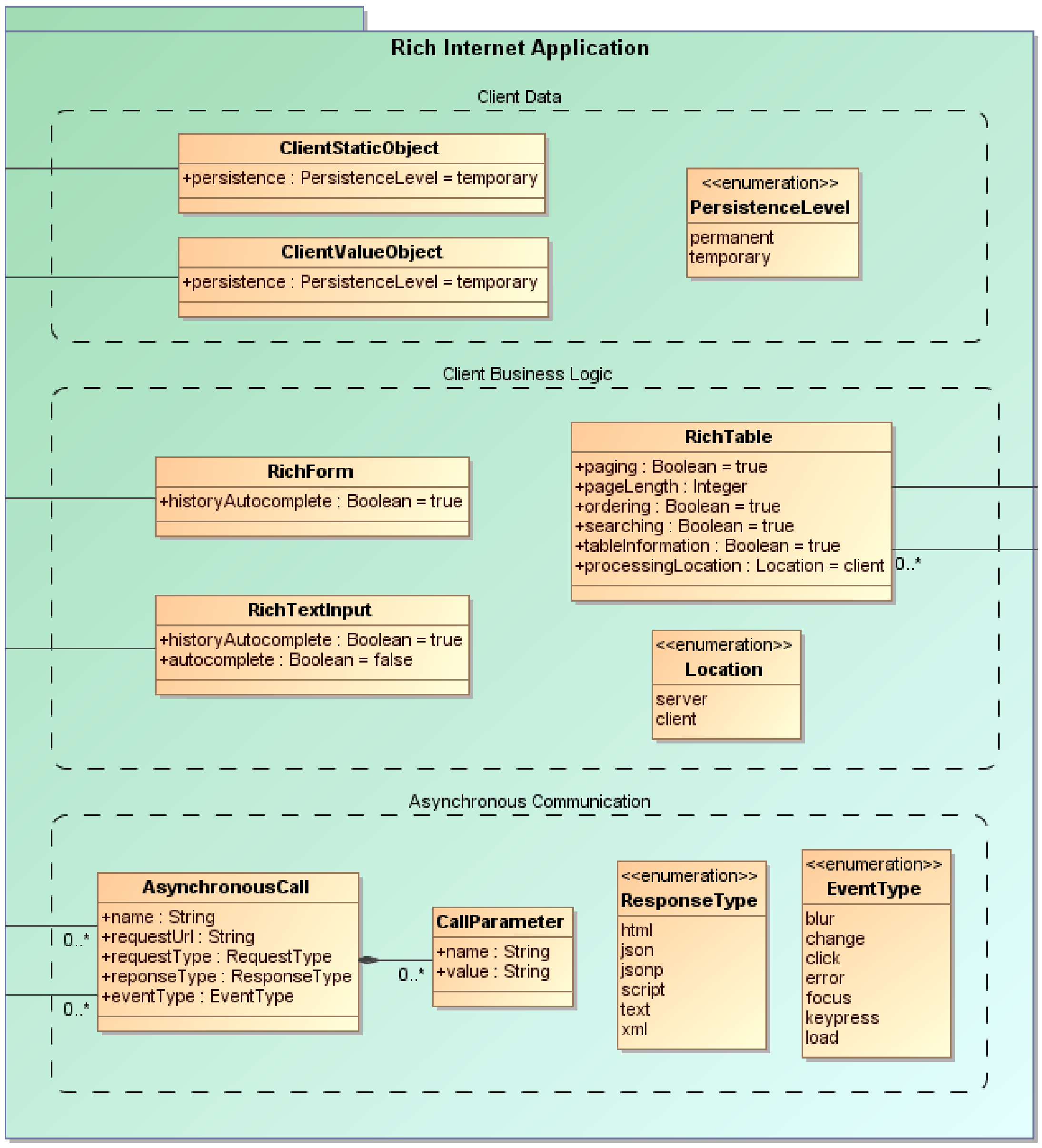


Figure 4.3: Metamodel used for the definition of an ASM for RIA

diagram that will be responsible for populating the table. In addition, it adds new properties to the table to allow paging (*paging*), specify number of records per page (*pageLength*), allow ordering by columns (*ordering*), look up words in records (*searching*), display information (index of current page, number of pages, number of records) of the table (*tableInformation*), and specify where these functions will be processed (*processingLocation*), either on the *server* or the *client*.

To allow asynchronous communications between client and server there were created the element called *AsynchronousCall*. This class is used to make a request for a service.

The *AsynchronousCall* is associated with one or many UI elements in the content diagram and zero or one service of the node diagram. When an event occurs on some UI element, a service is executed. The service can be one specified in the node diagram, or a service running from a URL outside the system (for example, a web service). The instance of the *AsynchronousCall* can be located in the content diagram along with its associated UI elements. The *Asnchronous-Call* element has a name (*name*); a URL (*requestUrl*), used in case an external service is requested; a type of request (*requestType*), which can be used to retrieve remote data (*retrieve*) or insert or update data (*insert/update*); a type of response (*responseType*), which can be *html*, *json*, *jsonp*, *script*, *text* or *xml* formats; and a type of event (*eventType*), that is applied to the UI element and can take the values *blur*, *change*, *click*, *error*, *focus*, *keypress* or *load*. In addition, the *AsynchronousCall* may attach certain parameters to the request. Each parameter consists of a property of the class that has a name and a value.

Figure 4.4 presents the UML profile that corresponds to the described ASM RIA metamodel.

4.3.2.2.2 Metamodel and Profile in MoWebAMobile4FC

The metamodel and profile for the network communication proposal extend the logic diagram of MoWebA (see section 3.2.4). Such logic diagram enables the definition of logic processes. In fact, we consider the network communication as a logic process. The logic diagram contains *TProcesses*, which are the logic processes defined. Such processes include *Services*, which are procedures doing a specific task. Also, the logic diagram has *ValueObjects*, which group attributes of entities and enable the access to the entities' data.

The extensions for obtaining the ASM are based on the REST architecture and on the four types of network communication functions, presented in section 4.3.2.1.2 (light-data, load-image, download-files, upload-files). Such elements belong to the metamodel showed in Figure 4.5.

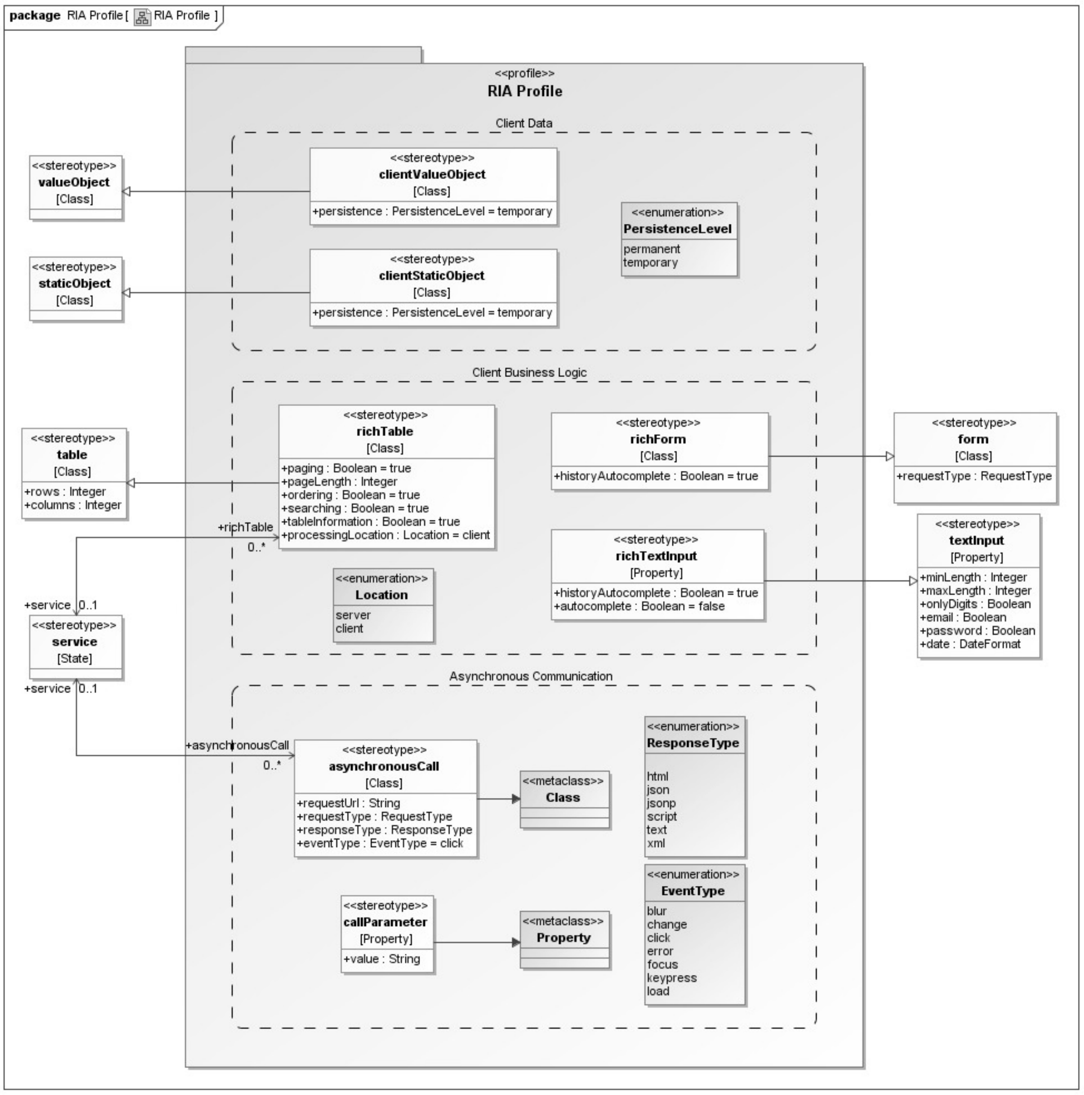

package RIA Profile [RIA Profile]
<<profile>>
RIA Profile
Client Data
<<stereotype>> valueObject [Class]
<<stereotype>> clientValueObject [Class]
+persistence : PersistenceLevel = temporary
<<enumeration>> PersistenceLevel
permanent
temporary
<<stereotype>> staticObject [Class]
<<stereotype>> clientStaticObject [Class]
+persistence : PersistenceLevel = temporary
Client Business Logic
<<stereotype>> table [Class]
+rows : Integer
+columns : Integer
<<stereotype>> richTable [Class]
+paging : Boolean = true
+pageLength : Integer
+ordering : Boolean = true
+searching : Boolean = true
+tableInformation : Boolean = true
+processingLocation : Location = client
+richTable
0..*
<<stereotype>> richForm [Class]
+historyAutocomplete : Boolean = true
<<stereotype>> form [Class]
+requestType : RequestType
<<stereotype>> richTextInput [Property]
+historyAutocomplete : Boolean = true
+autocomplete : Boolean = false
<<stereotype>> textInput [Property]
+minLength : Integer
+maxLength : Integer
+onlyDigits : Boolean
+email : Boolean
+password : Boolean
+date : DateFormat
<<enumeration>> Location
server
client
+service 0..1
<<stereotype>> service [State]
+service 0..1
Asynchronous Communication
+asynchronousCall
0..*
<<stereotype>> asynchronousCall [Class]
+requestUrl : String
+requestType : RequestType
+responseType : ResponseType
+eventType : EventType = click
<<metaclass>> Class
<<enumeration>> ResponseType
html
json
jsonp
script
text
xml
<<stereotype>> callParameter [Property]
+value : String
<<metaclass>> Property
<<enumeration>> EventType
blur
change
click
error
focus
keypress
load

Figure 4.4: ASM RIA Profile for MoWebA

Firstly, it is defined a *CloudServer*, which is accessed through a *Domain*. The *Domain* is provided by a service *Provider*, which enables the recognition of the particular configurations required by each cloud service provider. Then, each *CloudServer* contains a set of logic processes, where each one of them is called *RestProcess*. Such processes modularize the network communication design. Subsequently, each *RestProcess* includes a set of resource interfaces, where each *ResourceInterface* is associated with a *Resource*, one at a time. A *Resource*

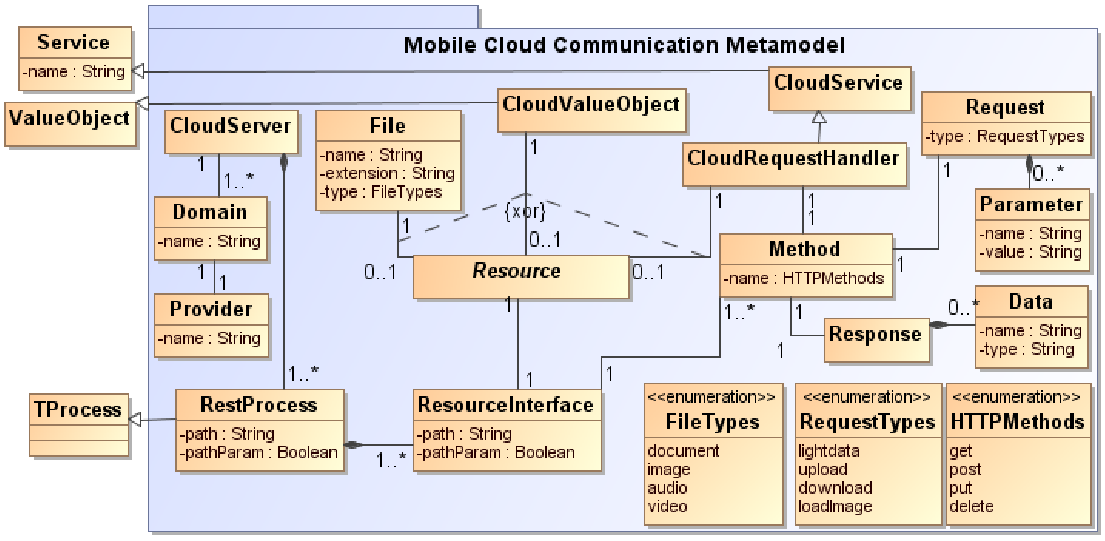


Figure 4.5: Metamodel for the network communication ASM

can be a MoWebA's *ValueObject* called *CloudValueObject* because it is residing in the cloud, a *File* stored or to be store in the cloud or a MoWebA's *Service* called *CloudRequestHandler*, which is executed in the cloud.

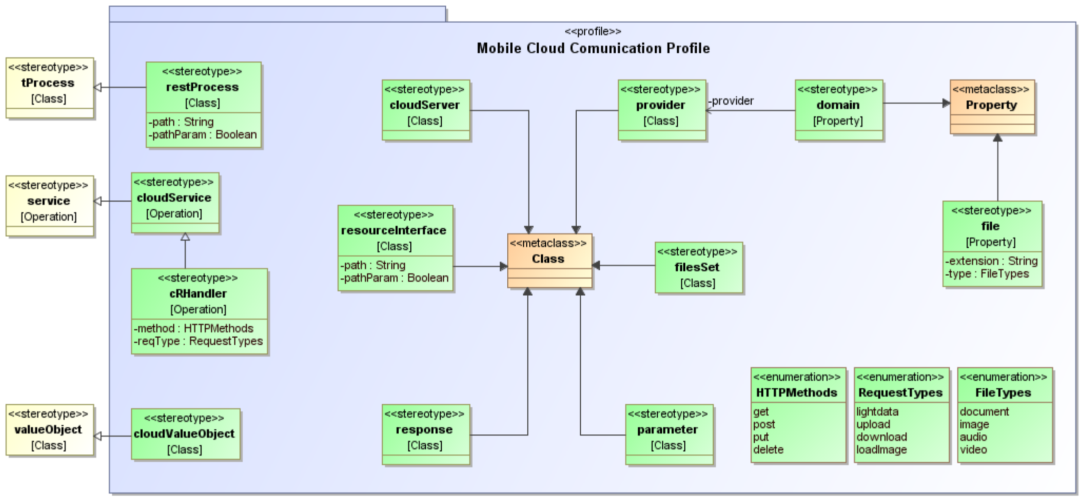
Each *ResourceInterface* is associated to a set of methods, where each *Method* defines the operation to be performed on a *Resource*. At the same time, each *Method* is associated to the object *Request* and, in some cases, to the object *Response*. On one side, each *Request* can be associated to a set of parameters, which contains each *Parameter* of the request. On the other side, each *Response* is associated to a set of data expected to receive from a *light-data Request*, operated by the *Method get*.

Following, we will explain the attributes of representative elements described so far. The *RestProcess* has a relative *Path*, which makes the process accessible, and a boolean flag which establish if the *Path* is used as additional data in each *Request*. For instance, the *Path* could be a user identification. The same case is for the attributes of the *ResourceInterface*. About *File*, its attributes specify the *name*, *extension* and the file *type*. The *Name* of *Method* defines the type of HTTP method (*get, post, put, delete*). The four HTTP methods considered are the most common ones.

The *Request type* (*lightdata, download, upload, loadImage*) defines the call to be done. In case of *lightdata*, the associated *Method* can be any of the mentioned HTTP methods. Nevertheless, *download* (*get*), *upload* (*post*) and *loadImage* (*get*) have, each one, a predefined *Method*. The *name* and the *value* of the *Parameter* describe the different parameters of the *Request*. Finally, the *name* of *Data* specify the data to be received from the *Request lightdata* and the *Method get*. Such *name* specifies an attribute of the *CloudValue Object* associated to the respective resource interface.

The described elements of the ASM are represented concretely in the respective profile of figure 4.6.

The profile contains the definition of the elements (stereotypes, tag values and enumerations), which enable the modeling.



<<profile>>
Mobile Cloud Comunication Profile
<<stereotype>> tProcess [Class]
<<stereotype>> restProcess [Class]
-path : String
-pathParam : Boolean
<<stereotype>> cloudServer [Class]
<<stereotype>> provider [Class]
-provider
<<stereotype>> domain [Property]
<<metaclass>> Property
<<stereotype>> service [Operation]
<<stereotype>> cloudService [Operation]
<<stereotype>> resourceInterface [Class]
-path : String
-pathParam : Boolean
<<metaclass>> Class
<<stereotype>> filesSet [Class]
<<stereotype>> file [Property]
-extension : String
-type : FileTypes
<<stereotype>> cRHandler [Operation]
-method : HTTPMethods
-reqType : RequestTypes
<<enumeration>> HTTPMethods
get
post
put
delete
<<enumeration>> RequestTypes
lightdata
upload
download
loadImage
<<enumeration>> FileTypes
document
image
audio
video
<<stereotype>> valueObject [Class]
<<stereotype>> cloudValueObject [Class]
<<stereotype>> response [Class]
<<stereotype>> parameter [Class]

Figure 4.6: UML Profile for the network communication ASM

4.3.2.2.3 Metamodel and Profile in MoWebAMobile4Persistence

Considering the well-defined layered structure and our main aim, we focus on MoWebA's Data Access level (see section3.2.2). Here, the Entity Diagram allows defining the structure and static relationships between the classes in the problem domain at the PIM level. Thus, this diagram was our clear choice to extend in order to cover more conceptual elements to the structural type, which might facilitate the modeling of mobile applications data persistence. Moreover, for data provider modeling, we extended a UML class.

Then, the definition of the mobile ASM for persistence started with the extension of MoWebA's Entity Diagram at the PIM to take advantage of conceptual elements of the structural type.

MoWebA presents its Entity Diagram to define the structure and static relationships between the classes identified in the problem domain at the PIM level. We extended this diagram to cover more conceptual elements of the structural type, which will later facilitate the modeling of mobile applications. We summarize two types of modifications: an extension of the entity properties (*EntityProperty*), and an addition of specific data types these properties can have (*dataType*). The aggregated properties allow: specify a data type (*DataType*), whose values can be *bool*, *date*, *time*, *datetime*, *int*, *real*, *string* and *text*; select a maximum data size (*size*); restrict the field to being null (*notNull*); indicate a default value (*defaultValue*); and, establish if the property is constituted as the identifier of the entity (*id*).

According to the extensions made, we present the specific mobile architecture model or mobile ASM. From this model, we can define mobile applications with the established functionalities: the local data persistence and the design of data provider components. Starting with Figure 4.7, we observe the metamodel of the proposal. From the figure, we can highlight the following: two sections are distinguished in particular, the persistence of data (within the box with dashed lines: *Persistence Data*) and the data providers (within the box with dotted lines: *Data Provider*). In data providers, we find three distinguishable subsections:

1. other applications as data providers (within the box with dotted lines: *MobileAppData*);
2. external data providers (inside the box with dotted lines: *External Data*); and,
3. internal data providers (within the box with dotted lines: *Internal Data*).

Concerning the UML profile, the mobile profile model of figure 4.8 presents the same functionalities definition as the metamodel, only with the definition of the elements (stereotypes, tag values, and enumerations), which enable the modeling.

