

**Development of a personalization model for web applications
in a context of model-driven development**



Luz Viviana Cobaleda Estepa

Advisor

PhD. John Freddy Duitama Muñoz

PhD Thesis

Doctorate in Electronic Engineering program

Faculty of Engineering

University of Antioquia

2017

Abstract

Technological advances and new practices in web systems allow gathering more detailed information from users, this in turn triggers a fast evolution of the web personalization field. Web personalization is considered as the process of adapting the content, the structure of a website and the services that it provides, for different user groups as well as for concrete users, to provide a customized experience. Although personalization has showed advantages in web applications, there remains some issues that complicate its inclusion in a business environment: the difficulty to evolve personalized web applications; the technical complexity to incorporate the personalization strategies; and, the limited support of model-driven approaches. The aim of this dissertation is to develop and validate a maintainable approach to improve the modifiability of personalized web applications and to reduce the technical complexity to integrate personalization strategies in a short time in a business environment. As results and contributions, we developed a Software Reference Architecture to face the maintainability problem. This reference architecture, adopts a software component weaving process, and encapsulates personalization strategies in independent specialized components with defined interfaces. We implemented, in context of the model-driven development (MDD), the framework MAMPA (**Model-driven Approach to enhance the Modifiability of Personalized Web Applications**) to reduce the technical complexity of weaving personalization strategies (specialized software components), and its subsequently code generation. We proposed a new modeling language: The Personalized Web Application Modeling Language (PWML) defined as an UML Profile to specify the personalization model. In addition, MAMPA provides a transformation process based on templates and using *Acceleo* as the transformation language. The framework tool is built on technologies for Eclipse. Finally, we executed two controlled experiments to evaluate i) the software modifiability of the proposed Reference Software Architecture, and ii) the effectiveness of the MAMPA approach. The experiments made use of personalization strategies extracted from a real Brazilian e-commerce enterprise.

Keywords: personalization, personalized web applications, reference software architecture, modifiability, model-driven development, model transformations, personalization modeling language.

Acknowledgments

I would like to thank my family for their support and their time.

I would like to thank my PhD advisor, Dr. Freddy Duitama, for his guidance during my research.

I would like to thank the members of the *Centre de Recherche en Informatique* (CRI) at the Panthéon Sorbonne University, and especially to Raúl Mazo and Camille Salinesi for their valuable discussions and encouragement.

I would like to thank Dr. Jorge Risco, for his assistance during the internship at the University of Sao Paulo.

I want to thank *Lamon Chapman* for his reviews for the publication of articles.

I want to thank *Walter Arboleda* for his effort and patient that made possible the implementation of the MAMPA approach.

I would like to thank all the people who contributed in some way to the elaboration of the present dissertation.

Finally, I wish to acknowledge VTEX enterprise for providing useful information for this work, This research was supported by University of Antioquia through the committee for the research development - CODI (CODI- PRG13-2-01).

Contents

Part I: Overview	1
Chapter 1. Introduction	1
1.1 Motivation	1
1.2 Proposal	4
1.3 Contributions	5
1.4 Papers	6
1.5 Thesis organization	6
Chapter 2. State of the Art	8
2.1 Evolution of web personalization	8
2.2 Motivation	10
2.3 Literature Review Process	11
2.3.1 The research questions	11
2.3.2 The search process	12
2.3.3 The study selection	12
2.4 Revision	13
2.4.1 Garrigós, Gomez, & Houben, 2010	13
2.4.2 Ceri, Daniel, Matera, & Facca, 2007	16
2.4.3 De Virgilio, 2012	18
2.4.4 De Bra et al., 2016	20
2.4.5 Montes Garcia, De Bra, Fletcher, & Pechenizkiy, 2014	22
2.4.6 A. I. Cristea, Smits, Bevan, & Hendrix, 2009	22
2.4.7 Sadat-Mohtasham & Ghorbani, 2008	23
2.4.8 Fan, Hussain, Younas, & Hussain, 2015	26
2.5 Summary	28
Part II: Proposal	31

Chapter 3. Reference Software Architecture to improve modifiability of Personalized Web Applications	31
3.1 Introduction	31
3.2 Analysis of software modifiability: Related work	32
3.3 An e-commerce system – the running example	34
3.4 Reference architecture	35
3.4.1 Reference architecture description	35
3.4.2 Methodology to apply the reference architecture	38
3.5 Conclusion	42
 Chapter 4. A controlled experiment to validate the Reference Software Architecture	 44
4.1 Experiment process	44
4.2 Design of the experiment	45
4.3 Experimental units – Test Case	46
4.3.1 Change scenario 1	46
4.3.2 Change scenario 2	46
4.3.3 Change scenario 3	47
4.3.4 Change scenario 4	47
4.3.5 Change scenario 5	47
4.4 Metrics	47
4.5 Prototype implementation	48
4.6 Threats to validity	48
4.7 Results	50
 Chapter 5. MAMPA: A model-driven approach to enhance the modifiability of personalized web applications	 53
5.1 Running Example	53
5.2 Framework to Adopt the MAMPA Approach	55
5.2.1 Framework description	56

5.2.2	PWML: Personalized Web Application Modeling Language	58
5.2.3	Transformations	66
5.2.4	Prototype	67
5.3	Related Work	68
Chapter 6. A controlled experiment to validate MAMPA approach		72
6.1	Scoping	72
6.2	Research questions	72
6.3	Experimental units	74
6.4	Experiment design	74
6.5	Selection of variables	75
6.6	Threats to validity	75
6.7	Results	76
Part III: Final Considerations		79
Chapter 7. Conclusions		79
7.1	Contributions	81
7.2	Ongoing and Future Work	82
Appendix A.		83
Appendix B.		87
Appendix C.		89
Appendix D.		94

List of figures

Figure 1.	AWAC Architecture. Taken from (Garrigós Fernández, 2008)	14
Figure 2.	Architecture of an application developed with WebRatio. Taken from (Molinaroli, 2012)	17
Figure 3.	FAWIS Architecture. Taken from (De Virgilio, 2012)	19

Figure 4.	Architecture of an adaptive course under an open model adaptation	21
Figure 5.	Architecture of the adaptive web (aWeb) framework	24
Figure 6.	SaaS Personalization Framework	27
Figure 7.	History of recent products visited by customer	34
Figure 8.	Reference architecture for personalized web applications	36
Figure 9.	PMAAdmin with multiples web applications, represented with UML component diagram	38
Figure 10.	Process to apply the reference architecture represented with a UML activity diagram	39
Figure 11.	Interaction between modules and personalization components	41
Figure 12.	Architecture for the change scenario 4 according to the reference architecture	49
Figure 13.	Framework to support MAMPA approach	56
Figure 14.	Meta-model for PWML	59
Figure 15.	Model of the Application Layer corresponding to our running example using PWML	61
Figure 16.	Model of the User Interface Layer corresponding to our running example using PWML	62
Figure 17.	Use of tag values in the User Interface Layer corresponding to our running example	63
Figure 18.	Model of the Component Layer corresponding to our running example using PWML	64
Figure 19.	Architecture for the change scenario 1 according to the reference architecture	83
Figure 20.	Architecture for the change scenario 2 according to the reference architecture	84
Figure 21.	Architecture for the change scenario 3 according to the reference architecture	85
Figure 22.	Architecture for the change scenario 5 according to the reference architecture	86
Figure 23.	Product list with a discount ticket of 50%	87
Figure 24.	Product detail with a discount by day	88
Figure 25.	Product detail with recommended products	88
Figure 26.	Packages for the definition of meta-model, models, and transformations	89
Figure 27.	Meta-model definition	89
Figure 28.	Models definition screenshot	90
Figure 29.	Packages of the transformations project	90
Figure 30.	Transformation for the controller module in the reference architecture	91
Figure 31.	Transformation for the personalization controller module in the reference architecture	91
Figure 32.	Transformation for the view	92
Figure 33.	Product detail page generated for the personalization strategy # 7	92
Figure 34.	Personalization controller* generated for the personalization strategy # 4	93
Figure 35.	Packages of the final web application with the assembled personalization strategies	93

List of tables

Table 1.	Web modeling languages with personalization support in a MDD context	29
Table 2.	Results of the execution of change scenarios under standard MVC Architecture	50
Table 3.	Results of the execution of change scenarios under reference architecture	51
Table 4.	Changes on the PMAAdmin module in the execution of change scenarios	51
Table 5.	Summary of the defined stereotypes and properties in the UML profile for PWML	65
Table 6.	MDE work features	70
Table 7.	Error density	77
Table 8.	Coverage percentage of the MAMPA approach	78

List of Abbreviations and Acronyms

ATL	ATLAS Transformation Language
AWBS	Adaptive Web-Based Systems
AWS	Adaptive Web systems
A-OOH	Adaptive OO-H
AML	Adaptation Modeling Language
AWAC	Adaptive Web Applications Creator
BNF	Backus-Naur Form
EBNF	Extended BNF
CBSE	Component-Based Software Engineering
CSS	Cascading Style Sheets
CMS	Content Management Systems
DSL	Domain Specific Language
ECA	Event-Condition-Action
EJB	Enterprise Java Beans
EMF	Eclipse Modeling Framework
FAWIS	Flexible adaptation of web information systems
GAL	Generic Adaptation Language
GALE	The GRAPPLE Adaptive Learning Environment
GQM	Goal-question-metric

ICT	Information and Communication Technologies
IFML	Interaction Flow Modeling Language
JSF	Java Server Faces
JSP	Java Server Pages
JPA	Java persistence API
M2T	Model-to-Text transformation
M2M	Model-to-Model transformation
MAMPA	A Model-driven Approach to enhance the Modifiability of Personalized web Applications
MDD	Model-driven development
MDSD	Model-driven software development
MVC	Model View Controller
LOC	Lines of Code
OCL	Object Constraint Language
OO-H	Object Oriented Hypermedia
OMG	Object Management Group
PIM	Platform Independent Model
PSM	Platform Specific Model
PWML	Personalized Web Application Modeling Language
QoS	Quality of service
QVT	Query/View/Transformation Language
UWA	Ubiquitous Web Applications
WebML	Web Modeling Language

Part I: Overview

Chapter 1.

Introduction

This chapter introduces the web personalization area and motivates the present dissertation. Section 1.1 explains the motivation and problem statement. The goals of the present work are presented in Section 1.2. Section 1.3 presents the main contributions. Section 1.4 lists the papers written about this dissertation. Finally, Section 1.5 summarizes the structure of the dissertation.

1.1 Motivation

Technological advances and new practices in web systems allow gathering more detailed information from users, this in turn triggers a fast evolution of the web personalization field. The digitalization of everyday life, the fast evolution of Information and Communication Technologies (ICT), and an increasing number of datapoints to the Internet, are excellent conditions for prospering of web personalization (Salonen & Karjaluoto, 2016).

Personalization can be understood, as the process of tailoring the business information and services to needs, interests, preferences, context, behavior and specific requirements of an individual or community. It provides a customized environment with an increased value to the customer and to the business (Baldoni, Baroglio, & Henze, 2005; Brusilovsky, 1996, 2001; Brusilovsky, Kobsa, & Nejd, 2007; Brusilovsky & Nejd, 2004; L. Cobaleda & Duitama, 2009; Karat, Brodie, Karat, Vergo, & Alpert, 2003). Personalized software systems support the dynamics of enterprises to embrace different kinds of audiences, and to respond more quickly to market demands. Particularly, in the web domain, definitions for personalization are more close to technological or implementation aspect. Web personalization is “the process of changing the content and structure of a website to adapt it to the specific needs, goals, interests and preferences of each user taking advantage of the user’s navigational behavior and context” (Garrigós, Gomez, & Houben, 2010; Garrigós Fernández, 2008). “The adaptation considers different and orthogonal components of the context and operates over the selection of the most

suitable contents, the organization of the hypertext structure of the response, and the layout of the final pages” (De Virgilio, 2012). In this direction, web personalization is considered as the process of adapting the content, the structure of a website and the services that it provides, for different user groups as well as for concrete users, to provide a customized experience.

The field of web personalization has matured quickly and is on the rise (Salonen & Karjaluoto, 2016). Some authors consider “personalization” as the most effective tool for achieving business success online (Cao & Li, 2007). In e-commerce domain for example, personalization is intended to increase customer fidelity (Kwon & Kim, 2012), and to filter the information that users need most. It enables the users to receive only the information they can handle within the available time and to locate what they need most. personalization is considered as the major driver of marketing efficiency (Kalaiganam, Kushwaha, & Varadarajan, 2008). Personalized software takes user data to adapt the offered information and services to the preferences, interests, needs, context and behavior of the individual users or communities. Personalized software also provides to users an experience of added value to them (Brusilovsky & NejdI, 2004; Karat et al., 2003).

Although personalization has showed advantages in web applications (Alotaibi, 2013; Kwon & Kim, 2012), there remains some issues that complicate its inclusion in a business environment:

1. The difficulty to evolve personalized web applications, issue originated from the low level of modifiability of software code.
2. The technical complexity to integrate personalization strategies in a short time in a business environment.
3. The lack of tools to support personalization modeling and generation of code.

Problem #1. The difficulty to evolve personalized web applications, issue originated from the low level of modifiability of software code.

Modern web systems are in constant evolution; they are becoming more and more complex and deal with continuous changes derived from business needs. Even more, the context is

becoming a significant requirement to web mobile applications because of its heterogeneity characteristic (Bettini et al., 2010). This makes the strategies of personalization more complex. For this reason, it is required that specification of the personalization to be updated in an easy and fast way (modifiability), with possibilities of extension and reuse in similar software systems. The issues associated with maintainability, extensibility and reusability in software is due to the low level of abstraction in the specification of personalization (Garrigós et al., 2010; Perez & Correal, 2011). Because of that, the personalization code in web applications is intermingled with the basic functionality (De Virgilio, 2012; Garrigós et al., 2010), and the subsequent process to add or modify a personalization strategy in a web application is difficult. Different approaches (De Virgilio, 2012; Garrigós et al., 2010) have been explored in order to permit the continuous update of personalization strategies, and to reduce the complexity in the implementation, however, these issues have not been sufficiently studied.

Problem #2. The technical complexity to integrate personalization strategies in a short time in a business environment.

Derived from problem #1, the technical complexity to integrate personalization strategies in short time and the frequent update of strategies for each business, make challenging to include them in a business environment. Usually, organizational interests and market evolution motivate the continuous modifications. For example, in an e-commerce domain, personalization strategies may vary even in few days or hours as a natural result of getting closer to diverse audiences—to which we need for instance some particular discounts or recommendations strategies.

Problem #3. The lack of tools to support personalization modeling and generation of code.

Code generation is a known and effective means to increase the productivity and quality of software applications (Jiang, Ying, Wu, & Jin, 2008) as well as to reduce the development effort (Stahl & Voelter, 2006); even more, the visual tools specially favor the quick development. Model-driven software development (MDS) is an approach that considers application models as the primary artifact of the software development process, and uses them systematically to automate the software production and allowing the (semi)automatic generation of code (Brambilla, Cabot, & Wimmer, 2012; Stahl & Voelter, 2006).

Related to a high productivity in the development of personalized software, it is found that current tools to support the web development lack of efficient mechanisms to support the specification of personalization and its subsequent automatic generation of code. Although exist some tools in the research area (Ceri, Daniel, Matera, & Facca, 2007; De Virgilio, 2012; Garrigós Fernández, Glorio, Hernández, & Maté Morga, 2009; Khambati, Grundy, Warren, & Hosking, 2008; Sadat-Mohtasham & Ghorbani, 2008), they are limited to authors particular methodology.

1.2 Proposal

To tackle the aforementioned problems, we develop and validate a maintainable approach to improve the modifiability of personalized web applications and to reduce the technical complexity to integrate personalization strategies in a short time in a business environment.

The proposed solution includes the establishment of the Reference Software Architecture to improve modifiability of Personalized Web Applications (L. V. Cobaleda, Mazo, Becerra, & Duitama, 2016) that has the software modifiability as the main architectural drive, and is based on component weaving process. As well as the proposal and implementation of the framework called MAMPA¹ (L.-V. Cobaleda, Mazo, Mejía, & Duitama, 2017), that follows a model-driven approach to enhance the modifiability of personalized web applications embracing the Reference Software Architecture previously proposed. The framework allows modeling the personalization strategies within a personalization model using a specific modeling language called *Personalized Web Application Modeling Language* (PWML), and its subsequent generation of code using a model-driven approach.

The framework MAMPA tackles the problems described above in the following way:

Solution #1. Maintainability difficulties

The maintainability problem is faced with a Software Reference Architecture, which adopts a software component weaving process, and encapsulates personalization strategies in

¹ A Model-driven Approach to enhance the Modifiability of Personalized web Applications (MAMPA)

independent specialized components with defined interfaces. A controlled experiment verifies the maintainability of the proposal (L. V. Cobaleda et al., 2016).

Solution #2. The technical complexity to integrate personalization strategies in a short time in a business environment.

The complexity in the integration of personalization strategies in a business environment, is faced with the framework supporting MAMPA approach. This framework reduces the technical complexity of weaving the specialized software components, by giving support for the modeling of personalization strategies, and subsequently with the code generation using M2T transformations. Consequently, those features are directed to reduce the time to respond to market changes. (L.-V. Cobaleda et al., 2017).

Solution #3. Tools to support personalization modeling and generation of code.

As solution, we give a framework tool as an alternative to enhance the evolution and modifiability of personalized web applications making use of the model-driven methodology, called MAMPA. The framework makes use of a new modeling language: The Personalized Web Application Modeling Language (PWML) defined as an UML Profile to specify the personalization model. In addition, it provides a transformation process based on templates and using *Acceleo* as the transformation language. The framework tool is built on technologies for Eclipse. We faced a key problem of a CASE tool, the maintainability, by focusing on the changes of components. Each component implements a personalization strategy from business. We validated the MAMPA effectiveness through the completeness and the accuracy, using a controlled experiment with a collection of personalization strategies provided by a Brazilian company (L.-V. Cobaleda et al., 2017).

1.3 Contributions

The main contributions of this work are the following:

- A Reference Software Architecture to support the modifiability of personalized web applications.
- A methodology to apply the Reference Software Architecture.

- A Model-driven Approach to enhance the Modifiability of Personalized web Applications (MAMPA).
- A Personalized Web Application Modeling Language (PWML) to specify the personalization strategies.
- A transformation process based on templates to guarantee the conformity of the generated code with the Reference Software Architecture.
- A group of code transformation templates implemented with *Acceleo* as the transformation language.
- A framework tool, supporting the MAMPA approach, built on technologies for Eclipse.

1.4 Papers

Cobaleda, Luz-Viviana., Mazo, Raúl., Risco, Jorge-Luís., Duitama, John-Freddy. Reference Software Architecture for improving modifiability of Personalized Web Applications - A controlled experiment. *Int. J. of Web Engineering and Technology*, 2016 Vol.11, No.4, pp.351 – 370 DOI: 10.1504/IJWET.2016.081768

<http://www.inderscience.com/info/inarticle.php?artid=81768>

Cobaleda, Luz-Viviana., Mazo, Raúl., Mejía, Jayson., Duitama, John-Freddy. MAMPA: A Model-Driven Approach to Enhance the Modifiability of Personalized Web Applications. Submitted to *International Journal on Software and Systems Modeling Journal - SoSyM*, 2017.

1.5 Thesis organization

Chapter 2 presents the exploration of different approaches to support the personalization into the web applications in a MDD context.

Chapter 3 presents a reference software architecture as an alternative to solve the maintainability problems faced in the evolution of personalized web applications.

Chapter 4 presents the controlled experiment to evaluate the software modifiability of the proposed reference software architecture, using a real business case taken from a Brazilian e-commerce enterprise.

Chapter 5 presents a **Model-driven Approach** to enhance the **Modifiability of Personalized web Applications (MAMPA)** to improve the modification of personalization strategies in web applications.

Chapter 6 presents the controlled experiment to evaluate the effectiveness in supporting software evolution under the MAMPA approach.

Chapter 7 presents conclusions, contributions and future work.

Chapter 2.

State of the Art

This chapter presents different approaches to support the personalization into the web applications in a MDD context; it mainly focus on topics as the specification of the personalization, the maintenance of the personalization strategies in an evolution scenario, the process to integrate the personalized behavior in business environment and the support given from a MDD context. Section 2.1 introduces the evolution of web personalization area. Section 2.2 presents the motivation and the goal of the exploration. The literature review criteria and process are presented in section 2.3. Section 2.4 describes the analyzed software architectures, focusing on analysis criteria. Finally, section 2.5 summarizes the main features of the exploration.

2.1 Evolution of web personalization

The field of web personalization has matured in the last two decades; it has remained actively during the period of 2005-2015 and continues to be an important topic in the top 20 marketing and information systems (IS) journals (Salonen & Karjaluo, 2016).

Personalization has its origins in adaptive hypermedia systems traced back to the early 1990s (Brusilovsky, 1996, 2001). “Adaptive hypermedia systems build a model of the goals, preferences and knowledge of each individual user, and use this model throughout the interaction with the user, in order to adapt to the needs of that user” (Brusilovsky, 2001). From the adaptive hypermedia area, personalization can be understood, as the process of tailoring the business information and services to needs, interests, preferences, context, behavior and specific requirements of an individual or community. It provides a customized environment with an increased value to the customer and to the business (Brusilovsky, 1996, 2001; Brusilovsky & Nejd, 2004; Karat et al., 2003). The typical kinds of adaptive hypermedia systems were educational hypermedia, on-line information systems, on-line help systems, and information retrieval hypermedia (Brusilovsky, 2001).

Later, and due to technological advances, especially to the mobile communication devices, emerged the context-awareness research area. Context-awareness refers to systems that can both sense and react based on their environment, thus, the web applications are expected to respond appropriately to the context of users. Whereas a conventional adaptive hypermedia system adapts an environment in response to user-generated events, context-awareness responds to context-triggered actions. For example, a mobile phone may detect that a user is sitting or walking, and the web application reacts in correspondence with the user state. In these cases, besides the wide variety of devices, their continuous technological evolution demands the permanent development of new personalization strategies (Ceri et al., 2007; Hoyos, García-Molina, & Botía, 2013). Context is any information that characterizes the situation of a person, place, or object relevant to the interaction between a user and an application, as well as the environment where such interaction occurs, and the user and software system themselves. For example, physical context properties as temperature, location, time, illumination and noise conditions. Device context properties include the device itself, network and sensors properties: For instance, CPU, memory, size of the screen, energy of the mobile device, or network bandwidth, or Sensors and its states. Also, it includes user features and preferences (Ceri et al., 2007; De Virgilio, 2012). As context definition preserves features from user itself, context-based adaptation may be considered as a natural evolution of personalization concept (Ceri et al., 2007).

The context is a significant prerequisite to enable ubiquitous web applications because of the heterogeneity of its characteristic (Bettini et al., 2010; Schauerhuber, Schwinger, Retschitzegger, Wimmer, & Kappel, 2008). Advances in hardware and software technologies make possible non-intrusive human-computer interaction in ubiquitous (or pervasive) computing as well as multiple research fields (Kakousis, Paspallis, & Papadopoulos, 2010). In ubiquitous web domain, where the applications cope with many aspects any time and any place and from any device, not only the user information is the clue for tailoring a personalized experience. Consequently, the context plays an important role, and allows the creation of personalization strategies more varied and complex (Bettini et al., 2010; Kakousis et al., 2010; Schauerhuber et al., 2008). Adaptation in ubiquitous computing also is understood as “the reactive process triggered by a specific event or a set of events in the context, with an ultimate goal to improve the QoS perceived by the end-user” (Kakousis et al., 2010).

The personalization concept has evolved over time, and has crossed through different research areas. Personalization is closely related to customization and adaptation. Consequently, in the literature, personalization (considered as company-initiated), adaptation (considered as context-initiated) and customization (considered as user-initiated) concepts, sometimes have been used as synonymous (Salonen & Karjaluoto, 2016) or are considered as part of the natural evolution of the concept (Ceri et al., 2007).

2.2 Motivation

Personalization is considered an effective tool for achieving business success online (Cao & Li, 2007), a key driver of marketing efficiency (Kalaiganam et al., 2008) and a valuable tool in the business, because it provides to users an experience of added value to them (Brusilovsky & Nejd, 2004; Karat et al., 2003). Usually, organizational interests and market evolution motivate the continuous modifications of personalization strategies. In this context, the support to the frequent update of personalization strategies into web application, in order to respond in short time to the changing market has a significant impact for enterprises. This scenario of frequent update of personalization strategies and quickly response to market leading to an evolution of the web application became a challenge of personalized web applications.

The support to personalization (specification and implementation) into web applications has been handled with the use of high level modeling languages and in some cases with MDD approaches. In order to explore the support to the personalization (specification and implementation) into the web applications, and to identify how the different approaches deal with an evolution scenario in a context of MDD, this chapter presents a literature review process and analyzes limitations and strengths of some approaches. Specifically, this chapter analyzes the specification of the personalization strategies, the process of constant modification/addition, the integration process into a production environment and the tools and MDD approaches adopted.

2.3 Literature Review Process

This section describes the literature review process and the criteria used throughout it. The process followed was based on (Kitchenham & Charters, 2007). The steps followed are:

- a) The specification of the research questions: it sets the focus for the identification of the primary studies.
- b) The description of the search process: it includes the definition of the strategy used to search for primary studies including search terms and resources to be searched, like digital libraries or specific journals, and conference proceedings.
- c) The study selection criteria: it is used to determine which studies are included in, or excluded from, a systematic review; it also includes quality criteria.
- d) Finally, the data extraction and data synthesis is reported.

Our search process was iterative and executed in three steps:

- 1- First, an initial search in the bibliographic databases with the search strings, during the selected period was done. All searches were based on title, keywords and abstract. Then we carried out a screening process based on title, abstract and keywords, applying the inclusion / exclusion criteria.
- 2- Then, we read the entire article and applied again the inclusion / exclusion criteria and quality criteria.
- 3- Finally, a deep search was done only for broaden the knowledge in specific works or authors. We applied “snowballing” that means to search for primary studies based on references to and from other studies.

2.3.1 *The research questions*

The main focus of analysis is about the architectures to support personalization in web applications; specifically, the maintenance of the personalization strategies in an evolution scenario, the process to integrate the personalized behavior in business environment in a MDD context, and how is the specification of the personalization (models/languages). In order to analyze these aspects, particularly we have stated the following research questions (RQ):

RQ1. How are the architectures/frameworks to support web personalization?

RQ2. How are the specifications (models/languages) in the architectures/frameworks to support web personalization in a MDE context?

2.3.2 The search process

In the search strategy, the bibliographic databases explored were Science Direct, Scopus, Springer Link, ACM Digital Library, IEEE *Xplore* Digital Library. Articles of 10 years backward were included in order to broaden the scope of the review. All searches were based on title, keywords and abstract. For all sources, the search words used were: (architecture) and (Web personalization); (specification) and (Web personalization); (architecture) and (adaptive web); (specification) and (adaptive hypermedia); (specification) and (adaptive web); (framework) and (Web personalization); (framework) and (adaptive web); (modeling language) and (Web personalization); (modeling language) and (adaptive hypermedia); (modeling language) and (adaptive web).

The search strings were complemented, in cases to be necessary, with restrictions to the discipline: Computer science and sub discipline: Software Engineering.

2.3.3 The study selection

After integrating the results for the different searches, we carried out a screening of the 827 papers found, based on title, abstract and keywords. We applied the inclusion /exclusion criteria finding 67 papers.

The inclusion criteria were:

- Any article focusing on software architectures (including its language) for web personalization or web adaptation.
- Any article focusing on the specification of web personalization or web adaptation (languages).
- Only full papers (not a PowerPoint presentation or extended abstract).
- Only papers written in English.
- Any paper in computer science and software engineering domain.

The exclusion criteria were:

- Papers that were obviously irrelevant, or duplicates

- Any paper unrelated with frameworks, architectures or modeling languages for web personalization.
- Any paper that was not in a web domain, although they tackle context-aware challenges or IoT (Internet of Things) matters.
- Any paper that presented only algorithms of prediction, selection, recommendations, retrieval process, security and not focusing on web architectures.