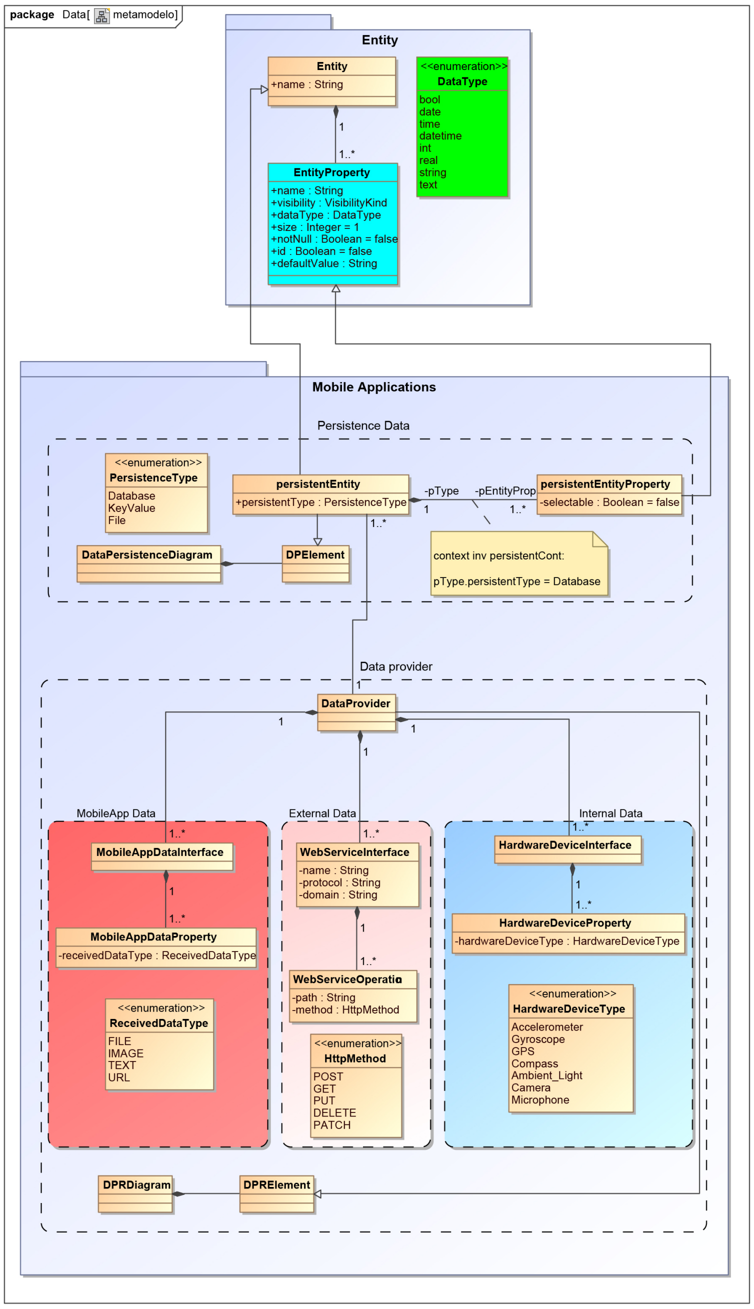


Figure 4.7: Metamodel used for the definition of an ASM for mobile applications.

Regarding data persistence, we introduce elements to identify what data will be stored on the device (*persistentEntity* tag value), and what type of persistence will be used (*persistentType* tag value). The first *persistentType* (*Database*

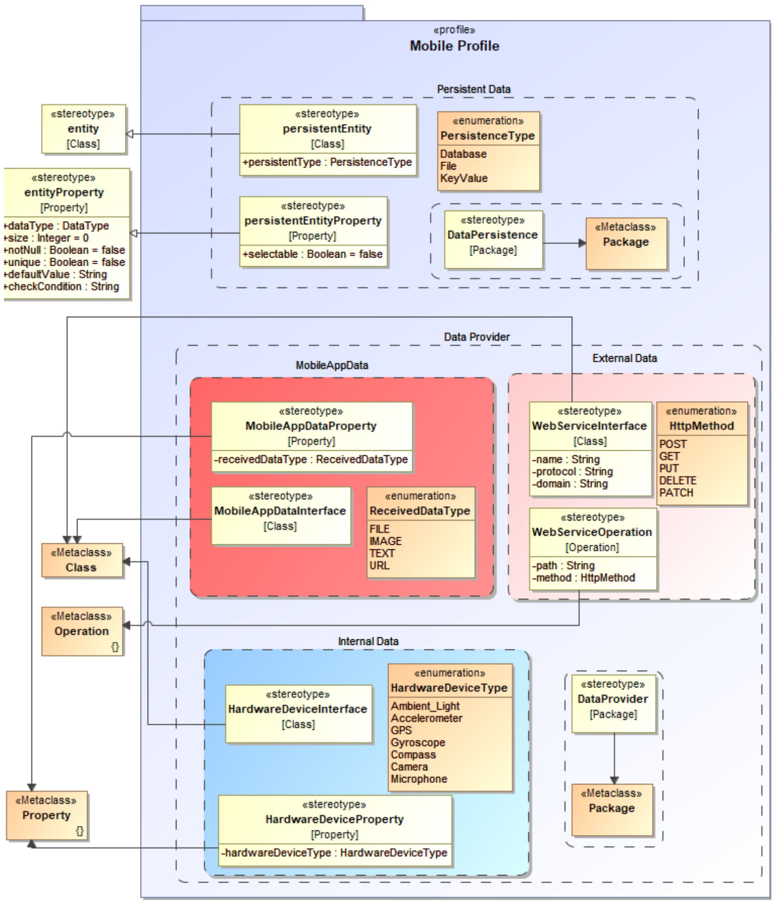

«profile»
Mobile Profile
Persistent Data
«stereotype» entity [Class]
«stereotype» entityProperty [Property]
+dataType : DataType
+size : Integer = 0
+notNull : Boolean = false
+unique : Boolean = false
+defaultValue : String
+checkCondition : String
«stereotype» persistentEntity [Class]
+persistentType : PersistenceType
«enumeration» PersistenceType
Database
File
KeyValue
«stereotype» persistentEntityProperty [Property]
+selectable : Boolean = false
«stereotype» DataPersistence [Package]
«Metaclass» Package
Data Provider
MobileAppData
«stereotype» MobileAppDataProperty [Property]
-receivedDataType : ReceivedDataType
«stereotype» MobileAppDataInterface [Class]
«enumeration» ReceivedDataType
FILE
IMAGE
TEXT
URL
External Data
«stereotype» WebServiceInterface [Class]
-name : String
-protocol : String
-domain : String
«enumeration» HttpMethod
POST
GET
PUT
DELETE
PATCH
«stereotype» WebServiceOperation [Operation]
-path : String
-method : HttpMethod
«Metaclass» Class
«Metaclass» Operation
Internal Data
«stereotype» HardwareDeviceInterface [Class]
«enumeration» HardwareDeviceType
Ambient_Light
Accelerometer
GPS
Gyroscope
Compass
Camera
Microphone
«stereotype» HardwareDeviceProperty [Property]
-hardwareDeviceType : HardwareDeviceType
«stereotype» DataProvider [Package]
«Metaclass» Package
«Metaclass» Property

Figure 4.8: UML Profile used for the definition of an ASM for mobile applications.

type) allows generating a database and uses the name of the persistent entity as a table. *File*, another *persistentType*, enable us to save the persistent entity in files through functions that facilitate to handle files (e.g., reading, writing, and others). The last *persistentType* (*KeyValue* type), allows storing the persistent entity in key-value pairs, through functions that we provide for the management of this data. In turn, each attribute of a persistent entity has properties with a stereotype *«PersistentEntityProperty»*. At the same time, each property can be enriched with sub-properties, defined as tagged values. Apart from those mentioned, we define the property *selectable*, used for the persistence with databases (*persistentType* = *Database*). This property allows indicating if an attribute of the persistent entity will be used as a key for data selection, to execute *delete* and *update* operations in the database. Finally, we include the stereotype *«Dat-*

aPersistence» to identify the package where the persistent data model will be included. According to the transformation rules, this property allows identifying we are working with the persistent data model.

Concerning data providers, we introduce the following interfaces: *WebServiceInterface*, *HardwareDeviceInterface*, and *MobileAppDataInterface*, for the representation of external providers (via web-services), internal providers (through commonly supported sensors among platforms) and interoperability with other applications (data types defined for communication between applications), respectively.

External providers provide data by communicating the mobile application with servers or remote databases. This communication is made via web-services following the *REST* architecture. The class *WebServiceInterface*, through tagged values, allows defining the URL base connection: [*protocol*]: [*domain*]. With *WebServiceOperation*, we define the functions or services in the class *WebServiceInterface.* The name of the function corresponds to the name of the service, to which, through tagged values, we add the *HTTP* method (*method*) and the service access path (*path*). The *HTTP* method possible selected values are *POST*, *GET*, *PUT*, *DELETE*, and *PATCH.*

Smartphones have sensors and specific hardware resources (e.g., camera, microphone, among others) that provide data flow; we consider these as internal data providers. The *HardwareDeviceInterface* class allows defining which sensors and hardware resources to use through *HardwareDeviceProperty* and its tagged value *hardwareDeviceType.* As possible options, we have an *Accelerometer*, *Gyroscope*, *GPS*, *Compass*, *AmbientLight*, *Camera*, and *Microphone.*

The mobile application can interoperate with other applications installed on the same mobile device, sending or receiving data. Through the class *MobileAppDataInterface*, we can represent data that a mobile application could receive from other applications. With the tag value *ReceivedDataType* of the property (*MobileAppDataProperty*), we select what data type the application may receive and handle. *File* (for file exchange), *Image* (to receive images), *Text* (to receive texts), and *Url* (to receive links) are available as options. Finally, we include the stereotype *«DataProvider»* to identify the package where the data provider model will be included. According to transformation rules, this property allows identifying we are working with the model of data providers.

4.3.2.3 PIM-ASM Mapping Rules

The next step in the ASM definition process is the PIM-ASM mapping. In this activity, a mapping between the elements of the PIM and the elements of the ASM has been made, seeking to identify which elements of the PIM metamodel must be transformed and, above all, to which elements of the ASM metamodel.

This mapping was performed through a visual analysis of the defined profile (e.g. see Figure 4.4 for RIA profile).

During the mapping phase we noticed that, in general, when the relation between elements from source and target profiles is an inheritance, the transformation that allows to obtain the corresponding target element is very simple and consists of the application of the correct stereotype.

The following are the mappings made between elements of the PIM and ASM metamodel elements that were applied directly for RIA PIM-ASM mapping:

- A *Table* element of the PIM, becomes a *RichTable* in the ASM.
- A *Form* element of the PIM, becomes a *RichForm* (see Figure 4.3.2.3).
- A *TextInput* of the PIM, is transformed into a *RichTextInput*.
- A *StaticObject* element of the PIM, becomes a *ClientStaticObject* (see Figure 4.3.2.3).
- A *ValueObject* of the PIM, is transformed into a *RichValueObject*.

In MoWebaMobile4FC mapping, we identified the following Mappings: *tProcess* from the PIM becames a *restProcess* in the ASM, a *service* from the PIM becames *cloudService* in the ASM and a *valueObject* from the PIM becames a *cloudValueObject* in the ASM.

In MoWebAMobile4Persistence mapping, we could detect that a class Entity in the PIM model, becomes a *PersistentEntity* class in the ASM model (see Figure 4.10) and that property *entityProperty* in the PIM model is transformed into a *persistentEntityProperty* in the ASM model (see Figure 4.11).

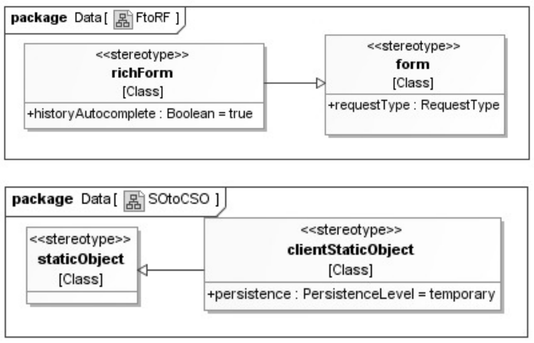


Figure 4.9: Mapping-RIA-Inheritance

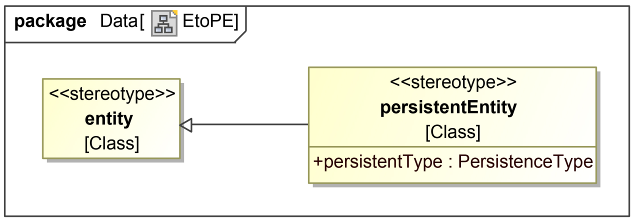


Figure 4.10: A class Entity in the PIM model, becomes a *PersistentEntity* class in the ASM model.

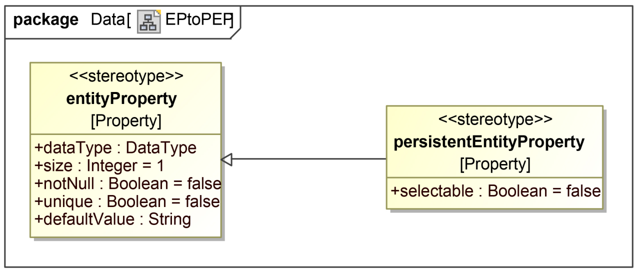


Figure 4.11: A property *entityProperty* in the PIM model is transformed into a *persistentEntityProperty* in the ASM model.

When the relation between two elements in one profile is not an inheritance, the mapping becomes a bit more difficult. In the case of the RIA profile, we found two relations of this type, one between the *ServiceState* and *AsynchronousCall* elements, and the other between the *ServiceState* and *RichTable* elements. Analyzing these relationships, we could conclude that the *AsynchronousCall* must

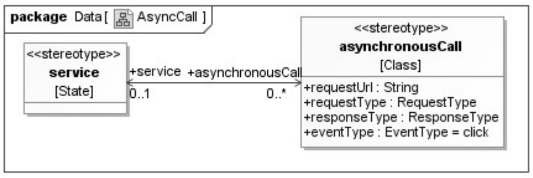


Figure 4.12: Mapping-RIA-Asynchronous service

be created when it is related to a *ServiceState*, which, in addition, must represent an asynchronous service (see Figure 4.12). This condition can be detected automatically, allowing the automation of the transformation.

4.3.2.4 PIM to ASM Transformation Rules

For the definition of the M2M transformation rules, we have chosen ATL[6] (Atlas Transformation Language) over QVT[7] (Query/View/Transformation).

This choice was based, above all, on the fact that ATL is considered one of the most widely used transformation languages, both in academia and industry, and a mature tool support is available [25].

Another very important choice was the selection of the engine execution mode of the ATL transformation, which has two modes of execution: default and refining [49]. When the source and target metamodels are different, it is mandatory to use the default execution mode, but when the metamodels of the source and target models are the same you can opt for either of the two execution modes [49].

In our case, both, the metamodels of source and target models are the same because MoWebA's implementation is based on profiles [57].

Moreover, the M2M transformation from PIM to ASM fits conceptually better to the refining mode, since the PIM is justly refined to obtain the ASM. For all this, we opted for the refining execution mode. This choice has greatly reduced the number of necessary transformation rules. However, this also led to some complications, since the application of profiles in ATL is performed in the imperative block and turns out this option when using refining mode.

[6]ATL.`http://www.eclipse.org/atl/`
[7]QVT.`http://www.omg.org/spec/QVT/About-QVT/`

For the above reason, we were forced to change the compiler. The default compiler in Eclipse is the EMF-specific Virtual Machine (EMF-specific VM), but it also provides other compilers. We opted for the EMF Transformation Virtual Machine (EMFTVM). Although the main reason for this choice was that it allows the use of the imperative block in the refining mode, which is necessary to work with profiles, there are some other advantages. For example, its performance is roughly 80% better than the EMF-specific VM [62], and allows us to invoke native Java methods [61].

Another important aspect for the transformation rules definition are the Configuration files. Configuration files allow the designer to control some transformation aspects, allowing the achievement of two important capabilities: on the one hand, the possibility of capturing specific design decisions of the system under design that would otherwise not be automated, and on the other hand, the possibility that the designer has some level of influence over the transformation rules.

Two different configuration files have been considered. One that allows to indicate which elements of the model are transformed and which are not, called "ArchConfTransformacion.yaml", and another one that allows to specify some properties for the classes created automatically when executing the M2M transformation rules, called "ArchConfAsynchronousCall.yaml".

The "ArchConfTransformacion.yaml" configuration file has two modes of operation (see Listing 4.1). Mode 1 specifies which elements of the PIM must be transformed, and mode 2 specifies which elements must not. If the file does not exist, all the PIM's elements referenced by the M2M transformation rules are transformed. Its structure is straightforward. It has three sections, and the first indicates the operation mode, the second, the affected classes, and the third, the affected properties. A class is referenced indicating the package containing it and its name, while a property, indicating the package and class containing it and the property name.

Listing 4.1: Configuration files that allow the designer to control some transformation process aspects.

```
mod: 1
```

```
classes:
    - package : package_containing_the_class
      class : class_name
properties:
    - package : package_containing_the_property
      class : class_containing_the_property
      property : property_name
```

The “ArchConfAsynchronousCall.yaml” configuration file specifies the properties to be added to the automatically created asynchronous classes.

It has a single section that allows to identify the classes to which the properties should be linked. Thus, since these classes must be specified before they are created, we identify them from elements of the PIM (i.e., the class and attribute that contains the asynchronous service that justifies the creation of the asynchronous class) (e.g., Listing 4.2).

Listing 4.2: ArchConfAsynchronousCall.yaml - Example.

```
classes:
    - classPIM : class_name_in_the_PIM_model
      atributePIM: property_name_in_the_PIM_model
      properties:
    - name : name_of_the_new_property
          stereotype: stereotype_of_the_new_property
```

Among the challenges of the M2M transformation that have been tackled, we can mention:

1. The inability to work with profiles in the refining mode, which forced us to use a compiler different from the one proposed by default.

2. The asynchronous services identification, indispensable for automating the creation of asynchronous classes.

3. The correct configuration of the IDE in order to access native java methods created expressly for the processing of the configuration files.

Next, we show some representative examples of M2M transformation rules defined for this project.

4.3.2.4.1 Headers The header section is an essential section of transformation rules as it details the compiler, the metamodels, the mode of execution, and the profiles used (see Listing 4.3).

Listing 4.3: Header section of the M2M transformation rules.

```
-- @atlcompiler emftvm
-- @nsURI UML2=http://www.eclipse.org/uml2/2.0.0/UML

module ReglasM2M;

create OUT : UML2 refining IN : UML2,
                        MOBILE_PROFILE : UML2,
                        CONTENT_PROFILE : UML2;
```

4.3.2.4.2 Called Rules The *called rules* is another vital section of transformation rules. The called rule named *applyMobileStereoTypes* allow changing the stereotypes in the target model (see Listing 4.4 below).

Listing 4.4: *applyMobileStereoTypes* called rule that change the stereotypes in the target model.

```
rule applyMobileStereoTypes(required : Boolean, s : UML2!Element, t :
    UML2!Element)
{
    using {
        new_stereotype : UML2!Stereotype = '';
        change_stereotype : Boolean = false;
        container : String = s.namespace.name.toString();
        element : String = s.name.toString();
        package : String = s.getNearestPackage().name;
    }
    do
    {
        for (stereotype in s.getAppliedStereotypes())
        {
            if (stereotype.getName() = 'entity')
            {
                new_stereotype <-
                    thisModule.getStereotype('persistentEntity');
            }
            else if (stereotype.getName() = 'entityProperty')
            {
                new_stereotype <- thisModule.getStereotype('persistentEntityProperty');
            }
            --- It is decided whether or not to change the stereotype
            if (container = package) {
                --- if container is equal to package, then the element is a class
                change_stereotype <- let object :
                "#native"!"atl::conf::ConfM2M" =
                    "#native"!"atl::conf::ConfM2M".newInstance()
                        in object.aplicarEstereotipo(container,element);
            } else {
                --- if container is NOT equal to package, then the element is a property
                change_stereotype <- let object : "#native"!"atl::conf::ConfM2M" =
                    "#native"!"atl::conf::ConfM2M".newInstance()
                        in object.aplicarEstereotipo(package,container,element);

            }

            if ( new_stereotype <> '' and change_stereotype)
            {
                --if the UML!Element does not get its stereotype automically applied (i.e., UML not required)
                --then we must apply it manually
                --note:surprisingly the "required" property doesn't seem to be exposed by UML2.
                --maybe look into this more
                if (not required)
                {
                    t.unapplyStereotype(stereotype);
                    t.applyStereotype(new_stereotype);
                }
                for (property in stereotype.getAllAttributes())
                {
                    --apply the value if there is one.
                    --don't apply the base type property as ATL cannot handle this.
                    --also don't touch read-only properties.
                    if(s.hasValue(stereotype,property.name)
                        and not property.name.startsWith('base_')
                        and not property.isReadOnly())
                    {
                        t.setValue(stereotype,property.name,s.getValue(stereotype,property.name
                    }
                }
                --marks the 'selectable' property as 'true' for the first
                    property of the class
                if (new_stereotype.getName() = 'persistentEntityProperty')
                {
                    if (not
                        thisModule.Selectable_Control.includes(t.class.name))
                        {
                        t.setValue(new_stereotype,'selectable',true);
                        thisModule.Selectable_Control <-
                            thisModule.Selectable_Control.including(t.class.name);
                    }
                }
                for (property in new_stereotype.getAllAttributes())
                {
                    if (property.name.toString() = 'dataType')
                    {
                        if (s.refGetValue('type').name = 'Integer') {
                            t.setValue(new_stereotype,'dataType',
                                thisModule.getDataType('int'));
                        }
                        else if (s.refGetValue('type').name = 'Real') {
                            t.setValue(new_stereotype,'dataType',
                                thisModule.getDataType('float'));
                        }
                        else {
                            t.setValue(new_stereotype,'dataType',
                                thisModule.getDataType('text'));
                        }
                        t.refSetValue('type', OclUndefined);
                    }
                }
            }
        }
    }
}
```

4.3.2.5 ASM to Code Generation

After the generation of the ASM of the application it is possible to go through the process of generating code. To do this, we have defined transformation rules from model to text using the Acceleo[8] tool for the three extensions.

These transformation rules follow a template-based approach in which text templates are specified with entries for data to be extracted from the model diagrams. The MTL[9] language was used for the definition of the templates, and OCL[10] to make queries to the model. In addition, services defined in Java were used to extend MTL with greater functionalities.

In the case of MoWebA4RIA, these rules are responsible for transforming elements defined in the logic and content diagrams to HTML5, Javascript, jQuery, jQuery code and Datatables libraries. The tools developed for this extension are available at `http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/mowebaria/`.

In MoWebAMobile4FC we have built a service in Java to extend the functions of the MTL. We have built the transformation rules based on the classes, properties and operations characterized by the respective stereotypes, tag values and enumerations defined in the ASM's profile. In this sense, such rules perform a mapping between the model elements defined and the target code to be generated. Basically, the generation for both side, mobile and cloud, depends on each combination of a *CloudServer*, a *RestProcess*, *ResourceInterface* and *CloudRequestHandler*. The target code generated consists, on one side, in native mobile code written in Java[11] for Android, in Swift[12] for iOS, and on the other side, in open source code written in Javascript[13] with Node.js for the Openshift and Amazon Web Services platforms. Additionally, our cloud implementation is based on Docker,[14] which is a container where an application runs. Moreover, Docker is an emerging method developed by the open source community for easing the portability of cloud applications. The tools developed for MoWebAMobile4FC are available at `http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/moweba-para-mobileapps-fc/`.

In MoWebAMobile4Persistence, Listing 4.5 presents the main template rule of

[8]Acceleo.`https://eclipse.org/acceleo/`

[9]MTL.`http://www.omg.org/spec/MOFM2T/1.0/`

[10]OCL.`http://www.omg.org/spec/OCL/`

[11]Java, link: `https://goo.gl/hGBggw`

[12]Swift, link: `https://developer.apple.com/swift/`

[13]Javascript, link: `https://www.javascript.com/`

[14]Docker, `https://www.docker.com/what-docker`

the Acceleo's transformations rules for the application generation. The generated target codes for Android and Windows Phone are Java and C#, respectively. Additionally, GUI code is also generated for Android (XML [2]) and Windows Phone (XAML [6]). This generated code is ready to be executed for both mobile platforms, Android and Windows Phone, previous compilation in their respective IDEs.

Listing 4.5: Main template of the transformation rules in Acceleo.

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/5.0.0/UML')]
[comment]Imports ...[/comment]

[template public generateElement(model: Model)]

[comment @main/]
[let aPackages: Sequence(Package) = model.eAllContents(Package) ]

[generateGeneralAndroidClasses(model)/]
[generateGeneralWindowsClasses(model)/]

[for (aPackage : Package | aPackages)]
[let aClasses: Set(Class) = aPackage.ownedElement->filter(Class) ]
[let p : Package = aPackage.ancestors(Package)->first()]
[comment]We go through the existing packages in the model, specifically:
    DataPersistence and DataProvider[/comment]

    [comment]If the DataPersistence package exists[/comment]
    [if (aPackage.hasStereotype('DataPersistence'))]

        [comment]Generate the beans or models of the application[/comment]
        [beansGenAndroid(aPackage, p.name.toLower())/]
        [beansGenWindows(aPackage)/]

        [comment]We only create this file if there is at least one
            Database type entity[/comment]
        [if
            (aPackage.isPackageHasThisPropertyStereotype('persistentEntity',
            'persistentType', 'Database'))]
            [generateDBForAndroid(aPackage, p.name.toLower())/]
            [generateDBForWindows(aPackage, p.name.toUpperFirst())/]
        [/if]

        [comment]We only create this file if there is at least one File
            entity type[/comment]
        [if (aPackage.isPackageHasThisPropertyStereotype('persistentEntity', 'persistentType', 'File'))]
            [generateFilesForAndroid(aPackage, p.name.toLower())/]
            [generateFilesForWindows(aPackage, p.name.toUpperFirst())/]
        [/if]

        [comment]We only create this file if there is at least one KeyValue entity type[/comment]
        [if (aPackage.isPackageHasThisPropertyStereotype('persistentEntity', 'persistentType', 'KeyValue'))]
            [generateKVForAndroid(aPackage, p.name.toLower())/]
            [generateKVForWindows(aPackage, p.name.toUpperFirst())/]
        [/if]
    [/if]

    [comment]If the DataProvider package exists[/comment]
    [if (aPackage.hasStereotype('DataProvider'))]
        [for (aClass : Class | aClasses)]
            [comment]If the class has the WebServiceInterface package[/comment]
            [if (aClass.hasStereotype('WebServiceInterface'))]
                [generateRestAndroid(aClass, p.name.toLower())/]
                [generateRestWindows(aClass, p.name.toUpperFirst())/]
            [/if]