Finally, in order to only complement the data extraction process, we executed a deep search for specific works or authors.

2.4 Revision

Software architectures to support personalization in web are described and analyzed in this section under the research questions; particularly the scenarios of evolution and integration in a business environment (enclosed in a MDD context) are studied. The features related to architecture are highlighted with the “(RQ1-architecture)” label. The specification matters are highlighted with the “(RQ2-Spec)” label, and all aspects related to MDD context are highlighted with the “(RQ2-MDD)” label.

2.4.1 Garrigós, Gomez, & Houben, 2010

The Garrigós et al.’s proposal extended the OO-H (Object Oriented Hypermedia) design method to support personalization becoming A-OOH (Adaptive OO-H) (Garrigós et al., 2010). It provides an abstract rule language named *Personalization Rules Modeling Language (PRML)* to specify the personalization in web applications (RQ2-Spec). In this way, the designer specifies the personalization at design time to be performed at runtime. The personalization model (textual model) consist of a group of Event-Condition-Action (ECA) rules (Garrigós & Gómez, 2007; Garrigós et al., 2010). These rules are conceived both to update the knowledge about user, and to define the effects that the personalization causes to the presented content and navigation structure. The personalization model is highly tied to a four models: a *domain model*, which specifies the structure of the domain data; a *navigation model*, which defines the structure and behavior of the navigation over the domain model; a *presentation model*, which defines the layout of the presentation. And finally, to a *user model*, which describes the information needed

for personalization. Thus, the personalization model with PRML is highly dependent on the A-OOH design method.

Since PRML is proposed in a MDD context, the Garrigos' work created the transformations from PRML into the specifics of UWE and Hera design methodologies. (RQ2-MDD) The transformations were developed using Query/View/Transformation Language (QVT) which is a standard proposed by OMG. In the case of UWE, a PRML rule is transformed into an OCL expression; and in the Hera case, a PRML rule is transformed into a set of SeRQL² expressions.

Concerning architecture (RQ1-architecture) Garrigós et al. propose the *Adaptive Web Applications Creator* (AWAC), a three-layer architecture tool, to automatically generate an adaptive Web application from the A-OOH models (Garrigós Fernández, 2008; Garrigós Fernández et al., 2009). The generation process is done with .NET technologies. The proposed architecture holds personalization functionalities concentrated in two of three layers, as shown in Figure 1. It consists of three modules *Website Engine*, *PRML Manager*, and *PRML Evaluator*.

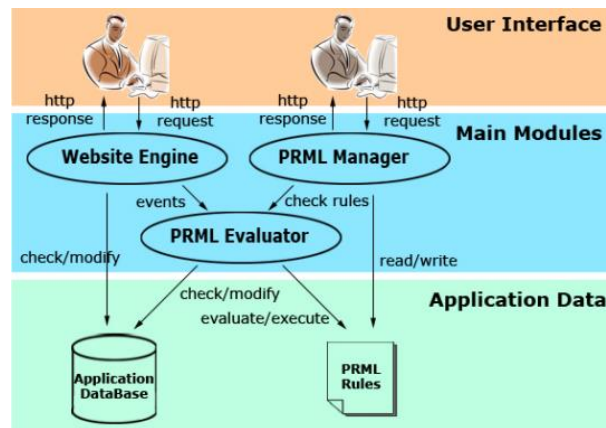


Figure 1. AWAC Architecture. Taken from (Garrigós Fernández, 2008)

The *Website Engine* module interacts with the user, gathering the requests and giving back the response. In addition, it loads the models (from the Application Database) of the particular user when s/he starts a new session, and captures the events that the user performs with his/her

² An RDF query language. RDF means Resource Description Framework.

browsing actions and sends them to the *PRML Evaluator module*. (Garrigós Fernández, 2008) (Page 153). The *PRML Evaluator* module is responsible for evaluating and performing the personalization rules attached to the events. When a rule is triggered, this module evaluates the rule conditions and performs the proper actions. The adaptive actions are only performed once during a session not to overwhelm the user. The *PRML manager* module allows reading and updating rules at runtime. PRML rules are defined in a separate text file. This schema supports the content, navigation support and presentation personalization, because the rules may concern those three aspects. Finally, the application data layer consists of both a text file containing the set of rules defining personalization policies on the website, and the application database.

Concerning our aspects of analysis, the definition of PRML as a high-level language, allows the specification of the personalization before selecting the target platform, in order to take more clear decisions on personalization requirements (RQ2-Spec). In this way, this proposal provides a good support to the activities for writing and updating the strategies in an evolution scenario. But, although Garrigós' approach considers different modules to tailor the response to the user, the personalization actions are so fine-grained causing the creation of many rules to achieve a single strategic personalization goal. As a result, it is hard to manage all of them. Moreover, in time, the user model accumulates valuable information, this approach lacks a clear way to reuse the user model in other applications. Also, Garrigós' approach does not support the integration of different approaches using techniques such as collaborative-filtering or content-based analysis (RQ1-architecture). On the other hand, the process of integrating a personalized behavior in a business environment in an evolution scenario may be time-consuming (RQ1-architecture). The AWAC tool generates the entire web application according to a set of A-OOH models and PRML rules, and generates the three modules in its architecture. Nevertheless, in an evolution scenario, when the personalization strategies involve new elements from models, the process to generate again the application and its deployment in a business environment may become a time-consuming process that limits the fast response to market (RQ1-architecture). Moreover, the personalization strategies are written as textual rules and there is a lack of tools to speed up the correct writing process. Finally and regarding MDD, on one hand, the Garrigós' proposal presents the use of M2M transformations from PRML into Hera and UWE design methods, just as an optional step in the development process. Alternatively, the code generation

is not provided by a transformation language, by contrary, it is contained into a module, which is highly dependent from web application (RQ2-MDD). In our scenario of evolution, these features are limitations for the fast response to market changes.

2.4.2 Ceri, Daniel, Matera, & Facca, 2007

Ceri et al. proposed the *Web Modeling Language* (WebML) that is a visual notation for specifying data intensive web applications embodied in a complete design process (Ceri et al., 2007; Politecnico di Milano., n.d.) (RQ2-Spec). WebML has been evolving from academia since 2003 and currently is supported by the commercial tool WebRatio (www.webratio.com/). This language has evolved up to this time, becoming *Interaction Flow Modeling Language* (IFML) that is designed for expressing the content, user interaction and control behavior of the front-end of software applications. Both languages, WebML and IFML, are supported by WebRatio platform. The WebML design process expresses the structure of a web application with a high-level description and was extended by additional concepts addressing context-aware (Ceri et al., 2007).

The personalization support in WebML consists of the definition of different views according to user profile data or browsing device, based on the User-Group-Module pattern. This process is done at design time. This pattern consists of associating users to groups, and associate groups to modules. It is a way to indicate that a user, who belongs to a group, has access to that specific module. In this way, when user login, the modules he could access, are those associated to groups the user belongs (Garrigós Fernández et al., 2009; Kęsik & Żyła, 2010; Martinenghi, 2014). The process to fill of the personalization entities is done manually (Kęsik & Żyła, 2010). The applications generated with WebRatio are based on classical three-tier architecture, called Model View Controller (MVC) (Molinaroli, 2012) (RQ1-architecture) (RQ2-MDD). The Figure 2 shows the architecture common to all the applications developed with WebRatio in the J2EE architecture.

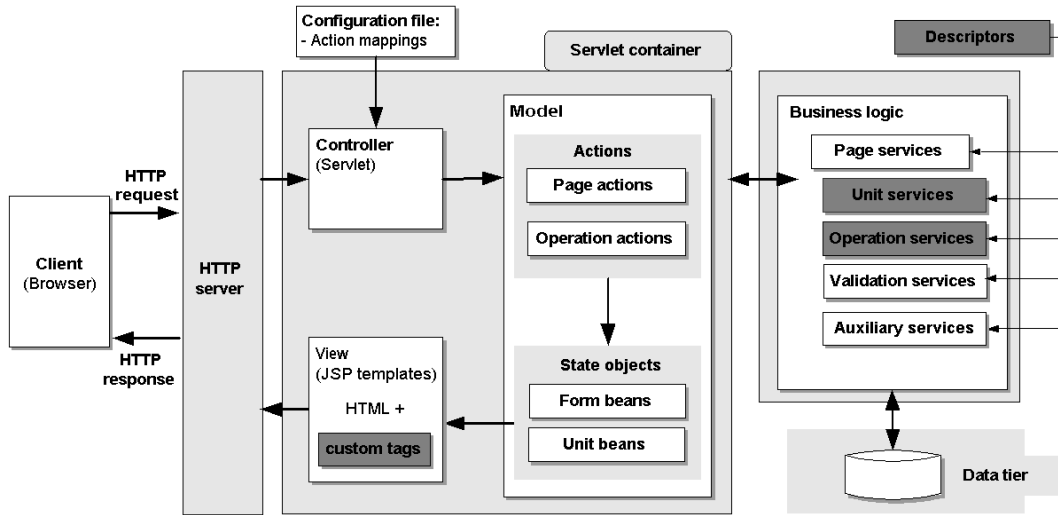


Figure 2. Architecture of an application developed with WebRatio. Taken from (Molinaroli, 2012)

The generation of such applications in WebRatio is similar to the generation of a traditional Web application (without personalization support) (Garrigós Fernández et al., 2009) (RQ2-MDD).

Related to IFML, it is important to stress that it does not consider concerns about modeling display content such as layout, style and look & feel in the application front-end, neither consider computer based bi or tri dimensional graphics, video games or high interactivity applications. IFML is dedicated to support data-intensive business applications (Brambilla & Butti, 2014).

Concerning our aspects of analysis, the personalization in WebML is limited to restrict the user and communities' access to resources, and it does not consider content personalization nor the personalization in the navigation or presentation (RQ1-architecture). Due to the personalization scope in WebML, an evolution scenario only would be possible in the personalized access, but this process in WebML does not take advantages from abstract modeling (RQ2-Spec) nor MDD capabilities (RQ2-MDD), because it is done manually.

2.4.3 *De Virgilio, 2012*

De Virgilio proposed the *Adaptation Modeling Language* (AML) as a high level abstraction language to specify complex adaptive web applications at the conceptual level (De Virgilio, 2012) (RQ2-Spec).

AML is proposed for the domain of data-intensive web applications, where the context is an important factor. The specification of the adaptation is made according to different context aspects, like the capacity of devices, quality of service (QoS) and user preferences (De Virgilio, Torlone, & Houben, 2007; De Virgilio, 2012). This proposal adopts a classical organization of a Web resource (content, navigation, and presentation), which is tied to the specification of the adaptation. AML is based on three main elements: the profile, the configuration, and the adaptation rule. The profile is a conceptual model for the description of various aspects of a context, such as the user, the device, and the location. The configuration describes, in abstract terms, a suitable adaptation. Each configuration has a content view, a hypertext definition, and a logical style sheet. The adaptation rule expresses the matching relationship between configurations and profiles. The profile activates an adaptation rule and this rule generates the configuration. If it is possible substitute the profile parameters into the rule, having the conditions in true, then, the profile activates the adaptation rule.

A tool called *flexible adaptation of web information systems* (FAWIS) supports the language and the design process; it follows an extensible architecture presented in Figure 3 (De Virgilio, 2012) (RQ1-architecture). A set of production rules enables the automatic adaptation of content delivery, and allows specifying declaratively how to build a configuration satisfying the adaptation requirements for a given profile. The proposed architecture consists of four modules, as shown in Figure 4.

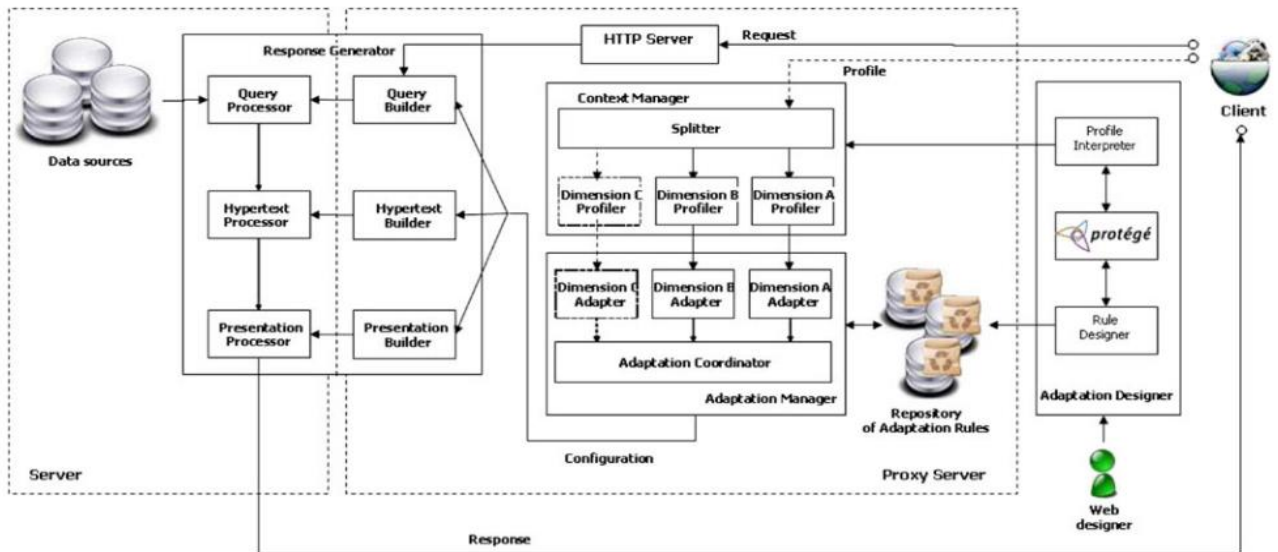


Figure 3. FAWIS Architecture. Taken from (De Virgilio, 2012)

The *Context Manager* (CM) is able to capture and classify a description of the client characteristics (the context). The *Adaptation Manager* (AM) takes as input the context of the client and generates a suitable adaptation configuration. The AM communicates with a repository of adaptation rules. Three modules, one for each level of the response, compose the *Response Generator* (RG): presentation, navigation and information recovery. RG generates all the components of a response to deliver over the Web and appropriate for the client profile (RQ2-MDD). It generates a set of adaptation statements that are the implementation of the configuration in specific languages. For example, adaptation statements can be SQL queries at the content level, XHTML statements at the navigation level, and CSS style sheets at the presentation level. The *Adaptation Designer* (AD) communicates with a repository of adaptation rules and allows the designer to define new adaptations of a previously unpredicted event. This way, it permits to extend the functionality of the tool. The web designer has to define manually a set of adaptation rules, collecting a large amount of possible configurations corresponding to the adaptation tasks to expose in the system.

Concerning our aspects of analysis, the use of a language with a high level of abstraction may drive the appropriated specification of the adaptation in an evolution scenario. Even so, the

large number of configurations to cover the possible adaptation cases may result in possible conflicts between them and in high complexity to the designer, resulting in a limitation to respond quickly to market demand. Regarding the deployment in a business environment, FAWIS generates adaptation statements that are portions of code, and as shown in the FAWIS architecture, a processor take these portions to build the client response. It shows the high interdependence between the web application and the modules to rationale about the adaptations, restricting the evolution of each application. Moreover, although this approach considers different modules for each type of personalization technique, it lacks ways to communicate with external modules like those provided by recommended systems, and lacks strategies to enable the reuse of context model by other applications.

2.4.4 De Bra et al., 2016

De Bra et al., proposed a tool for create adaptive courses with the Generic Adaptation Language (GAL) (Paul De Bra et al., 2016; Smits & De Bra, 2011, 2012). GAL is a Generic Adaptation Language for describing adaptive hypermedia, developed within the GRAPPLE project (Smits & De Bra, 2011). Subsequently, the GRAPPLE Adaptive Learning Environment (GALE) was developed as a generic and general purpose adaptation platform (RQ2-Spec). The Generic Adaptation Language and Engine (GALE) has its roots in AHA! (P. De Bra, Smits, & Stash, 2006) and redesign that proposal (P. De Bra et al., 2006) where not only the resources (pages) to perform the adaptation may be stored on remote web servers, but also the definition of the adaptation can be distributed (RQ1-architecture). This is called *open model adaptation*.(Van Der Sluijs, Hidders, Leonardi, & Houben, 2009). The content creation for the adaptive course (e. g. learning material) is completely separate from defining the adaptation and can be done using any HTML authoring tool (RQ1-architecture). Each adaptation model is stored in a different file, and possibly containing also the domain model and the relations. Figure 4 presents the architecture of an adaptive course under an open model adaptation. The author A and B may create his course in different servers, and the specification of the adaptation is taken from another remote server.

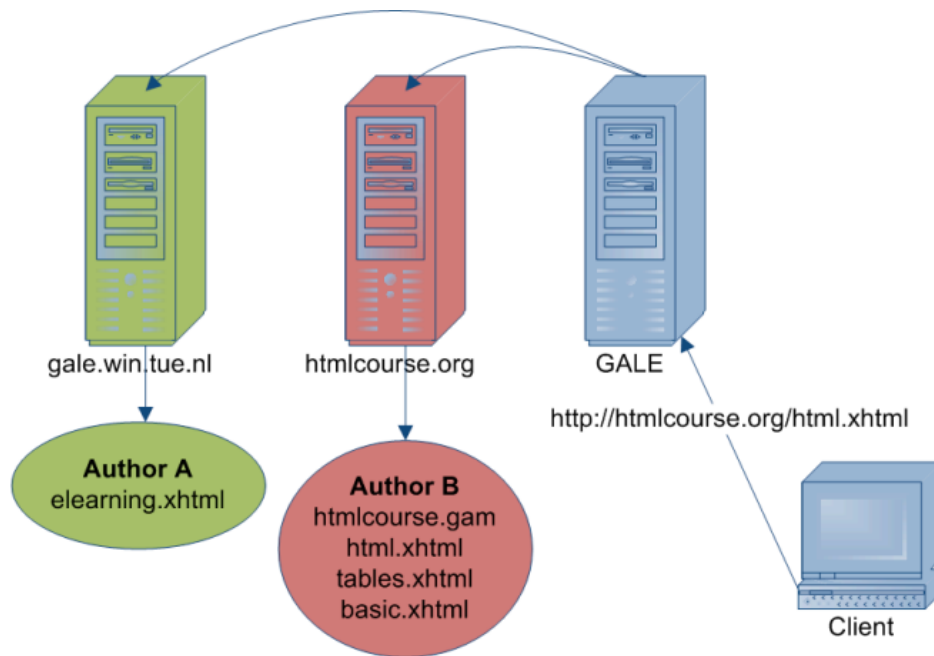


Figure 4. Architecture of an adaptive course under an open model adaptation

Only the individual adaptation based in his own profile is considered in GALE, group adaptation, or collaborative filtering, are not considered (Paul De Bra et al., 2016). GALE do not capture all the adaptation techniques (Montes Garcia, De Bra, Fletcher, & Pechenizkiy, 2014). Although the proposal is highly configurable, may present difficulties with the integration into a business environment, when the changes are few but continuous.

The authoring tool for GALE is ALAT (Paul De Bra et al., 2016). ALAT allows the authoring process for the domain and adaptation model of an application (course). This process is completely visual defining concepts in a hierarchical way, and attributes, relations and resources. The “adaptation model” in ALAT is generated from small code fragments shown in a blueprint (template). Blueprints define a structure common to all concepts of an application, and special concept types with additional structure for those.

The generation process of the domain and adaptation model in ALAT is not achieved with a transformation language but directly with a programming language, producing a file (RQ2-MDD). When the conditions to adapt resources changes, it is necessary to generate again the adaptation model.

2.4.5 Montes Garcia, De Bra, Fletcher, & Pechenizkiy, 2014

Montes Garcia proposed a framework to provide personalization in Adaptive web-based systems (AWBS) balancing personalization and privacy (García, 2014; Montes Garcia, De Bra, Fletcher, & Pechenizkiy, 2014). The framework performs within-browser adaptation and uses a user model fragmented in the client and the server (RQ1-architecture). The user selects the information to be stored in local machine or the data to be sent to the server. This framework is browser dependent but it does not need to install additional software on client. Montes Garcia et al. proposed a Domain Specific Language (DSL) to express adaptation in the development of AWBS (Montes Garcia et al., 2014) (RQ2-Spec). This way adaptation is decoupled from the business logic and by doing so, easier integration with existing applications is enabled.

The DSL uses CSS (Cascading Style Sheets) and is based on AHAM model that includes a presentation layer (Paul De Bra, Houben, & Wu, 1999; de Vrieze, van Bommel, & van der Weide, 2004). The adaptation effects are determined by the adaptation rules specified with CSS and JavaScript. This DSL supports effects like hiding fragments, append, insert, re-order, delete-note, etc. (RQ2-Spec)

This DSL only considers the personalization on the presentation, thus, it is a constraint to the evolution of personalization strategies. It lacks of the expressions to involve a user model limiting further an evolution scenario. Regarding to the process of integration into a business environment, it is necessary to have a JavaScript interpreter highly coupled with the adaptive application, limiting the fast response to market. This work lacks of a MDD support (RQ2-MDD).

2.4.6 A. I. Cristea, Smits, Bevan, & Hendrix, 2009

Layers of Adaptation Granularity (LAG) is a high-level adaptation specification language designed to support the authoring of courses (A. Cristea & Verschoor, 2004). The LAG language is defined as a grammar, and the version 2.0 is defined in (A. I. Cristea, Smits, Bevan, & Hendrix, 2009) (RQ2-Spec). The adaptive delivery engine to support the LAG version 5.0, is ADE: Adaptive Display Environment. This version improve functionality adding basic support for list objects, group selection of concepts and adaptation of specific navigational elements without increasing the difficulty of authoring that functionality (Scotton, 2013). This version

includes the support for adaptation techniques including adaptation of content fragments via content tags, links in the content as well as navigational link annotation and the emphasizing/ deemphasizing/ hiding/ disabling of links in the navigational elements of a course. With LAG it is possible to generically specify the adaptation strategies instead of specific condition-action rules which are tied to the course content. The LAG adaptation strategies are created separately from course content. LAG language is designed for adaptive systems which use the LAOS framework, which divides an adaptive system into five layers: presentation, adaptation, user, goal and domain model.

The adaptive strategies written in LAG are used in the AHA! delivery system (A. I. Cristea, Smits, Bevan, & Hendrix, 2009). Thus, the architecture of the adaptive application is based on the AHAM model, comprising three layers: component, storage (domain, adaptation and user model), and run-time layer (RQ1-architecture). This proposal does not take advantage of an MDD approach (RQ2-MDD). Concerning the evolution scenario, the LAG language offers a good level of modifiability, because the authoring tool permits the update relatively easy. However, the user must be an expert in the language, because of the fine-grained specification.

2.4.7 Sadat-Mohtasham & Ghorbani, 2008

Sadat-Mohtasham & Ghorbani proposed the adaptive web (aWeb) framework for building adaptive web systems (AWS) (Sadat-Mohtasham & Ghorbani, 2008). This work considers the adaptation based on individual user information (preferences, browsing history), environment parameters (geo-location, temporal), social parameters (user groups' profiles/group browsing patterns), and technology parameters (access and browsing technologies-devices). Sadat-Mohtasham & Ghorbani propose an adaptive process called "Synthesis", which create automatically a web page based on information fragments. The page is not previously in the server; it is built dynamically when the user request information.

The architecture of *aWeb* framework as shown in Figure 5, comprise two servers: a web server in charge of receiving all the HTTP request, and redirect the request to an adaptation server (RQ1-architecture). The Adaptation server is responsible for processing individual requests, querying different information sources, and forming the final response. It has four

components: 1) the *User Model Manager* that keeps users' personal information, preferences, and past (browsing) behavior. 2) *Device Profile Manager* that recognizes different devices and extract their characteristics, limitations, and capabilities, to provide the adapted presentation. 3) *Concept-Relationship Knowledge Provider* that uses concepts and the relationships between them, to better provide related information. 4) *Information Providers* that collects information scattered across various sources.

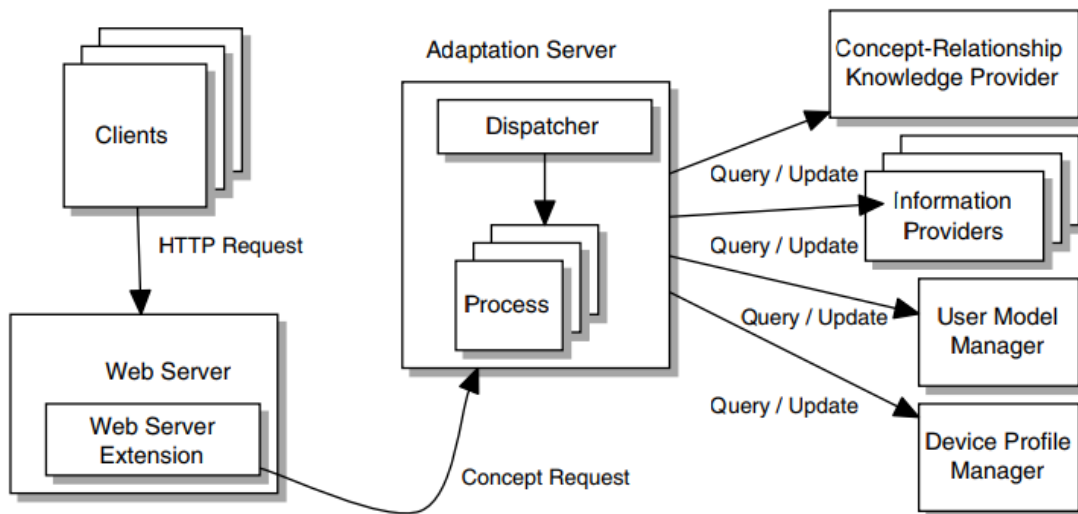


Figure 5. Architecture of the adaptive web (aWeb) framework

Since *aWeb* framework is defined in the context of family-based approach, the reuse is the main principle, however it lacks of mechanisms to enforce the reuse or indicate the way to apply it (RQ1-architecture).

The Adaptive Web Language (AWL) is the DSL proposed for the *aWeb* framework, to develop adaptive web systems and domain libraries (RQ2-Spec). It is conceived in the software product-line approach in order to increase the productivity, and follows the Aspect-Oriented Programming (AOP) paradigm as a mechanism to increase reusability. A domain library is a set of reusable artifacts and models developed in AWL. The *aWeb* framework uses AWL specifications and domain library artifacts to develop a new adaptive web application. AWL is designed to support instructions at fine-grained level to specify presentation web fragments. Examples of fragment types are text, image. It comprises a set of five modules: *Concepts and*

relationships are attributes that can represent an application domain entity/notion, or the structure of a piece of information; *Presentation* module used to describe how information is presented to the user; *Adaptation* module to abstract the adaptation strategies; *user model* to specify the structure of user model; and *functions and extended elements* as mechanisms to make domain libraries.

Adaptation strategies are specified in adaptation modules, through targets and adaptation statements (*pointcut* and *advice* in AOP terminology). A set of fragments are called *targets*. The adaptation is based on the concept of “separation of concerns” to foster quality attributes such as reusability and modifiability. The user based adaptation in the adaptation module uses the framework’s User Profile Manager (UPM) component to access users’ profile information, for example a browsing history. On the other hand, user model module consists of a set of attributes defined as overlay model (one attribute in the user model has a vector of values); and events are used to specify when and how the user model is updated.

In order to generate the final web adaptive application, *aWeb* framework uses the AWL compiler that translates AWL programs to the aWeb framework’s input specifications. Also, a Synthesis Engine (SE), that is a web server, interprets the site description given in RDF (W3C, 2004), transforms it to obtain a browser-ready code, and update tracking records with web-systems (RQ2-MDD). Hence, the generation process to get the final application do not uses any transformation language.