            [comment]If the class has the HardwareDeviceInterface package[/comment]
            [if (aClass.hasStereotype('HardwareDeviceInterface'))]
                [generateSensorsAndroid(aClass, p.name.toLower())/]
                [generateSensorsWindows(aClass, p.name.toUpperFirst())/]
            [/if]
        [/for]
    [/if]
[/let]
[/let]
[/for]
[/let]
[/template]
```

As a summary of the main aspects of the generated code in MoWebAMobile4Persistence we have:

- Respect to data persistence, we used specific data persistence mechanisms for these platforms. Firstly, the generated database is SQLite for both platforms. Secondly, files are saved in the device, and specific functions are generated for each platform that allows executing *read* and *write* operations. Finally, key-value pairs enable handling *SharedPreferences* and *LocalStorage*, for Android and Windows Phone, respectively.

- Considering the data providers, we generate code from each of these identified providers. Firstly, the communication with external providers (e.g., servers or remote databases) is done via web-services following *REST* architecture. Secondly, we identified a series of common options for sensors and resources of hardware in smartphones: *Accelerometer*, *Gyroscope*, *GPS*, *Compass*, *AmbientLight*, *Camera*, and *Microphone*. Finally, we identify different data types commonly used in the exchange of data among applications (*ReceivedDataType*): *File*, for file exchange in general; *Image*, to receive images; *Text*, to receive text; and *URL*, to receive links (*links*).

- To get the first sight and test the generated functionalities, we generate a set of basic and general screens. These screens are forms defined from each persistent entity to load data and perform CRUD operations on them. Moreover, we generated a screen to verify the values thrown by each sensor and test the device hardware resources (e.g., GPS and camera).

- In addition to this application, we provide auxiliary classes (or *helpers*), equipping the user with functionalities to handle the different aspects mentioned.

The tools developed for MoWebAMobile4Persistence are available at `http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/mowebamobile/`.

4.3.2.6 An Overview of the Development Process with the MoWebA Extensions

Figure 4.13 illustrates the proposed process for the development of RIA applications with MoWebA4RIA. In the figure we can see that the process begins with the PIM definition with MagicDraw tool[15] using the corresponding UML profiles.

[15]MagicDraw.`http://www.nomagic.com/products/magicdraw.html`

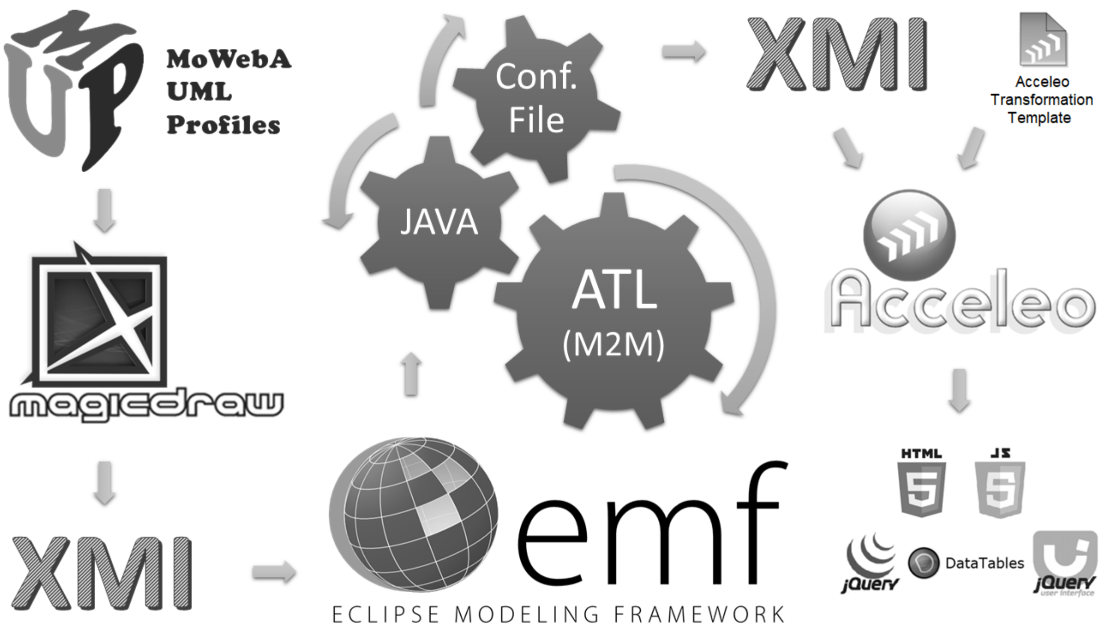


Figure 4.13: RIA development process

The PIM model is then exported to a XMI file, which is imported into the EMF[16] tool, where, through the mapping of the defined M2M transformation rules, it is transformed into a new XMI with diagrams corresponding to the ASM. The new XMI is imported into the Acceleo tool. The latter uses the transformation rules to perform transformations from model to text, generating the code in HTML5, Javascript, jQuery, Datatables and jQuery UI technologies, corresponding to the final implementation of the RIA.

In Figure 4.14, we observe the development process with MoWebA Mobile. We start modeling with MagicDraw [3], using the UML profiles of MoWebA. Then, M2M transformation is made from PIM to ASM using ATL language with ATL EMFTVM (ATL EMF Transformation Virtual Machine). The result of this step is the ASM Model, which could be adjusted manually. Then, to generate code from the performed models, we export these in XMI v2.1 format (XML of Metadata Exchange) and import them in Acceleo. Using defined transformation rules (Acceleo spreadsheets), we generate the source code of the mobile application function in the cloud (in the case of MoWebAMobile4FC) and data layer

[16]Eclipse Modeling Framework.`https://www.eclipse.org/modeling/emf/`

implementation (in the case of MoWebAMobile4Persistence). We define transformation rules to generate this implementation. In MoWebAMobile4Persistence, we have also developed a set of user interfaces to show the data layer functionalities. Finally, to obtain a native application for both platforms, the generated code must be compiled in their respective IDEs: Android Studio [1] and Visual Studio [5].

The tools defined for the three MoWebA extensions, which include the metamodels, UML profiles, PIM-ASM transformation rules, and ASM-code transformation rules, are available on the *MDD+ project* website [17].

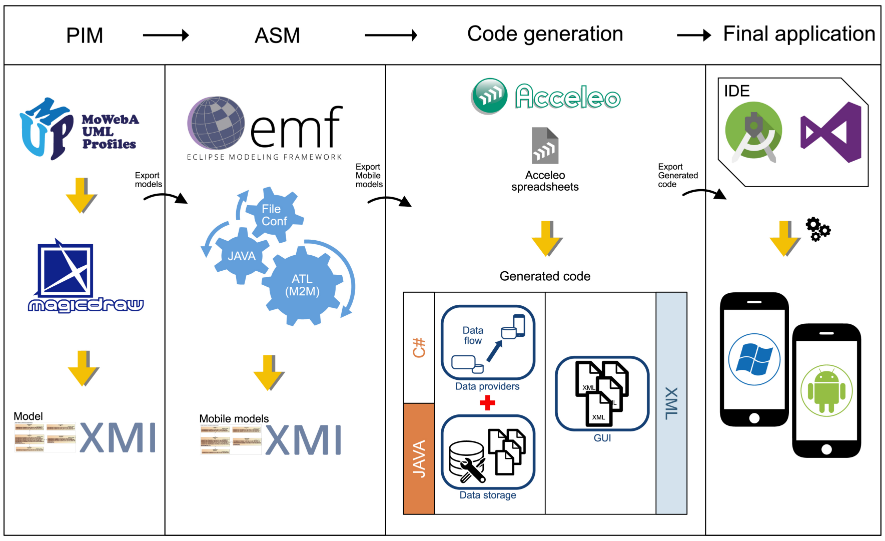


Figure 4.14: MoWebA Mobile development process

.

4.3.2.7 Proof of Concepts

The following are three proof of concepts developed with the MoWebA extensions during the research project mentioned in section 4.3. We followed the development process presented in section 4.3.2.6 which include the PIM and ASM modeling, and the transformation rules to obtain the final code.

MoWebA4RIA has been applied to a case called "Employee Marking". The application consisted of an employee marking system in which a guest user can

[17]http://www.dei.uc.edu.py/proyectos/mddplus/herramientas/

register as an employee and then login and be able to mark in or out. If the user is a supervisor, he/she is able to observe the clock-ins and clock-outs performed by the employees. Figures 4.15 and 4.16 show part of the ASM models for the Employee Marking application.

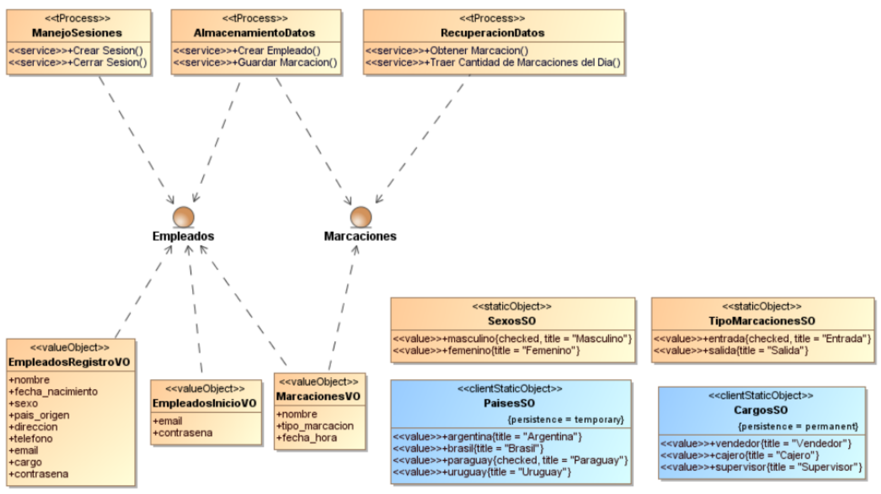

<<tProcess>>
ManejoSesiones
<<service>>+Crear Sesion()
<<service>>+Cerrar Sesion()
<<tProcess>>
AlmacenamientoDatos
<<service>>+Crear Empleado()
<<service>>+Guardar Marcacion()
<<tProcess>>
RecuperacionDatos
<<service>>+Obtener Marcacion()
<<service>>+Traer Cantidad de Marcaciones del Dia()
Empleados
Marcaciones
<<staticObject>>
SexosSO
<<value>>+masculino{checked, title = "Masculino"}
<<value>>+femenino{title = "Femenino"}
<<staticObject>>
TipoMarcacionesSO
<<value>>+entrada{checked, title = "Entrada"}
<<value>>+salida{title = "Salida"}
<<valueObject>>
EmpleadosRegistroVO
+nombre
+fecha_nacimiento
+sexo
+pais_origen
+direccion
+telefono
+email
+cargo
+contrasena
<<valueObject>>
EmpleadosInicioVO
+email
+contrasena
<<valueObject>>
MarcacionesVO
+nombre
+tipo_marcacion
+fecha_hora
<<clientStaticObject>>
PaisesSO
{persistence = temporary}
<<value>>+argentina{title = "Argentina"}
<<value>>+brasil{title = "Brasil"}
<<value>>+paraguay{checked, title = "Paraguay"}
<<value>>+uruguay{title = "Uruguay"}
<<clientStaticObject>>
CargosSO
{persistence = permanent}
<<value>>+vendedor{title = "Vendedor"}
<<value>>+cajero{title = "Cajero"}
<<value>>+supervisor{title = "Supervisor"}

Figure 4.15: ASM Logic Diagram for "Employee Marking" application

.

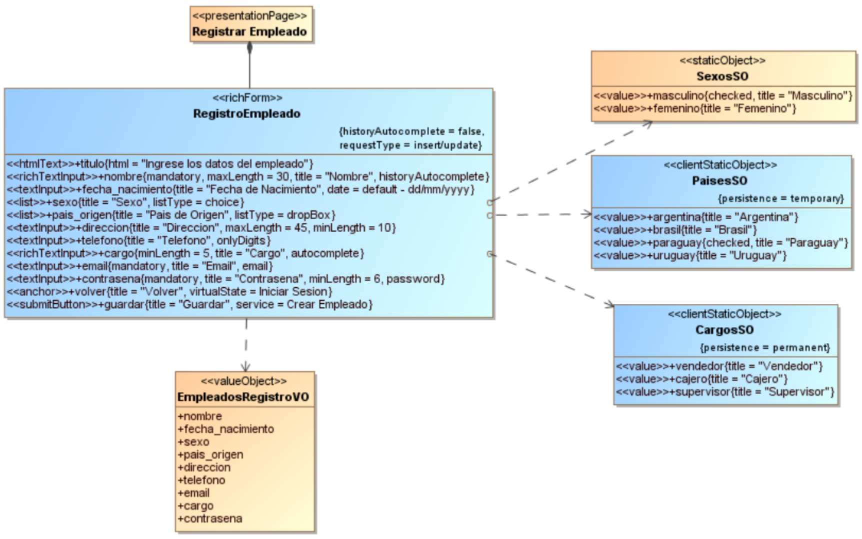

<<presentationPage>>
Registrar Empleado
<<richForm>>
RegistroEmpleado
{historyAutocomplete = false, requestType = insert/update}
<<htmlText>>+titulo{html = "Ingrese los datos del empleado"}
<<richTextInput>>+nombre{mandatory, maxLength = 30, title = "Nombre", historyAutocomplete}
<<textInput>>+fecha_nacimiento{title = "Fecha de Nacimiento", date = default - dd/mm/yyyy}
<<list>>+sexo{title = "Sexo", listType = choice}
<<list>>+pais_origen{title = "Pais de Origen", listType = dropBox}
<<textInput>>+direccion{title = "Direccion", maxLength = 45, minLength = 10}
<<textInput>>+telefono{title = "Telefono", onlyDigits}
<<richTextInput>>+cargo{minLength = 5, title = "Cargo", autocomplete}
<<textInput>>+email{mandatory, title = "Email", email}
<<textInput>>+contrasena{mandatory, title = "Contrasena", minLength = 6, password}
<<anchor>>+volver{title = "Volver", virtualState = Iniciar Sesion}
<<submitButton>>+guardar{title = "Guardar", service = Crear Empleado}
<<staticObject>>
SexosSO
<<value>>+masculino{checked, title = "Masculino"}
<<value>>+femenino{title = "Femenino"}
<<clientStaticObject>>
PaisesSO
{persistence = temporary}
<<value>>+argentina{title = "Argentina"}
<<value>>+brasil{title = "Brasil"}
<<value>>+paraguay{checked, title = "Paraguay"}
<<value>>+uruguay{title = "Uruguay"}
<<clientStaticObject>>
CargosSO
{persistence = permanent}
<<value>>+vendedor{title = "Vendedor"}
<<value>>+cajero{title = "Cajero"}
<<value>>+supervisor{title = "Supervisor"}
<<valueObject>>
EmpleadosRegistroVO
+nombre
+fecha_nacimiento
+sexo
+pais_origen
+direccion
+telefono
+email
+cargo
+contrasena

Figure 4.16: ASM Content Diagram example for "Employee Marking" application

.

In MoWebAMobile4FC the application consisted in a virtual shop, which includes a set of products to be offered and a set of users, who are the potential purchasers of such products. This application required the implementation in mobile and cloud platforms. In our case, the task was focused on the functions of network communication for the data exchange. Basically, the data to be exchanged was about the user (e.g., name, phone, email, address) and about the products (e.g., name, provider, pictures). Figures 4.17 and 4.18 present part of the ASM model for the virtual shop application.

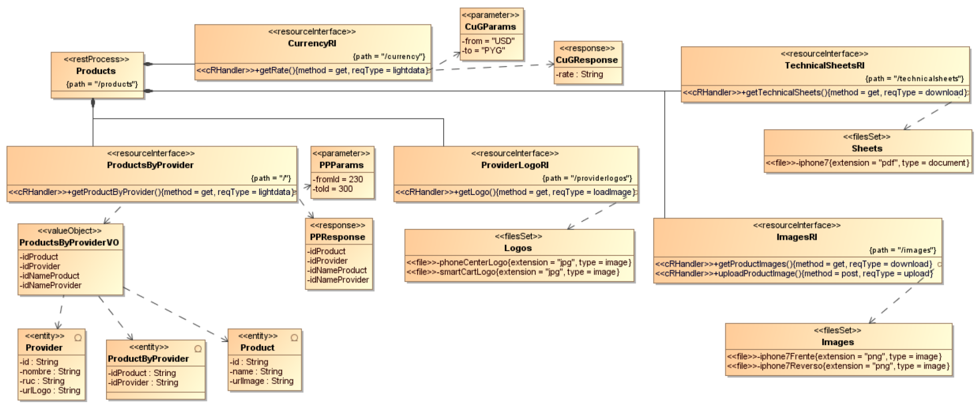


Figure 4.17: Example of an application for offering products. Products

.

In MoWebA4Persistence we used an example of modeling a mobile application called "e-market". The application consisted of a virtual store that makes home deliveries. One has available a product catalog and all the necessary information about each product. If one wishes to purchase a particular product, has to selects that product and adds it to the shopping cart. Once the selection of products is completed, the purchase is finalized, with the possibility of indicate whether or not you want the delivery to be made at home. The user needs to be registered to enter the application. The application must be available at all the time, even without a permanent Internet connection ("offline" mode). Figures 4.19 and 4.20 present part of the ASM model for the e-market application.

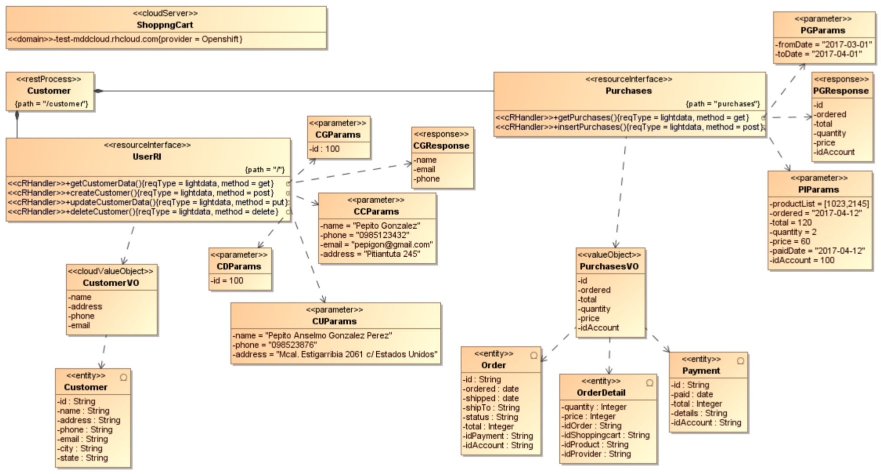


Figure 4.18: Example of an application for offering products. Customer, purchases diagram

.

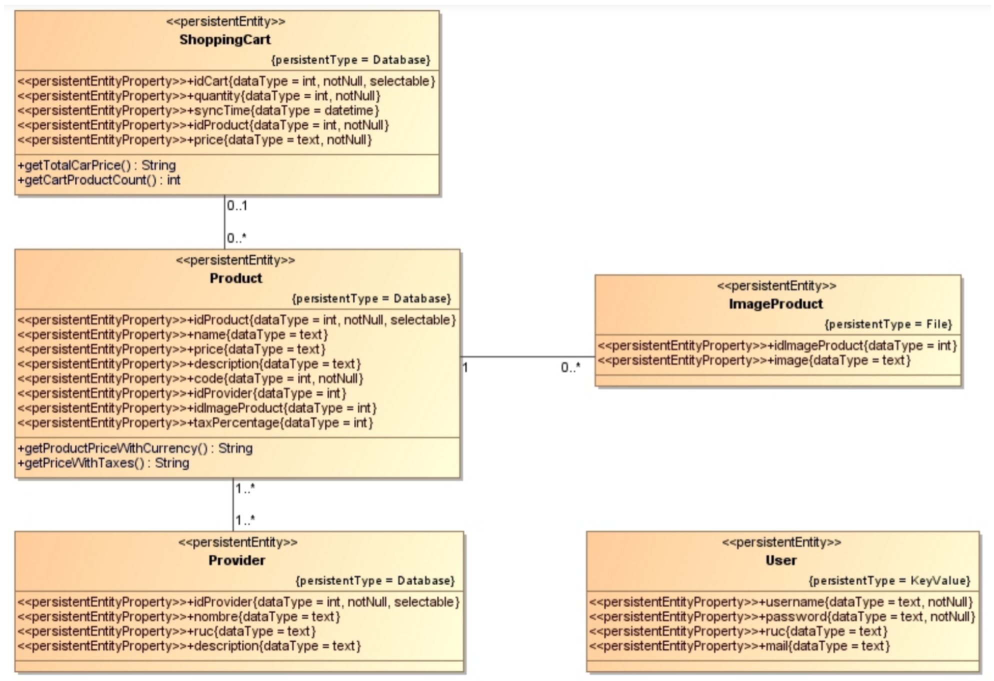
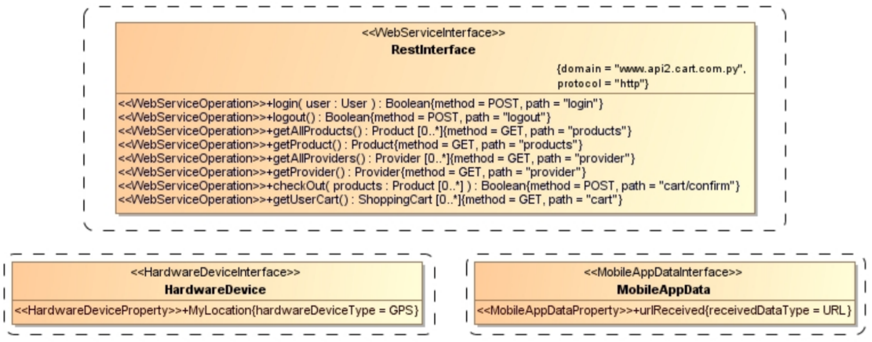

<<persistentEntity>>
ShoppingCart
{persistentType = Database}
<<persistentEntityProperty>>+idCart{dataType = int, notNull, selectable}
<<persistentEntityProperty>>+quantity{dataType = int, notNull}
<<persistentEntityProperty>>+syncTime{dataType = datetime}
<<persistentEntityProperty>>+idProduct{dataType = int, notNull}
<<persistentEntityProperty>>+price{dataType = text, notNull}
+getTotalCarPrice() : String
+getCartProductCount() : int
0..1
0..*
<<persistentEntity>>
Product
{persistentType = Database}
<<persistentEntityProperty>>+idProduct{dataType = int, notNull, selectable}
<<persistentEntityProperty>>+name{dataType = text}
<<persistentEntityProperty>>+price{dataType = text}
<<persistentEntityProperty>>+description{dataType = text}
<<persistentEntityProperty>>+code{dataType = int, notNull}
<<persistentEntityProperty>>+idProvider{dataType = int}
<<persistentEntityProperty>>+idImageProduct{dataType = int}
<<persistentEntityProperty>>+taxPercentage{dataType = int}
+getProductPriceWithCurrency() : String
+getPriceWithTaxes() : String
1
0..*
<<persistentEntity>>
ImageProduct
{persistentType = File}
<<persistentEntityProperty>>+idImageProduct{dataType = int}
<<persistentEntityProperty>>+image{dataType = text}
1..*
1..*
<<persistentEntity>>
Provider
{persistentType = Database}
<<persistentEntityProperty>>+idProvider{dataType = int, notNull, selectable}
<<persistentEntityProperty>>+nombre{dataType = text}
<<persistentEntityProperty>>+ruc{dataType = text}
<<persistentEntityProperty>>+description{dataType = text}
<<persistentEntity>>
User
{persistentType = KeyValue}
<<persistentEntityProperty>>+username{dataType = text, notNull}
<<persistentEntityProperty>>+password{dataType = text, notNull}
<<persistentEntityProperty>>+ruc{dataType = text}
<<persistentEntityProperty>>+mail{dataType = text}

Figure 4.19: E-market application data persistence ASM model

.



<<WebServiceInterface>>
RestInterface
{domain = "www.api2.cart.com.py",
protocol = "http"}
<<WebServiceOperation>>+login(user : User) : Boolean{method = POST, path = "login"}
<<WebServiceOperation>>+logout() : Boolean{method = POST, path = "logout"}
<<WebServiceOperation>>+getAllProducts() : Product [0..*]{method = GET, path = "products"}
<<WebServiceOperation>>+getProduct() : Product{method = GET, path = "products"}
<<WebServiceOperation>>+getAllProviders() : Provider [0..*]{method = GET, path = "provider"}
<<WebServiceOperation>>+getProvider() : Provider{method = GET, path = "provider"}
<<WebServiceOperation>>+checkOut(products : Product [0..*]) : Boolean{method = POST, path = "cart/confirm"}
<<WebServiceOperation>>+getUserCart() : ShoppingCart [0..*]{method = GET, path = "cart"}
<<HardwareDeviceInterface>>
HardwareDevice
<<HardwareDeviceProperty>>+MyLocation{hardwareDeviceType = GPS}
<<MobileAppDataInterface>>
MobileAppData
<<MobileAppDataProperty>>+urlReceived{receivedDataType = URL}

Figure 4.20: E-market application data provider ASM model

.

4.3.2.8 Validations with the Extensions

4.3.2.8.1 MoWeba4RIA

We have performed two validation experiences with MoWebA4RIA focusing on usability. The results of the former experience were presented in [92], and the latter were presented in [91].

Using the GQM template (Goal-Question-Metric) [16], the goal of this validation was stated as follows: *analyze* the MoWebA approach for the development of RIA, *for the purpose of* assessing its usability, *with respect to* effectiveness, efficiency and satisfaction, *from the viewpoint of* the developer, *in the context of* last year Computer Science students at the Catholic University (Asunción, Paraguay).

Based on this goal, the following research questions were established:

- **RQ1.1**: What effectiveness, efficiency and satisfaction does the PIM modeling process of the proposed approach present?
- **RQ1.2**: What effectiveness, efficiency and satisfaction does the PIM-ASM transformation process of the proposed approach present?
- **RQ1.3**: What effectiveness, efficiency and satisfaction does the code generation process of the proposed approach present?

The case consists of a RIA development project using MoWebA4RIA. The requested application consists of a system for enrolling students in the Computer

Table 4.5: Usability measurements for the PIM modeling process

Process	**Average Success Rate**	**Average Completion Time**	**Average Satisfaction**
Modeling	95.2%	17 min.	2.87

Science degree, composed of two pages with RIA elements. The experience was performed by ten students of the last year of the Computer Science degree at the Catholic University (Asunción, Paraguay) who were asked to model the necessary PIM diagrams, execute the PIM-ASM transformation rules to obtain the ASM models, make manual adjustments to the ASM models, generate code from them, and make manual adjustments to the generated code. These students have skills in model-based processes, acquired in previous courses, and in model-driven processes, acquired in the course in which this validation experience was carried out.

The models and code generated and adjusted by the students were corrected in order to obtain success rates (in relation to efficiency) for the modeling process and for the code generation process. Furthermore, the 10 students answered three After Scenario Questionnaires (ASQ) [69], one for each task (PIM modeling, PIM-ASM transformation, and code generation). These questionnaires allow us to know the perception of the satisfaction of the participants with respect to the accomplishment of every task.

Next we discuss the experience and the results according to the research questions.

1. **RQ1.1**: What effectiveness, efficiency and satisfaction does the PIM modeling process of the proposed approach present?

   Table 4.5 presents the usability measurements for the PIM modeling process carried out during the experience. On average, we observed that students completed the application model with a success rate of 95.2%, in 17 minutes, and with an ASQ score of 2.87 (ASQ scores range from 1 to 7, and values closer to 1 are those that indicate a higher level of satisfaction).

   Analyzing these results, we can see that a satisfactory success rate was obtained. The biggest difficulty at the time of modeling was given by misapplying some labeled values, stereotypes and forgetting to include a submit button on the forms. In these cases, intervention from researchers were made in order to correct errors.

Regarding modeling time, although we can not affirm with certainty that it corresponds to a favorable or unfavorable time, because it is necessary to compare this approach against another, from our experience in modeling and development, it seems to us that this time is reasonable.

The score obtained from the ASQ questionnaire reflects a good level of satisfaction from the students using the proposed modeling process.

From the above, we can derive that good effectiveness, efficiency and satisfaction were obtained in the modeling process.

2. **RQ1.2**: What effectiveness, efficiency and satisfaction does the PIM-ASM transformation process of the proposed approach present?

   The PIM-to-ASM transformation phase of the experience included two tasks: applying the ATL rules to automatically generate the ASM, and performing some manual adjustments to the ASM model in order to change default values established by the transformation rules (e.g., change the value of the autocomplete tagged valued to TRUE, since the default value of the transformation rule is set to FALSE). The first task was performed without researchers intervention, but the second task needed some help from researchers, in order to identify the tagged values that needed to be changed.

   Table 4.6 presents the usability measurements for the PIM-ASM transformation process of the experience. On average, the automatic MTM transformation was completed with a success rate of 100%, in 9.2 minutes, and with an ASQ score of 2.70.

   From the above, we can derive that very good levels of effectiveness, efficiency and satisfaction were obtained in the PIM-ASM transformation process.

3. **RQ1.3**: What effectiveness, efficiency and satisfaction does the code generation process of the proposed approach present?

Table 4.6: Usability measurements for the PIM-ASM transformation process

Process	**Average Success Rate**	**Average Completion Time**	**Average Satisfaction**
Modeling	100%	9.2 min.	2.70

Table 4.7: Usability measurements for the code generation process

Process	**Average Success Rate**	**Average Completion Time**	**Average Satisfaction**
Code Generation	90%	8.1 min.	2.87

Table 4.7 presents the usability measurements for the code generation process of the experience. On average, automatic code generation was completed with a success rate of 90%, in 8.1 minutes, and with an ASQ score of 2.87.

From these results, we can note that the success rate is just as satisfactory as the success rate obtained for the modeling process. However, we believe that the success rate for the code generation process could have obtained a better score, since this was directly affected by the input model used. Although several lines of code were generated from all models, these lines were not complete, due to imperfections of the models developed by participants, used as input for the code generator. We emphasize that in-situ corrections of the elaborated models were carried out, but not all the necessary corrections were detected due to the limitation of time in the working session.

In relation to the time of code generation, we can observe that a quite reduced time was achieved. This time could be generalized for the generation of all types of applications since it requires strictly mechanical and predefined steps, without great variation in the intervention of the developer. Also, this time is independent from the size of the application to be generated, because an increase in the size of the application would not generate a significant increase in time.

From these observations we can understand that the code generation process was carried out with good effectiveness, efficiency and satisfaction for these group of students.

4.3.2.8.2 MoWebAMobile4FC

MoWebAMobile4FC was analysed by a comparative study. In such study, we have measured the effort related to the development of the MobileApps-FC's network communication functions. Details of this comparative study can be found at [121].

In order to do the study, basically, we have proceeded as follows: i) we have selected an application as an example to develop its network communication; ii) we have used MoWebAMobile4FC, WebRatio Mobile Platform ([29]) and the traditional approach to develop independently the network communication; iii) we have measured and registered the development times taken by each alternative; iv) in case of the MDD approaches, we have registered and analyzed the modeling and generation differences which could affect the development effort. Following, we describe the study in more details.

We have considered a application similar to one presented in Brambilla et al. [29]. The application is a virtual shop for selling products. It is going to be deployed on tablets and cell phones for field agents, i.e., salesman that go to customers for selling the products. In this case, the task is focused on the functions of network communication for the data exchange. Basically, the data to be exchanged is about the products (e.g., images, technical sheets, providers). Part of the modeling is shown in Figure 4.21.

The application was developed using the three alternatives. The development have been done in an academic environment. The developer was a computer science student in his last year at the university. Moreover, the resources used to guide and support the development were those available on-line.

In this comparative study, we proposed the following research questions:

- **RQ2.1**: How much time of development do the traditional approach, WebRatio Mobile Platform and MoWebA Mobile require to obtain the network communication implementation?

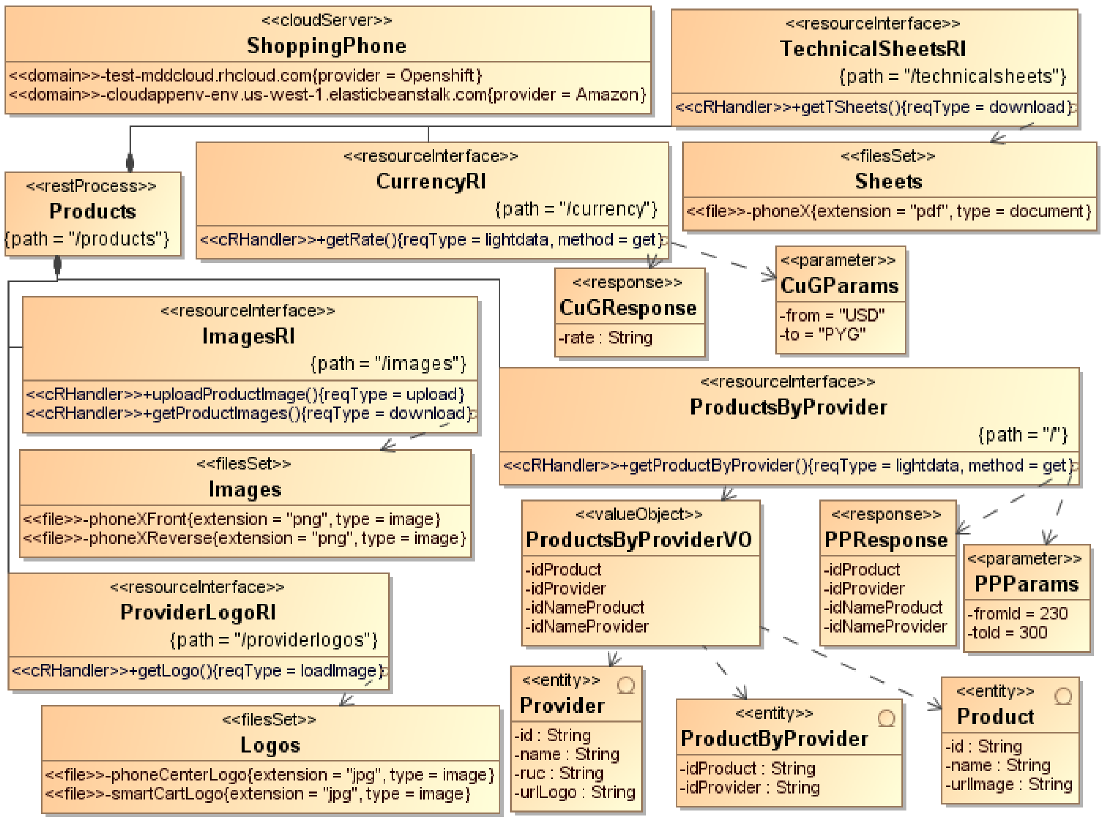


Figure 4.21: Example of an application for offering products

- **RQ2.2**: What differences in the modeling process between *A* and *B* could affect the effort required to develop MobileApps-FC?

- **RQ2.3**: Respectively, how many platforms do A and B generate code for (mobile and cloud)?

In this sense, we have compared MoWebA Mobile against the traditional approach and WebRatio Mobile Platform to answer RQ2.1. Similarly, we have compared exclusively both MDD approaches, MoWebA Mobile (A) and WebRatio Mobile Platform (B) to answer RQ2.2 and RQ2.3.

Following, we present the results by each research question defined.

1. **RQ2.1:** How much time of development do the traditional approach, WebRatio Mobile Platform and MoWebA Mobile require to obtain the network

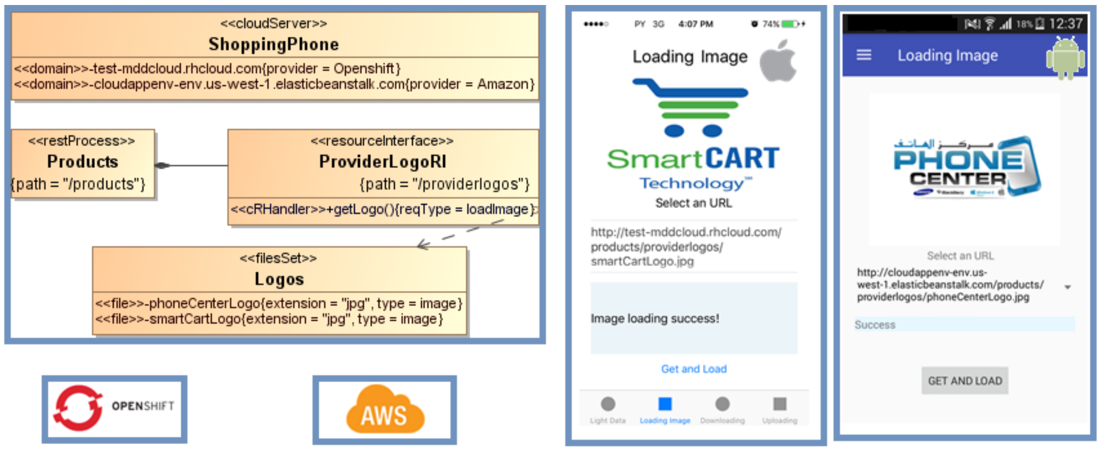

<<cloudServer>>
ShoppingPhone
<<domain>>-test-mddcloud.rhcloud.com{provider = Openshift}
<<domain>>-cloudappenv-env.us-west-1.elasticbeanstalk.com{provider = Amazon}
<<restProcess>>
Products
{path = "/products"}
<<resourceInterface>>
ProviderLogoRI
{path = "/providerlogos"}
<<cRHandler>>+getLogo(){reqType = loadImage}
<<filesSet>>
Logos
<<file>>-phoneCenterLogo{extension = "jpg", type = image}
<<file>>-smartCartLogo{extension = "jpg", type = image}
OPENSHIFT
AWS
Loading Image
SmartCART Technology
Select an URL
http://test-mddcloud.rhcloud.com/products/providerlogos/smartCartLogo.jpg
Image loading success!