Concerning our aspects of analysis, *aWeb* framework has a good support to an evolution scenario because the adaptation model is a separated model and the AWL language is based on the concept of “separation of concerns”, however, the adaptation is reached only in the presentation specification, limiting variety of personalization strategies possible to specify. Furthermore, the concept model is not well connected to the other models in the language, and AWL lacks of means to let the designer use external code or cooperate with the existing web languages (Sadat-Mohtasham & Ghorbani, 2008). Those features limit the process to integrate efficiently personalization strategies in a business environment.

2.4.8 *Fan, Hussain, Younas, & Hussain, 2015*

Fan et al. proposed a software as a service (SaaS) personalization framework that is a cloud-enabled personalization framework which facilitate the collection of preferences and the delivery of corresponding SaaS services (Fan, Hussain, Younas, & Hussain, 2015). It is called “SaaS Personalization Framework”. This framework chooses suitable services automatically and configures them to offer personalized services.

SaaS Personalization Framework is shown in Figure 6 and comprises four fundamental layers (RQ1-architecture): Client Layer, Business Model Layer, Data Access Layer, and Data Layer. Client layer has a dedicated user model to collect and persist the user information, and supports a semantic client-side approach.

A Business Model Layer is responsible for receiving user data, storing data, processing data, generating patterns, building personalization configurations and recommending services. This layer is organized in some modules:

- Edge Component: it communicates with client-side components and synchronizes data between the client side and the server side.
- Observer Module: it gathers user preferences and environment information.
- Recommendation Engine: it applies data mining techniques on user data to discover recommendation patterns and predict user personalization.
- Personalization Engine: it receives the recommendation patterns and creates personalization configurations.
- Dynamic Service Composition: it is responsible for choosing and configuring the cloud services that will be allocated to users.

A Data Access Layer builds up data entities and performs task manipulation for the data system.

A Data Layer contains database systems and file systems that store data including user profiles, user preferences, user activities, service directory and ontology.

The framework also has a Cloud Services tier that is located in a remote cloud environment which supplies a group of candidate cloud services.

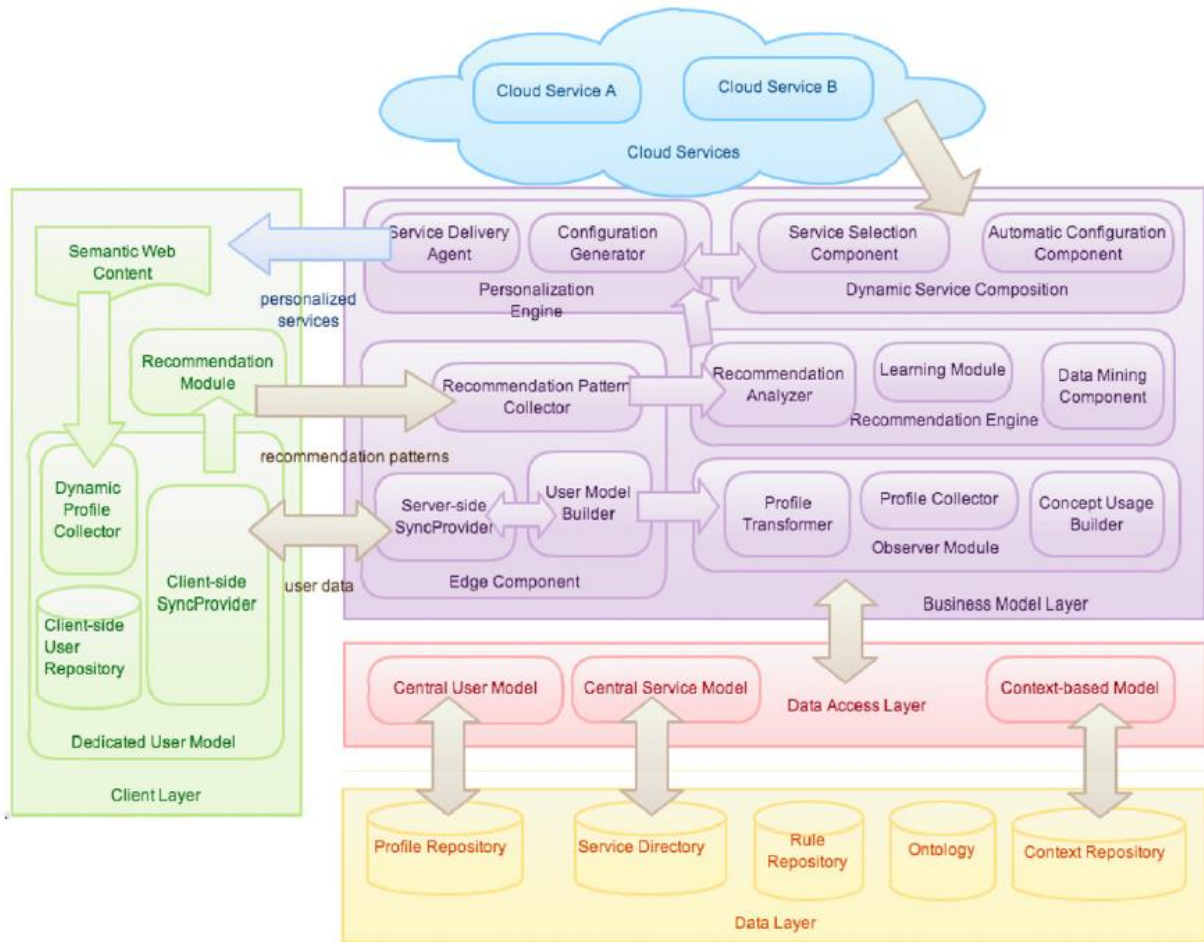


Figure 6. SaaS Personalization Framework

The framework performance was evaluated using two metrics: load time and reasoning time, in a prototype application of “personalized music”. *Load time* measures how long it takes for the prototype to capture user data and *Reasoning Time* measures how long it takes the system to generate rule-based personalized recommendations. The framework showed relatively low in load time and reasoning time. Concerning an evolution scenario, and a process to integrate the personalized behavior in business environment in a MDD context, the personalization process in the SaaS Personalization Framework is well supported by the recommendation engine implementing data mining procedures, however, the advices from experts that are continuously updated, are quite difficult to introduce in order to respond in short time to market. Moreover, this approach does not take advantage of an MDD approach (RQ2-MDD).

2.5 Summary

Table 1 summarizes the software architectures to support web personalization analyzed, and shows some features regarding a scenario of frequent updating of personalization strategies and the quickly response to market leading, titled as “Evolution” and “Integration into a business environment” labels respectively. Also describes some features of the MDD approach adopted such as the modeling language adopted, the transformation type (M2M/M2T), the transformations modeling languages, and finally the tool supporting each architecture in a MDD context.

As shown in Table 1, most of the software architectures analyzed do not offer a good support concerning the constant update/addition of personalization strategies; some of them only include partial personalization, only in the access (Ceri et al., 2007) or only in the presentation (Montes Garcia et al., 2014; Sadat-Mohtasham & Ghorbani, 2008). Others proposals use a fine-grained specification that may lead to create a lot of rules to cover different personalization scenarios causing conflicts and being unmanageable by designer (A. I. Cristea et al., 2009; De Virgilio, 2012; Garrigós et al., 2010). Or works that base the personalization reasoning to data mining techniques (Fan et al., 2015) resulting in a complex intervention of update.

It is notorious to improve the integration of personalized behavior into a business environment, looking for a less time-consuming process. It is also essential look for an approach in favor of the modifiability, stressing the importance of having a convenient separation of the specification of the personalization from the functional specification. It could be valuable to explore different approaches in the web applications development like component-based approaches. Lastly, it is also important to admit that the use of MDD practices could give clear benefits to the evolution scenario.

Work	Architecture	Approach	Evolution	Integration into a business environment	MDD			
					Language	Language definition	Model transformation type / transformation language	Tool
(Garrigós et al., 2010)	three-layer	Rule-based (Object-oriented)	Good	Limited	PRML	Grammar - BNF	M2M / QVT	AWAC
(Ceri et al., 2007)	three-tier	Model-based (data-intensive web applications)	Manually	Manually	WebML	--	--	--
(De Virgilio, 2012)	three-tier four-layer	Rule-based (data-intensive web applications)	Limited	Limited	AML	Own modeling primitives	M2T	FAWIS
(Paul De Bra et al., 2016)	distributed	Rule-based	Good	Limited	GALE	--	M2T	ALAT
(Montes Garcia, De Bra, Fletcher, & Pechenizkiy, 2014)	--	Rule-based	Limited	Limited	A DSL based on CSS	CSS Java script functions	No uses transformations	--
(A. I. Cristea et al., 2009)	three-layer	Textual	Good	--	LAG 5.0	Grammar	No uses transformations	--
(Sadat-Mohtasham & Ghorbani, 2008)	two-tier	Aspect-oriented programming (data-driven applications)	Limited	Limited	AWL	Grammar - EBNF	No uses transformations	<i>aWeb</i>
(Fan et al., 2015)	four-layer	Cloud services (SaaS-based cloud services)	Limited	Limited	--	--	--	--

Table 1. Web modeling languages with personalization support in a MDD context

Finally, the approach presented in this dissertation can be seen as a complement for the previously described architectures in several senses. The difference between our approach, and the approach of Garrigós and Virgilio is that they use a fined-grained personalization instruction like add/remove links, or show/hide texts, and we conceive components implementing a personalization goal instead of just focusing on one personalization technique. Additionally, we consider an external module dedicated to managing the information of users, contexts and groups, which makes possible the reuse of other personalized systems in the same enterprise; additionally, we permit the extension of personalization strategies by binding external components.

Chapter 3.

Reference Software Architecture to improve modifiability of Personalized Web Applications

This chapter presents the reference software architecture as an alternative to solve the maintainability problems faced in the evolution of personalized web applications. Section 3.1 introduces the modifiability and reference software architecture concepts. Section 3.2 presents related work in the modifiability assessment approaches. Section 3.3 describes the running example used throughout the chapter to explain the proposed reference architecture. Finally, section 3.4 describes the reference architecture, and presents the methodology to apply it.

3.1 Introduction

Personalization has demonstrated advantages in web applications (Alotaibi, 2013; Kwon & Kim, 2012), however, two factors complicate the inclusion of personalization strategies into a business environment: the frequent change of personalization strategies for each business (modifiability), and the technical complexity to integrate these strategies in a short time in a business environment.

In the area of software engineering, a reference software architecture is a consistent set of architectural best practices, which are designed with the aim of providing a template solution for a particular domain. It gathers the learning experiences from past projects, and offers guidance for future developments. It also enables reuse of architectural assets in a particular domain (Bengtsson, Lassing, Bosch, & Van Vliet, 2004; Kazman & McGregor, 2012; Reed, 2002; SEI, n.d.). A reference architecture is necessary to support increased complexity, scope and size of software systems; as well as, to support the dynamics of enterprises that need to respond more quickly to market demands.

We adopted the software modifiability definition as “the ease with which it [software system] can be modified to changes in the environment, requirements or functional specification” (Bengtsson et al., 2004). This chapter proposes a reference software architecture that has the software modifiability as the main architectural drive, and is based on component weaving process.

3.2 Analysis of software modifiability: Related work

Managing software evolution have been addressed in research for many years. Some of the more well-known modifiability assessment approaches include the software architecture analysis method (SAAM) (Bass, Clements, & Kazman, 1998) and the Oman taxonomy (Oman & Hagemester, 1992). The research community has used these assessment approaches to conduct experiments on modifiability analysis of web applications, and has proposed architectures intended to respond to the need of managing the personalization in web applications. Below, we briefly describe and analyze some of these assessment approaches.

Stella et al. compare the modifiability of a web application from the same requirements on three platforms [J2EE, .NET and Ruby on Rails (RoR)] (Stella, Jarzabek, & Wadhwa, 2008). In order to do so, they conduct a change propagation analysis on each implementation of the web application, and use three modifiability metrics (number of modified files, number of modified lines of code, and development time in man-hours to incorporate the change) to compare the extent to which each platform facilitates modifiability. Stella et al. observe that the web application developed on .NET required more modifications to source code, and more effort for implementing enhancements than the ones implemented with RoR and Spring-Hibernate. These results can be attributed mainly to two reasons: the enhancement of the .NET application required hand code mapping from the database to the entity object, whereas J2EE Hibernate and RoR ActiveRecord automated the mapping process. The second reason could be that .NET offers tighter coupling between concerns versus the cleaner separation of concerns in J2EE and RoR.

The model for assessing the maintainability proposed by Di Lucca et al. and the taxonomy of metrics presented by Oman and Heigemaster are other works related with the measurement of modifiability as a mean to compute the maintainability of web applications. Both are intended

to estimate the maintainability of traditional software, and establish other metrics specific to web applications, such as web page data coupling (Di Lucca, Fasolino, Tramontana, & Visaggio, 2004; Oman & Hagemaster, 1992). Besides metrics presented by Oman and Hagemaster there are several other methods supporting the analysis of software modifiability; such as, the SAAM (Bass et al., 1998), architecture level modifiability analysis (ALMA) (Bengtsson et al., 2004), and aspectual software architecture analysis method (ASAAM) (Tekinerdogan, 2004). SAAM takes several quality attributes as key issues: performance, security, availability, functionality, usability, portability, reusability, testability, integrability, and modifiability. Bass et al. (1998) categorize modifications as follows: extending or changing capabilities, deleting unwanted capabilities, adapting to new operating environments, and restructuring. Based on the quality attributes presented, Bass et al. (1998) propose different architectural styles that then are employed in the SAAM. ALMA is a scenario-based evaluation method focused on modifiability issues that provides quantitative predictions (via metrics and change impact analysis) about the modifiability of a system when it is confronted with different scenarios. ASAAM is an extension and refinement to SAAM intended to include explicit mechanisms for identifying architectural aspects and components. Similar to SAAM, ASAAM takes as input a problem description, requirements and architecture descriptions for:

1. developing a candidate architecture to provide a design that will be analyzed with respect to the required quality factors and aspects
2. developing scenarios from different stakeholders.

Stafford and Wolf propose an automated technique for architecture dependency analysis that builds graphs of architectural components and captures their static and behavioral relationships (Stafford & Wolf, 2001). Stafford and Wolf's approach is implemented on top of an architectural description language, and it serves mainly to support architects in the navigation and analysis of the set of components related to a given particular concern.

According to the presented exploration of modifiability assessment approaches, our evaluation approach can be seen as a complement to the application of the aforementioned architectural assessment methods to test offering cases and change scenarios. We present the evaluation in the next chapter.

3.3 An e-commerce system – the running example

To illustrate our approach, we provide a real scenario in the e-commerce domain from a Brazilian enterprise called VTEX. This enterprise is a leader in e-commerce technology in Latin America, and is dedicated to the commercialization of software as a service. VTEX offers solutions to enterprises that have websites in different market segments.

Taking into account the experience and knowledge of the e-commerce business by VTEX, we extracted various scenarios of personalization to define our running example. The functionalities included are in the categories of product discounts and product recommendation. In an e-commerce domain, it is common to offer product discounts to attract new clients and to gain their loyalty. The discounts category offers customers a specific discount (e.g., 10%) for accumulated purchases greater to a fixed value over a period; and, highlights the discount in the web page. The recommendation category is adopted to increase the conversion rate. The conversion rate is the percentage of website visitors who actually purchase a product on the site.

In our scenario, the recommendation functionality consists of several modalities: the history of recent products visited by customers with a link to the detailed product description (Figure 7), and recommend products based on similarity measures between users and/or products. The products recommended to a user are those preferred by similar users, or those similar to the product that the user is searching.

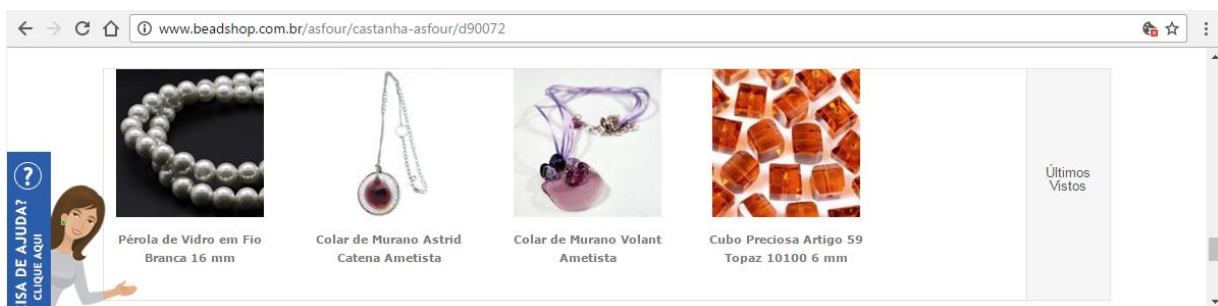


Figure 7. History of recent products visited by customer

3.4 Reference architecture

This section presents a conceptual view of the reference architecture, and makes use of our running example. It also presents a methodology to apply the architecture.

The reference architecture designed to support personalization on web applications has the software modifiability as the main architectural drive. We presented an example in the e-commerce domain to illustrate the importance of the modifiability feature in personalized web software. To increase the customer's loyalty, an enterprise may want to define periodically new personalization strategies like different types of product discounts. These discounts may last from a few hours to many days, so it is valuable to permit the change of personalization strategies in a short period. Therefore, web applications should have the capacity to include, or discard different personalization strategies in a short period; that is, software should be modifiable. This reference architecture proposes the use of component weaving as an alternative to tackle the challenges of including personalized behavior.

3.4.1 Reference architecture description

The standard MVC web application architecture supports the proposal. It separates an application into three main logical layers: the model, the view, and the controller. Objects in the model layer encapsulate the data specific to an application; view objects are the user graphical interfaces; and the objects in the controller layer are intermediaries between view objects and model objects, and coordinate tasks for an application. In this chapter, we refer to the model layer as 'persistence'. Our approach manages each personalization strategy as a specialized component, which can be added or removed from basic application. Additionally, we add three specific modules to facilitate the integration of personalization strategies as specialized components:

1. the personalization controller (PC) module
2. the connector to personalization administrator (PMAAdmin connector) module
3. the personalization model administrator (PMAAdmin) module.

Figure 8 shows the reference architecture where bold lines mark the modules to manage personalization process.

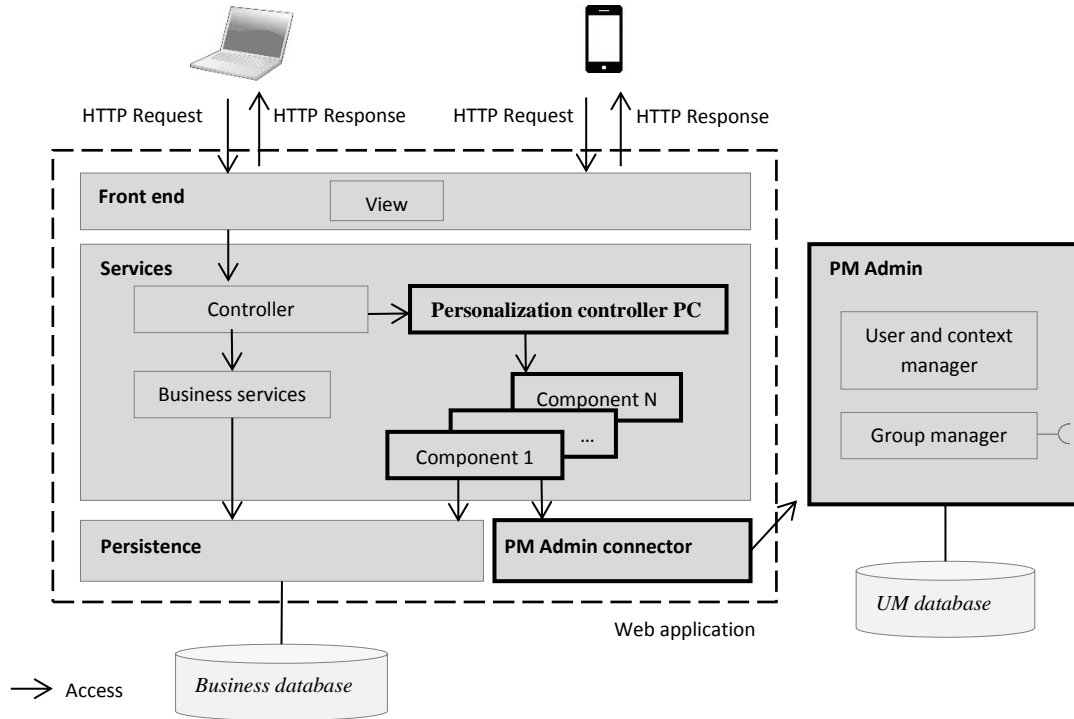


Figure 8. Reference architecture for personalized web applications

Regarding to the set of specialized components (component 1, ..., component N), each component implements a personalization goal or strategy, and possibly each component uses different personalization techniques. For instance, in our running example, the ‘*discount by accumulated value*’ component may use techniques like ‘adaptive selection’ to capture customer purchases in a period and the ‘link annotation’ technique to emphasize the discount. In the same way, in the *recommendation category*, the associated component may use techniques like ‘filter collaborative’ to find similar products and/or users. Note that each specific personalized component implements a personalization goal as a key feature instead of just focusing on one personalization technique.

The *PC module* is responsible for processing events detected by the web application, and coordinates the personalization effects by accessing the specialized components. PC module manages the specific functionality of personalization to avoid mixing the web application’s basic functionality with the personalized behavior; in this way, PC simplifies the modifiability of personalized behavior. In our running example, as a strategy to offer a personalized discount

via accumulated purchases, the PC module identifies the relevant events to achieve this purpose, and interacts with the appropriate component. In this case, PC module identifies when a customer browses a product list, and demands to the ‘*discount by accumulated value*’ component for a personalized discount value and the type of visual emphasis. The personalized discount component obtains the value of the customer’s accumulated purchases to determine the discount percentage and the highlighting mode.

The *connector to personalization model administrator (PMAAdmin connector)* serves as a bridge between specialized components and the external *PMAAdmin* module. The specialized components send the data request to the *PMAAdmin* module through this connector. The *personalization model administrator (PMAAdmin)* is an independent module placed outside the web application. It manages two types of information: inferred data, and redundant data.

To build the inferred data, the system usually gathers records from diverse sources as transactional databases or unstructured files. Periodically, an ETL (extract, transform and load) process collects and stores these records in a data warehouse; after that, diverse data mining or machine-learning techniques build the relevant information about users, context or items. In our running example, the inferred data could include user profile, product profile, user groups, product groups, similar users, similar products, and prediction models.

The web personalized system gathers information while users interact with the system, such as type of user device, geographical position, visited products or the navigation track. This information allows us to establish user behavior. *PMAAdmin* is intended to manage a minimum amount of redundant data to establish user behavior, and to allow their use through a personalization processes. Note that *PMAAdmin* excludes transactional information. As this module uses its own database, at design time, analysts must decide which strategies to utilize as a means to maintain consistency among replicas.

This reference architecture separates the transactional process module from the personalization functionalities. In this way, both personalization strategies and user data gather techniques run in parallel with respect to transactional operations. This separation facilitates software maintainability tasks, such as the addition of new gathering mechanisms, the addition of new personalization strategies, and the change prediction models. This facility gains value in changing and dynamic scenarios as e-commerce.

On the other hand, several software web-personalized applications can share the *PMAAdmin* module within the same enterprise (Figure 9); i.e., applications like telesales, call centers, and logistics. A unique *PMAAdmin* module reduces the development effort for each personalized application, and improves the collection of more precise information over time. The more users' information the system has, the greater are the possibilities of offering a personalized experience.

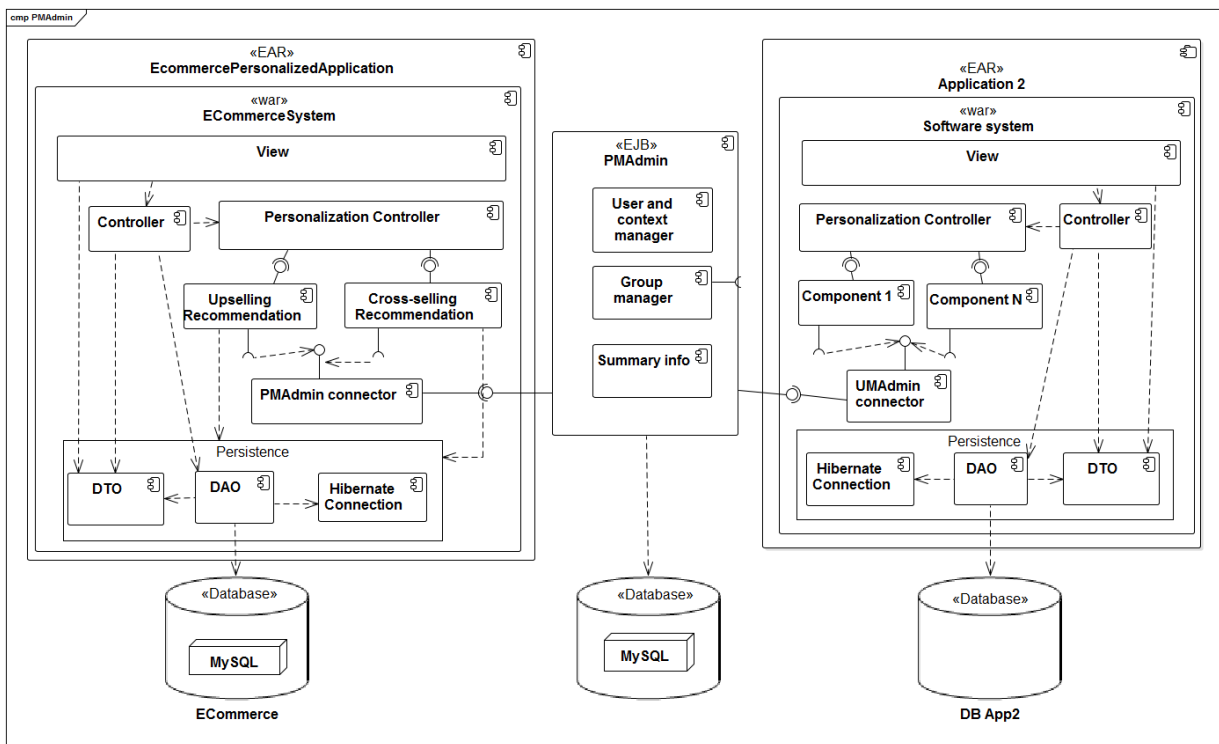


Figure 9. PMAAdmin with multiples web applications, represented with UML component diagram

Finally, notice that the persistence layer serves both the web application, and the specialized personalization components.

3.4.2 Methodology to apply the reference architecture

This section establishes a process to guide the developers in the adoption of the reference architecture. The web application can previously exist, or be designed independently; in

consequence, the next step is to integrate personalized behavior. Figure 10 shows the process systematically which is represented using a UML activity diagram.

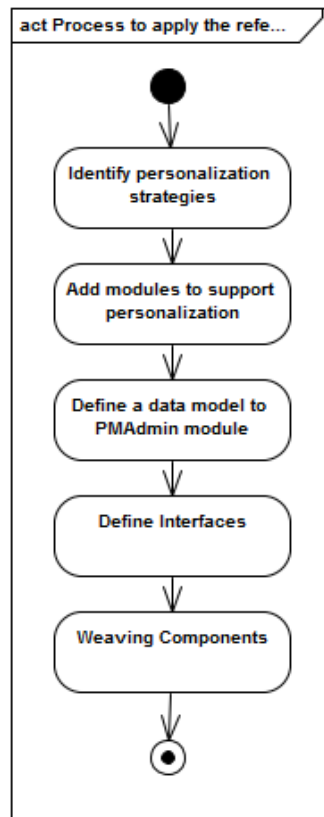


Figure 10. Process to apply the reference architecture represented with a UML activity diagram

3.4.2.1 Identify personalization strategies

The goal of this step is to identify the personalization strategies as different portions of functionality, encapsulate them in independent specialized components with defined interfaces, and identify the information sources to support the strategies.

One or several components materialize each strategy. Components may require data from the persistence layer in web application, from the *PMAAdmin* module, or from the user interaction.

In our running example, the functional requirements correspond to product discounts and product recommendation categories: the *discount by accumulated value* and *product history* components encapsulate this functionality. Note here that the functional decomposition is

mainly about the identification of business strategies instead of the selection of a particular personalized technique. Designers can introduce, or remove strategies from global web applications. Additionally, to recommend products based on similarity measures between users, the system may use techniques such as collaborative-filtering or content-based analysis. Furthermore, to offer the discount strategies, the system requires accumulated user purchases. Thus, both types of recommendations require information from the *PMAdmin* module.

3.4.2.2 Add modules to support personalization

This step consists of adding three modules to enable the ensemble of personalization strategies. PC, and *PMAdmin* connector modules are created inside the web application, whereas *PMAdmin* module is created as an external component.

3.4.2.3 Define a data model to PMAdmin module

It is required that the designer determines which type of information from *PMAdmin* module is necessary for each specialized component. For example, which user, items and context information will be extracted from the *PMAdmin* module. After, designers define the *PMAdmin* data model, and ways to gather its information; i.e., ETL processes from transactional databases, social networks or another web sources or application. Note that each personalization strategy can require an additional batch process to provide the intended functionality, i.e., machine learning algorithms, data mining approaches or another technique could run periodically over the *PMAdmin* module.

Although the goal is to minimize data redundancy, in some cases, the *PMAdmin* module could contain redundant data with respect to the storage level on the web personalized system. In these cases, designers must define synchronous or asynchronous replication mechanisms such as online triggers or periodic batch processes.

3.4.2.4 Define interfaces

This step allows the establishment of components that will interact with other parts of the application. Designers must specify the functionality that each *specialized component* will provide or require from other modules. Several interfaces allow specifying these interactions:

the relation between PC and *specialized components*, the relation of specialized components with persistence layer and/or PMAAdmin connector, the relation between the *PMAAdmin* connector and the *PMAAdmin* module.

In the running example, one interaction is identified between PC module and two specialized components: *discount by accumulated value* and *product history* components. Thus, it should have an interface that guarantees the input data for first component, the *username*, and the *username* and *customer visited page* for the second one. A second interaction is between the two specialized components and the *PMAAdmin* connector. It should guarantee the connection to *PMAAdmin* module. A third interaction is between the *PMAAdmin* connector and the *PMAAdmin* module. The interface in this case should guarantee the retrieval of the *total value of customer purchases* over a period for the first component, and search the *last customer viewed* products for the second one. The interaction between modules and personalization components corresponding to our running example is presented in Figure 11.

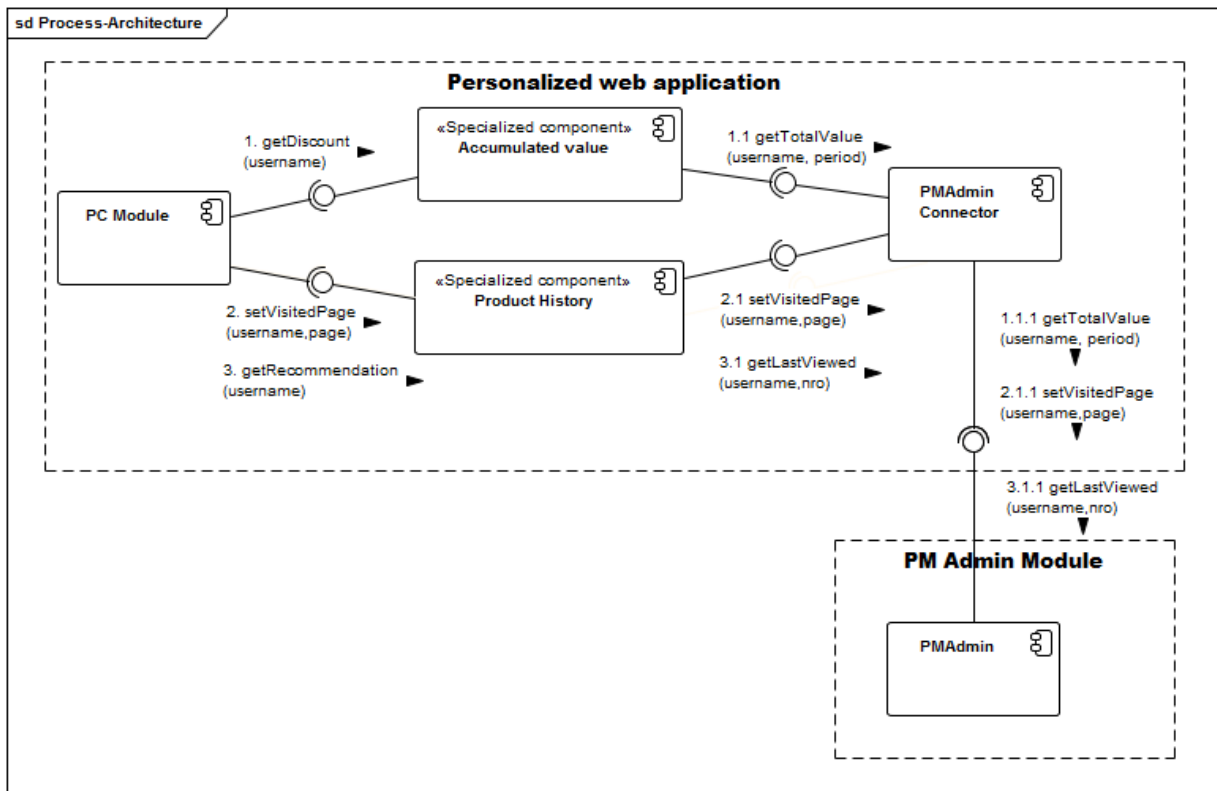


Figure 11. Interaction between modules and personalization components

3.4.2.5 Weaving components

The aim of this step is to compose a connected personalized web application. To do that, designers must attach the web application with the specialized components, with PC module and with PMAAdmin connector.

Thus, it is necessary to add the specialized components previously identified, and adjust the defined interfaces of the PC and PMAAdmin connector. In our running example, it means to add the discount by accumulated value and product history specialized components to the global proposed web structure and to update the PC and UM Admin connector architectural modules according to the defined interfaces.

3.5 Conclusion

This chapter presented the reference software architecture as an alternative to solve the maintainability problems faced in the evolution of personalized web applications. It mainly focused in scenarios of constant update/addition of personalization strategies (modifiability), and the technical complexity to integrate these strategies in a short time in a business environment.

The proposed reference software architecture presented the following benefits or advantages: This reference architecture manages each personalization strategy as a specialized component. Each component may implement different personalization techniques inside (Brusilovsky, Kobsa, & Nejd, 2007) (instead of just focusing on one personalization technique). This feature allows to easily add or remove the personalization features from basic web application and diminish the complexity originated in the management of fined-grained personalization instructions.

Also, this reference architecture proposes a personalization controller (PC) module that coordinate the access to specialized components avoiding to mix the web application's basic functionality with the personalized behavior, boosting the modifiability of personalized behavior.

A connector to personalization administrator (PMAAdmin connector) module is also proposed, as a single module to communicate the web application with a specialized external module that

manage the user information (PMAAdmin). This feature reduce the complexity to access to user information.

This reference architecture also includes a personalization model administrator (PMAAdmin) module as an independent module placed outside the web application, in order to administrate personal information, context and user group information, as inferred and redundant data to support the personalization strategies. Due to user and context information is essential to support personalized applications, this feature permits that several personalized web applications can share the PMAAdmin module within the same enterprise, reducing the development effort for each application, and improving the collection of more precise user information over time.

This reference architecture separates the transactional process module from the personalization functionalities, allowing personalization strategies and user data gather techniques run in parallel. This separation makes easy software maintainability tasks, such as the addition of new gathering mechanisms, the addition of new personalization strategies, and the change prediction models. This feature is valuable in changing and dynamic scenarios as e-commerce.

Finally, a methodology to apply the reference architecture in a business environment is provided.

Although the proposed reference software architecture tackles the modifiability issue and the complexity to integrate personalization strategies in a business environment, is still necessary to validate the proposal with real enterprise information, and explore the possible benefits from model driven development (MDD) approaches and technologies to automatically derive the web applications code.

The next chapter present a controlled experiment to evaluate the software modifiability of the proposed reference software architecture using a group of personalization strategies taken from a Brazilian e-commerce enterprise.

Chapter 4.

A controlled experiment to validate the Reference Software Architecture

This chapter presents the controlled experiment to evaluate the software modifiability of the proposed reference software architecture. We used a real business case taken from a Brazilian e-commerce enterprise. Section 4.1 presents the reference process used to design and conduct the experiment. Section 4.2 presents the design of the experiment. Section 4.3 describes the experimental units. Metrics are determined in section 4.4. Section 4.5 shows the implementation of the prototype. Finally, threats to validity and results are presented in sections 4.6 and 4.7 respectively.

4.1 Experiment process

Experimentation in software engineering includes to prepare, conduct and analyze experiments properly. This chapter follows the experiment process presented by Wohlin et al. (2012). The experiment process presented is divided in five activities.

Scoping: It consist in scoping the experiment in terms of problem, objective and goals. Specifically the following aspects are defined:

- a. Object of study (What is studied?)
- b. Purpose (what is the intention?)
- c. Quality focus (which effect is studied?)
- d. Perspective (Whose view?), and
- e. Context (Where is the study conducted?)

As a result of this activity, the goal definition is stated. Wohlin et al. (2012) suggests a goal definition template to ensure that important aspects of an experiment are defined before the planning and execution take place. The template is as follows:

*Analyze <Object(s) of study>
for the purpose of <Purpose>
with respect to their <Quality focus>
from the point of view of the <Perspective>
in the context of <Context>*

Planning: The design of the experiment is determined in this activity. It includes to state the personnel, the environment, the hypothesis, the variables and its measurement scale, the experiment design and if is necessary the instrumentation, and the threats to the experiment. As a result of this activity, the experiment design is established.

Operation: This activity is about three steps: preparation, execution and data validation. The subjects and the material needed is prepared. The execution is done following the planning stated before, and the validation consist in make sure that the collected data is correct and valid to the experiment. As a result of this activity, the experiment data is obtained.

Analysis and interpretation: In this activity, the data is analyzed and interpreted. The conclusions are obtained.

Presentation and package: This activity is concerned with presenting and packaging of the findings. The results are documented and present in a proper way. Consider that an experiment will never provide the final answer to a question.

4.2 Design of the experiment

Following the goal-question-metric (GQM) suggested in Wohlin et al. (2012), we stated that the goal of the experiment was to *analyze* the reference architecture *for* evaluating it *with respect* to its modifiability *from the point of view of* the software developer and *in the context of* personalization strategies taken from a Brazilian e-commerce enterprise.

The *context* of the experiment was a test case composed of five change scenarios, and a software application implemented under two architectures: experimental and control architecture. The experimental architecture is proposed in this chapter; and control architecture corresponds to a standard model-view-controller (MVC) web application architecture.

The test case comprises five change scenarios described in terms of application changes involving personalization strategies. These scenarios allow us to evaluate the support of future changes, and consequently, evaluate software modifiability under an architecture. The strategies were taken from a real Brazilian e-commerce enterprise (<http://www.vtex.com/>). A software engineer implemented the same test case in both architectures under the supervision of a senior engineer, resulting in two web applications: experimental and the control web applications, respectively. Later, the senior engineer counted the number of needed changes in both web applications to accomplish the five change scenarios and finally we compared the results. The

experiment was performed off-line (not in an industrial software development) and staffed by a software engineer. With the experiment, we answered the primary research question (RQ):

RQ: To what extent is the reference architecture able to allow modifiability?

4.3 Experimental units – Test Case

To compare architectures, and assess their modifiability, we use a test case as the same point of comparison. The test case consists of a set of representative change scenarios. Here, a change scenario is the representation of the modifiability requirements when a web application needs to include personalization strategies. The change scenarios should reveal differences in the architecture.

In the definition of scenarios process, first we interviewed two stakeholders at VTEX enterprise: The Strategy Manager and the Project Manager. The idea was to focus on possible and repetitive modifications related to personalization. Later in the process, we (the researchers) discussed the scenarios and selected the most relevant. The selection included those scenarios with an impact in different architecture layers, those requiring and updating user information as well as context information, and those that may use computational intelligence or collaborative filtering in the personalization strategy.

The scenarios we found were the following:

4.3.1 Change scenario 1

The broad audience of the system demands a wide visual support for different customers, especially to middle-aged adults that present visual limitations. Thus, the software needs to be modified to adapt to the user's visual limitations. The software should identify if the registered customer has visual limitations (if he/she belongs to a specific group) and if so, the software should increase size letters and update colors.

4.3.2 Change scenario 2

According to ubiquitous web applications paradigm, this kind of application may adapt its services and software structure to user context; including the device, network, time and location context. Thus, the software needs to be changed to adapt its services and presentation to device

context and user location. Specifically, the software should give a discount if the user is located near to a store; also, the site's graphical interface should adjust according to size of the device.

4.3.3 Change scenario 3

The personalized recommendation methods are typical strategies in e-shop. Thus, the system shall include a recommendation method showing the last five products visited by the user.

4.3.4 Change scenario 4

The recommendation strategies need to be more meaningful for customers. The recommendation method needs to be changed to implement techniques like collaborative filtering and content-based filter. The system shall implement two recommendation methods: upselling recommendation and cross-selling recommendation. Upselling recommendation consists of offering the customer additional or complementary products for purchase. The system uses collaborative filtering techniques to find products bought by similar customers but having higher cost. Cross-selling recommendation consists of offering the customer alternative products for purchase. This time, a content-based analysis allows finding similar products to one that the customer is searching.

4.3.5 Change scenario 5

The software needs a personalized discount strategy. Thus, the software shall implement a product discount strategy by accumulated value. This kind of discount offers customers a specific discount percentage for accumulated purchases greater to a fixed value over a period.

4.4 Metrics

To answer the RQ, we counted the number of changes needed to complete the implementation of the change scenario, in terms of:

- the number of files added or removed
- the number of methods added
- the number of code lines added or modified.

4.5 Prototype implementation

Experimental and control architectures were used to implement all the change cases, resulting in two web applications: the experimental and the control, respectively. Both web applications were implemented using Java programming language and Java Server Pages (JSP) technology, Hibernate framework for managing the data persistence, MySQL 5.6 as a database engine and Wildfly 8.2 as the application server. In the experimental web application, we implemented the PMAdmin module under Enterprise Java Beans (EJB) technology and used the Java persistence API (JPA) for managing data persistence.

Figure 12 shows the architecture for the implementation of the change scenario 4 according to the reference architecture³, where *upselling recommendation* and *cross-selling recommendation* are the specialized components. The architectures for the implementation of the changes scenarios 1, 2, 3, and 5 are shown in appendix A. Appendix B shows some screenshots of the implemented personalized web application.

4.6 Threats to validity

The main threat to internal validity in this experiment is the subject experience. This threat was alleviated by considering another engineer with good experience (senior) on web programming that cross-checked the results. The main threat to external validity of the experiment is the generalization of the results. This threat was alleviated by selecting representative change scenarios extracted from a real enterprise. However, it is not possible to generalize the results because we only worked in the context of one enterprise, and only performed a comparison against a standard method. The main threat to construct validity is the misunderstanding of the ‘modifiability’ concept. To alleviate this threat, we defined the metrics based on the definition of modifiability from Bengtsson et al. that is a work focused on the modifiability of software architectures (Bengtsson et al., 2004). The metrics selected address our RQ in a direct way. Finally, and regard to conclusion validity, we present the results as preliminary validation since we do not use statistical validation.

³ The architecture and the implementation of the remained scenarios are available on <http://telesalud.udea.edu.co/Ecommerce/>

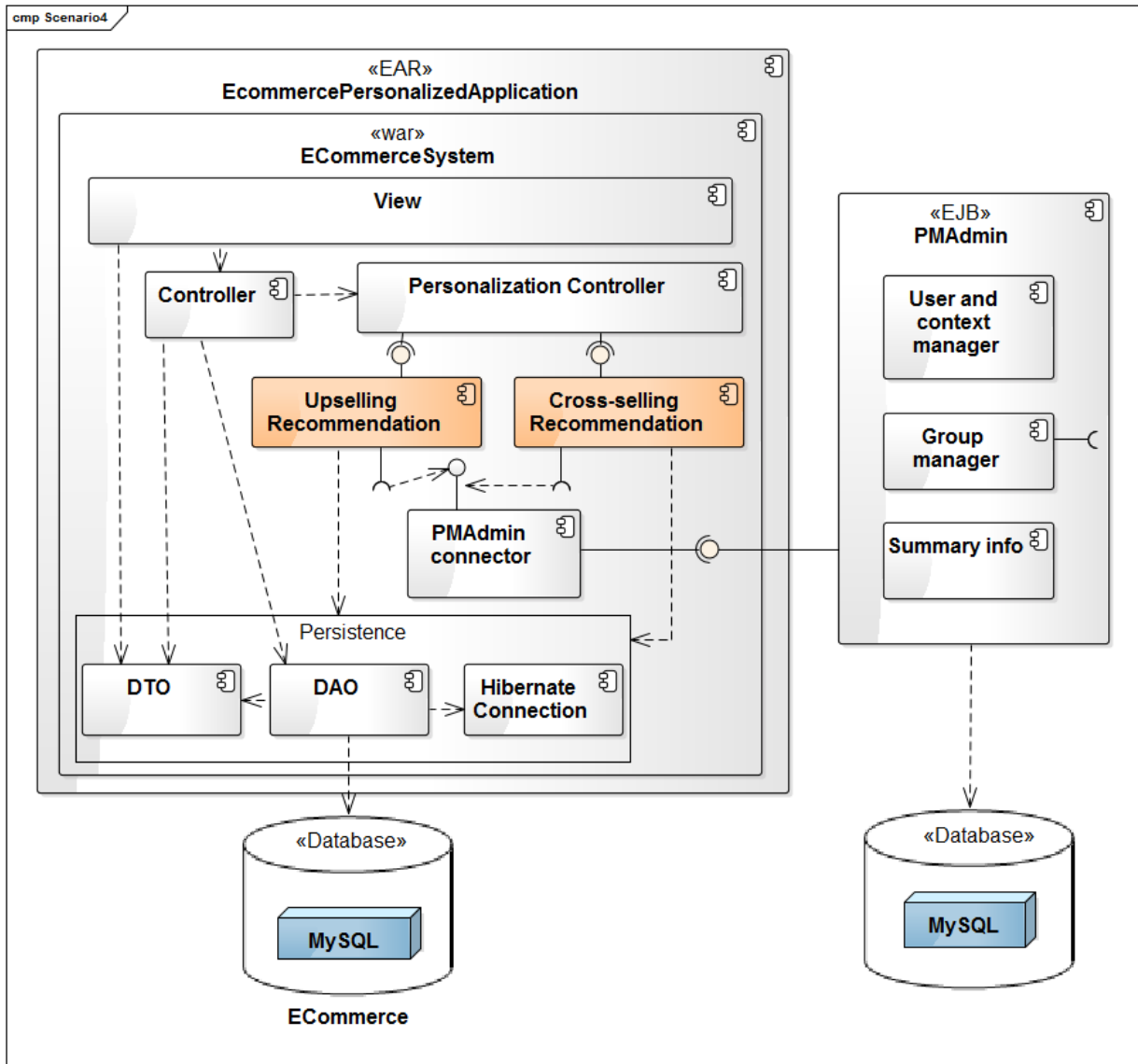


Figure 12. Architecture for the change scenario 4 according to the reference architecture

4.7 Results

Tables 2 and 3 show the result of counting the needed changes to achieve the test scenarios under the implementation of experimental and control architectures respectively⁴. Table 4 shows the changes made in the PMAAdmin module corresponding to experimental architecture. We counted the number of files added or removed, differentiating classes (including interfaces) from configuration files; the number of methods added or removed; the number of code lines added or removed differentiating classes (including interfaces) from configuration files.

The fourth change scenario reports two counts because this scenario has two parts: (–) removing an existing recommendation strategy and (+) adding two new different recommendation strategies.

No. change scenario	No. of files added or removed		No. of added or removed methods	No. of code lines added or modified		Total
	Classes	Configuration files		Classes	Configuration files	
1	6	2	3	12	2	24
2	6	0	11	136	0	153
3	4	1	4	31	1	41
4 (+)	4	0	5	56	0	65
4 (-)	-4	-1	-4	-31	-1	-41
5	3	1	3	12	1	20

* Files, methods or lines removed have a negative number.

Table 2. Results of the execution of change scenarios under standard MVC Architecture

We create the PMAAdmin module before implementing the change scenarios with the experimental architecture. However, this effort was not included in Table 3, because the enterprise makes this task once, and it is not part of the personalization strategies in a particular web application. Additionally, we add the PC module and PMAAdmin connector inside the web application, with its interface and its implementation class. Table 3 excludes these changes, because they only are done once to prepare the web application to accept various personalization strategies.

⁴ The table with a detailed description of changes in the experimental and control architectures are available on <http://telesalud.udea.edu.co/Ecommerce/>

We used the data obtained to answer our RQ.

RQ: To what extent is the reference architecture able to allow modifiability?

Focusing on the column labelled ‘*No. of files added or removed*’ in tables 2 and 3, we observed that the number of classes was reduced in the reference architecture. It might be explained because in the control architecture we needed to create files mapping persistence matters; by contrary, in the experimental architecture those functionalities were managed inside the *PMAAdmin* module.

No. change scenario	No. of files added or removed		No. of added or removed methods	No. of code lines added or modified		Total
	Classes	Configuration files		Classes	Configuration files	
1	3	0	3	13	0	18
2	6	0	9	136	0	151
3	2	0	6	33	0	41
4 (+)	4	0	7	58	0	69
4 (-)	-2	0	-6	-33	0	-41
5	2	0	3	13	0	18

* Files, methods or lines removed have a negative number.

Table 3. Results of the execution of change scenarios under reference architecture

No. change scenario	<i>PMAAdmin</i>			
	Number of elements added or modified			
	Classes	methods	Code lines	Total
1	3	1	1	5
2	0	0	0	0
3	2	2	2	6
4 (+)	2	2	2	6
4 (-)	-2	-2	-2	-6
5	1	1	1	3

* Files, methods or lines removed have a negative number.

Table 4. Changes on the *PMAAdmin* module in the execution of change scenarios

In addition, as opposed to control architecture, in the experimental architecture was not necessary to add or modify configuration files. It may be explained because in the experimental architecture those configurations are already in the *PMAAdmin* module. An additional advantage in the development process can result of reducing the configuration files changes in the experimental architecture and transferring them to centralized *PMAAdmin* module, because it

may potentially reduce the error introduction that always appears in the modification of software.

Regarding data for the number of added or removed methods, we can observe that the number of changes has been slightly increased in the experimental architecture. This fact occurs because this architecture proposes the PC module to filter the personalization petitions. Thus, the methods augmented.

Concerning changes in code lines and the number of methods, the experimental architecture reports a small increase. The control architecture includes methods for implementing functionalities, and for working on persistence matters. By contrary, the experimental architecture only includes methods for implementing the functionalities because *PMAAdmin* module manages the persistence matters. However, the calls to the PC module could explain the increase in code lines in the experimental architecture.

Analyzing the overall results, the reduction of configuration files, and the number of classes in the experimental architecture was meaningful. It leads to show advantages in software modifiability. In the control architecture, the code line number was slightly greater but not significant. It could be explained because the experimental architecture demands separate personalization strategies in different components, and adds new modules that demand the creation, and modification of new files and methods.

Finally, we have tested the experimental architecture against a control architecture, but it is necessary to extend the experiment to contrast the number of changes when the enterprise has more than one application to be adapted with personalization strategies. We believe that in that scenario, the benefits from *PMAAdmin* could be more visible.

This chapter presented the controlled experiment to evaluate the software modifiability of the proposed reference software architecture. As an improvement for the adoption of the reference architecture, we argue that it is possible to enhance the modifiability and evolution of personalized web applications using an MDD approach. The generation of code could diminish the handwritten code lines, and decrease the number of possible errors in the update process of personalization strategies.

The next chapter presents a model-driven approach based on the proposed reference software architecture that automatically derive the web applications code permitting an opportune update of personalization strategies, and consequently respond in short time to the changing market.

Chapter 5.

MAMPA: A model-driven approach to enhance the modifiability of personalized web applications

With the aim of permitting an opportune update of personalization strategies, and consequently respond in short time to the changing market, this chapter presents a **Model-driven Approach to enhance the Modifiability of Personalized Web Applications (MAMPA)**; this approach is intended to improve the modification of personalization strategies in web applications. The MAMPA approach is based on the reference software architecture described in chapter 3 (L. V. Cobaleda et al., 2016), and in particular, it resolves the difficulties caused by the manual procedures existent in the process of applying the reference software architecture. Section 5.1 presents the running example used through the chapter. Next, the framework to adopt the MAMPA approach is described in Section 5.2. Finally, the contribution of the MAMPA approach is presented and compared with the related works in section 5.3.

5.1 Running Example

This chapter uses personalization strategies adapted from a Brazilian enterprise in the e-commerce domain. This enterprise is a leader in e-commerce technology in Latin America, and is dedicated to the commercialization of software as a service. This software offers solutions to enterprises that have web sites in different market segments.

A common strategy used in the e-commerce domain consists in using different types of recommendations to increase the conversion rate. The conversion rate is the percentage of website visitors who really purchase a product on the web site. Hence, personalization strategies may vary in quantity or in time, while they are enabled to users, or about the user or context information required to generate a recommendation. Regarding the quantity, a software system may include one kind of recommendation or include several of them at the same time. Regarding the time, personalization strategies may last from some minutes to several months. Lastly,

regarding the required information, the recommendation strategy may or may not require user and context information.

Some of the recommendation strategies implemented by the Brazilian company are: (i) recommendations based on similarity measures between users and/or products; that is, the products recommended to a user are those preferred by similar users, or those similar to the product that the user is searching. In that case, the system uses collaborative filtering techniques to find products bought by similar customers (ii) Upselling recommendation, which consists in offering to the customer additional or complementary products for purchase. (iii) Cross-selling recommendation, which consists in offering to the customer alternative products for purchase. In this case, a content-based analysis allows finding products like these that the customer is searching. (iv) An alternative method of recommendation may involve user information like the customer navigation history. For example, suggesting the last products visited by the customer itself or by a group of users similar to the customer. And, (v) the use of diverse types of discounts is another typical strategy to personalize an e-commerce application.

Also, those strategies may vary in quantity, time, and user and context information required. For example, the use of a personalization strategy for offering discounts depending on the number of previous purchases of customers, and possibly varying the percentages according to customer fidelity.

As a running example, we take the following change scenario to illustrate a possible occurrence in e-commerce domain: Initially, the web application named “*Vtex-ecommerce*” has one recommendation technique implemented as a reusable component, and for the example, it is the “*upselling recommendation*”.

The change consists in:

- a) replacing the actual recommendation strategy for a different one showing the last products viewed in the customer navigation history. This new recommendation is called “*navigation history*”. And,
- b) adding a personalized discount strategy. The discount strategy gives a fixed discount percentage if the customer has accumulated a specific value in the purchases of the last month. This discount strategy is called “*accumulated value discount*”.

5.2 Framework to Adopt the MAMPA Approach

This chapter proposes a framework as an alternative to enhance the evolution and modifiability of personalized web applications making use of the model-driven methodology. The proposed model-driven approach, MAMPA, is based on the reference software architecture described in chapter 3. MAMPA improves the previously explained reference software architecture taking into account the key activities in a model-driven process such as modeling and transformation activities. The framework provides a process, a tool, and some other assets to integrate personalization strategies into the reference architecture. As the framework follows a model-driven approach, it supports the continuous and reliable changes of personalization components in an enterprise. Concretely MAMPA reduces the handwritten code lines, and decreases the number of possible errors in the update process of personalization strategies.

The framework is envisioned in the contexts in which an enterprise needs to add personalized behavior into an existent web application.