Get and Load
Light Data
Loading Image
Downloading
Uploading
Loading Image
PHONE CENTER
Select an URL
http://cloudappenv-env.us-west-1.elasticbeanstalk.com/products/providerlogos/phoneCenterLogo.jpg
Success
GET AND LOAD

Figure 4.22: Example of an application for offering products. Load-image function

communication implementation?

We have compared MoWebA Mobile against the traditional approach and a consolidated MDD tool to analyze the required effort differences. Obviously, a MDD tool will improve the effort needed following a traditional approach (i.e., the manual development). However, in this case, the aim was to understand how big the difference is.

WebRatio Mobile Platform, is used in the industry and it is the most representative MDD tool for the development of the MobileApps-FC [120]. Therefore, we have compared it with MoWebA Mobile to see how much difference of effort exists with such kind of tool.

The development effort, measured through the development time, is presented in Figure 4.23.

It is worth noting that there exists a substantial difference of effort among the development times of the MDD approaches against the traditional one. For obtaining the same implementation, using WebRatio Mobile Platform and MoWebA Mobile have been necessary 0.93 hours (55 minutes) and 1.33 hours (1 hour 20 minutes), respectively. The traditional approach has taken 160 hours.

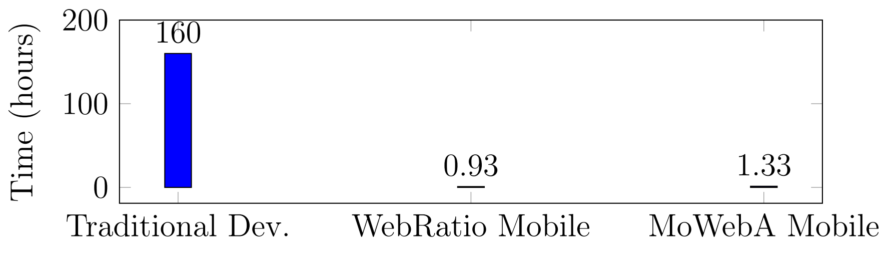


Figure 4.23: Comparison of development times

On one side, since, the network communication development implies, necessarily, working with several platforms and cloud service providers. In this sense, the developer had to face difficulties like the transition between different development environments (mobile, cloud), the use of different frameworks and programming languages. Such difficulties slowed down the development using the traditional approach.

On the other side, about the MDD approaches, they have several properties which support such difference of time. Firstly, the abstraction of specific details of the different platforms through the models. We highlight two advantages of such abstraction. On one side, it prevents dealing with the difficulty of working with different technologies. On the other side, it allows developers without specific platform and communication knowledge to get specific platform implementation of the network communication of the MobileApps-FC. Secondly, the automatic generation of platform specific code from the built model, which save most of the development time. Third, the generated code is already tested, so the probability to spend time in fixing bugs decreases.

Therefore, we could say that there is a considerable improvement in saving effort using a MDD approach for the development of the network communication of MobileApps-FC.

Comparing the MDD approaches, there is a difference that favors to one of them. First of all, we have to say that WebRatio Mobile Platform has a specific development environment, which eases, makes simple and faster the modeling comparing with MoWebA Mobile. Furthermore, WebRatio Mobile Platform is a robust and mature platform used in the industry. In contrast, in MoWebA Mobile the modeling and the generation are made with tools of general purpose which slowed down the development process. Therefore, we suppose that if we build a MoWebA Mobile's specific tool, it could help to make more simple and faster the process of modeling and generation. Such eventual reduction could imply equalizing, or even improving, the network communication development time of WebRatio Mobile Platform. Following, we present further differences which reinforce the possible improvement of development time using MoWebA Mobile through a more specific tool. We refer to MoWebA Mobile as *A* and to WebRatio Mobile Platform as *B*.

2. **RQ2.2:** What differences in the modeling process between *A* and *B* could affect the effort required to develop MobileApps-FC?

   On one side, *A* prescribe all the modeling and configurations in a unified model, while *B* works with two models and projects. One project for mobile, another project for back-end. In case of *A*, the purpose of working with the same project and model for designing the communication is to abstract the developer from individuals settings and modelings by each side (mobile and cloud). In other words, since the communication implies two sides (mobile and cloud), from a unified model, at the moment of generation, the design and settings are replicated in both sides. Therefore, the developer "works once" instead of twice or more, which is the case of *B*. For instance, if a url is modified, then, the developer does not modify it for the cloud side and for the mobile side, the url modification is made only once in the model. Afterwards, thanks to the transformation rules, the change is replicated in both sides, the mobile and cloud ones. In this sense, *A* saves effort in the modeling process and consequently, in the overall process of developing MobileApps-FC.

   On the other side, the main difference of MoWebA regarding other MDD approaches is the inclusion of three design aspects which could help to address the portability challenge. Since, in this work, where we have focused on the network communication aspect, we consider one of such aspects, which is the ASM. The relevance of the ASM is to improve the portability of the PIM regarding the different architectures. Even though, WebRatio considers as well the ASM, MoWebA considers, additionally, model to model (M2M) transformation rules, which enable the semi-automatic transition from the PIM to ASM. In fact, there exist such rules for other architectures.[18] Regarding the network communication aspect, M2M transformation rules were made manually. Such saving of time implies as well a saving of effort in the process of developing MobileApps-FC.

3. **RQ2.3:** Respectively, how many platforms do A and B generate code for (mobile and cloud)?

   On the mobile side, both, *A* and *B*, generate code for iOS and Android, the most popular platforms. While *A* generates native mobile code, *B* generates code for hybrid applications. On the cloud side, *A* generates code for two providers (Openshift and Amazon). *B* generates a Java application just for one provider, that is its own cloud service platform.[19] With *B*, thanks to its development tools, the application can be automatically deployed in the cloud. In case of A, it generates an implementation to run in two different clouds service provider's (Openshift and Amazon). Even though, we highlight that the code generated using A, is based on Docker,[20] which is a method developed by the open source community. Precisely, one of the main goals of Docker is to ease the cloud application portability. Therefore, the code generated could be ported, more easily, to other providers which include Docker in their services.

   A brief summary of the comparative study is shown in Table 4.8.

4.3.2.8.3 MoWebAMobile4Persistence

MoWebAMobile4Persistence was analysed considering the usability and portability aspects.

For this validation, we were guided by the activities suggested by Wohlin et al. [149] for case studies. Details of this validation can be found at [94].

[18]M2M transformation rules for RIA, `https://goo.gl/8Lsy6n`
[19]WebRatio Cloud Plans, link: `https://goo.gl/ByQgMp`
[20]Docker, link: `https://www.docker.com/what-docker`

Table 4.8: Brief summary of the comparative study

Aspects	**Approaches**			
	MoWebA M. (A)	**WebRatio M. (B)**	**Traditional**	
Dev. time	1 h 20 min	55 min	160 hs	A's time can be improved using a specific tool
Modeling diff.	Unified model	One model for mobile, another one for cloud	—	In A, the network communication is designed in only one model which saves design effort
	It considers M2M semi-automatic rules from PIM to ASM	It does not consider such M2M rules	—	The semi-automatic M2M rules could help to save effort in the modeling process
Number of Gen. Platf.	2 mobile platforms, 2 cloud platforms	2 mobile platforms, 1 cloud platform	—	The cloud code generated by A is based on Docker, which eases the code portability

The evaluation focuses on validating the M2T transformation rules that allow obtaining the final code from the ASM. First of all, we focus on evaluating the usability of our approach to get the first evaluations of the end user's experience. Secondly, we are interested in evaluating the portability of our approach. The fragmentation problem increases the effort of developing mobile applications as each mobile platform handle data persistence mechanisms differently.

To define the goal of the evaluation, we used the GQM paradigm (*Goal Question Metric*) [16]. We established the modeler and developer profiles. The first is a person with enough knowledge in modeling with MoWebA. The second is a person with experience in the development of mobile applications, able to understand and make changes to the generated code.

The defined **goal**(G) is: *Analyze* the MDD approach for the development of native mobile applications focused on the data layer: MoWebA Mobile, *for the purpose of* a better understanding *in regard to* the usability and portability *from the viewpoint* of the modeler and developer *in the context of* mobile application development.

The **case analysis** consisted of the development of a *E-market* application for online purchases.

The complete development of the mobile application can be divided into the following **development stages**:

1. modeling with MoWebA Mobile;

2. code generation from the model;

3. generation of the application from the generated code; and,
4. modifications to the generated application.

From the aspects mentioned above, and to achieve the stated goals, we established **research questions** (RQ).

- **RQ3.1:** What effectiveness, efficiency, and satisfaction does the modeling process of the proposed approach present?
- **RQ3.2:** What effectiveness, efficiency, and satisfaction does the code generation process of the proposed approach present?
- **RQ3.3:** What perception of satisfaction does the proposed MDD approach present?
- **RQ3.4:** What effectiveness, efficiency, and satisfaction does the process of modifying the generated application present?
- **RQ3.5:** What perception of portability does the proposed MDD approach present?

From the established goals, we classify the **participants of the experience** with two profiles:

- *Modelers*: six students of the 8th semester of the Computer Science career at the Catholic University of Asuncion. At the time of the experience, they were finishing the Software Engineering 1 subject, with enough knowledge in modeling with MoWebA after they had been evaluated for a semester developing a complete application with MoWebA.
- *Developers*: five mobile developers, four of them with experience working with Android applications, and one developer with Windows Phone applications development experience. Using networking, we contacted a group of mobile developers active in the industry at that time. We started with our laboratory colleagues, and they, in turn, invited their acquaintances. The average development experience was 1.5 years in the industry.

According to the stages of this evaluation experience, the modeler was in charge of the modeling process and code generation. On the other hand, the developer was in charge of generating the application from the generated code and making modifications to the application.

The experience was carried out in three work sessions. In the first and second sessions, we worked with modelers, and in the third session, with the developers. A researcher was in charge of conducting and supporting the experience.

In the first session (150 minutes long), the researcher presented MoWebA Mobile to the modelers, and together they worked in a mobile application case example using MoWebA Mobile, showing modeling and code generation processes.

In the second session (140 minutes long), the researcher presented the application *E-market* to the modelers. The modelers received a document with the system requirements. Modelers were asked to obtain the data persistence and data provider models. Then, they started the modeling process. This activity was divided into two stages: first, modeling without the support of the researcher, and then modeling with the support of the researcher. This whole process was timed to obtain data related to effectiveness. At the end of modeling, the modelers answered an *After Scenario Questionnaire* (ASQ). This questionnaire focuses on the measurement of the satisfaction of a person concerning the accomplishment of a task [12], with a score ranging from 1 to 7, where values closer to 1 indicate a higher value of satisfaction [135]. Afterward, the modelers continued with the code generation process from the models they have made. Again, this process was timed. At the end of the code generation, the modelers had another ASQ. Concluding this session, the modelers answered a *System Usability Scale* (SUS) questionnaire about the entire session, which focuses mainly on the measurement of user satisfaction, covering a variety of the system usability aspects [12].

Finally, in the third session (203 minutes long), the researcher presented MoWebA Mobile and the application *E-market* to the developers. The process to import the generated code to the IDEs was explained. The developers received a document with the requirements of the system. They were asked to import the generated code to the respective IDEs and generate the application. This process was timed to obtain data related to effectiveness. At the end of this process, the developers answered an ASQ. In this case, and a difference with the previous session, we complement this ASQ with additional questions in order to collect more information, specifically asking about the comments and opinions of the participants. Subsequently, as a new task, the developers tested the generated application, made verification of the generated functionalities in the different mobile phones, and experienced the generated application in Android and Windows Phone platforms. This process focused on to get the first approximations about the portability of the proposed approach, for which we employed a *questionnaire with open questions* and asked the developers to answer it. With this questionnaire, we collected opinions from the developers regarding the possible differences in the functionalities and mechanisms of persistence, in the different platforms, and the usefulness of the approach for the automatic generation of mobile applications for different platforms. Afterward, the developers made modifications to the generated application, which consisted of changing the type of persistence mechanism of an entity. The developers were divided into two groups. The first group made modifications directly to the generated application (manual modifications). The second group made changes to the models (automatic modifications) and followed all the development steps with MoWebA Mobile to generate the application with the changes required. This modification process was timed, and the modifications made were saved separately. At the end of this process, the developers answered an ASQ. Concluding this session, the developers responded to a SUS questionnaire about the entire session.

From the experience with modelers and developers, quantitative and qualitative data were obtained. These data allowed answering the research questions and improving the understanding of the perception collected.

The following are the answers to the request questions:

1. **RQ3.1:** What effectiveness, efficiency, and satisfaction does the modeling process of the proposed approach present?

   As we mentioned, this process was divided into two stages: first, modeling without the support of the researcher, and then modeling with the support of the researcher.

   Considering the tasks in this process, we remark that all modelers completed the data persistence model. Difficulties were found during the data provider modeling, especially with the services modeling. Common errors found were: some tag values misapplying and stereotypes names missing. In general, 33% of modelers were able to complete the entire process without asking for the support of the researcher. The remaining modelers completed the task but asked for some help at some point during this process.

Based on our perception, most of the modelers were able to complete this process satisfactorily. Furthermore, looking at the quantitative analysis presented in Table 4.9, the average success rate was 82%, which we consider as a notable success rate.

Table 4.9: Usability measurements of the modeling process.

Process	**Average Success Rate**	**Average Completion Time (minutes)**	**Average Satisfaction (ASQ score)**
Modeling	82%	64.17	2.89

In regard to the completion time, we measured that more time was required during the first stage, where the modelers, with the help of the documentation provided, worked on the modeling process. After this, some questions arose in order to complete the task. As a result, the average completion time of the complete process was 64.17 minutes. We consider this number as reasonable, taking into account the time lost because of some errors during modeling. Nevertheless, as a first experience working with MoWebAMobile4Persistence, this modest performance draws attention to the fact they had informational materials, and the complexity of modeling was reduced. This performance gives us clue that more training could be necessary to reduce the learning curve.

From all we mentioned above, we concluded MoWebAMobile4Persistence was accepted among the modelers, thanks to their experience using MoWebA, but more training was necessary to achieve higher levels of success rate and efficiency. The score obtained from the ASQ questionnaire reflects an acceptable level of satisfaction of the modelers with the modeling process.

2. **RQ2:** What effectiveness, efficiency, and satisfaction does the code generation process of the proposed approach present?

In this process, all the modelers were able to complete the tasks, generating the code application satisfactorily in Acceleo. Nevertheless, errors were detected during the code generation. In total, 33% of modelers presented errors in this process due to imperfections found in their models from the modeling process. Time was lost detecting such errors, increasing the average completion time, causing a number significantly high for an automatic generation. From Table 4.10, the average completion time was 15.33 minutes. Despite this fact, we noticed good results considering the efficiency of the modelers.

Table 4.10: Usability measurements of the code generation process.

Process	**Average Success Rate**	**Average Completion Time (minutes)**	**Average Satisfaction (ASQ score)**
Code generation	100%	15.33	1.94

From our perspective, this automatic code generation process obtained good results. We noticed a compensatory experience once the modelers generated code from their models, seeing the effort of the previous task materialized. The score obtained from the ASQ questionnaire for this code generation process, and comparing with the obtained in the modeling process, reflects a very good level of satisfaction for the modelers.

From the view point of the developers we did the following analysis. As we mentioned in *RQ3.1*, there were errors during the modeling process. In order to all the developers could reach a unified generated application and do not drag previous errors in this process, we decided to provide a 100% complete model to start the code generation process.

From the collected comments during this activity, the main problems mentioned by the developers were related to the IDE. First, problems with the configuration of the IDE, loosing time in the installation of updates and import of required libraries. This setback could have been avoided with necessary adjustments during the preparation of the experience. It is not considered as an error of the developers. Similarly, problems also occurred importing the generated code, a process that required creating a new project in the IDE, and then locating the generated code in folders.

Although, with an almost perfect success rate (see Table 4.11), the drawback with the use of the IDE got our attention because, despite the work experience the developers had, the tasks of creating the project and then adapting the generated code to it took longer than expected. From these results, we analyzed methods to improve the fulfillment of this process: a complete IDE project structure generation or even the generation of the final application.

Table 4.11: Usability measurements of the application generation process from the generated code.

Process	**Average Success Rate**	**Average Completion Time (minutes)**	**Average Satisfaction (ASQ score)**
Application generation	98%	51.6	2.20

We noticed that the developers were very engaged with the experience of validation, willing to finish the tasks, participatory, and striving to achieve good performance. Furthermore, in general, we perceived a good level of satisfaction during the application generation process (also reflected in the ASQ results).

Concluding, we present additional data collected to be used in later analysis. The average of generated lines of code (LoC) for Android and Windows Phone were 6411 and 3828, respectively. This number of lines corresponds to the total amount of lines of each developer's project, so there were certain variations in each case. The lines of code counted include blank lines, comments, and the *imports* in each class.

3. **RQ3:** What perception of satisfaction does the proposed MDD approach present?

   The modelers' average score of perception of satisfaction of MoWebA Mobile was 50 in the SUS, with a standard deviation of 16.89. Converting this score to Sauro and Lewis percentile rank[21], we obtain a value of 13%. This result

[21] About the SUS questionnaire, Sauro and Lewis [125] mention that the best way to interpret

is below average.

In general, we noticed some doubts in the use of the proposed approach from the modelers. With the introduction of new concepts of the approach, and considering this as the first working experience with the approach, we believe this complexity perception could improve with more training, which could minimize the learning curve.

On the other hand, we obtained the developers' opinion regarding the general development with our approach and the coverage of the initial requirements by the generated application. Many agreed to say that since the modification process could not be completed, it was difficult for them to give an opinion on the validation of the initial requirements. Regarding the suggestions and opinions about MoWebAMobile4Persistence, they stressed that it is an easy to use approach, and interesting things could be achieved with some adjustments. The suggested adjustments include the automation of this process, that means improving the import of the generated code to the IDE.

The mentioned above is reflected in the developers' average score of perception of satisfaction of MoWebAMobile4Persistence, which was 70 with a standard deviation of 15.2. In a percentile rank, the result is above average with a value of 56%. Furthermore, compared to the perception of satisfaction obtained by the modelers (50 points in the SUS), the score of the developers is high and may have been even higher if not for the errors mentioned in *RQ3.4* and *RQ3.5*. However, in general, analyzing the SUS questionnaires of the developers, and unlike the modelers, the proposed approach was well accepted, considering it consistent and not complicated.