The framework has three assumptions:

- (i) the enterprise has, in its web application, a scenario of a constant update process of personalization strategies as part of a business strategy.
- (ii) The diverse personalization strategies are implemented as software components. And,
- (iii) the enterprise needs to add or update, in short time, the personalization strategies supported by the software, as a way to rapidly adapt itself to market.

The framework to adopt MAMPA approach considers two typical situations in enterprises: when it is their first time adding personalization strategies, and when they need to support continuous changes of personalization strategies to quickly adapt the strategies to the clients or to the market. Supporting continuous changes in personalization strategies has a significant impact for enterprises.

For instance, in e-commerce domain, some personalization strategies are well known like personalized discounts or personalized recommenders, and their impact depends on the time in which they are released, on the targeted market segment, and on the recent market behavior. Thus, in those cases, the immediate and reliable changes of personalization strategies are vital for an enterprise to select the best personalization strategy to apply at that moment. The MAMPA approach description in this chapter focuses on software evolution supporting

addition, update and continuous change of personalization strategies. The following subsections present a detailed description of the framework, introduce the modeling language to specify the personalization strategies, describe the transformation process adopted, and describe the corresponding prototype we implemented in order to validate the approach.

5.2.1 Framework description

Figure 13 depicts the framework to support the MAMPA approach; to facilitate its explanation this Figure is divided into four regions, each region is marked with letters A, B, C and D:

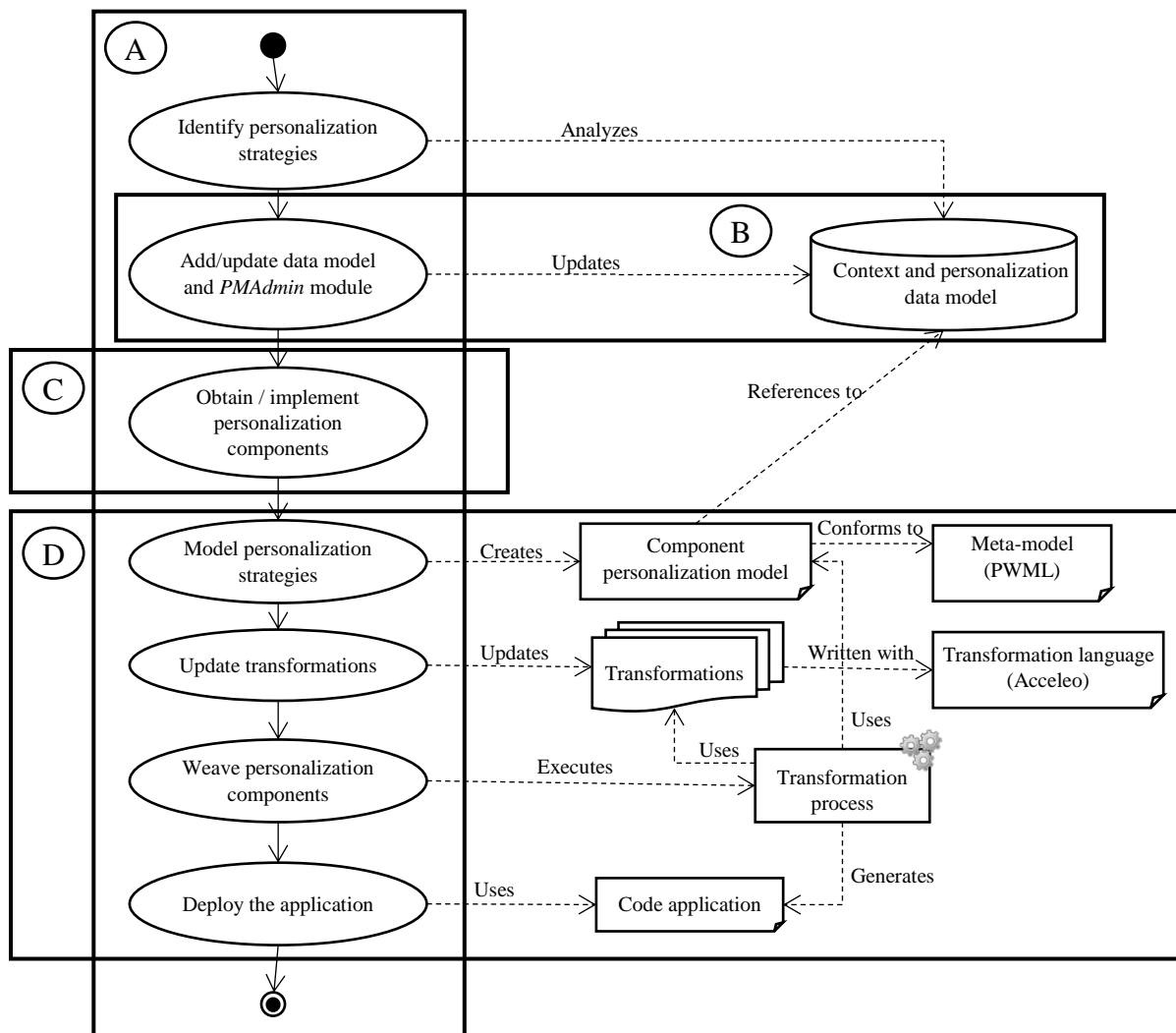


Figure 13. Framework to support MAMPA approach

- The region A shows the process of seven activities to support the integration of personalization strategies into an existent web application on permanent evolution. This process extends an initial process (five activities) to apply the reference architecture presented in Figure 8, having into account a MDD approach. The first activity in this region, “Identify personalization strategies,” remains with the initial intention of recognizing the set of personalization strategies that will be encapsulate and integrated in the architecture of the personalized web application. In our running example, the two new personalization strategies identified in region A are the recommendation strategy named “*navigation history*,” and the discount strategy named “*accumulated value discount*.” The subsequent activities are explained in the other regions.
- Region B highlights a preparation phase for supporting the personalization of web applications. This region contains the activity “Add/update data model and PMAAdmin module,” in which the *PMAAdmin* module is adjusted to support the subsequent personalization strategies, both, in the data model as in the module itself. The *PMAAdmin* module is in charge of managing the user and context information and sharing it throughout the enterprise. This module is transversal to the enterprise because other applications in the same enterprise could need the same user information and therefore, they could use this module. In our running example, the information added to context and personalization data model is the product page visited by the customer to support the recommendation strategy, and the value of purchases accumulated by the customer to support the discount strategy.
- Region C containing the activity called “Obtain / implement personalization components,” and reflects the assumption of having software components. In the case that components do not exist, then they should be implemented. Concerning our running example, the personalization strategies are implemented and are available to use them as reusable components.

- Region D encloses activities and artifacts making evident the application of the model-driven paradigm. This region consist of four activities:
 - “*Model personalization strategies*”: In this activity, different personalization strategies are specified in a high-level of abstraction using the modeling language: *Personalized Web Application Modeling Language* (PWML). The personalization model conforms to a meta-model defined as an extension of UML. The next section explains PWML in detail.
 - “*Update transformations*”: In this activity, the user specifies or updates in the transformations, the source code needed to integrate the personalization components into the web application. The update is written in Acceleo language.
 - “*Weave personalization components*”: In this activity, the user executes the transformations to generate the application code that intertwine the personalization components.
 - “*Deploy the application*”: Finally, the whole code is prepared to deploy the final application.

The activities corresponding to region D are explained in detail in the following sub-sections.

The MAMPA framework focuses on technical factors and omits considerations about organizational, managerial and social factors that are also important for technology adoption. The MDD approach does not define exactly which models to use through the development process, neither the abstraction level nor the process to adopt them. Consequently, for each domain is indispensable to define its own methodology in accordance with the organization’s context and requirements (Brambilla et al., 2012).

5.2.2 PWML: Personalized Web Application Modeling Language

The overall process of a model-driven approach comprises the modeling tasks of the application and the realization of a series of transformations until the executable code achieved, thus, modeling languages are essential in the conceptual modeling tasks. There are two classes of modeling languages: Domain-Specific Modeling Languages (DSML or in short DSL), which are designed specifically for a particular domain, context or company; and General-Purpose

Modeling Languages (GPML), which can be applied to any sector or domain (Brambilla, Cabot, & Wimmer, 2012). The definition of a new DSL is justified because it provides specialized means for some specific domain in which development would take much more effort otherwise. The Unified Modeling Language (UML) is widely known and adopted in software engineering. Moreover, the use of UML extension mechanisms like profiles, are commonly used in MDD approaches.

The framework to adopt the MAMPA approach deals with the evolution of personalized web applications, and contains an UML profile called *Personalized Web Application Modeling Language* (PWML) to achieve this purpose.

Figure 14 shows the meta-model that represents the abstract syntax of the PWML as an UML profile. The stereotypes defined in the PWML covers the specification of a personalization model. This model comprises three layers: Application, User Interface and Component Layer. They are stereotyped as «*ApplicationLayer*», «*UILayer*» and «*ComponentLayer*» respectively, and all of them extend the *Package* meta-class.

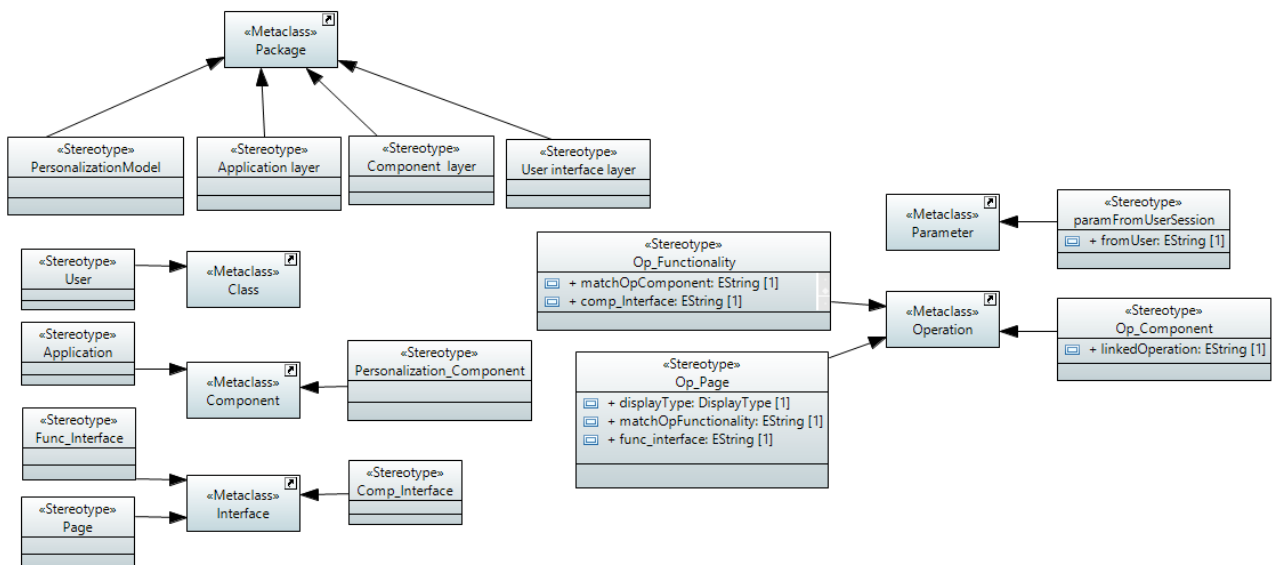


Figure 14. Meta-model for PWML

The application layer

The main stereotype in the Application Layer is «*Application*», which is defined as an extension of the *Component* UML meta-class, and represents the entire personalized web application; therefore, in the personalization models should exist only one instance of it. In our running example, the component having the «*Application*» stereotype is our e-commerce application named “Vtex-ecommerce.”

The group of functionalities that provides a web application is represented with the stereotype «*Func_Interface*» (functionality interface) extending the UML meta-class *Interface*. In consequence, an element stereotyped with «*Application*» only may have associations to elements stereotyped as «*Func_Interface*». In our running example, since our web application “Vtex-ecommerce” offers two personalized strategies, then, we have two interfaces as «*Func_Interface*» called “Recommender” and “Discounts,” corresponding to the recommendation and discount strategies respectively.

An interface consists of a group of operations. Hence, «*Func_Interface*» includes operations stereotyped as «*Op_Functionality*» (operation functionality), defined by the *Operation* meta-class. In our running example, the “Recommender” interface has an operation called “getRecommendedProducts” to support the recommendation strategy, and the “Discounts” interface has an operation called “getTotalDiscount” to support the discount strategy.

The «*Op_Functionality*» stereotype has two tag values: <*matchOpComponent*> indicating the operation name from a personalization component in charge of providing the implementation of the functionality; and <*comp_Interface*> indicating the name of the component interface that specifies the operation. In our running example, the value for <*matchOpComponent*> tag is “getDiscount,” and the value for <*comp_Interface*> is “Discount_Interface.” Note that the name of the operations in the interfaces stereotyped as «*Func_Interface*» can be different to the operations name from personalization components, since those two tag values, <*matchOpComponent*> and <*comp_Interface*>, make the match between the functionalities offered by web application and the personalization components. This feature permits the model to guarantee the flexibility in the weaving of components.

Figure 15 shows the use of the stereotypes defined for the Application Layer model in our running example.

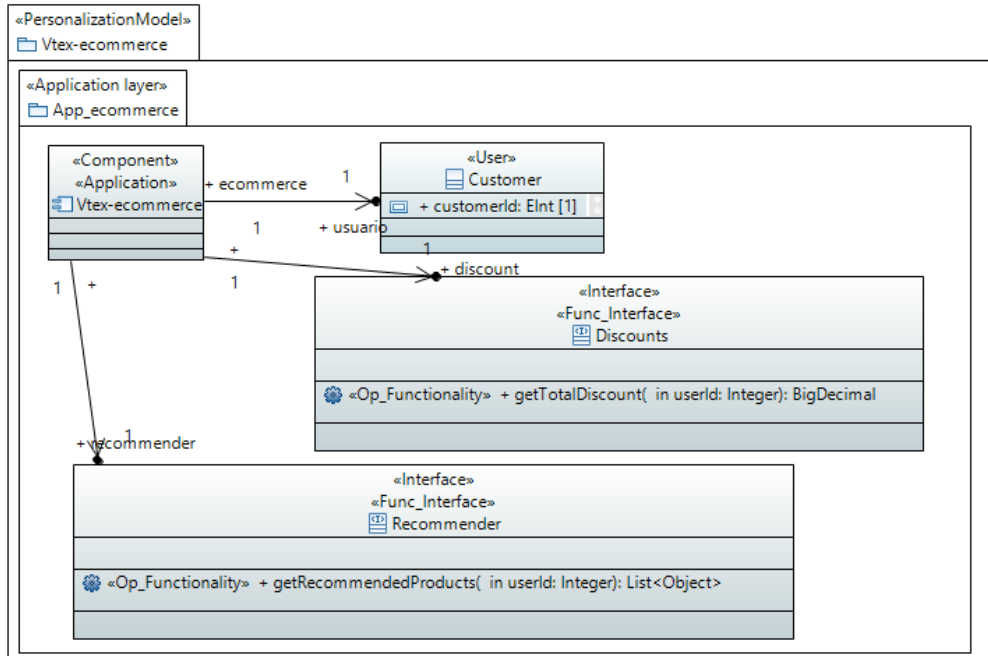


Figure 15. Model of the Application Layer corresponding to our running example using PWML

In the modeling of a personalized application it is fundamental to have the user information (the customer information in our running example); therefore, the «*User*» stereotype contains the personal features related to the personalization strategies employed in the web application. This stereotype extends the UML meta-class *Class*, and the features are represented as class *attributes*.

The web pages that display the personalized information are marked with the «*Page*» stereotype, which extends the *Interface* meta-class. In our running example, both, the information of the recommended products and the discount percentage is displayed in a web page named “Details.” The interfaces having the «*Page*» stereotype belong to the User Interface Layer as presented in Figure 16. Also, these interfaces have operations in charge of retrieving information to display the results in the web page. Those operations are tagged with «*Op_Page*» (operation page) stereotype.

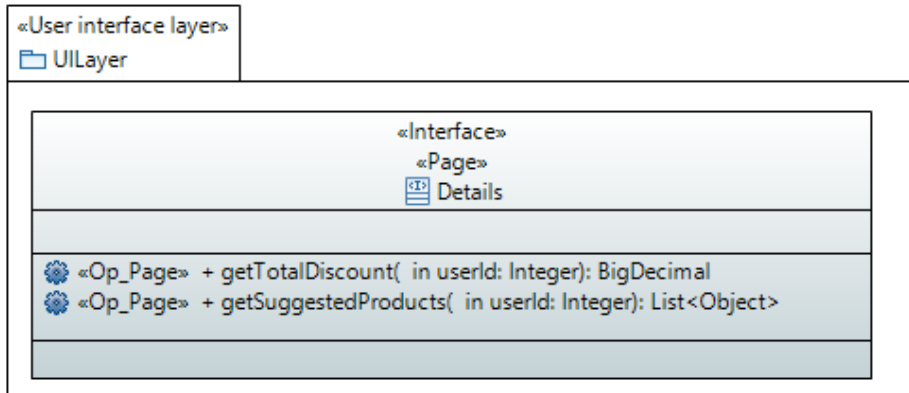


Figure 16. Model of the User Interface Layer corresponding to our running example using PWML

In our running example, the web page named “Details” has two operations tagged with «Op_Page»: the “getSuggestedProducts(userId)” operation to show the list of products given by the “navigation history” recommendation strategy. And the “getTotalDiscount(userId)” to display the user discount percentage given by the “accumulated value discount” strategy.

The «Op_Page» stereotype has three tagged values: <displayType>, <matchOpFunctionality> and <func_interface>. The <displayType> tag value indicates how the information is visually presented in the page. For instance, in our running example, the “getSuggestedProducts(userId)” operation has three tagged values. The <displayType> tag value will be “Photo-Roll” in order to indicate that the list of recommended products would appear as a Photo-Roll. The possible values for <displayType> are implemented as a data type Enumeration, with the values: Photo-Roll, List, Table, etc. The <matchOpFunctionality> tag value makes a match between the operation from a web page, «Op_Page» and the system functionality «Op_Functionality». Thus, the <matchOpFunctionality> tag value has the name of an operation tagged as «Op_Functionality». In our running example, the <matchOpFunctionality> tag value will have “getRecommendedProducts.” Lastly, the <func_interface> tag value indicates the name of the functionality interface that contains the operation indicated by <matchOpFunctionality>. In our running example, the <func_interface> tag value will have “Recommender.” Figure 17 shows the use of the tag values in the User Interface Layer corresponding to our running example.

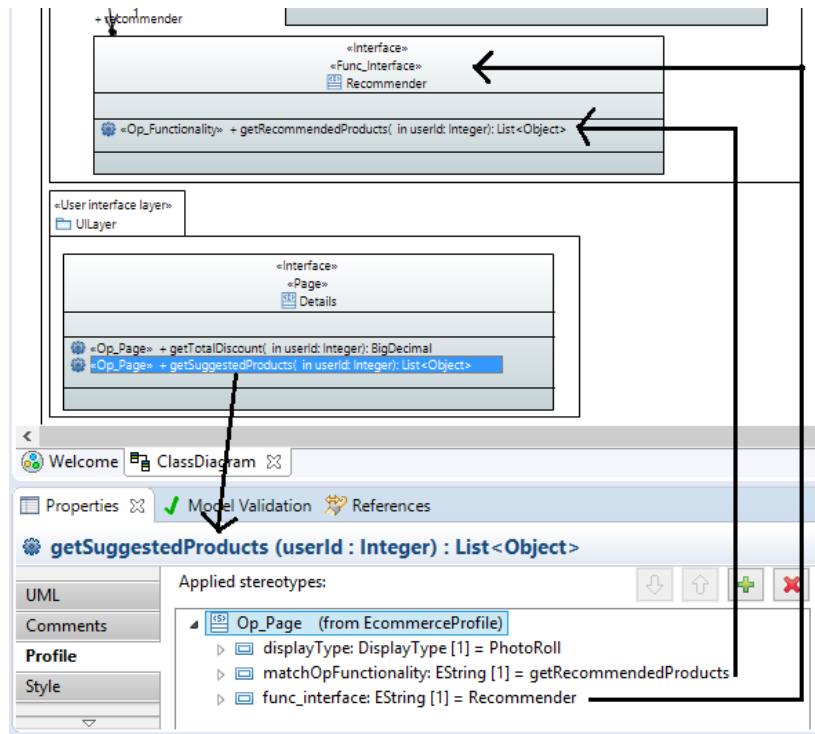


Figure 17. Use of tag values in the User Interface Layer corresponding to our running example

The component layer

Once the basic functionalities are identified for the web application, it is needed to define what components will offer the personalized behavior. Those components are defined in the Component Layer. The *«personalization_Component»* stereotype represents the software components that implement a personalized behavior and extend the *Component* meta-class. The personalization component interface is stereotyped as *«Comp_Interface»* (Component Interface) which extends the *Interface* meta-class. In addition, *«Op_Component»* (Component Operation) stereotype represents the operations offered by the personalized component in its interface.

In our running example, there are two personalized components stereotyped as *«personalization_Component»*: the component to provide the “navigation history” recommendation strategy that is named “NavigationHistory,” and the component to provide the “accumulated value discount” strategy, that is called “AcummulatedDiscount.” The

“NavigationHistory” component has an interface stereotyped as «*Comp_Interface*», and called “NH_Interface,” with two operations “getSuggestedProduct(userId)” and “setVisitedProduct(userId, idProd),” stereotyped as «*Op_Component*». Similarly, the “SpecificDiscount” personalized component has an interface stereotyped as «*Comp_Interface*» and called “Discount_Interface,” with one operation “getDiscount(userId),” stereotyped as «*Op_Component*». The personalization components are modeled in a Component Layer as presented in Figure 18.

In the personalized components, some operations may be required by others. For example, in our running example, and regarding “navigation history” recommendation strategy, the web application should save the pages visited by the customers to keep the navigation history and use it in subsequent recommendations. Thus, the «*Op_Component*» stereotype has a tag value <linkedOperation> pointing to another Component Operation in the same interface. In our running example, the “getSuggestedProduct(userId)” operation has in the <linkedOperation> tag value, a value of “setVisitedProduct.”

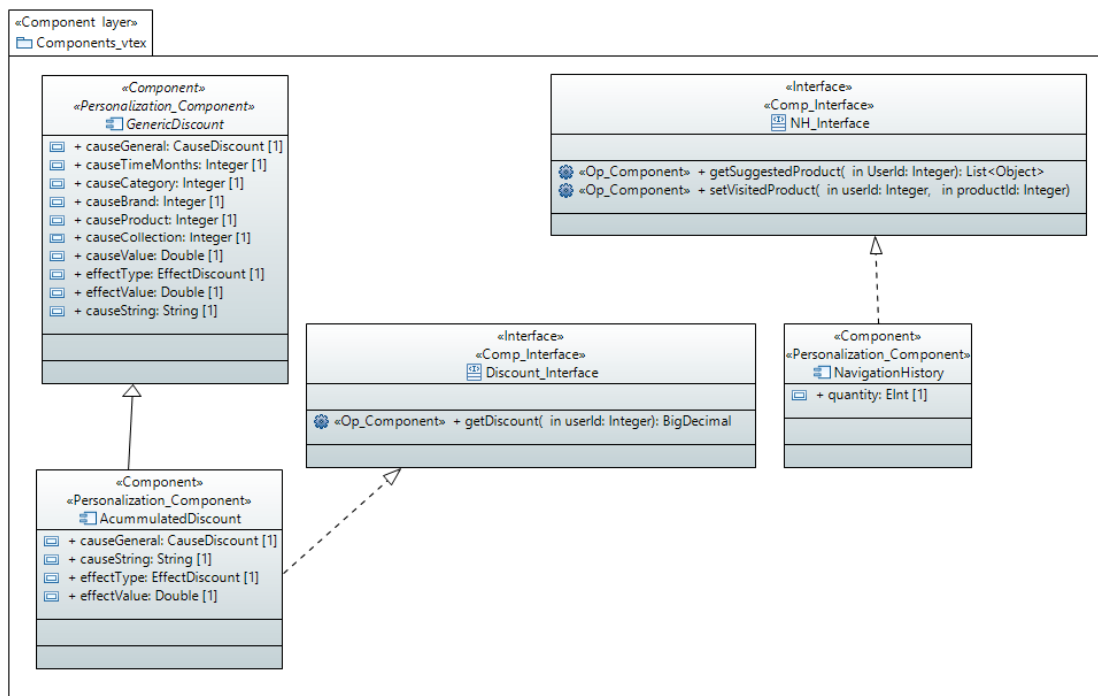


Figure 18. Model of the Component Layer corresponding to our running example using PWML

In some cases, an operation has parameters that come from information in the current user session. Those parameters are stereotyped as «*paramFromUserSession*», extending the *Parameter* meta-class. This stereotype has a tag value called <*fromUser*> and its value is the name of an attribute from the «*User*» stereotype.

Figures 15, 16 and 18 show the model corresponding to our running example. Table 5 presents a summary of the defined stereotypes and properties in the UML profile for PWML.

UML Meta-class	Stereotype	Tagged values
Package	<i>PersonalizationModel</i>	
Parameter	<i>paramFromUserSession</i>	<i>fromUser</i>
Package	ApplicationLayer	
Component	<i>Application</i>	
Interface	<i>Func_Interface</i>	
Operation	<i>Op_Functionality</i>	<i>matchOpFunctionality</i> <i>comp_Interface</i>
Class	<i>User</i>	
Package	ComponentLayer	
Component	<i>Personalization_Component</i>	
Interface	<i>Comp_Interface</i>	
Operation	<i>Op_Component</i>	<i>linkedOperation</i>
Package	UserInterfaceLayer	
Interface	<i>Page</i>	
Operation	<i>Op_Page</i>	<i>DisplayType</i> <i>matchOpFunctionality</i> <i>func_Interface</i>

Table 5. Summary of the defined stereotypes and properties in the UML profile for PWML

Note that some constraints cannot be defined by the graphical meta-model, so additional constraints using OCL (Object Constraint Language) as well-formedness rules were defined. Those constraints make the language more precisely defined leading to models with higher quality (Brambilla et al., 2012).

Examples of constraints are, that an interface stereotyped as «*Comp_Interface*» can only be defined for a component stereotyped as «*Personalization_Component*», and the operation with the stereotype «*Op_Component*» can only be defined into an interface with «*Comp_Interface*» stereotype, in the Component layer. Similarly in the Application Layer, an interface stereotyped as «*Func_Interface*» can only be defined for a component stereotyped as «*Application*», and the operation with the stereotype «*Op_Functionality*» can only be defined into an interface with

«*Func_Interface*» stereotype. Finally, in the User Interface layer, an operation stereotyped as «*Op_Page*» can only be defined into an interface stereotyped as «*Page*».

5.2.3 Transformations

Model-to-Text (M2T) transformations are generally employed to automate the code generation from models. “Code-generation may be described as the vertical transition from models on a higher-level of abstraction to lower-level artifacts” (Brambilla et al., 2012). The MAMPA framework uses a M2T transformation language that is template-based, to generate the code of the web applications. A template separates the static code (simple text fragments) from dynamic code (meta-markers). The static code refers to code that is generated as the same way for every model element, and the dynamic code refers to which is obtained according to the model specification. The template-based approach helps in the maintainability and evolution of the transformation. Another benefit is that the template represents the structure, which contains the generated code. Although there are several template-based languages like XSLT⁵, JET⁶, Xpand⁷ and MOFScript⁸; MAMPA uses Acceleo⁹ which works for EMF-based models that provides a full OCL support and offers a template-based language for defining code-generation templates (Brambilla et al., 2012).

In the transformation process with Acceleo, once the source model and the rules in the transformation template are defined, the Acceleo tool analyzes automatically the model and applies the transformation to each element from source model according to the transformation rules defined in the template, and obtains the text file as output, in the format required by the user. For example, the text file could be .java, .txt, .css, .html.

The Listing 1 shows as an example, an excerpt of the transformation template with a transformation rule written in Acceleo, corresponding to the *Controller* module in the Reference software architecture. This transformation rule generates the calls to the personalization

⁵ <https://www.w3.org/TR/xslt20/>

⁶ JET: The Java Emitter Template https://eclipse.org/articles/Article-JET/jet_tutorial1.html

⁷ <https://eclipse.org/modeling/m2t/?project=xpand>

⁸ <https://eclipse.org/gmt/mofscript/>

⁹ <http://www.eclipse.org/acceleo/>

components with the appropriate parameters, according to the web functionalities defined in the Application Layer from the personalization model. In Listing 1 the changeable elements that depend on the meta-model elements are enclosed in brackets. An example of the transformation code with Acceleo is:

All the elements from a package in the model are examined with instructions like:

```
[for (aPackage : Package | aModel.eAllContents(Package))]
```

The package stereotype is verified with the instruction:

```
[if (aPackage.getAppliedStereotypes() -> exists(name.equalsIgnoreCase('Application Layer')))]
```

The applied stereotype to a model element, for example, to an interface, is verified this way:

```
[if (ifunctionality.getAppliedStereotypes() ->
exists(name.equalsIgnoreCase('Func_Interface')))]
```

The generation of code is based on elements from model. For example, this line set in session, an object named as the result from an operation. The static code is in black.

```
session.setAttribute("[optPage.returnResult().name/]", [optFuntionality.returnResult().name/]);
```

5.2.4 Prototype

The framework was implemented using current technologies like Papyrus¹⁰ and Acceleo¹¹ on top of Eclipse¹² (Neon.2). Papyrus is a UML modeling tool that provides a support for Domain Specific Languages and OCL. Acceleo is an implementation of the M2T transformation standard of the OMG (Object Management Group) for EMF (Eclipse Modeling Framework) to support code generation. Appendix C shows some screenshots of the MAMPA framework, including screenshots of the meta-model definition, models, transformations, generated files, and the final web application with the assembled personalization strategies.

In the next chapter, we present a controlled experiment to evaluate the MAMPA approach, specifically in the reduction of manual interventions in the generated code, and the level of accuracy.

¹⁰ <https://eclipse.org/papyrus/>

¹¹ <http://www.eclipse.org/acceleo/>

¹² <https://eclipse.org/>

```

[for (aPackage : Package | aModel.eAllContents(Package))]
[if (aPackage.getAppliedStereotypes() -> exists(name.equalsIgnoreCase('Application Layer')))]
  [for (ifunctionality : Interface | aPackage.eAllContents(Interface))]
    [if (ifunctionality.getAppliedStereotypes() ->
exists(name.equalsIgnoreCase('Func_Interface')))]
      [for (optPage : Operation | interface.eAllContents(Operation))]
        [for (steOptPage : Stereotype | optPage.getAppliedStereotypes())]
          [if (ifunctionality.eAllContents(Operation) -> exists(name =
optPage.getValue(steOptPage, 'matchOpFunctionality')))]
            [if (ifunctionality.name = optPage.getValue(steOptPage, 'func_interface'))]
              [for (optFuntionality : Operation | ifunctionality.eAllContents(Operation) ->
select(name = optPage.getValue(steOptPage, 'matchOpFunctionality')))]

[optFuntionality.type.name/] [optFuntionality.returnResult().name/] = new
[optFuntionality.type.name/] ();
[optFuntionality.returnResult().name/] = innerpersonalization.[optFuntionality.name/](
[for
(param : Parameter | optFuntionality.inputParameters())separator(',')] [if
param.getAppliedStereotypes() -> exists(name = 'paramFromUserSession')] [for (steParam :
Stereotype | param.getAppliedStereotypes())separator(',')] [if
steParam.name.equalsIgnoreCase('paramFromUserSession')]cc.get[param.getValue(steParam,
'fromUser')] / () [ /if] [ /for] [else] [param.name/] [ /if] [ /for] );
session.setAttribute("[optPage.returnResult().name/]",
[optFuntionality.returnResult().name/]);
  [ /for]
[ /if]          [ /if]          [ /for]          [ /for]          [ /if]          [ /for] [ /if] [ /for]

```

Listing 1. Excerpt of the transformation template with a transformation rule written in Acceleo, corresponding to the Controller module in the Reference software architecture

5.3 Related Work

This section presents an overview of different proposals in the web application domain using a model-driven web engineering approach to face the generation of web applications from scratch or in their evolution.

Most of the proposals have adopted different model-driven web engineering methods to generate web applications from scratch. The diversity of those methods is due to the need to adapt, extend or redefine different contexts. For instance, some proposals are oriented to support RIA applications (Agustin, 2015), Web mobile applications (Achilleos, Paspallis, & Papadopoulos, 2011; Vera, Pons, Giulianelli, & Rodríguez, 2012), Web 2.0 (Guzmán, López, Valverde, & Panach, 2012), Content Management Systems (CMS)-based Web Applications

(Trias, 2012), or are linked to a particular web design method like WSDM (Mukhtar, Hassan, Jaafar, & Rahim, 2013), WCF (Agustin, 2015), or Spring Roo (Castrejón, López-Landa, & Lozano, 2011). However, those proposals focus on the initial creation of the application and do not consider how to deal with modifiability and software evolution.

For example, Agustin proposes a UML profile to design an application using a Component-Based Software Engineering (CBSE) approach, and a generation tool to semi-automatically transform a UML design into a Web Component Framework (WCF) application (Agustin, 2015). Mukhtar et al. provide a mechanism called WSDMDA to enhance the Web Site Design Method (WSDM) from a conceptual modeling approach to a Model-Driven Architecture (MDA) modeling approach; which profiles the conceptual model of WSDM with a new user-interest profile (Mukhtar et al., 2013). This work uses QVT model transformation language to automate the mapping specification from Platform Independent Model (PIM) to Platform Specific Model (PSM). Vera et al. present a proposal based on the Object Oriented Hypermedia Design Method (OOHDM) and MDA to generate web applications specially designed for mobile devices with one view in XHTML and other in HTML5 (Vera et al., 2012). This proposal uses a conceptual model, a component diagram and a state diagram to specify the application, and uses XMI files to generate the database scripts and the application code, following the Model-View-Controller (MVC) pattern. As a proposal, the details of MDA are incipient. Achilleos et al. use a Model-Driven Engineering (MDE) approach in the development of device-aware Web Services (Achilleos et al., 2011). This work focuses on the development of client's (mobile applications in different technologies, like Android, Windows mobile, J2ME), which can consume web services directly via the Web platform. The client comprises the Graphical User Interfaces GUIs and the necessary proxy classes. This work proposes the Presentation Modeling Language (PML), which is defined as an EMF meta-model in order to design the GUIs. Moreover, this work uses existing code generation tools to enable the transformation of models written with Web Services Description Language (WSDL) into platform-specific proxy classes. Guzmán et al. with the aim of supporting the challenges from Web 2.0, propose a Web 2.0 pattern that includes a functionality model and an interaction model, which is designed to improve the end-user involvement in Web sites (Guzmán et al., 2012). Also, this work introduces a M2M transformation strategy to integrate these Web 2.0 patterns into the models of any model-driven Web Engineering method. Trias in the context of the Content Management

Systems (CMS), proposes a CMS Common Metamodel that captures the key concerns required to model and implement CMS-based Web applications (Trias, 2012). Castrejón et al. propose an MDA approach called Model2Roo for web application development. This work proposes a meta-model extension mechanism based on Ecore annotations and UML profiles, and the transformation process uses ATL query (ATLAS Transformation Language) to transform Ecore models into Spring Roo scripts (Castrejón et al., 2011). This proposal is integrated with the Eclipse Modeling Framework (EMF).

Regarding to evolution and maintenance of web applications, we found the work of Bernardi et al. which uses a reverse engineering phase to recover missing and out-of-date models of a software system, and then, a model-driven engineering phase to generate the application (Bernardi, Lucca, Distanto, & Cimitile, 2013). The process recovers the models according to the Ubiquitous Web Applications (UWA) design methodology, and then using the Atlas Transformation Language (ATL) they are transformed into a Java Server Faces (JSF) implementation model following a MVC pattern.

Table 6 summarizes the MDE features of the previously described works: the type of transformation involved (i.e., Model-to-Text (M2T) or Model-to-Model (M2M)); the abstraction level (i.e., Platform Independent Model (PIM), Platform Specific Model (PSM) or a programming language code); the transformation approach (i.e., relational, hybrid or by templates); and the corresponding tool.

Work	Transformation type	Abstraction levels	Approach and transformation tool
(Agustin, 2015)	M2T	PIM, code	Templates / Acceleo
(Mukhtar et al., 2013)	M2M	PIM, PSM	Relational / MediniQVT
(Bernardi et al., 2013)	M2M	PIM, PSM	Hybrid / ATL
(Vera et al., 2012)	M2T	PIM, code	N/E
(Trias, 2012)	N/A	PIM	N/A
(Guzmán et al., 2012)	M2M	PIM	N/E
(Castrejón et al., 2011)	M2M	PIM, PSM	Hybrid/ATL
(Achilleos et al., 2011)	M2T	PIM, code	Template/Xpand

* N/E: does not exist. N/A: not applicable.

Table 6. MDE work features

All the described proposals accept the benefits of using MDE, specifically in quality and productivity, and most of them apply MDE to generate web applications from scratch. However, there is not participation of those works in the domain of personalized web applications, and they face the software evolution marginally. Besides incorporating a CBS approach, and based on a reference architecture supporting the modifiability of personalized web applications, the MAMPA approach is intended to take advantage of the MDE to increase the quality of software, reduces the effort in the development process and enhances the modifiability of personalized web applications. In the next chapter, we present a controlled experiment to evaluate the effectiveness in supporting software evolution under the MAMPA approach.

Chapter 6.

A controlled experiment to validate MAMPA approach

This section presents the controlled experiment to evaluate the effectiveness in supporting software evolution under the MAMPA approach. This controlled experiment evaluates the reduction of manual interventions specifically in the code generated after applying the MAMPA approach. The results show an improvement to the modifiability of personalized web applications with a good level of accuracy of 99.92% and a good degree of completeness an average of 75.16%. Section 6.1 presents the experiment scope. Section 6.2 states the research questions. The experimental units are identified in section 6.3. The experiment design and the selection of variables are described in section 6.4 and 6.5 respectively. Finally, sections 6.6 and 6.7 present the threats to validity and results.

6.1 Scoping

In order to establish our evaluation goal, we followed the Goal-Question-Metric (GQM) template (Wohlin et al., 2012). We stated that the goal of the experiment is to analyze the MAMPA approach, for the purpose of evaluating it, with respect to the effectiveness in supporting the software evolution under this approach, specifically when there are changes of the personalized strategies. This evaluation is from the point of view of researchers and practitioners; developed in the context of a software engineer updating different personalization strategies into an existent web application.

We adopted the *Effectiveness* concept as is defined in the ISO/IEC 25010:2011 in its product quality model: “*accuracy and completeness with which users achieve specified goals.*”

6.2 Research questions

The general research question is *How effective is the MAMPA approach?* To analyze the *effectiveness* of the MAMPA approach, we identified two more detailed research questions (RQ):

RQ1. How *accurate* is the MAMPA approach?

RQ2. To what extent is the degree of *completeness* of the MAMPA approach?

According to ISO/IEC 25010:2011 in their software product quality model, the functional correctness (referred in ISO/IEC 9126-1 as "Accuracy") is defined as the "Degree to which a product or system provides the correct results with the needed degree of precision" (ISO - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011). Therefore, we associated the *accuracy* concept with metrics related to the number of errors in the generation, compilation and integration of code.

In addition, we have defined the *completeness* concept based on following works in software development area. Matook & Indulska considered the completeness as the "degree to which all the components of the reference model are present under a predefined scope" (Matook & Indulska, 2009). In Moody & Shanks's work, "completeness refers to whether the data model contains all user requirements." Moody & Shanks evaluated the completeness in terms of number of requirements missing from the model, and expressed it as a percentage of the total user requirements (Moody & Shanks, 2003). According to the software product quality model of the ISO/IEC 25010:2011 international standard, the functional completeness is defined as the "degree to which the set of functions covers all the specified tasks and user objectives" (ISO - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011).

In reference to the MAMPA approach, its goal is to enhance the modifiability of personalized web applications reducing the manual intervention when the personalization strategies are applied. Therefore, in the experiment to evaluate MAMPA, we tailored the *completeness* concept to the extent in which the MAMPA approach covers the modeling and generation of code that represents the application of personalization strategies in a personalized web application developed according to the reference architecture. Therefore, the RQ2 can be re-defined as follows:

RQ2: Which is the MAMPA approach coverage percentage?

This question lets us evaluate the degree in which it reduces the manual intervention when we are evolving a software, updating or adding personalization strategies.

6.3 Experimental units

We use a set of 18 personalization strategies¹³ tailored from an industrial e-commerce environment. The personalization strategies can be classified in three groups:

- *Inclusion / exclusion of components*: focusing on the quantity of personalization strategies included in the final application.
- *Component selection based on user features*: grouping strategies that involve some user features like purchase behavior, age, and preferences.
- *Component selection based on user context features*: grouping strategies that involve some user context features like location and behavior.

6.4 Experiment design

The experiment corresponds to a *multi-object variation study* (Wohlin et al., 2012) because it is conducted on a single subject across a set of objects. The set of personalization strategies used were tailored from a Brazilian e-commerce enterprise. The experiment was executed from an existing web application of e-commerce implemented with the reference software architecture, in which all the personalization strategies were implemented in two ways: manually and under the MAMPA approach. The MAMPA approach was applied in the modeling and code generation activities for each personalization strategy. The generated code was assembled into the e-commerce application to test the complete functionality. Finally, both personalized applications, manually and under the MAMPA approach, were compared to obtain the defined metrics, with the supervision of a software engineer.

¹³ The personalization strategies list is available at <http://telesalud.udea.edu.co/Ecommerce/MAMPA> and in the appendix D. The personalization strategies are in Spanish because they were tailored, analyzed and implemented by the researches and programmers who's native language is Spanish.

6.5 Selection of variables

The *accuracy* and the *coverage percentage* were measured in order to answer both research questions. We understood the accuracy as the degree of correctness, or the degree to which the method is applied without mistakes. Thus, this experiment measures the density of defects in defects per 100 Lines of Code (LOC) (Bunse, Gross, & Peper, 2009).

The defects are established in terms of:

- *Generation errors*: the number of errors found in the generation of the application code.
- *Compilation errors*: the number of syntax errors found in the compilation of the application.
- *Integration errors*: the number of errors found in the execution of the final application due to failures in the integration process.

The experiment also measures the percentage of generated lines of code by the MAMPA approach; regarding to the number of lines in the application developed manually. Thus, the *coverage percentage* is stated as:

$$\text{Coverage percentage} = \frac{\text{\# of lines generated by the MAMPA approach} * 100\%}{\text{\# of lines in the application developed manually}}$$

Equation 1. Coverage percentage

6.6 Threats to validity

We identified and handled different threats to the validity of the experiment. Regarding the internal validity, one important threat is the subject experience and the natural variation in human performance. We handled the threat by selecting a last semester student in the System Engineering program at the University of Antioquia to execute the experiment. In addition, we consulted an experienced engineer for reviewing the experiment and its results. The external validity is concerned with conditions that limit the ability to generalize the results of the experiment to industrial practice. Therefore, making the experimental environment as realistic as possible is essential to reduce threats to this validity (Easterbrook, Singer, Storey, & Damian, 2008; Wohlin et al., 2012). One threat to the external validity is about having a population, not

representative of the population we want to generalize. In our case, the MAMPA target population are software engineers in charge of integrating the personalization strategies into the existent enterprise e-commerce. Again, we alleviated this threat by selecting a last semester student and an experienced engineer as a reviewer. Another significant threat is not having representative materials for the experiment execution. In this case, we used up-to-date personalization strategies extracted from a real industrial case. However, we cannot generalize the results because we only used the data from one enterprise in the domain; nevertheless, we conducted the experiment in an adequate environment, keeping the features of industrial scenario.

Concerning construct validity, a *mono-operation bias* threat exists because we just use a set of personalization strategies from one enterprise, and it is possible that the experiment under-represents the construct. An *interaction of testing and treatment* threat also exists, as we measure the errors generated after applying the MAMPA approach. Therefore, it may cause the programmer to be more sensitive or receptive to the treatment; thus, the programmer would be more aware of their errors made, and thus will try to reduce them. However, we diminished this threat by adopting a well, acceptable and international standard, ISO/IEC 25010:2011, to define the constructs, before transforming them to measures. Moreover, the number of errors and the MAMPA approach coverage percentage represent exactly what they are defining. Finally, threats to conclusion validity are related to issues that affect the capacity to draw the correct conclusion. In our experiment, the sample size for personalization strategies was representative, having into account that each personalized system did not include a lot of personalization strategies at the same time. In addition, the implementation and measurement of our defined variables was done carefully.

6.7 Results

We developed and completed the process of modeling and execution of the transformations under the MAMPA approach with 18 of 25 strategies¹⁴. Those strategies properly support the

¹⁴ The table with the detailed results of the experimental evaluation is available at <http://telesalud.udea.edu.co/Ecommerce/MAMPA>

evaluation, because they are a representative in number, and we extracted them from an industrial context. Appendix C shows some screenshots of the implementation of the MAMPA framework, including views of the meta-model definition, models, transformations, generated files, and the final web application with the assembled personalization strategies.

In relation to the MAMPA approach accuracy, we obtained the errors number density for each personalization strategy modelled and implemented, and we summarized the error density metric for all strategies as shown in Table 7.

<i>Type of strategy</i>	<i>Total of errors</i>	<i>Density of errors</i>
Generation errors	35	0,08
Compilation errors	0	0,00
Integration errors	0	0,00

Table 7. Error density

We found a low density of errors in the execution of transformations, and zero errors in the compilation and integration errors. This result gives the idea that the MAMPA approach is accurate. The errors in the generation task were fixed properly after the test. After the generation task, we added or fixed some code lines manually. In future execution of the MAMPA approach, this manual task could introduce errors, therefore, they must be detected in the compilation and integration activities. Zero errors in the compilation and integration metrics are explicable because of the use of CB (component-based) paradigm. Under this paradigm, the components used to build an application were previously tested.

Regarding the MAMPA approach coverage percentage, we collected the results for each personalization strategy, as shown in Table 8. The first column, “Number of lines in manual application (1)” reports the number of code lines involved in the integration of the personalization strategies in the application developed manually. The second column “Number of generated lines (2)” reports the number of code lines generated executing the MAMPA approach. The third column “Number of manually modified lines (3)” reports the number of code lines generated that were manually modified. The fourth column “Number of added lines

manually (4),” similar to the previous column, reports the number of lines that were added manually. Finally, the last column “Coverage percentage (5)” reports the metric for the coverage percentage.

As shown in Table 8, the MAMPA approach shows a good degree of completeness, evidenced with the coverage percentage, which in average is 75.16%. It shows clear benefits by automating a large percentage of code that represents the component integration. Even more, although in some cases, there is a need to modify some generated lines or add new lines manually, that is a low percentage, of 11.54% and 1.67% respectively. Also, we worked with strategies from only one enterprise and considered a specific domain: e-commerce. It can be considered as a limitation of the results, but also and an opportunity to improve our approach.

<i>No. of strategy</i>	<i>Number of lines in the manual application (1)</i>	<i>Number of generated lines (2)</i>	<i>Number of manually modified lines (3)</i>	<i>Number of manually added lines (4)</i>	<i>Coverage percentage (5)</i>
1	39	28	3	0	71,79%
2	71	54	5	0	76,06%
3	36	25	5	0	69,44%
4	66	48	9	0	72,73%
5	72	51	7	0	70,83%
5	124	99	11	9	79,84%
7	171	143	15	0	83,63%
8	45	34	3	0	75,56%
9	23	18	1	0	78,26%
10	22	17	4	4	77,27%
11	40	29	5	0	72,50%
12	34	23	4	0	67,65%
13	35	24	4	0	68,57%
14	36	25	4	0	69,44%
15	36	25	4	0	69,44%
16	41	30	4	0	73,17%
17	94	90	1	0	95,74%
18	21	17	1	0	80,95%

Table 8. Coverage percentage of the MAMPA approach

Finally, the proposed reference software architecture has evidenced the improvement in the software modifiability, and the MAMPA approach has progressed also in this direction, showing clear benefits in the automation of code, reduction of manual interventions and clear benefits in the accuracy level; all of that directed to permit an appropriated evolution of software and consequently a timely response to the market.

Part III: Final Considerations

Chapter 7. Conclusions

This section presents a summary of the work, the main contributions and the ongoing and future work.

Personalization is considered an effective tool for achieving business success online (Cao & Li, 2007), and a valuable tool in the business context (Brusilovsky & Nejdl, 2004; Karat et al., 2003). Although personalization has demonstrated advantages in web applications (Alotaibi, 2013; Kwon & Kim, 2012), there are factors that complicate the inclusion of personalization strategies into a business environment.

This dissertation has investigated the personalization architectures (models) for web applications in a MDD context. Specifically, this work studied the difficulty to evolve personalized web applications, when the personalization strategies are updated constantly (modifiability); the technical complexity to integrate the personalization strategies in a short time in a business environment; and, the lack of tools to support personalization modeling and generation of code in a context of MDD.

We tackled the aforementioned problems in two steps:

First, we faced the modifiability and maintainability issues, and as a result, we proposed a Reference Software Architecture for personalized web applications. This architecture conceives the personalization strategies as a specialized component, proposes different modules to avoid mixing the web application's basic functionality with the personalized behavior, and diminishes the technical complexity of integrate the personalization strategies in a web application. This proposal favor the reusability and maintainability aspects. Even more, it proposes an external module to administrate the user, context and group information, with possibilities to be shared with other applications, and to integrate data mining or machine-learning techniques to build the relevant information about users. In order to validate the proposal, we executed a controlled

experiment to evaluate the software modifiability of the proposed Reference Software Architecture, using a set of personalization strategies taken from a Brazilian e-commerce enterprise.

Although the proposed reference software architecture showed benefits about modifiability and maintainability in a personalized web application, there remained repetitive manual tasks in the programming activities, making the inclusion of personalization strategies in a business environment complicated. At this point, we considered to explore the MDD approach to reduce efforts in the manual procedures existent in the process of applying the reference software architecture.

Then, as the second step, we integrated the reference architecture designed in the first step into a framework using model-driven approach. The framework proposed is called **MAMPA: a Model-driven Approach to enhance the Modifiability of Personalized Web Applications**. The MAMPA framework reduces the technical complexity of weaving personalization strategies (specialized software components), and its subsequently code generation. All of that, directed to reduce the time to respond to market changes.

In order to validate the framework, we executed a controlled experiment to evaluate the effectiveness in supporting software evolution under the MAMPA approach, also using personalization strategies extracted from a real Brazilian e-commerce enterprise. Specifically, the experiment examined how accurate is the MAMPA approach, and to what extent is the degree of completeness of the MAMPA approach. Finally, the experiment demonstrated a good level of accuracy and completeness.

As a summary, the definition of the Reference Software Architecture for personalized web applications, and the adoption of an MDD approach, both integrated in the MAMPA approach, allowed to face effectively the modifiability issues when the personalization strategies are updated constantly, and the technical complexity to integrate them in a short time in a business environment.

7.1 Contributions

The main contributions of this dissertation are the following:

- A Reference Software Architecture to support the modifiability of personalized web applications where the software modifiability is the main architectural drive.
 - It uses software component weaving as an alternative to tackle the challenges of including personalized behavior into web applications.
 - It considers an external personalization model administrator (*PMAAdmin*) module, an independent module placed outside the web application and responsible for managing information about the users, context and groups of users.
- A methodology to apply the reference architecture.
 - It establishes a process to guide the developers in the adoption of the reference architecture.
- A Model-driven Approach to enhance the Modifiability of Personalized web Applications (MAMPA).
 - The MAMPA approach supports the continuous update of personalization strategies into web applications and reduces the time to respond to market changes.
- A Personalized Web Application Modeling Language (PWML) to specify the personalization strategies within a personalization model. PWML is defined as an UML Profile.
- A transformation process based on templates to guarantee the conformity of the generated code with the reference architecture.
- A group of code transformation templates implemented using Acceleo as the transformation language.
 - The transformation process gives homogeneity to the generated code and decreases the number of errors in the update of personalization strategies, enhancing the modifiability and evolution of web application.
- A framework tool supporting MAMPA built on technologies for Eclipse.

7.2 Ongoing and Future Work

A complete validation of the Reference Software Architecture is part of the future work. In particular, it is necessary to test more scenarios, experiment with different domains such as e-health or e-learning. This work does not address other important issues, such as, the downstream economic benefits of using the reference software architecture for developing personalized web applications. For example, one could raise the question “How does fast and personalized web development really benefit software engineering at large?” “How much does it cost to do it early on as compared to later on?” These complex issues have yet to be investigated.

Moreover, future works include the testing of the MAMPA approach with personalization strategies from other enterprises in the e-commerce domain, and from other domains. For example, testing MAMPA in the e-learning domain, using pedagogic strategies with machine learning based systems that need to be retrained or to change the prediction methods.

It is also valuable to evaluate the software architecture in systems involving prediction and transactional issues, where the prediction strategies change constantly. A particular case, for example, are the fraud detection systems for credit card purchase systems; the fraud techniques evolve every day and the detection strategies should be adapted at the same pace. In that case, there is a transactional system supporting purchases, and another “module” to apply strategies based on rules or machine learning techniques in order to detect fraudulent transactions.

Appendix A.

Architectures for the implementation of the changes scenarios

The following figures are the architectures used for the implementation of the changes scenarios 1, 2, 3, and 5 in the controlled experiment to validate the Reference Software Architecture, reported in chapter 4.