4. **RQ4:** What effectiveness, efficiency, and satisfaction does the process of modifying the generated application present?

   The requested modification consisted of changing the type of persistence mechanism of an entity: *key-value* instead of *database* as a new persistence mechanism. We worked in two groups: the manual development involved

the results is normalizing and obtaining the percentile rank. Any result with a percentile rank less than 50% is, by definition, below the average, and anything above 50% is above average.

three developers (they developed for Android platform), and the development with MoWebA Mobile approach involved two developers (one developer working with Android and another with Windows Phone platform).

Analyzing the results of the first group mentioned above, and regarding efficiency, we can see a lower performance in comparison with previous processes. Only 33% of developers could fully complete the requested modification. The other two developers reached only 50% of the development. These developers made the requested modification but could not show the data from the database on the screen. For this group, the main problem commented was that the designated time to finish all the requested changes for this task was insufficient, but they encouraged saying that with a little more practice, they could do it without inconvenience. The previous discomfort is reflected in the level of satisfaction the ASQ questionnaire returned (see Table 4.12).

Table 4.12: Usability measurements on the process of modifying the generated application: code modifications.

Process	**Average Success Rate**	**Average Completion Time (minutes)**	**Average Satisfaction (ASQ score)**
Modifications to the code	73.3%	57	3

Concerning the results of the second group, firstly, only one of the developers, who worked with the Windows Phone application, could fully complete the requested modification and test successfully the changes made. The other developer, who worked with the Android application, was able to generate the application code successfully with the requested changes, but commenting some problems creating the project and importing folder to the IDE, so he could not test the application. In this case, during the evaluation, we could test this application, obtaining excellent results. The developers who made the requested modifications to the code highlighted the readability and proper structure of the generated code. Secondly, the average activity time of this second group was 38.5 minutes. This average time reflects the drawbacks mentioned above with one of the developers, but it is worth noting that this developer was able to complete the activity successfully in 20 minutes (see Table 4.13).

Finally, we observed the developers using MoWebAMobile4Persistence, more confident and satisfied with the experience. Unlike the score obtained in the manual development, the ASQ score for this group reflects a very good level of satisfaction.

Table 4.13: Usability measurements on the process of modifying the generated application: model modifications.

Process	**Average Success Rate**	**Average Completion Time (minutes)**	**Average Satisfaction (ASQ score)**
Modifications to the model	72%	38.5	1.67

To summarize, all developers were able to accomplish the requested task in this stage, changing the type of persistence of the entity, but only 40% of them could complete and test the changes made.

Next, we highlight some interesting points found during the analysis of both development groups:

- *Time*: we notice a considerable difference comparing the completion time of those who could fully finish this task. It can be said that the development with MoWebAMobile4Persistence allowed making the requested modification 2.85X faster than developing it manually.
- *LoC of the requested modifications*: comparing the development for the Android platform, we found a nontrivial difference: on average, 309 lines were added developing manually, and 226 lines were added using MoWebAMobile4Persistence. We highlight the conciseness of the transformation rules, in the context of the proposed exercise.
- *ASQ*: both groups experienced different types of errors during the task, but the difference in both ASQ scores reflects a better level of satisfaction using MoWebAMobile4Persistence.
- The same modification was made manually and by the proposed approach. We emphasize that the changes to the model did not re-

quire any extra manual modification, highlighting the expressiveness of MoWebAMobile4Persistence, in the context of the proposed exercise.

5. **RQ5:** What perception of portability does the proposed MDD approach present?

   We emphasize the activities about portability were carried out successfully by all the developers. By loading data, the developers were able to test the application and the data persistence mechanisms available. Then, they carried out the same tests in different mobile OSs.

   Through the questionnaire, we were able to obtain the perception of the developers about the portability of MoWebA Mobile. Here is a summary of the answers gathered:

   - *Did you notice any difference concerning the functionalities?* In general, all agreed there were no problems with data persistence, and they were able to test the different data persistence mechanisms successfully in the different platforms. However, they highlighted that there were problems with some data providers.
   - *Do you find MoWebA Mobile useful?* Everyone found the proposal useful, noting the savings in effort and time in code generation is remarkable in this type of development.
   - *The problems in this scenario* were with the data providers. These drawbacks were presented using some sensors like GPS and gyroscope, and some specific hardware such as the camera.

The experience towards the first approach of evaluation of portability was positive. The comments from developers allowed us to verify the data persistence mechanisms worked correctly in different versions of the mobile OS. We highlight that there were problems with some data providers and some interface details, which challenges us to continue improving to achieve a more robust approach.

4.3.3 Data Collection

In this experience, data collection can be classified as of first degree, 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, complemented with open weekly meetings, observations and focus groups.

The meetings were approximately 120 minutes in length. Notes were taken during meetings, but the main sources of data were results/outputs expected for each stage. Focus groups and interviews were done in a same meeting.

We considered as **information sources** for data collection:

1. the *project documentation*, which includes PIM metamodels, ASM metamodels, PIM-ASM transformation rules, ASM-Code generation, proof of concept and all the documentation generated for each validation extensions;
2. the work sessions' *timesheets* of each validation extension; and,
3. *questionnaires* for validation of MoWebA4RIA and MoWebAMobile4Persistence.

The **quantitative data** were collected from these three information sources. On the other hand, the **qualitative data** were obtained from the comments and opinions of the participants.

In the first place, the **project documentation** allowed us to determine the success rate of the participants. Therefore, the **timesheets** permitted us to determine the completion time of each process in the validation extensions.

In the MoWebA4RIA and MoWebAMobile4Persistence validation experiences we used three types of **questionnaires**:

1. ASQ, to calculate the average satisfaction for each process;
2. SUS, to determine the measurement of user satisfaction for each session; and,
3. a questionnaire with open questions, to get first approximations about the perception of portability of the approach.

Table 4.14. presents a summary of data collected during the experience.

Table 4.14: Summary of data collected during the experience.

Data Collection method	**Materials**	**Outputs**
1. Definition of the scope to be considered for extensions.		
Focus Group, Open and semi-structured interviews	Scientific documentation (articles, proceedings, books)	Theoretical framework and state of the art related to each architecture; Document with scope definition.
2. Revision and adjustments to the PIM taking into account the architecture established for the extension.		
Observation (category 3); Archival data	Magic Draw; MoWebA specification; Internet	PIM metamodel modified; PIM modification report.
3. Metamodel and UML profile: definition of architecture metamodels, considering the Brambilla et al. framework [25]. 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; Explanatory report
4. PIM-ASM mapping.		
Observation (category 3); Archival data	Magic Draw	Mapping rules identification
5. PIM-ASM transformation rules definition with ATL.		
Observation (category 3); Archival data	ATL tool	Transformation rules in ATL; Explanatory report
6. Definition of transformation rules from the ASM model to the final code. Generation of source code.		

Table 4.14: Summary of data collected during the experience.

Data Collection method	**Materials**	**Outputs**
Observation (category 3); Archival data	Acceleo tool	Transformation definition in Acceleo; Documentation
7. Proof of Concept development.		
Focus group; Open and semi-structured interviews	MagicDraw; ATL tool; Acceleo	PIM Models; ASM models, code generated
8. Validation of each extension.		
8.1. MoWebA4RIA validation.		
Work sessions; Focus group; Open and semi-structured interviews	MagicDraw; ATL tool; Acceleo; timesheet; questonaries;	PIM models; ASM models; code generated; timesheets; ASQ questionnaires answers
8.2. MoWebAMobile4FC validation.		
Observation (category 3); Archival data	MagicDraw; Acceleo; WebRatio; IDEs; timesheets;	PIM models; WebRatio Models; code generated; code developed
8.3. MoWebAMobile4Persistence validation.		
Work sessions; Focus group; Open and semi-structured interviews	MagicDraw; ATL tool; Acceleo; IDEs; timesheets; questonaries;	PIM models; ASM models; code generated; timesheets; ASQ and SUS questionnaires answers

4.3.4 Data Analysis

The analysis carried out consists of a qualitative analysis and judgment for each research questions presented in section 4.3.1.2, based on the data collected and the monitoring throughout the process. We begin the section with table 4.15, which presents a summary of the activities carried out and the achievements reached for each of the extensions.

Table 4.15: Resume of activities of ASM extensions development

Task	**RIA**	**Mobile for Function and the Cloud**	**Mobile for Persistence**
Scope for the extensions	client data, client business logic and asynchronous communication between client and server	light-data, load-image, download-files, upload-files	data persistence mechanisms: databases, files, and key-value pairs providers: external, internal, other applications
Modifications to the PIM meta-model	Logic Diagram: *staticObject* and *value* attribute. Content Diagram: *requestType* attribute of Form element, *name* attribute of List element, *submitButton* as a specialization of Button	none	Entity Diagram: *EntityProperty* element and *Datatype*
PIM Diagrams extended	Logic and Content Diagrams	Logic Diagram	Entity Diagram

Task	**RIA**	**Mobile for Function and the Cloud**	**Mobile for Persistence**
PIM elements extended into the ASM metamodel	6 elements: *valueObject* and *staticObject* into the Client Data; *table*, *form* and *textInput* into the Client Business Logic; and *service* in the Asynchronous Communication)	3 elements: *valueObject*, *service* and *TProcess*	2 elements: *persistenceEntity* and *persistenceEntityProperty*
Number of elements defined in the ASM metamodel	11 elements: 3 for Client Data, 4 for Client Business Logic and 4 for Asynchronous Communication	18 elements: for Mobile Cloud Communications	17 elements: 5 for data persistence y 12 for data providers
PIM-ASM Mapping	5 direct mapping elements from inheritance and 1 mapping from associations	3 direct mapping elements from inheritance	2 direct mapping elements from inheritance
PIM-ASM Transformation	ATL with refining mode and 2 configuration files	manually	ATL with refining mode and 1 configuration file
M2T Code Generation	**M2T Tool:** Acceleo **Final code:** HTML5, Javascript, jQuery, jQuery UI, and Datatables libraries	**M2T Tool:** Acceleo **Final Code:** Java (Android), Swift (iOS), Node.js , Docker	**M2T Tool:** Acceleo **Final Code:** Java (Android) and C# (Windows), GUI code in XML (Android) and XAML (Windows)

Task	**RIA**	**Mobile for Function and the Cloud**	**Mobile for Persistence**
Extension's Validation	**Participants:** 10 students of last year of career **Type:** structured guided development **Data collected:** project documentation, work sessions' timesheets and questionnaires	**Participants:** 6 students and 5 mobile developers **Type:** Comparative Study **Data collected:** project documentation	**Participants:** 1 student of last year of career **Type:** structured guided development **Data Collected:** project documentation, work sessions' timesheets and questionnaires

The following sections discuss each of the research questions.

4.3.4.1 RQ1: To what extent the evolution and adaptability of the extension mechanism proposed by MoWebA to incorporate new architectures are achieved?

From the experience made to the three different extensions we can conduct the following analysis:

- All three extensions have been successfully developed, starting from the same PIM metamodel, with a reduced number of adjustments to the already defined at PIM level. However, we had to make some decisions for the development of the extensions, which have involved some minor adjustments to the PIM metamodel, and changes in the environments to be used for the definition of the transformation rules (e.g. the ATL tool used for M2M transformations explained in section 4.3.2.4).

- The MoWebA proposal for adaptation to other architectures can be done by users with knowledge of MDD and use of standards. This observation is made considering the fact that the extensions have been developed by undergraduate students whose knowledge and experiences have been the fundamentals, tools and standards of MDD. They have learned the MoWebA proposal during their degree studies and as part of the work carried out in the project.

- One aspect to consider is that the greater the number of elements of the ASM metamodel that are not related to elements of the PIM metamodel (either through inheritance or association), the degree of PIM-ASM automation decreases. Therefore, more effort is required in manual adjustments to the ASM model prior to code generation. However, the configuration files have allowed to introduce design decisions prior to the transformation processes and thus increased the degree of automation.

- Finally, regarding the number of elements added in the ASM metamodels, we consider that it corresponds to a reasonable and manageable amount to include new concepts to a methodological proposal (11 for RIA, 18 for Cloud Communications and 17 for persistence). On the other hand, the fact of considering the modeling of a specific architecture (ASM) in a different level of abstraction is not mandatory and should be included only in case the architecture must be specified in the modeling process. Doing so, the PIM remains independent of the proposed extensions.

4.3.4.2 RQ2: How independent is the MoWebA PIM for use in the modeling stage prior to ASM?

The extensions made to MoWebA have involved revisions to the elements defined in the PIM metamodel and the definition of the ASM metamodel for a specific architecture. In some cases these revisions implied a more detailed specification of certain existing elements, and in other cases the inclusion of new no contemplated elements into the PIM. In all three cases the extensions were made based on diagrams already defined in the PIM, since each extension aims to consider aspects of an architecture that includes one or more layers of an application. It should be noted that for the three metamodels defined, there are elements that correspond to specializations of elements from the PIM (e.g. *richForm* is a specialization of *form* in the RIA metamodel, *CloudService* as a specialization of *Service* in the FC extension, or *persistenceEntity* as a specialization of *entity* in the persistence ASM).

Table 4.15 shows some details of how the extensions were made. From this Table we have conducted the following analysis:

- The first has to do with the extended diagrams. In this regard, it should be noted that the three extensions have been based on diagrams already existing in the PIM, which leads us to believe that the PIM has the necessary independent modeling elements to carry out these extensions.
- Another aspect worth mentioning is that specializations of existing concepts have also been made in the three extensions, i.e., elements have been redefined or specialized in order to orient the elements towards specific architectures. It gives us evidence that the generic or independent concepts are included in the PIM metamodel.
- In two of the three extensions, new elements have been proposed or, existing elements have been redefined in the PIM metamodels. This is due to the fact that in addition to being required for the extensions, they have been considered as generic concepts, i.e., independent of the architecture, so we decided to include them in the PIM metamodels (e.g., the *staticObjects* identified in RIA, or *EntityProperty* added to the PIM during the persistence extension). It should be noted, however, that the number of new elements included in the PIM has been minimal (2 classes and 3 attributes in RIA, no elements in Cloud, 1 class and 1 enumeration in Persistence), which again confirms the fact that the PIM is independent enough to be extended, but at the same time has the necessary elements to PIM modeling.

4.3.4.3 RQ3: To what extent the automation can be obtained with MoWebA's model-to-model and model-to-code transformation rules?

ASM-PIM transformation has been completed for ASM elements derived from PIM by inheritance.

In cases where additional information has been required from the user to include design decisions, the configuration files have allowed to enter such information prior to performing the transformations, thus improving the degree of automation.

The PIM-ASM transformation experiences described in [91] [94] have shown that although it is possible to include manual adjustments to the ASM models generated from the PIM, the configuration files have helped to reduce these percentages considerably.

It should also be noted that the following manual adjustments have been made in the experiences of using the extensions:

- The ASM models obtained automatically from the PIM in order to include certain elements that could not be derived from the PIM.
- To modify the final code related to the user interface of the applications and other adjustments for generation of the application from the code generated in Acceleo.
- In addition, intentional modifications have been made at the final code level and at the modeling level for validation purposes.

4.3.4.4 RQ4: To what extent the user's satisfaction is achieved with the use of the MoWebA proposal?

Some of the most relevant results of user's satisfactions in the use of the MoWebA4RIA (A) and MoWebAMobile4Persistence (B) extensions are:

- The PIM modeling process of the A presented an ASQ score of 2.87 and B 2.89. In both results, we can see that a satisfactory success rate was obtained. Then, the score obtained from the ASQ questionnaire reflects a good level of satisfaction from the modelers using the proposed modeling process. Furthermore, looking at the quantitative analysis presented in Table 4.9 of B, the average success rate was 82%, which we consider as a notable success rate.

- The PIM-ASM transformation process of A presented an ASQ score of 2.70. With this score, we can derive that very good levels of satisfaction were obtained in the PIM-ASM transformation process.

- The code generation process of A presented an ASQ score of 2.87 and B 1.94. From these result, we can note that the success rate in A is just as satisfactory as the success rate obtained for the modeling process. However, as mentioned in section 4.3.2.8.1 we believe that the success rate for the code generation process could have obtained a better score, since this was directly affected by the input model used. In B, all the modelers were able to complete the tasks, generating the code application satisfactorily in Acceleo. Nevertheless, errors were detected during the code generation. In total, 33% of modelers presented errors in this process due to imperfections found in their models from the modeling process.

- The B extension presented a perception of satisfaction of 50 in the SUS, with a standard deviation of 16.89. Converting this score to Sauro and Lewis percentile rank [125], we obtain a value of 13%. This result is below average. We noticed some doubts in the use of the proposed approach from the modelers.

- The process of generating the application of B presented an ASQ score of 2.20. Although, with an almost perfect success rate (see Table 4.11, the drawback with the use of the IDE got our attention because, despite the work experience the developers had, the tasks of creating the project and then adapting the generated code to it took longer than expected. We noticed that the developers were very engaged with the experience of validation, willing to finish the tasks, participatory, and striving to achieve good performance. Furthermore, in general, we perceived a good level of satisfaction during the application generation process (also reflected in the ASQ results).

- The process of modifying the generated application with manual adjustment in B presented an ASQ score of 3.0. Analyzing the results of the first group mentioned above, and regarding efficiency, we can see a lower performance in comparison with previous processes. Only 33% of developers could fully complete the requested modification. The other two developers reached only 50% of the development. The previous discomfort is reflected in the level of satisfaction the corresponding ASQ questionnaire (see Table 4.12). The developers who made the requested modifications to the code highlighted the readability and proper structure of the generated code.

- We observed the developers using B extensions for modifications, were more confident and satisfied with the experience. Unlike the score obtained in the manual development, the ASQ score for this group reflects a very good level of satisfaction (1.67). We emphasize that the changes to the model did not require any extra manual modification, highlighting the expressiveness of B, in the context of the proposed task.

A common issue we found during the modeling process with MoWebA, is related to the modeling tool. In this sense, the fact of using a tool that is not MoWebA's own, but a standard one, has generated some additional difficulties during the modeling process. This leads us to believe that the degree of satisfaction could even be improved if a specific modeling tool is developed for the proposal.

4.3.5 Threat to Validity

Some aspects that may have attempted against the validity of this evaluation and how they were mitigated, are discussed below.