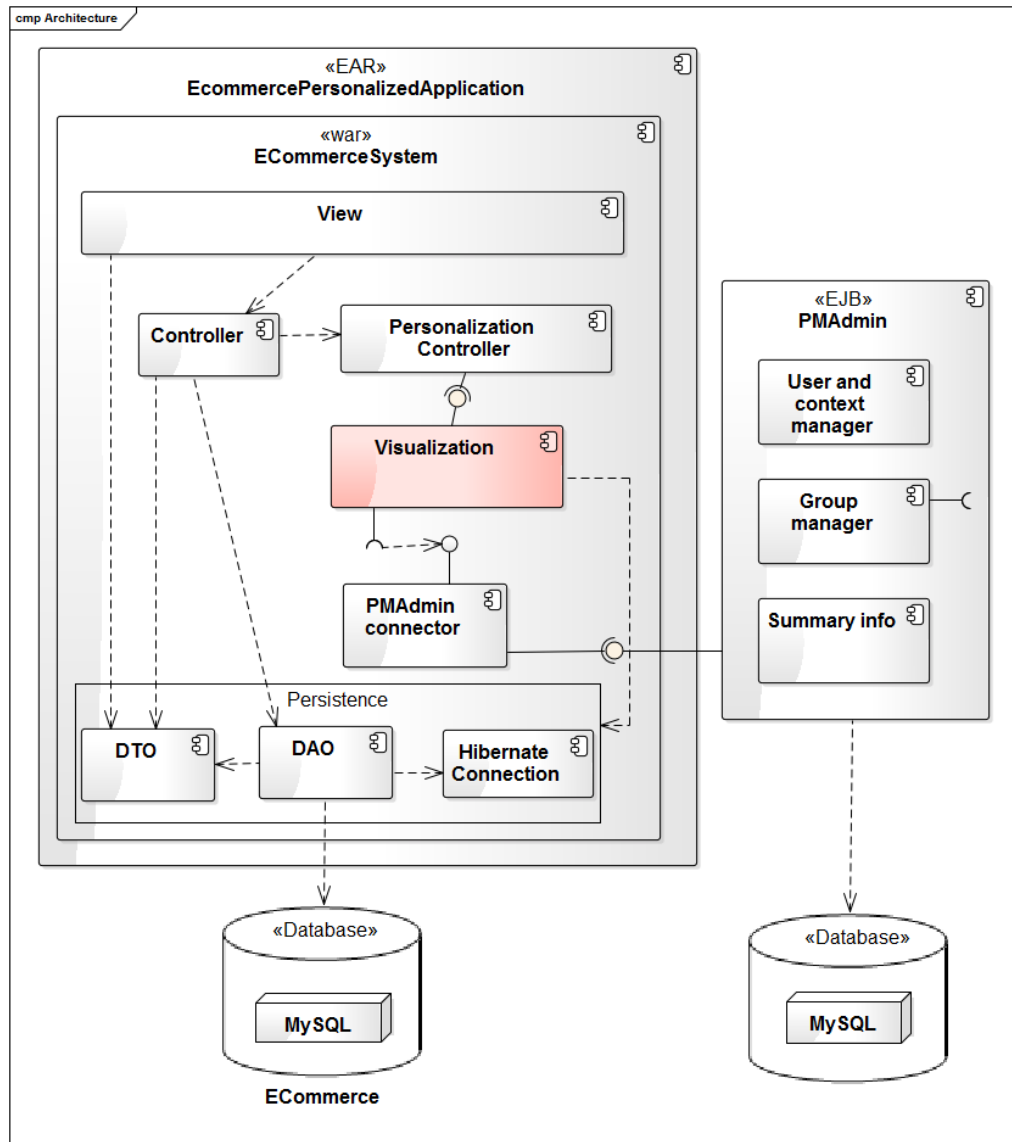


Figure 19. Architecture for the change scenario 1 according to the reference architecture

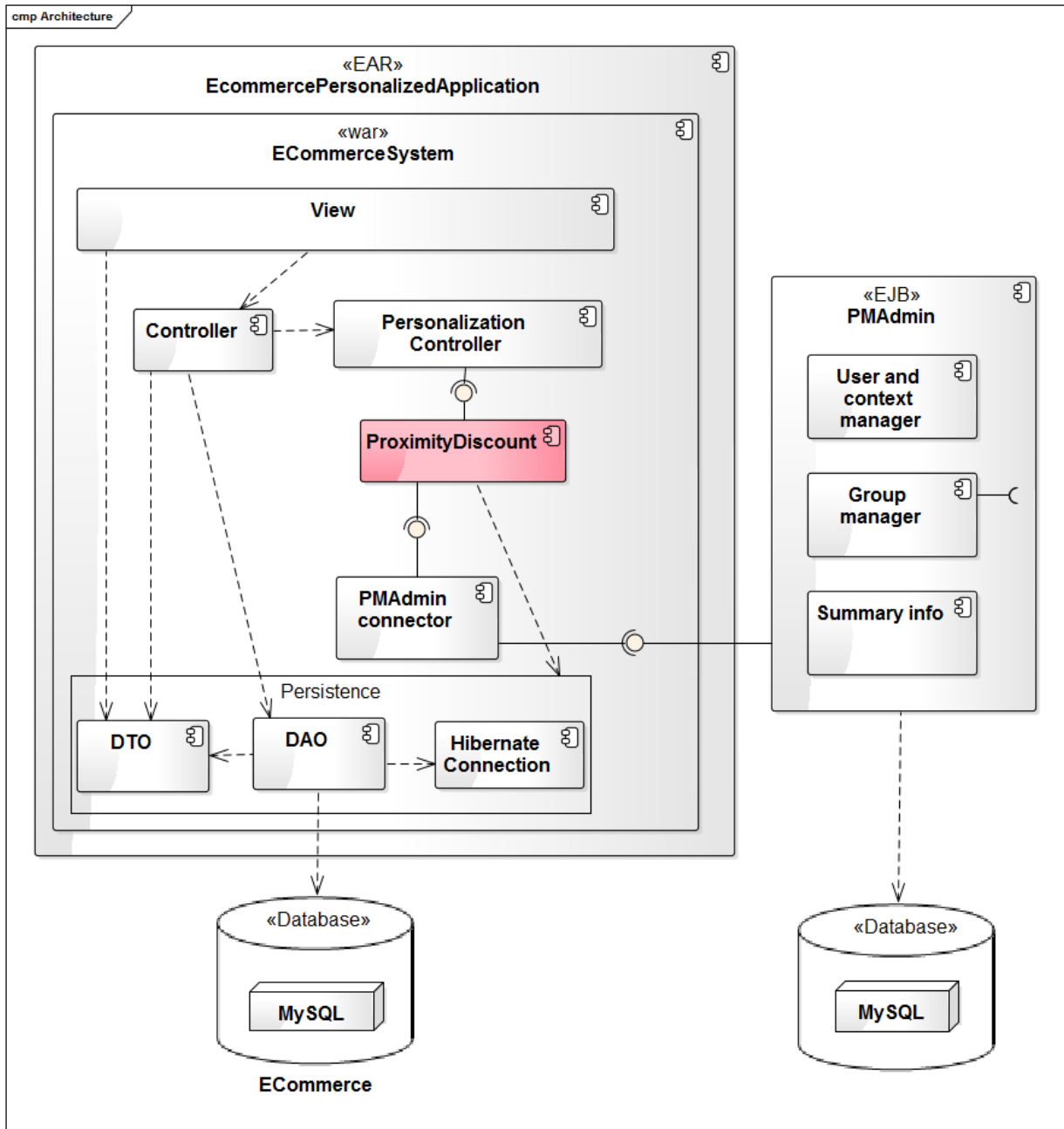


Figure 20. Architecture for the change scenario 2 according to the reference architecture

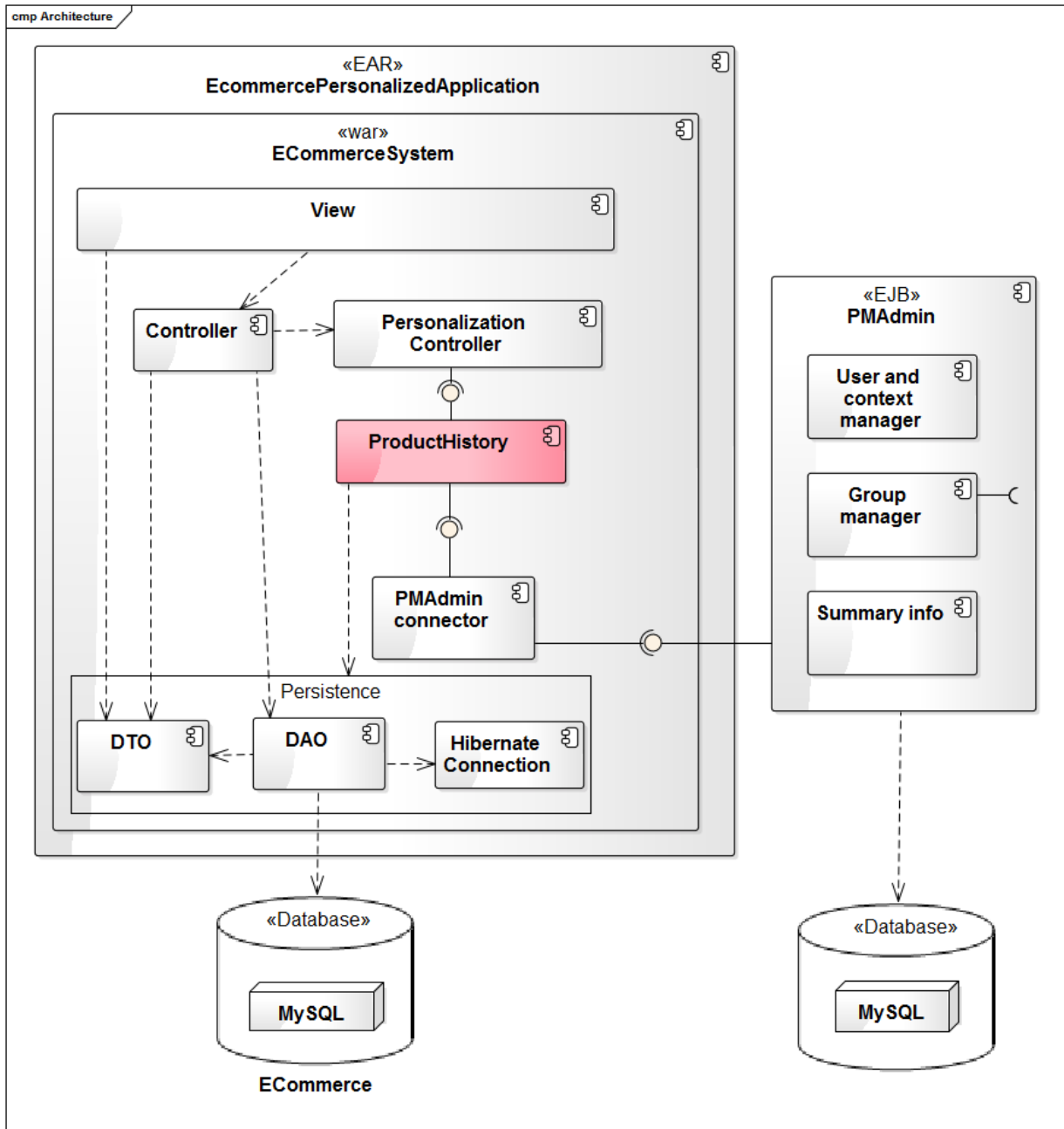


Figure 21. Architecture for the change scenario 3 according to the reference architecture

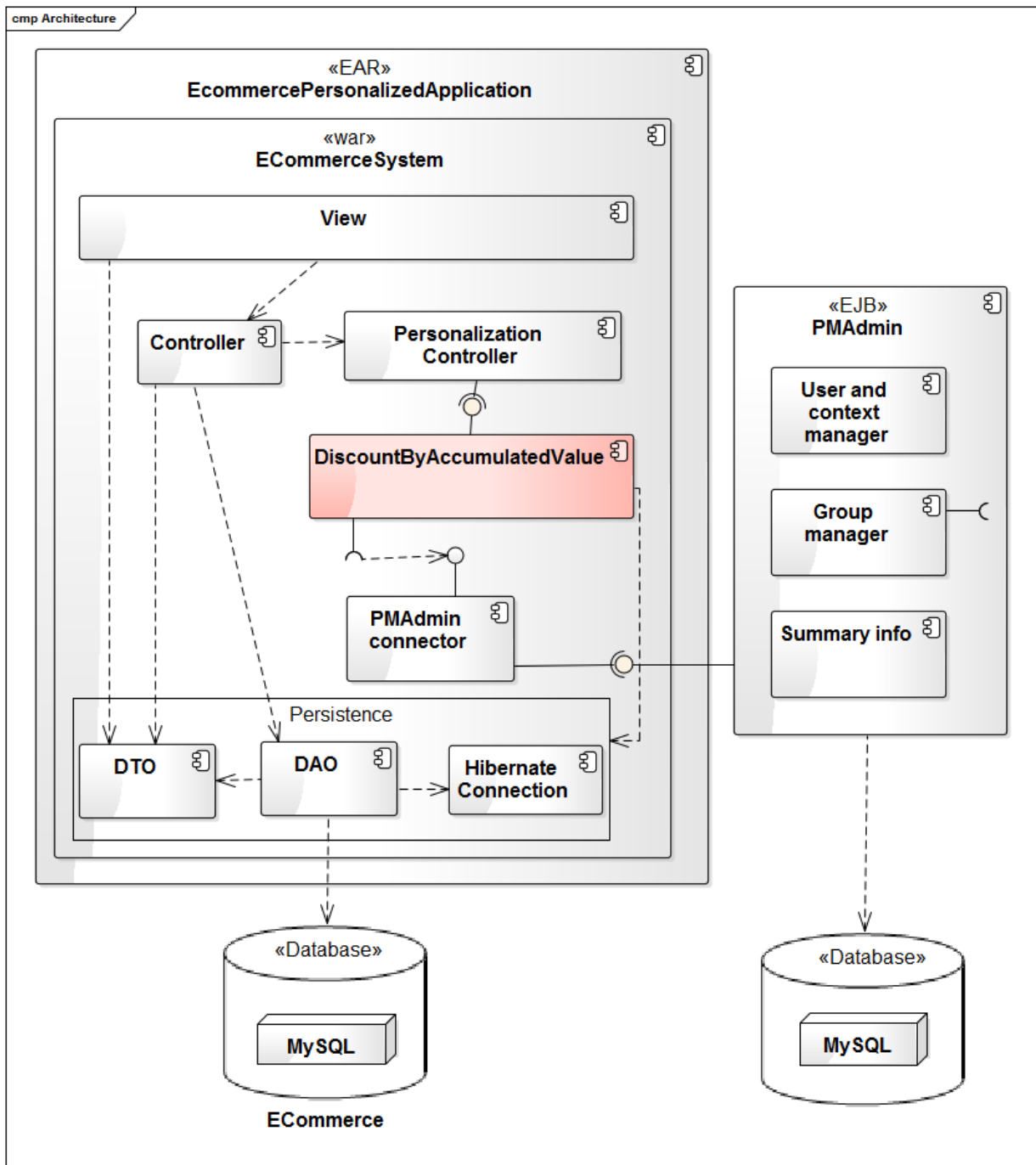


Figure 22. Architecture for the change scenario 5 according to the reference architecture

Appendix B.

Screenshots of the implemented personalized web application

The following figures are screenshots of the implemented personalized e-commerce.

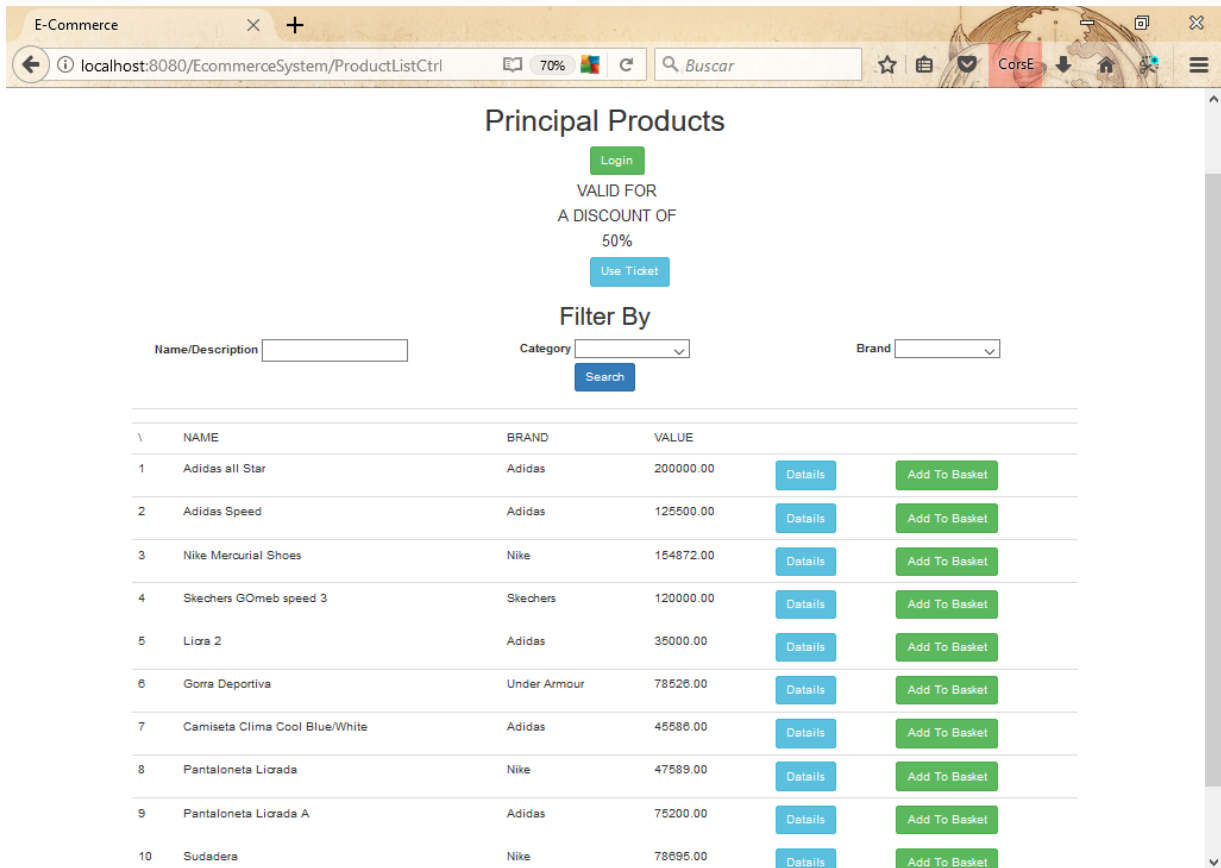


Figure 23. Product list with a discount ticket of 50%


E-Commerce

localhost:8080/EcommerceSystem/ProductListCtrlDetail

70%

Buscar

Nike Mercurial Shoes



Id	3
Product	Nike Mercurial Shoes
Description	Especiales para la práctica de deportes sobre grama
Brand	Nike
Category	Guayos
Value	154872.00

Today You have 30% of discount in the total value

[Add to Basket](#) [See more Products](#)

Figure 24. Product detail with a discount by day


E-Commerce

localhost:8080/EcommerceSystem/ProductListCtrlDetail

70%

Buscar

Under Armour Gemini



Id	4
Product	Under Armour Gemini
Description	Tenis especiales para realizar actividad física
Brand	Under Armour
Category	Calzado
Value	215688.00

Recommended Products

9	Cronometro
14	Sudadera
12	Pantaloneta Lirrada

[Add to Basket](#) [See more Products](#)

Figure 25. Product detail with recommended products

Appendix C.

Screenshots of the MAMPA framework

Meta-model definition screenshots

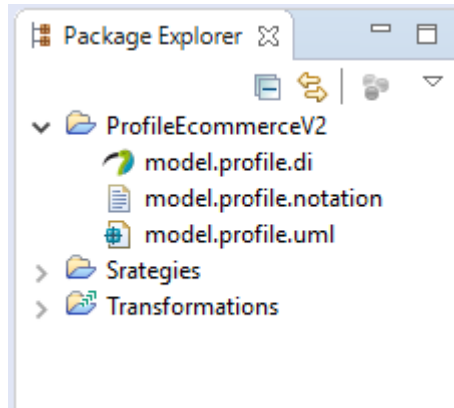


Figure 26. Packages for the definition of meta-model, models, and transformations

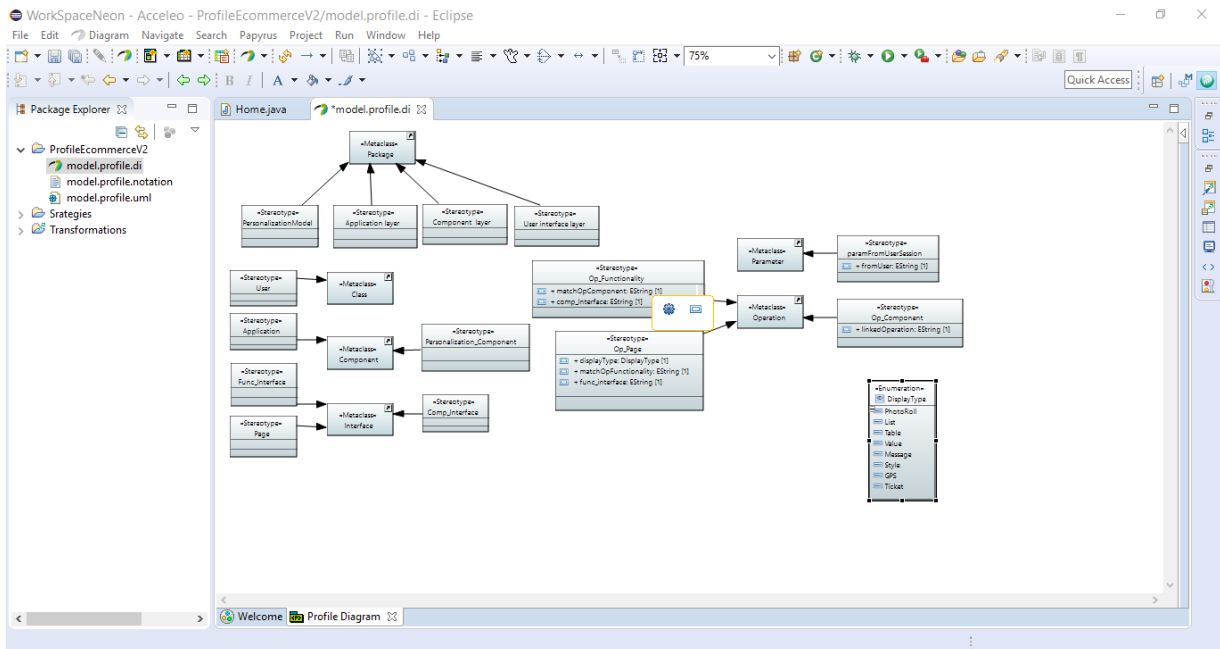


Figure 27. Meta-model definition

Models definition screenshot

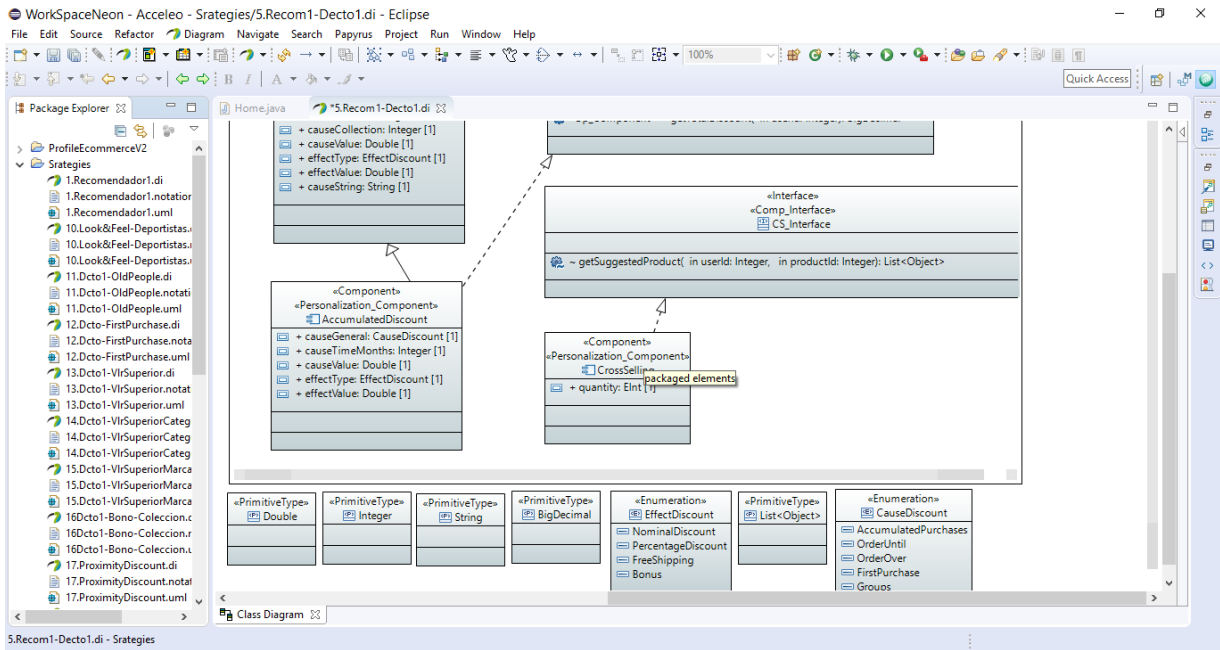


Figure 28. Models definition screenshot

Transformations screenshots

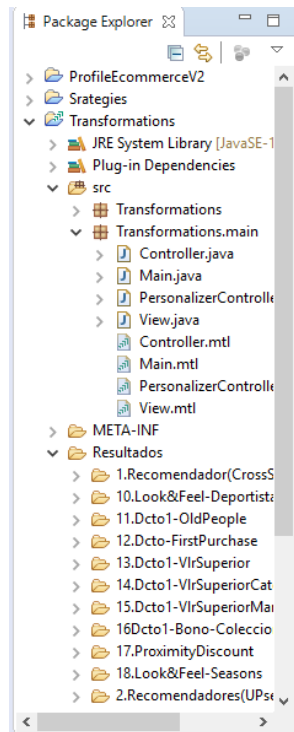


Figure 29. Packages of the transformations project

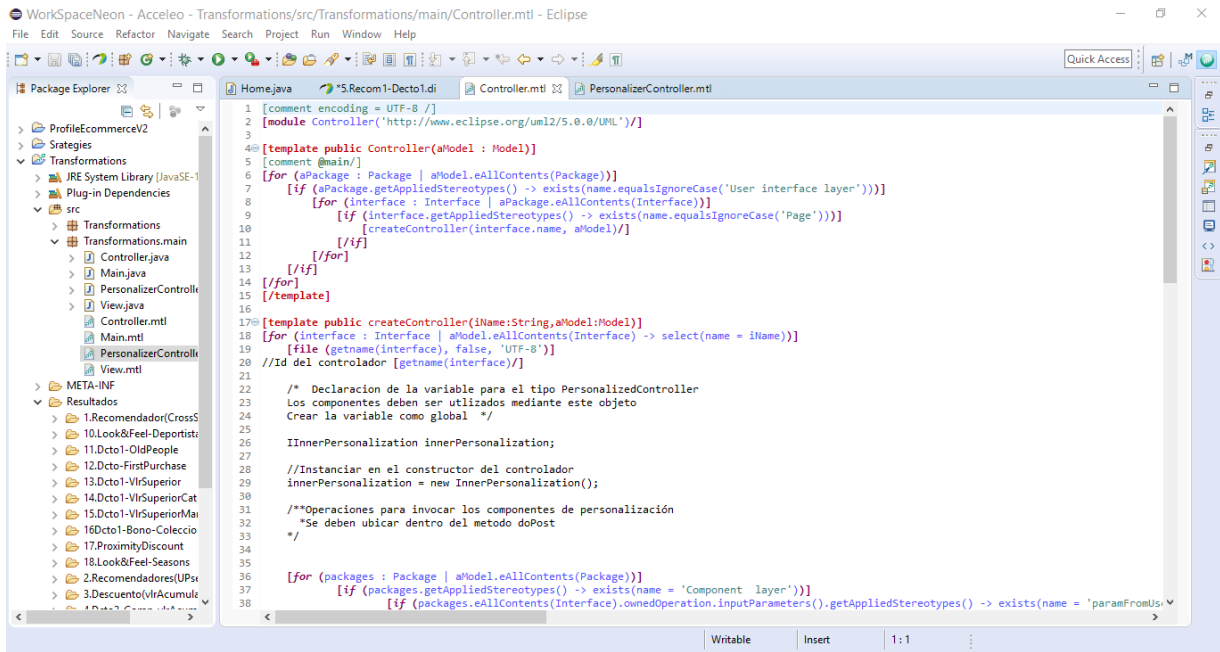


Figure 30. Transformation for the controller module in the reference architecture