Regarding **internal validity**, which has to do with the degree of confidence in a cause-effect relationship between the factors of interest and the observed results, it can be said that:

- The validation experience of MoWebA4RIA was carried out with students, who all have the same level of experience in terms of an MDD process, thus avoiding participants with unbalanced knowledge. To avoid plagiarism the students were supervised and communication between pairs was forbidden.
- In MoWebAMobile4FC we did a comparative study with one participant, supervised by MDD experts.

- In MoWebAMobile4Persistence two well-differentiated groups were formed, balancing the level and area of experience. First, the student/modelers (with sufficient knowledge in modeling with MoWebA). Second, the mobile developers (with an average experience of 1.5 years in the industry). The experience was carried out during Software Engineering I class time to avoid the absenteeism of the students/modelers during the sessions. The performed tasks were considered as possible topics for the Software Engineering I exam. Regarding the developers, we were in constant communication with them, organizing a schedule in which everyone could attend. Finally, to avoid plagiarism in the experience, each participant worked individually in different machines and was supervised to avoid communicating with each other.

The **external validity**, which represents the degree to which the results achieved can be generalized, is affected by the fact that:

- In MoWebA4RIA validation the experience was developed with students, which does not allow us to ensure that the results can be generalized to a target population corresponding to the web application developers that use MDD. In addition, the number of participants involved in this experience was ten, and although it does not correspond to a sufficient amount for statistical purposes, it reaches at least to get first judgments. In addition, the developed case consisted of a limited case, however, this case contemplates the development of a RIA with all the features that have been mentioned in this work.

- In MoWebAMobile4Persistence this aspect could be affected by the number of participants involved (six students/modelers and five mobile developers). Although this number is not statistically relevant, it is appropriate to issue an initial evaluation and initial judgments. Besides, the development case was simple, but not far from the requirements that an industry case could imply. This case contemplated the development of a mobile application taking into account all aspects covered by the proposal.

- In general, the number of participants involved in the experience (4 participant for extension definition, 10 participants for MoWebA4RIA validation, 1 participant for MoWebAMobiel4FC and 11 participants for MoWebAMobile4Persistence). Although this number is not statistically relevant, it is appropriate to issue an initial evaluation and judgments. More formal validations should be carried out later on (such as experiments and case studies) with a more significant number of participants, to obtain meaningful and precise conclusions.

Regarding the **validity** of the construct, which reflects the extent to which the measures have been adapted to what the researcher has in mind and what is being investigated:

- In MoWebA4RIA and MoWebaMobile4Persistence validations we selected data that are normally used to measure quality aspects. We also used standard questionnaires, which are considered reliable and valid [69; 124] to evaluate the students' perceptions without the intervention of the researcher: SUS [12] and ASQ [70].
- In MoWebAMobile4Persistence the measurement of portability could have been affected by not having standard questionnaires or accurate metrics, although we consider it appropriate to issue the first approximations to perform more formal validations in future works.

In relation to **reliability**, which indicates the dependence of the data and its analysis on a specific researcher and the ability to replicate the same study and obtain the same results, we respected the literalness of the data obtained in the documentation, in the measured times and in the questionnaires, avoiding the introduction of biases through interpretation.

4.4 Discussion and Summary of the Chapter

The contributions from the adoption of MoWebA in academic and real contexts for Web environments presented in section 4.1 have allowed to identify strengths and weaknesses of the PIM proposal, and to verify that the proposed notation covers the needs of different domains.

The experience presented in section 4.2 has allowed us to determine how feasible it is to adapt the proposal to other architectures, with an analysis that sought to answer the following questions: Can the same PIM model be used for different architectures?; Is it possible to specify clear limits between platform independent models (PIM) and architectural specific models (ASM)?; How does an architectural specific model facilitate the transformation rules definition?. In the experiences with students presented in this section, they have defined extensions for three different architectures and have verified the possibility of obtaining ASM models for these three architectures from the same PIM (ACA system). With this experience, we have proven that the same PIM model can evolve to different architectures through their specific ASM.

The results of the validations presented in section 4.3 consisted of extending MoWebA to three specific environments. This experience have allowed to carry out three complete extensions. Two of the extensions have include the PIM-ASM transformation rules, with configuration files. In one of the extensions, the PIM-ASM transformation was partially performed manually (MoWebAMobile4FC). In these experiences we could analysed the grade of adaptability of MoWebA and automation PIM-ASM, as well as the grade of independence of the PIM metamodel. We have also conducted some user's satisfaction experiences with modelers and developers.

We also carried out a validation analysis in academic environments and with professionals, as well as a comparative study with respect to another mature tool such as WebRatio and the traditional development.

5 Conclusion and Future Works

This chapter discusses the findings and contributions of the thesis, points out limitations of the current work, and also outlines directions for future research. In section 5.1 we present the main contributions of this PhD Thesis. Section 5.2 lists the different publications in Journals and Conferences as a result of the thesis. Section 5.3 mentions the collaborations carried out with researchers from other universities within the framework of research projects. The chapter ends with section 5.4 which sketches ongoing work and future directions.

5.1 Contributions

Various contributions can be highlighted as evidence of achieving the research goals, as well as answering the established research questions.

The main contributions are presented below:

1. We have conducted a state-of-the-art study related to Model Driven Engineering, Model Driven Web Engineering and Development Methods for Web applications. We also analysed the characteristics of Modern Web applications, and the mechanism in which Web methodologies contemplate aspects related to adaptation. We reviewed a series of academic and industry evidences on the adoption of MDE. It is worth noting that as a result of these reviews various publications has been presented in CLEI EJ Journal and Journal of System and Software [120] [22] [91] [94]. In addition, some papers were presented at conferences CLEI and CIbSE [92] [121].

2. We have identified three concerns that methodological approaches need to take into account for the development of Web applications. These concerns are related to navigation as a starting point for modeling, the use of

standards in modeling process, and the need for evolution of methodological approaches to better adapt to new environments. Part of this work has been published at the International Journal of Web Engineering and Technology [57].

3. We have proposed the MoWebA approach, defining the methodological aspects, phases, notation, metamodels, UML profiles, and transformation rules. The different results were presented in various international journals and conferences [54] [56] [57] [94] [91] [121].

4. We carried out a validation experience of MoWebA's ASM phase in an academic environment that was presented at RCIS International Conference [56].

5. We have performed the validation of the MoWebA extension mechanism by applying the ASM definition process to other environments (RIA, mobile for persistence and mobile for cloud). The results of these extensions experiences were presented at CIbSE 2018, CLEI EJ 2018, JSS 2020 [120] [91] [94].

6. We have accomplished a series of proof-of-concept experiences with MoWebA in academic and real environments: social network system, budget execution system, e-learning system, survey system, aquatic bird platform, among others. The overall results of these experiences have been presented in the IJWET 2016 Journal [57].

5.2 Publications

The results that were obtained during the development of this PhD thesis, especially those related to the development phase and validation experiences, have led to several publications. Despite other older publications achieved as part of the preliminary studies, the more relevant publications are presented in table 5.1.

Table 5.1: Relevant publications within this PhD thesis.

Authors	**Type**	**Title**	**Conference or Journal**	**Year**
Manuel Núñez, Daniel Bonhaure, Magalí González, Luca Cernuzzi [94]	Journal	A model-driven approach for the development of native mobile applications focusing on the data layer	Journal of System and Software	2020
Guido Nuñez, Daniel Bonhaure, Magalí González, Nathalie Aquino, Luca Cernuzzi [91]	Journal	A Model-Driven Approach to develop Rich Web Applications	Clei Electronic Journal	2018
Emanuel Sanchiz, Magalí González, Nathalie Aquino, Luca Cernuzzi [121]	Conference	MoWebA Mobile: Modeling and Generation of the Communication of Mobile Apps with their Functions in the Cloud	CIbSE	2018
Daniel Bonhaure, Magalí González, Nathalie Aquino, Luca Cernuzzi, Claudia Pons [22]	Journal	Exploring Model-to-Model Transformations for RIA Architectures by means of a Systematic Mapping Study	Clei Electronic Journal	2017
Emanuel Sanchiz, Magalí González, Nathalie Aquino, Luca Cernuzzi [120]	Journal	Development of Mobile Applications with Functions in the Cloud through the Model Driven Approach: A Systematic Mapping Study	Clei Electronic Journal	2017
Guido Nuñez, Magalí González, Nathalie Aquino, Luca Cernuzzi [93]	Conference	A model-driven approach to develop rich web applications	CLEI	2017

Authors	**Type**	**Title**	**Conference or Journal**	**Year**
Magalí González, Luca Cernuzzi, Oscar Pastor [57]	Journal	A navigational role-centric model oriented web approach - MoWebA	Internationa Journal of Web Engineering and Technology	2016
Iván López, Magalí González, Nathalie Aquino, Luca Cernuzzi [72]	Conference	Una Propuesta Basada en Model Driven Architecture para el Soporte de Rich Internet Applications	CIbSE	2016
Magalí González, Luca Cernuzzi, Nathalie Aquino, Oscar Pastor [56]	Conference	Developing web applications for different architectures: The MoWebA approach	RCIS	2016

5.3 Research Collaborations

One of the most important collaboration was with researchers from the Department of Electronic and Informatics Engineering (DEI) of Catholic University of Asunción (Paraguay) and researchers of the LIFIA Lab from La Plata University (Argentina). Specifically, we jointly collaborated in the international research project that has driven most of the validation experiences: "Mejorando el Proceso de Desarrollo de Software: Una propuesta basada en MDD"[1], grant 14-INV-056 of CONACYT (Consejo Nacional de Ciencias y Tecnología), Paraguay. The total number of participants in the project were 11 (five undergraduate students, one master student, two PhD students and three MDD experts). The experts where from Polytechnic University of Valencia (Spain), La Plata University (Argentina) and Catholic University of Asunción (Paraguay).

[1]`https://www.dei.uc.edu.py/proyectos/mddplus/`

5.4 Future Work

Along the thesis we identified several lines of research and opportunities to improve and extend the proposal. The following summarizes the research directions that are planned for the near future. The main goal of this future work will be to overcome some of the limitations of the work that has been developed so far.

1. In the modeling dimension, it is relevant the tools' support. Thus, we identified opportunities for: i) the development of modeling tool to simplify the use of MoWebA for PIM and ASM modeling; and ii) Revision and improvement of Model-to-Model and Model-to-Code transformation rules.

2. Spending more efforts with regards the need for adaptation of the methodologies, we envision the extensions to other environments and architectures.

3. Finally, usability experiences with MoWebA and further validation of the proposal (e.g., formal experiments or case studies) in industrial or commercial contexts are necessary steps to consolidate the approach.

Bibliography

[1] Android Studio | Android Developers. https://developer.android.com/studio. (Accessed on 08/16/2019).

[2] Extensible Markup Language (XML). https://www.w3.org/XML/. (Accessed on 08/16/2019).

[3] MagicDraw. https://www.nomagic.com/products/magicdraw. (Accessed on 02/02/2019).

[4] Most Widely Deployed SQL Database Engine. https://www.sqlite.org/mostdeployed.html. (Accessed on 08/03/2019).

[5] Visual Studio IDE, code editor, Azure DevOps and App Center - Visual Studio. https://visualstudio.microsoft.com. (Accessed on 08/16/2019).

[6] XAML overview (WPF) | Microsoft Docs. https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-overview-wpf. (Accessed on 08/16/2019).

[7] Roberto Acerbis, Aldo Bongio, Marco Brambilla, and Stefano Butti. Model-Driven Development Based on OMG's IFML with WebRatio Web and Mobile Platform. In Philipp Cimiano, Flavius Frasincar, Geert-Jan Houben, and Daniel Schwabe, editors, *Engineering the Web in the Big Data Era - 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings*, volume 9114 of *Lecture Notes in Computer Science*, pages 605–608. Springer, 2015.

[8] Roberto Acerbis, Aldo Bongio, Marco Brambilla, and Stefano Butti. Model-Driven Development of Cross-Platform Mobile Applications with Web Ratio and IFML. In *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*, pages 170–171, May 2015.

[9] Roberto Acerbis, Aldo Bongio, Marco Brambilla, Massimo Tisi, Stefano Ceri, and Emanuele Tosetti. Developing ebusiness solutions with a model driven approach: The case of acer EMEA. In Luciano Baresi, Piero Fraternali, and Geert-Jan Houben, editors, *Web Engineering, 7th International Conference, ICWE 2007, Como, Italy, July 16-20, 2007, Proceedings*, volume 4607 of *Lecture Notes in Computer Science*, pages 539–544. Springer, 2007.

[10] Silvia Alcaraz, Magalí González, and Luca Cernuzzi. Adaptatividad y adaptabilidad en el modelado de usuarios para aplicaciones web. In *Memorias de la XXXVII Conferencia Lationamericana de Informática*, 2011.

[11] 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.

[12] Aaron Bangor, Philip Kortum, and James Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of usability studies*, 4(3):114–123, 2009.

[13] Luciano Baresi, Sebastiano Colazzo, Luca Mainetti, and Sandro Morasca. W2000: A modelling notation for complex web applications. In Emilia Mendes and Nile Mosley, editors, *Web Engineering*, pages 335–364. Springer, 2006.

[14] Scott Barnett, Iman Avazpour, Rajesh Vasa, and John C. Grundy. A multi-view framework for generating mobile apps. In Zhen Li, Claudia Ermel, and Scott D. Fleming, editors, *2015 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2015, Atlanta, GA, USA, October 18-22, 2015*, pages 305–306. IEEE Computer Society, 2015.

[15] Scott Barnett, Rajesh Vasa, and John Grundy. Bootstrapping Mobile App Development. In Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum, editors, *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2*, pages 657–660. IEEE Computer Society, 2015.

[16] Victor R. Basili, Gianluigi Caldiera, and Dieter H. Rombach. *The Goal Question Metric Approach*, volume I. John Wiley & Sons, 1994.

[17] H. Benouda, M. Azizi, M. Moussaoui, and R. Esbai. Automatic code generation within MDA approach for cross-platform mobiles apps. In *2017 First International Conference on Embedded Distributed Systems (EDiS)*, pages 1–5, Dec 2017.

[18] Mario Luca Bernardi, Marta Cimitile, and Damiano Distante. Web applications design recovery and evolution with RE-UWA. *J. Softw. Evol. Process.*, 25(8):789–814, 2013.

[19] Mario Luca Bernardi, Giuseppe Antonio Di Lucca, and Damiano Distante. Model-driven fast prototyping of rias: From conceptual models to running applications. In *ICACCI*, pages 250–258. IEEE, 2014.

[20] Mario Luca Bernardi, Giuseppe Antonio Di Lucca, and Damiano Distante. Model-driven fast prototyping of rias: From conceptual models to running applications. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 250–258, 2014.

[21] Carlo Bernaschina, Sara Comai, and Piero Fraternali. Formal semantics of OMG's Interaction Flow Modeling Language (IFML) for mobile and rich-client application model driven development. *Journal of Systems and Software*, 137:239–260, 2018.

[22] Daniel Bonhaure, Magalí González, Nathalie Aquino, Luca Cernuzzi, and Claudia Pons. Exploring model-to-model transformations for RIA architectures by means of a systematic mapping study. *CLEI Electron. J.*, 20(3), 2017.

[23] Francis Bordeleau, Grischa Liebel, Alexander Raschke, Gerald Stieglbauer, and Matthias Tichy. Challenges and research directions for successfully applying MBE tools in practice. In Loli Burgueño, Jonathan Corley, Nelly Bencomo, Peter J. Clarke, Philippe Collet, Michalis Famelis, Sudipto Ghosh, Martin Gogolla, Joel Greenyer, Esther Guerra, Sahar Kokaly, Alfonso Pierantonio, Julia Rubin, and Davide Di Ruscio, editors, *Proceed-*

ings of MODELS 2017 Satellite Event: Workshops (ModComp, ME, EXE, COMMitMDE, MRT, MULTI, GEMOC, MoDeVVa, MDETools, FlexMDE, MDEbug), Posters, Doctoral Symposium, Educator Symposium, ACM Student Research Competition, and Tools and Demonstrations co-located with ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), Austin, TX, USA, September, 17, 2017, volume 2019 of *CEUR Workshop Proceedings*, pages 338–343. CEUR-WS.org, 2017.

[24] Alessandro Bozzon, Sara Comai, Piero Fraternali, and Giovanni Toffetti Carughi. Capturing RIA concepts in a web modeling language. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 907–908. ACM, 2006.

[25] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice, Second Edition.* Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2017.

[26] Marco Brambilla, Sara Comai, Piero Fraternali, and Maristella Matera. Designing web applications with webml and webratio. In *Web Engineering: Modelling and Implementing Web Applications*, pages 221–261. Springer, 2008.

[27] Marco Brambilla and Piero Fraternali. Large-scale model-driven engineering of web user interaction: The webml and webratio experience. *Science of Computer Programming*, 89:71–87, 2014. Special issue on Success Stories in Model Driven Engineering.

[28] Marco Brambilla, Andrea Mauri, and Eric Umuhoza. Extending the interaction flow modeling language (IFML) for model driven development of mobile applications front end. In *MobiWIS*, volume 8640 of *Lecture Notes in Computer Science*, pages 176–191. Springer, 2014.

[29] Marco Brambilla, Andrea Mauri, and Eric Umuhoza. Extending the interaction flow modeling language (IFML) for model driven development of

mobile applications front end. In *Mobile Web Information Systems - 11th International Conference, MobiWIS 2014, Barcelona, Spain, August 27-29, 2014. Proceedings*, pages 176–191. Springer International Publishing, 2014.

[30] Antonio Bucchiarone, Jordi Cabot, Richard F. Paige, and Alfonso Pierantonio. Grand challenges in model-driven engineering: an analysis of the state of the research. *Softw. Syst. Model.*, 19(1):5–13, 2020.

[31] Antonio Bucchiarone, Federico Ciccozzi, Leen Lambers, Alfonso Pierantonio, Matthias Tichy, Massimo Tisi, Andreas Wortmann, and Vadim Zaytsev. What is the future of modeling? *IEEE Softw.*, 38(2):119–127, 2021.

[32] Loli Burgueño, Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sébastien Mosser, Richard F. Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabriele Taentzer, Antonio Vallecillo, and Manuel Wimmer. Contents for a model-based software engineering body of knowledge. *Softw. Syst. Model.*, 18(6):3193–3205, 2019.

[33] Marianne Busch, Nora Koch, and Santiago Suppan. Modeling security features of web applications. In *Engineering Secure Future Internet Services and Systems*, volume 8431 of *Lecture Notes in Computer Science*, pages 119–139. Springer, 2014.

[34] Jordi Cabot, Robert Clarisó, Marco Brambilla, and Sébastien Gérard. Cognifying model-driven software engineering. In Martina Seidl and Steffen Zschaler, editors, *Software Technologies: Applications and Foundations - STAF 2017 Collocated Workshops, Marburg, Germany, July 17-21, 2017, Revised Selected Papers*, volume 10748 of *Lecture Notes in Computer Science*, pages 154–160. Springer, 2017.

[35] Sven Casteleyn, William Van Woensel, Kees van der Sluijs, and Geert-Jan Houben. Aspect-oriented adaptation specification in web information systems: a semantics-based approach. *New Rev. Hypermedia Multim.*, 15(1):39–71, 2009.

[36] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web modeling language (webml): a modeling language for designing web sites. *Comput. Networks*, 33(1-6):137–157, 2000.

[37] T. Channonthawat and Y. Limpiyakorn. Model Driven Development of Android Application Prototypes from Windows Navigation Diagrams. In *2016 International Conference on Software Networking (ICSN)*, pages 1–4, May 2016.

[38] Nacha Chondamrongkul and Nacha Chondamrongkul. Model-driven framework to support evolution of mobile applications in multi-cloud environments. *International Journal of Pervasive Computing and Communications*, 12(3):332–351, 2016.

[39] Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches. *IBM Syst. J.*, 45(3):621–646, 2006.

[40] Alberto Rodrigues da Silva. Model-driven engineering: A survey supported by the unified conceptual model. *Comput. Lang. Syst. Struct.*, 43:139–155, 2015.

[41] Yogesh Deshpande, San Murugesan, Athula Ginige, Steve Hansen, Daniel Schwabe, Martin Gaedke, and Bebo White. Web engineering. *J. Web Eng.*, 1(1):3–17, 2002.

[42] José Maria Duarte, Magalí González, Luca Cernuzzi, and Oscar Pastor. Evaluación del desarrollo de software mediante una herramienta MDA: un caso de estudio. In Maria Lencastre, João Falcão e Cunha, and Antonio Valecillo, editors, *Memorias de la XI Conferencia Iberoamericana de Software Engineering (CIbSE 2008), Recife, Pernambuco, Brasil, February 13-17, 2008*, pages 99–112, 2008.

[43] Rob Dunie, W Roy Schulte, M Cantara, and Marc Kerremans. Magic quadrant for intelligent business process management suites. *Gartner Inc*, 2019.

[44] David W. Embley, Stephen W. Liddle, and Oscar Pastor. Conceptual-model programming: A manifesto. In *Handbook of Conceptual Modeling*, pages 3–16. Springer, 2011.

[45] Siamak Farshidi, Slinger Jansen, and Sven Fortuin. Model-driven development platform selection: four industry case studies. *Software and Systems Modeling*, pages 1–27, 2021.

[46] Rita Francese, Michele Risi, Giuseppe Scanniello, and Genoveffa Tortora. Model-driven development for multi-platform mobile applications. In Pekka Abrahamsson, Luis Corral, Markku Oivo, and Barbara Russo, editors, *Product-Focused Software Process Improvement - 16th International Conference, PROFES 2015, Bolzano, Italy, December 2-4, 2015, Proceedings*, volume 9459 of *Lecture Notes in Computer Science*, pages 61–67. Springer, 2015.

[47] Piero Fraternali, Sara Comai, Alessandro Bozzon, and Giovanni Toffetti Carughi. Engineering rich internet applications with a model-driven approach. *ACM Trans. Web*, 4(2):7:1–7:47, 2010.

[48] Fabiano Freitas and Paulo Henrique Mendes Maia. JustModeling: An MDE Approach to Develop Android Business Applications. In *VI Brazilian Symposium on Computing Systems Engineering, SBESC 2026, João Pessoa, Paraíba, Brazil, November 1-4, 2016*, pages 48–55. IEEE Computer Society, 2016.

[49] Félix Óscar García Rubio, Juan Manuel Vara Mesa, and Cristina Vicente Chicote. *Desarrollo de Software Dirigido por Modelos: Conceptos, Métodos y Herramientas.* Ra-Ma Editorial, 2013.

[50] Sebastian Geiger-Prat, Beatriz Marín, Sergio España, and Giovanni Giachetti. A GUI modeling language for mobile applications. In *9th IEEE International Conference on Research Challenges in Information Science, RCIS 2015, Athens, Greece, May 13-15, 2015*, pages 76–87. IEEE, 2015.

[51] João Paulo Barbosa Glória and Adriana Pereira de Medeiros. xoohdm: Integrated development environment for creating and executing models in web applications design. In *SBES*, pages 273–278. ACM, 2019.

[52] Jaime Gómez, Alejandro Bia, and Antonio Párraga. Tool support for model-driven development of web applications. *Int. J. Inf. Technol. Web Eng.*, 2(3):65–78, 2007.

[53] Jaime Gómez, Cristina Cachero, and Oscar Pastor. Extending a conceptual modelling approach to web application design. In Benkt Wangler and Lars Bergman, editors, *Advanced Information Systems Engineering, 12th International Conference CAiSE 2000, Stockholm, Sweden, June 5-9, 2000, Proceedings*, volume 1789 of *Lecture Notes in Computer Science*, pages 79–93. Springer, 2000.

[54] Magalí González, Juan Bareiro, Rodney Rodriguez, Luca Cernuzzi, and Oscar Pastor. El enfoque navegacional para el desarrollo de sistemas web con mda. In *Memorias de la XIII Congreso Iberoamericano en Software Engineering - CIbSE*, 2010.

[55] Magalí González, Jorge Casariego, Juan José Bareiro, Luca Cernuzzi, and Oscar Pastor. A MDA approach for navigational and user perspectives. *CLEI Electron. J.*, 14(1), 2011.

[56] Magalí González, Luca Cernuzzi, Nathalie Aquino, and Oscar Pastor. Developing web applications for different architectures: The moweba approach. In *Tenth IEEE International Conference on Research Challenges in Information Science, RCIS 2016, Grenoble, France, June 1-3, 2016*, pages 1–11. IEEE, 2016.

[57] Magalí González, Luca Cernuzzi, and Oscar Pastor. A navigational role-centric model oriented web approach - MoWebA. *Int. J. Web Eng. Technol.*, 11(1):29–67, 2016.

[58] Henning Heitkötter, Sebastian Hanschke, and Tim Ar Majchrzak. Evaluating cross-platform development approaches for mobile applications. In *Web information systems and technologies*, pages 120–138. Springer, 2013.

[59] Henning Heitkötter, Herbert Kuchen, and Tim A. Majchrzak. Extending a model-driven cross-platform development approach for business apps. *Sci. Comput. Program.*, 97:31–36, 2015.

[60] Henning Heitkötter, Tim A. Majchrzak, and Herbert Kuchen. Cross-platform model-driven development of mobile applications with md^2. In Sung Y. Shin and José Carlos Maldonado, editors, *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*, pages 526–533. ACM, 2013.

[61] Frédéric Jouault and Dennis Wagelaar. ATL EMF Transformation Virtual Machine (research VM) - Invoking native Java methods. `https://wiki.eclipse.org/ATL/EMFTVM#Invoking_native_Java_methods`, 2017. Accessed at 19-02-2017.

[62] Frédéric Jouault and Dennis Wagelaar. ATL EMF Transformation Virtual Machine (research VM) - Performance. `https://wiki.eclipse.org/ATL/EMFTVM#Performance`, 2017. Accessed at 19-02-2017.

[63] William L. Kuechler Jr. and Vijay K. Vaishnavi. A framework for theory development in design science research: Multiple perspectives. *J. Assoc. Inf. Syst.*, 13(6):3, 2012.

[64] Minhyuk Ko, Yongjin Seo, Bup-Ki Min, Seung Hak Kuk, and Hyeon Soo Kim. Extending UML Meta-model for Android Application. In Huaikou Miao, Roger Y. Lee, Hongwei Zeng, and Jongmoon Baik, editors, *2012 IEEE/ACIS 11th International Conference on Computer and Information Science, Shanghai, China, May 30 - June 1, 2012*, pages 669–674. IEEE Computer Society, 2012.

[65] Nora Koch, Alexander Knapp, Gefei Zhang, and Hubert Baumeister. Uml-based web engineering - an approach based on standards. In Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina, editors, *Web Engineering: Modelling and Implementing Web Applications*, Human-Computer Interaction Series, pages 157–191. Springer, 2008.

[66] Nora Koch, Matthias Pigerl, Gefei Zhang, and Tatiana Morozova. Patterns for the model-based development of rias. In Martin Gaedke, Michael Grossniklaus, and Oscar Díaz, editors, *Web Engineering, 9th International*

Conference, ICWE 2009, San Sebastián, Spain, June 24-26, 2009, Proceedings, volume 5648 of *Lecture Notes in Computer Science*, pages 283–291. Springer, 2009.

[67] Dean Kramer, Tony Clark, and Samia Oussena. MobDSL: A Domain Specific Language for multiple mobile platform deployment. In *Proceedings of the 1st IEEE International Conference on Networked Embedded Systems for Enterprise Applications, NESEA 2010, November 25-26, 2010, Suzhou, China*, pages 1–7. IEEE Computer Society, 2010.

[68] 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.

[69] James R. Lewis. Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the asq. *ACM SIGCHI Bulletin*, 23(1):78–81, 1991.

[70] James R Lewis. Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the ASQ. *ACM SIGCHI Bulletin*, 23(1):78–81, 1991.

[71] Grischa Liebel, Nadja Marko, Matthias Tichy, Andrea Leitner, and Jörgen Hansson. Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. *Softw. Syst. Model.*, 17(1):91–113, 2018.

[72] Iván López, Magalí González, Nathalie Aquino, and Luca Cernuzzi. Una propuesta basada en model driven architecture para el soporte de rich internet applications. In Efraín R. Fonseca C., Cecilia Hinojosa, Juan Pablo Carvallo, Gleison Santos, Martín Solari, Marcos Kalinowski, Tayana Conte, José Ignacio Panach, Lidia López, Eduardo Kinder Almentero, Omar S. Gómez, Carlos Cares, and Sabrina Marczak, editors, *Proceedings of XIX Ibero-American Conference on Software Engineering, CIbSE 2016, Quito, Ecuador, April 27-29, 2016*, pages 25–38. Universidad de las Fuerzas Armadas ESPE, 2016.

[73] Esteban Robles Luna, Juan M. Sánchez-Begínes, José Matías Rivero, Leticia Morales-Trujillo, José Gonzalez Enríquez, and Gustavo Rossi. Challenges for the adoption of model-driven web engineering approaches in industry. *J. Web Eng.*, 17(3&4):183–205, 2018.

[74] González Magalí, Bareiro Juan, Rodriguez Rodney, and Cernuzzi Luca. Estudio comparativo de herramientas mda para el desarrollo de sistemas web. In *Memorias de la XXXV Conferencia Lationamericana de Informática*, 2009.

[75] Tim A. Majchrzak, Jan Ernsting, and Herbert Kuchen. Achieving Business Practicability of Model-Driven Cross-Platform Apps. *Open Journal of Information Systems (OJIS)*, 2(2):4–15, 2015.

[76] Esperanza Marcos, César J. Acuña, and Carlos E. Cuesta. Integrating software architecture into a MDA framework. In *EWSA*, volume 4344 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2006.

[77] Santiago Meliá, Jaime Gómez, and Nora Koch. Improving web design methods with architecture modeling. In Kurt Bauknecht, Birgit Pröll, and Hannes Werthner, editors, *E-Commerce and Web Technologies: 6th International Conference, EC-Web 2005, Copenhagen, Denmark, August 23-26, 2005, Proceedings*, volume 3590 of *Lecture Notes in Computer Science*, pages 53–64. Springer, 2005.

[78] Santiago Meliá, Jaime Gómez, Sandy Pérez, and Oscar Díaz. A model-driven development for gwt-based rich internet applications with OOH4RIA. In *ICWE*, pages 13–23. IEEE Computer Society, 2008.

[79] Stephen J. Mellor and Marc J. Balcer. *Executable UML - A Foundation for Model-Driven Architecture.* Addison Wesley object technology series. Addison-Wesley, 2002.

[80] Emilia Mendes and Nile Mosley, editors. *Web Engineering.* Springer, 2006.

[81] Tom Mens and Pieter Van Gorp. A taxonomy of model transformation. *Electron. Notes Theor. Comput. Sci.*, 152:125–142, 2006.

[82] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, December 2005.

[83] Tommi Mikkonen, Risto Pitkänen, and Mika Pussinen. On the role of architectural style in model driven development. In *EWSA*, volume 3047 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 2004.

[84] Bup-Ki Min, Minhyuk Ko, Yongjin Seo, Seunghak Kuk, and Hyeon Soo Kim. A UML metamodel for smart device application modeling based on Windows Phone 7 platform. In *TENCON 2011 - 2011 IEEE Region 10 Conference*, pages 201–205, Nov 2011.

[85] Nathalie Moreno, José Raúl Romero, and Antonio Vallecillo. *An Overview Of Model-Driven Web Engineering and the Mda*, pages 353–382. 01 2008.

[86] Siti Azreena Mubin, Azrul Jantan, Rusli Abdullah, and Azrina Kamaruddin. Uml-based conceptual design approach for modeling complex processes in web application. 11:4579–4585, 04 2016.

[87] Siti Azreena Mubin, Azrul Jantan, Rusli Abdullah, and Azrina Kamaruddin. Uml stereotypes for the development of process interaction-driven web applications. pages 81–88, 04 2016.

[88] San Murugesan, Yogesh Deshpande, Steve Hansen, and Athula Ginige. Web engineering: A new discipline for development of web-based systems. In San Murugesan and Yogesh Deshpande, editors, *Web Engineering, Software Engineering and Web Application Development*, volume 2016 of *Lecture Notes in Computer Science*, pages 3–13. Springer, 2001.

[89] Gunter Mussbacher, Daniel Amyot, Ruth Breu, Jean-Michel Bruel, Betty H. C. Cheng, Philippe Collet, Benoît Combemale, Robert B. France, Rogardt Heldal, James H. Hill, Jörg Kienzle, Matthias Schöttle, Friedrich Steimann, Dave R. Stikkolorum, and Jon Whittle. The relevance of model-driven engineering thirty years from now. In Jürgen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abrahão, and Emilio Insfrán, editors, *Model-Driven Engineering Languages and Systems - 17th International Confer-*

ence, MODELS 2014, Valencia, Spain, September 28 - October 3, 2014. Proceedings, volume 8767 of *Lecture Notes in Computer Science*, pages 183–200. Springer, 2014.

[90] Román Natalia, González Magalí, and Cernuzzi Luca. Análisis del enfoque mda: el caso de andromda. In *Memorias de la XXXVII Conferencia Lationamericana de Informática*, 2011.

[91] Guido Nuñez, Daniel Bonhaure, Magalí González, Nathalie Aquino, and Luca Cernuzzi. A model-driven approach to develop rich web applications. *CLEI Electron. J.*, 21(2), 2018.

[92] Guido Nuñez, Magalí González, Nathalie Aquino, and Luca Cernuzzi. A model-driven approach to develop rich web applications. In Héctor Monteverde and Rodrigo Santos, editors, *2017 XLIII Latin American Computer Conference, CLEI 2017, Córdoba, Argentina, September 4-8, 2017*, pages 1–10. IEEE, 2017.

[93] Guido Nuñez, Magalí González, and Luca Cernuzzi. Un enfoque MDD para el desarrollo de RIA. Proyecto final, Universidad Católica "Nuestra Señora de la Asunción", 2017. Available at `http://www.dei.uc.edu.py/proyectos/mddplus/documentos/`.

[94] Manuel Núñez, Daniel Bonhaure, Magalí González, and Luca Cernuzzi. A model-driven approach for the development of native mobile applications focusing on the data layer. *J. Syst. Softw.*, 161, 2020.

[95] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, page 35–46, New York, NY, USA, 2000. Association for Computing Machinery.

[96] José Ignacio Panach, Oscar Dieste, Beatriz Marín, Sergio España, Sira Vegas, Oscar Pastor, and Natalia Juristo. Evaluating model-driven development claims with respect to quality: A family of experiments. *IEEE Trans. Software Eng.*, 47(1):130–145, 2021.

[97] Oscar Pastor, Joan Fons, Vicente Pelechano, and Silvia Abrahão. Conceptual modelling of web applications: The OOWS approach. In Emilia Mendes and Nile Mosley, editors, *Web Engineering*, pages 277–302. Springer, 2006.

[98] Oscar Pastor, Jaime Gómez, Emilio Insfrán, and Vicente Pelechano. The oo-method approach for information systems modeling: from object-oriented conceptual modeling to automated programming. *Information Systems*, 26(7):507–534, 2001.

[99] Oscar Pastor and Juan Carlos Molina. *Model-driven architecture in practice - a software production environment based on conceptual modeling*. Springer, 2007.

[100] Microsoft Patterns. *Microsoft Application Architecture Guide.* Microsoft Press, 2nd edition, 2009.

[101] Roger Pressman and David Lowe. *Web Engineering: A Practioner's Approach.* McGraw-Hill, Inc., USA, 1 edition, 2008.

[102] Ajith H Ranabahu, Eugene Michael Maximilien, Amit P Sheth, and Krishnaprasad Thirunarayan. A domain specific language for enterprise grade cloud-mobile hybrid applications. In *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*, pages 77–84. ACM, 2011.

[103] André Ribeiro and Alberto Rodrigues da Silva. XIS-mobile: a DSL for mobile applications. In Yookun Cho, Sung Y. Shin, Sang-Wook Kim, Chih-Cheng Hung, and Jiman Hong, editors, *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014*, pages 1316–1323. ACM, 2014.

[104] Leonard Richardson, Mike Amundsen, and Sam Ruby. *RESTful Web APIs.* "O'Reilly Media, Inc.", 2013.

[105] Christoph Rieger and Herbert Kuchen. A process-oriented modeling approach for graphical development of mobile business apps. *Computer Languages, Systems & Structures*, 53:43–58, 2018.

[106] Christoph Rieger and Herbert Kuchen. A Model-Driven Cross-Platform App Development Process for Heterogeneous Device Classes. In Tung Bui, editor, *52nd Hawaii International Conference on System Sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8-11, 2019*, pages 1–10. ScholarSpace / AIS Electronic Library (AISeL), 2019.

[107] Jimmy Molina Ríos and Nieves Pedreira-Souto. Comparison of development methodologies in web applications. *Inf. Softw. Technol.*, 119, 2020.

[108] Villalba Rodrigo, Gonzán Magalí, Cernuzzi Luca, and Pastor Oscar. Enfoque mda para servicios: un tipo específico de comportamiento. In *Memorias de la XXXVI Conferencia Lationamericana de Informática*, 2010.

[109] Breno Lisi Romano and Adilson Marques da Cunha. A framework for web applications using an agile and collaborative model driven development (ac-mdd). *Acta Scientiarum. Technology*, 41:e38349–e38349, 2019.

[110] Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina, editors. *Web Engineering: Modelling and Implementing Web Applications*. Human-Computer Interaction Series. Springer, 2008.

[111] Gustavo Rossi and Daniel Schwabe. Modeling and implementing web applications with oohdm. In Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina, editors, *Web Engineering: Modelling and Implementing Web Applications*, Human-Computer Interaction Series, pages 109–155. Springer, 2008.

[112] Gustavo Rossi, Matias Urbieta, Damiano Distante, José Matías Rivero, and Sergio Firmenich. 25 years of model-driven web engineering: What we achieved, what is missing. *CLEI Electron. J.*, 19(3):1, 2016.

[113] Gustavo Rossi, Matias Urbieta, Jeronimo Ginzburg, Damiano Distante, and Alejandra Garrido. Refactoring to rich internet applications. A model-driven approach. In *ICWE*, pages 1–12. IEEE Computer Society, 2008.

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

[115] Anna Ruokonen, Lasse Pajunen, and Tarja Systä. On model-driven development of mobile business processes. In *SERA*, pages 59–66. IEEE Computer Society, 2008.

[116] John R Rymer, Rog Koplowitz, Salesforce Are Leaders, Kony Mendix, Salesforce are Leaders, GeneXus ServiceNow, Strong Performers, WaveMaker MatsSoft, and Thinkwise are Contenders. The forrester wave™: Low-code development platforms for ad&d professionals, q1 2019, 2019.

[117] Ayoub Sabraoui, Mohammed El Koutbi, and Ismaïl Khriss. A MDA-Based Model-Driven Approach to Generate GUI for Mobile Applications. *International Review on Computers and Software (IRECOS)*, 8(3):845–852, 2013.

[118] Pejman Sajjadi and Olga De Troyer. Revising web design to deal with current development practices. In *APCCM*, volume 165 of *CRPIT*, pages 103–108. Australian Computer Society, 2015.

[119] Daniel Sanchez and Hector Florez. Model Driven Engineering Approach to Manage Peripherals in Mobile Devices. In Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Elena N. Stankova, Carmelo Maria Torre, Ana Maria A. C. Rocha, David Taniar, Bernady O. Apduhan, Eufemia Tarantino, and Yeonseung Ryu, editors, *Computational Science and Its Applications - ICCSA 2018 - 18th International Conference, Melbourne, VIC, Australia, July 2-5, 2018, Proceedings, Part IV*, volume 10963 of *Lecture Notes in Computer Science*, pages 353–364. Springer, 2018.

[120] Emanuel Sanchiz, Magalí González, Nathalie Aquino, and Luca Cernuzzi. Development of mobile applications with functions in the cloud through the model driven approach: A systematic mapping study. *CLEI electronic journal*, 20(3), December 2017.

[121] Emanuel Sanchiz, Magalí González, Nathalie Aquino, and Luca Cernuzzi. Moweba mobile: Modeling and generation of the communication of mobile apps with their functions in the cloud. In Marcela Genero, Marcos Kalinowski, Jesús García Molina, Francisco Pino, Tayana Conte, Beatriz Marín, Isabel Brito, and Giovanni Giachetti, editors, *Proceedings of the*

XXI Iberoamerican Conference on Software Engineering, Bogota, Colombia, April 23-27, 2018, pages 312–325. Curran Associates, 2018.

[122] Marcos López Sanz, Valeria de Castro, and Esperanza Marcos. An architecture-centric process for service oriented systems development: Developing for the intelligent pavement. In *CLEI*, pages 1–9. IEEE, 2014.

[123] 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.

[124] Jeff Sauro. Measuring usability with the system usability scale (sus). 2011.

[125] Jeff Sauro and James R. Lewis. *Quantifying the User Experience: Practical Statistics for User Research.* Morgan Kaufmann, 2016.

[126] W. Schwinger and N. Koch. Modeling web applications. In G. Kappel, B. Pröll, S. Reich, and W.Retschitzegger, editors, *Web Engineering: a New Discipline for Development of Web-Based Systems*, chapter 3, pages 39–64. John Wiley & Sons, New York, 2006.

[127] Wieland Schwinger, Werner Retschitzegger, Andrea Schauerhuber, Gerti Kappel, Manuel Wimmer, Birgit Pröll, Cristina Cachero, Sven Casteleyn, Olga De Troyer, Piero Fraternali, Irene Garrigós, Franca Garzotto, Athula Ginige, Geert-Jan Houben, Nora Koch, Nathalie Moreno, Oscar Pastor, Paolo Paolini, Vicente Pelechano, Gustavo Rossi, Daniel Schwabe, Massimo Tisi, Antonio Vallecillo, Kees van der Sluijs, and Gefei Zhang. A survey on web modeling approaches for ubiquitous web applications. *Int. J. Web Inf. Syst.*, 4(3):234–305, 2008.

[128] João Seixas, André Ribeiro, and Alberto Rodrigues da Silva. A model-driven approach for developing responsive web apps. In *ENASE*, pages 257–264, 2019.

[129] Thomas Stahl, Markus Völter, Jorn Bettin, Arno Haase, and Simon Helsen. *Model-driven software development - technology, engineering, management.* Pitman, 2006.

[130] Dustin Steiner, Catalina Turlea, Cristian Culea, and Stephan Selinger. Model-driven development of cloud-connected mobile applications using dsls with xtext. In *EUROCAST (2)*, volume 8112 of *Lecture Notes in Computer Science*, pages 409–416. Springer, 2013.

[131] Victoria Torres, Vicente Pelechano, and Oscar Pastor. Building semantic web services based on a model driven web engineering method. In *ER (Workshops)*, volume 4231 of *Lecture Notes in Computer Science*, pages 173–182. Springer, 2006.

[132] Marino Linaje Trigueros, Juan Carlos Preciado, Rober Morales-Chaparro, Roberto Rodríguez-Echeverría, and Fernando Sánchez-Figueroa. Automatic generation of rias using rux-tool and webratio. In *ICWE*, volume 5648 of *Lecture Notes in Computer Science*, pages 501–504. Springer, 2009.

[133] Olga De Troyer and Sven Casteleyn. Exploiting link types during the conceptual design of websites. *Int. J. Web Eng. Technol.*, 1(1):17–40, 2003.

[134] Olga De Troyer and Tom Decruyenaere. Conceptual modelling of web sites for end-users. *World Wide Web*, 3(1):27–42, 2000.

[135] Tom Tullis and Bill Albert. Chapter 6 - Self-Reported Metrics. In Tom Tullis and Bill Albert, editors, *Measuring the User Experience (Second Edition)*, Interactive Technologies, pages 121 – 161. Morgan Kaufmann, Boston, second edition edition, 2013.

[136] Eric Umuhoza, Hamza Ed-douibi, Marco Brambilla, Jordi Cabot, and Aldo Bongio. Automatic code generation for cross-platform, multi-device mobile apps: Some reflections from an industrial experience. In *Proceedings of the 3rd International Workshop on Mobile Development Lifecycle*, MobileDeLi 2015, page 37–44, New York, NY, USA, 2015. Association for Computing Machinery.

[137] Francisco Valverde and Oscar Pastor. Applying interaction patterns: Towards a model-driven approach for rich internet applications development. In *7th International Workshop on Web-Oriented Software Technologies*, pages 13–18, 2008.

[138] Kees van der Sluijs, Geert-Jan Houben, Erwin Leonardi, and Jan Hidders. Hera: Engineering web applications using semantic web-based models. In *Semantic Web Information Management*, pages 521–544. Springer, 2009.

[139] Pedram Veisi and Eleni Stroulia. AHL: Model-Driven Engineering of Android Applications with BLE Peripherals. In Esma Aïmeur, Umar Ruhi, and Michael Weiss, editors, *E-Technologies: Embracing the Internet of Things - 7th International Conference, MCETECH 2017, Ottawa, ON, Canada, May 17-19, 2017, Proceedings*, volume 289 of *Lecture Notes in Business Information Processing*, pages 56–74, 2017.

[140] Paul Vincent, Kimihiko Iijima, Mark Driver, Jason Wong, and Yefim Natis. Magic quadrant for enterprise low-code application platforms. *Gartner report*, 2019.

[141] Karzan Wakil and Dayang Jawawi. A new adaptive model for web engineering methods to develop modern web applications. pages 32–39, 01 2018.

[142] Karzan Wakil and Dayang Jawawi. Extracting the features of modern web applications based on web engineering methods. *International Journal of Advanced Computer Science and Applications*, 10, 01 2019.

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

[144] Karzan Wakil and Dayang N. A. Jawawi. Comparison between web engineering methods to develop multi web applications. *J. Softw.*, 12(10):783–793, 2017.

[145] Karzan Wakil and DN Jawawi. Analyzing interaction flow modeling language in web development lifecycle. *International Journal of Advanced Computer Science and Applications*, 8(1):286–293, 2017.

[146] Karzan Wakil, Amirhossein Safi, and Dayang Jawawi. Enhancement of uwe navigation model: Homepage development case study. *International Journal of Software Engineering and its Applications*, 8:197–212, 01 2014.

[147] Stephan Weibelzahl. *Evaluation of adaptive systems*. PhD thesis, University of Trier, 2004.

[148] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014.

[149] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, and Björn Regnell. *Experimentation in Software Engineering*. Springer, 2012.

[150] R.K. Yin and SAGE. *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications, 2003.