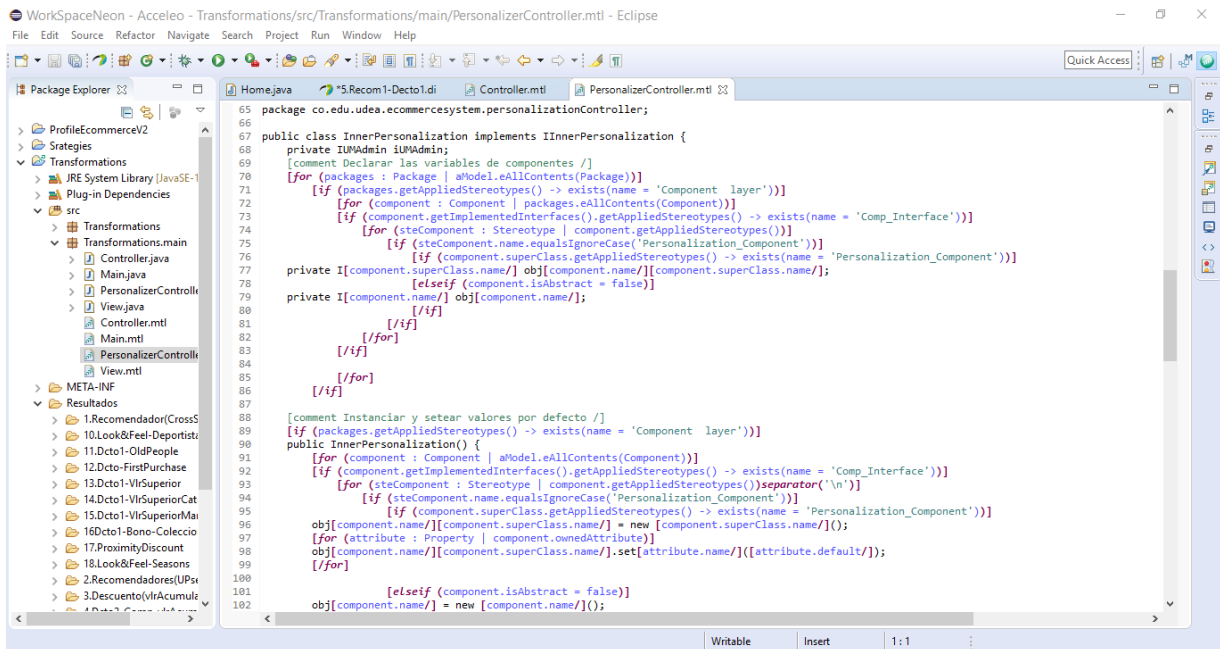


Figure 31. Transformation for the personalization controller module in the reference architecture

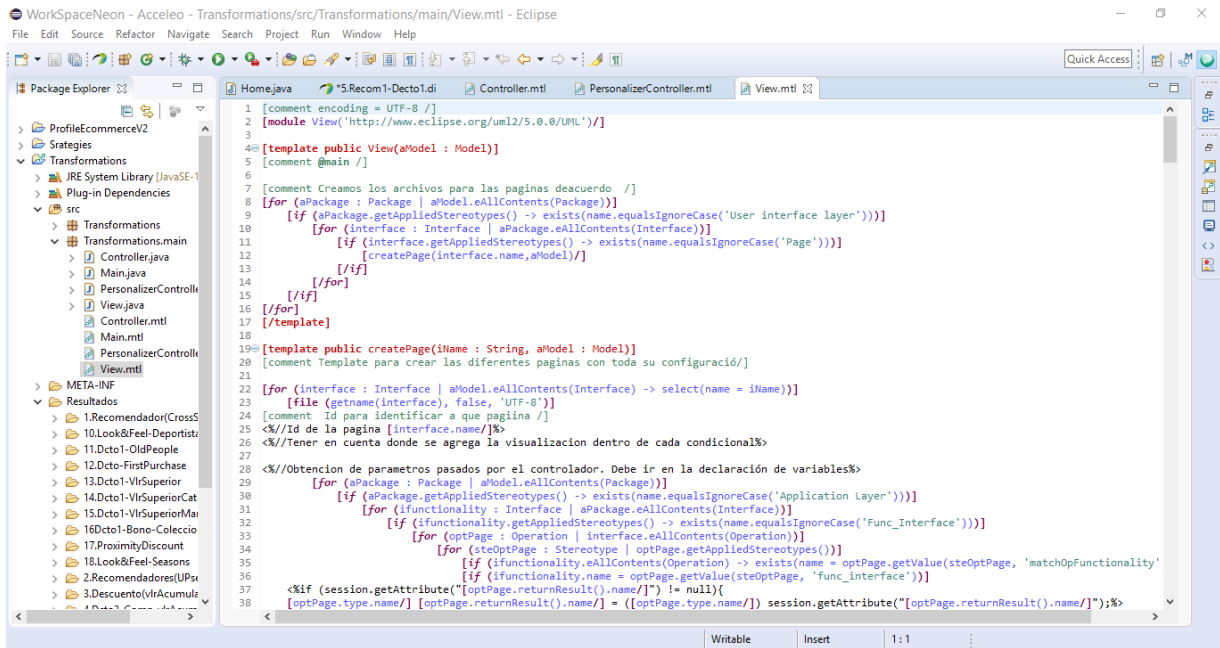


Figure 32. Transformation for the view

Screenshots of generated files

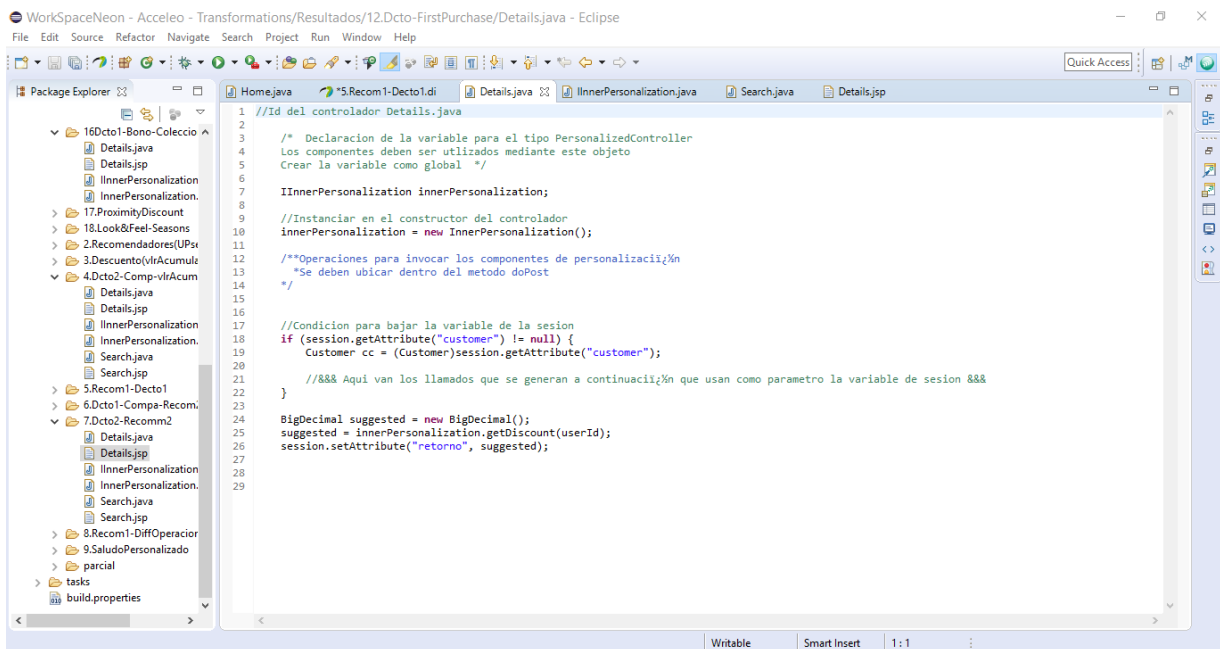


Figure 33. Product detail page generated for the personalization strategy # 7

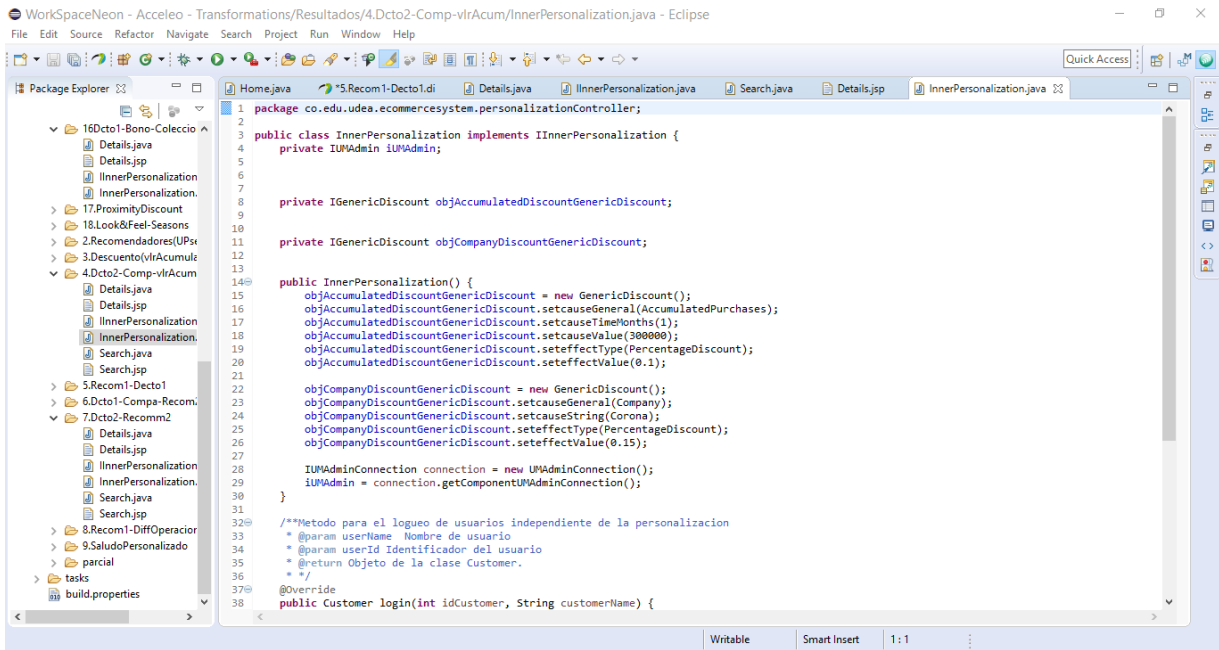


Figure 34. Personalization controller* generated for the personalization strategy # 4

* The personalization controller in this figure was implemented with the name InnerPersonalization.

Screenshot of the final web application with the assembled personalization strategies

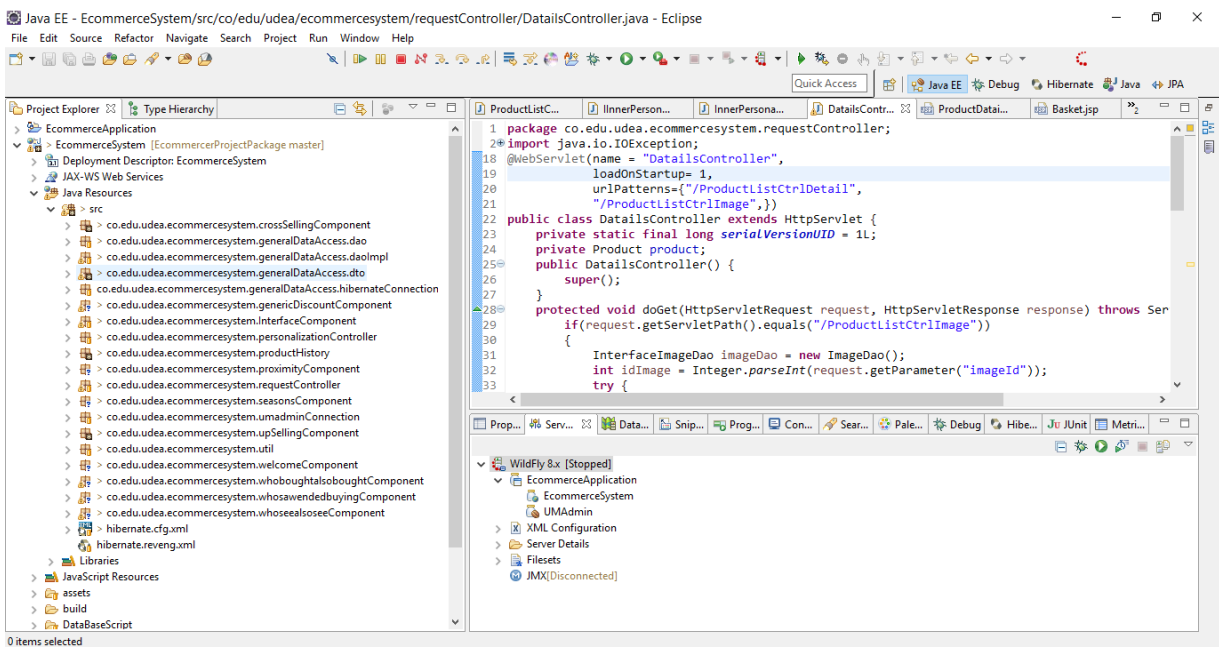


Figure 35. Packages of the final web application with the assembled personalization strategies

Appendix D.

List of personalization strategies

Nro	Palabras claves	Agrupación / Estrategia
Inclusión / Exclusión de componentes		
1	1 recom + param.	(1) Contar con el componente de recomendación por cross-selling. (VTEX-21-cross-selling)
2	2 recom.	(2) Contar con dos componentes de recomendación: a) Upselling. (VTEX-22-upselling) b) Histórico de últimos productos visitados por el cliente. (VTEX-12)
3	1 desc: \$acumulado, %	(3) Contar con un componente de descuento, el cual otorga un % de dcto si el usuario tiene como mínimo un valor acumulado en compras, en el último mes en toda la tienda. Valor mínimo: \$300.000. % dcto: 10%
4	2 desc: \$acumulado, % Compañía, %	(4) Contar con dos componentes de descuento para otorgar: a) Un % de descuento (15%) si el usuario es trabajador de una compañía específica. (VTEX-08) b) Un % de descuento si el usuario tiene como mínimo un valor acumulado en compras, en el último mes en toda la tienda. Valor mínimo: \$300.000. % dcto: 10% POLÍTICA: Se el usuario cumple con ambas condiciones se le otorga el descuento con mayor EFECTO.
5	1 recom + 1 desc.	(5) Contar con un componente de recomendación y uno de descuentos: Cross-selling y descuento por valor acumulado en compras. (Igual definición que el anterior)
6	2 recom + 1 desc.	(6) Contar con un componente de descuento y dos de recomendación: Descuento por compañía (VTEX-08), y recomendar así: a) Quien vio también vio; (VTEX-26) b) Quien compró, también compró.(VTEX-27)
7	2 recom + 2 desc.	(7) Contar con dos componentes de descuento y dos de recomendación, así: Descuento: a) Un % de descuento si el usuario es trabajador de una compañía específica. (VTEX-08) b) Un % de descuento si el usuario tiene como mínimo un valor acumulado en compras, en los últimos 3 meses en toda la tienda. Valor mínimo: \$300.000. % dcto: 10%

		Recomendación: a) Histórico de últimos productos visitados por el cliente. (VTEX-12) b) Quien compró, también compró.(VTEX-27)
8	nombre de métodos diff.	(8) Contar con un componente (de descuento o recomendación) que tenga establecidos métodos con otros nombres diferentes a los definidos en la aplicación base. (Homologación de operaciones) a) Quien vió, acabó comprando. (VTEX-28)
Selección según características del usuario		
9	Saludo	(9) Saludo personalizado: presentar un mensaje de saludo personalizado a los clientes registrados en el sitio. (VTEX-25)
10	look & feel - preferences	(10) Deportistas : desplegar una interfaz gráfica con gráficos y colores acordes al tema de preferencia del usuario que ingresa en la aplicación, en este caso con preferencia a los deportes. (VTEX-18. - look & feel)
10	look & feel - preferences	(11) Gamers : desplegar una interfaz gráfica acorde con un usuario con preferencia a juegos interactivos como vídeo juegos. (user with preference to play interactive games such as video games) (VTEX-18. - look & feel)
10	look & feel – Edad	(12) Adultos mayores : desplegar una interfaz gráfica adecuada para clientes con dificultades visuales, si el cliente que ingresa presenta estas dificultades. (VTEX-18. - look & feel)
10	look & feel – Edad	(13) Menores de edad : desplegar una interfaz gráfica adecuada si el cliente que ingresa es menor de edad. (VTEX-18. - look & feel)
11	Descuentos – grupo	(14) Ofrecer un descuento del 10% si el usuario pertenece al grupo de “ <i>adultos mayores</i> ”. (VTEX-16; VTEX-04)
12	Descuento	(15) Contar con un componente de descuento para otorgar: -efecto: descuento de \$50.000 - causa: si es la primera compra que realiza el usuario. (VTEX-04. – Causas y efectos)
13	Descuento	(16) Contar con un componente de descuento para otorgar: - efecto: descuento del 5% - causa: por compras superiores a \$300.000
14	Descuento	(17) Contar con un componente de descuento para otorgar: - efecto: descuento del 10% - causa: por compras superiores a \$300.000 en la categoría “x”
15	Descuento	(18) Contar con un componente de descuento para otorgar: - efecto: descuento del 10% - causa: si ha realizado pedidos por encima de \$200.000 en la marca “xy”

16	Descuento	<p>(20) Contar con un componente de descuento para otorgar: - efecto: bono de \$200.000 - causa: si mínimo ha realizado 3 pedidos de la Colección “de verano 2015” Cupón de descuento: Posibilidad de asociar las promociones a cupones de descuento a través de la definición automática del código del cupón, incluir varios números de cupón de descuento en una única acción, definir el uso único para cada cliente y con una cantidad total de uso ilimitado o predefinido.</p>
Selección según el contexto del usuario		
17	Contexto - ubicación	<p>(23) Descuento por proximidad: ofrecer el flete gratis para un usuario que se encuentre ubicado cerca del almacén físico. (VTEX-nro 85)</p>
18	Contexto - localidad	<p>(25) Estilos por estaciones del año: actualizar el estilo de la interfaz gráfica de la aplicación, según la estación del año, del sitio donde se encuentre el usuario que ingrese. (VTEX-18. - look & feel)</p>

References

- Achilleos, A., Paspallis, N., & Papadopoulos, G. A. (2011). Automating the Development of Device-Aware Web Services: A Model-Driven Approach. In *2011 IEEE 35th Annual Computer Software and Applications Conference* (pp. 535–540). <https://doi.org/10.1109/COMPSAC.2011.74>
- Agustin, J. L. H. (2015). Model-driven web applications. In *Proceedings of the 2015 Science and Information Conference, SAI 2015* (pp. 954–964). London, UK. <https://doi.org/10.1109/SAI.2015.7237258>
- Alotaibi, M. B. (2013). Adaptable and Adaptive E-Commerce Interfaces: An Empirical Investigation of User Acceptance. *Journal of Computers*, 8(8), 1923–1933. <https://doi.org/10.4304/jcp.8.8.1923-1933>
- Baldoni, M., Baroglio, C., & Henze, N. (2005). Personalization for the Semantic Web. *World Wide Web Internet And Web Information Systems*, 506779(506779), 173–212. https://doi.org/10.1007/11526988_5
- Bass, L., Clements, P., & Kazman, R. (1998). *Software Architecture in Practice*. (S. E. Institute, Ed.). Reading, MA: Addison-Wesley Longman.
- Bengtsson, P., Lassing, N., Bosch, J., & Van Vliet, H. (2004). Architecture-level modifiability analysis (ALMA). *Journal of Systems and Software*, 69(1–2), 129–147. [https://doi.org/10.1016/S0164-1212\(03\)00080-3](https://doi.org/10.1016/S0164-1212(03)00080-3)
- Bernardi, M. L., Lucca, G. A. Di, Distante, D., & Cimitile, M. (2013). Model driven evolution of web applications. In *2013 15th IEEE International Symposium on Web Systems Evolution (WSE)* (pp. 45–50). <https://doi.org/10.1109/WSE.2013.6642416>
- Bettini, C., Brdiczka, O., Henriksen, K., Indulska, J., Nicklas, D., Ranganathan, A., & Riboni, D. (2010). A Survey of Context Modelling and Reasoning Techniques. *Pervasive Mob. Comput.*, 6(2), 161–180. <https://doi.org/10.1016/j.pmcj.2009.06.002>
- Brambilla, M., & Butti, S. (2014). Quince años de Desarrollo Industrial Dirigido por Modelos de aplicaciones Front-End: desde WebML hasta WebRatio e IFML. *Novática: Revista de Asociación de Técnicos de Informática.*, 36. Retrieved from <http://www.ati.es/novatica/2014/228/nv228sum.html>

- Brambilla, M., Cabot, J., & Wimmer, M. (2012). *Model-Driven Software Engineering in Practice. Synthesis Lectures on Software Engineering* (Vol. 1). <https://doi.org/10.2200/S00441ED1V01Y201208SWE001>
- Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*. <https://doi.org/10.1007/BF00143964>
- Brusilovsky, P. (2001). Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11(1–2), 87–110. <https://doi.org/10.1023/A:1011143116306>
- Brusilovsky, P., Kobsa, A., & Nejdl, W. (2007). *The Adaptive Web: Methods and Strategies of Web Personalization. The Adaptive Web* (Vol. 4321). <https://doi.org/10.1007/978-3-540-72079-9>
- Brusilovsky, P., & Nejdl, W. (2004). Adaptive Hypermedia and Adaptive Web. In M. P. Singh (Ed.), *Practical Handbook of Internet Computing*. CRC Press. Retrieved from <http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2003/brusilovsky-nejdl.pdf>
- Bunse, C., Gross, H. G., & Peper, C. (2009). Embedded system construction - Evaluation of model-driven and component-based development approaches. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5421, pp. 66–77). https://doi.org/10.1007/978-3-642-01648-6_8
- Cao, Y., & Li, Y. (2007). An intelligent fuzzy-based recommendation system for consumer electronic products. *Expert Systems with Applications*, 33(1), 230–240. <https://doi.org/10.1016/j.eswa.2006.04.012>
- Castrejón, J. C., López-Landa, R., & Lozano, R. (2011). Model2Roo: A model driven approach for web application development based on the Eclipse Modeling Framework and Spring Roo. In *CONIELECOMP 2011, 21st International Conference on Electrical Communications and Computers* (pp. 82–87). <https://doi.org/10.1109/CONIELECOMP.2011.5749344>
- Ceri, S., Daniel, F., Matera, M., & Facca, F. M. (2007). Model-driven development of context-aware Web applications. *ACM Trans. Internet Technol.*, 7(1), 387–413. <https://doi.org/10.1145/1189740.1189742>
- Cobaleda, L.-V., Mazo, R., Mejía, J., & Duitama, J.-F. (2017). MAMPA: A Model-Driven Approach to Enhance The Modifiability of Personalized Web Applications. *Submitted to*

International Journal on Software Tools for Technology Transfer (STTT).

- Cobaleda, L., & Duitama, F. (2009). Personalización de contenidos en sistemas hipermedia educativos adaptativos : una revisión. *Rev. Fac. Ing. Univ. Antioquia*, 50(1), 217–226. Retrieved from <http://ingenieria.udea.edu.co/grupos/revista/revistas/nro050/Articulo20.pdf>
- Cobaleda, L. V., Mazo, R., Becerra, J. L. R., & Duitama, J. F. (2016). Reference software architecture for improving modifiability of personalised web applications - a controlled experiment. *International Journal of Web Engineering and Technology*, 11(4), 351. <https://doi.org/10.1504/IJWET.2016.081768>
- Cristea, A. I., Smits, D., Bevan, J., & Hendrix, M. (2009). LAG 2.0: Refining a Reusable Adaptation Language and Improving on Its Authoring. In U. Cress, V. Dimitrova, & M. Specht (Eds.), *Learning in the Synergy of Multiple Disciplines: 4th European Conference on Technology Enhanced Learning, EC-TEL 2009 Nice, France* (pp. 7–21). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04636-0_4
- Cristea, A., & Verschoor, M. (2004). The LAG grammar for authoring the adaptive Web. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.* (Vol. 1, p. 382–386 Vol.1). <https://doi.org/10.1109/ITCC.2004.1286484>
- De Virgilio, R., Torlone, R., & Houben, G.-J. (2007). Rule-based Adaptation of Web Information Systems. *World Wide Web*, 10(4), 443–470. <https://doi.org/10.1007/s11280-007-0020-2>
- De Bra, P., Houben, G.-J., & Wu, H. (1999). AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In *Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia : Returning to Our Diverse Roots: Returning to Our Diverse Roots* (pp. 147–156). New York, NY, USA: ACM. <https://doi.org/10.1145/294469.294508>
- De Bra, P., Smits, D., & Stash, N. (2006). The design of AHA! In *Proceedings of the seventeenth conference on Hypertext and hypermedia* (pp. 133–134). ACM. Retrieved from <http://portal.acm.org/citation.cfm?id=1149968>
- De Bra, P., Stash, N., Boereboom, W., Chen, C., Den Ouden, J., Kunstman, M., ... Verbakel, E. (2016). ALAT: Finally an Easy To Use Adaptation Authoring Tool. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media* (pp. 213–218). New York, NY,

USA: ACM. <https://doi.org/10.1145/2914586.2914627>

- De Virgilio, R. (2012). AML: a modeling language for designing adaptive web applications. *Personal Ubiquitous Comput.*, 16(5), 527–541. <https://doi.org/10.1007/s00779-011-0418-9>
- de Vrieze, P., van Bommel, P., & van der Weide, T. (2004). A Generic Adaptivity Model in Adaptive Hypermedia. In P. M. E. De Bra & W. Nejdl (Eds.), *Adaptive Hypermedia and Adaptive Web-Based Systems: Third International Conference, AH 2004, Eindhoven, The Netherlands, August 23-26, 2004. Proceedings* (pp. 344–347). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-27780-4_48
- Di Lucca, G. A., Fasolino, A. R., Tramontana, P., & Visaggio, C. A. (2004). Towards the definition of a maintainability model for Web applications. In *Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings. Eighth European Conference on* (pp. 279–287). <https://doi.org/10.1109/CSMR.2004.1281430>
- Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting Empirical Methods for Software Engineering Research. In F. Shull, J. Singer, & D. K. Sjøberg (Eds.), *Guide to Advanced Empirical Software Engineering SE - 11* (pp. 285–311). Springer London. https://doi.org/10.1007/978-1-84800-044-5_11
- Fan, H., Hussain, F. K., Younas, M., & Hussain, O. K. (2015). An Integrated Personalization Framework for SaaS-based Cloud Services. *Future Gener. Comput. Syst.*, 53(C), 157–173. <https://doi.org/10.1016/j.future.2015.05.011>
- García, A. M. (2014). WiBAF: Within browser adaptation framework. In *CEUR Workshop Proceedings* (Vol. 1181).
- Garrigós, I., & Gómez, J. (2007). Modelado de Aplicaciones Web Reactivas al Usuario. In *XII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2007), Zaragoza, Spain* (pp. 232–241).
- Garrigós, I., Gomez, J., & Houben, G.-J. (2010). Specification of personalization in web application design. *Information and Software Technology*, 52(9), 991–1010. <https://doi.org/10.1016/j.infsof.2010.04.001>
- Garrigós Fernández, I. (2008). *A-OOH: Extending Web Application Design with Dynamic Personalization*. University of Alicante. <https://doi.org/ISBN:978-84-691-7772-3>
- Garrigós Fernández, I., Glorio, O., Hernández, P., & Maté Morga, A. (2009). An Eclipse-based

- tool for model driven web applications with personalization support. *Revista de Informática Teórica E Aplicada (RITA)*, 16(2), 99–102. Retrieved from <http://rua.ua.es/dspace/handle/10045/25153>
- Guzmán, A. R., López, V., Valverde, F., & Panach, J. I. (2012). Web 2.0 patterns: A model-driven engineering approach. In *2012 Sixth International Conference on Research Challenges in Information Science (RCIS)* (pp. 1–2). <https://doi.org/10.1109/RCIS.2012.6240420>
- Hoyos, J. R., García-Molina, J., & Botía, J. A. (2013). A domain-specific language for context modeling in context-aware systems. *Journal of Systems and Software*, 86(11), 2890–2905. <https://doi.org/10.1016/j.jss.2013.07.008>
- ISO - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. (2011). Iso/Iec 25010. Retrieved May 25, 2016, from <http://iso25000.com/index.php/en/iso-25000-standards/iso-25010>
- Jiang, T., Ying, J., Wu, M., & Jin, C. (2008). A method for model-driven development of adaptive web applications. In *Computer Supported Cooperative Work in Design, 2008. CSCWD 2008. 12th International Conference on* (pp. 386–391). <https://doi.org/10.1109/CSCWD.2008.4537010>
- Kakousis, K., Paspallis, N., & Papadopoulos, G. A. (2010). A survey of software adaptation in mobile and ubiquitous computing. *Enterprise Information Systems*, 4(4), 355–389. <https://doi.org/10.1080/17517575.2010.509814>
- Kalaignanam, K., Kushwaha, T., & Varadarajan, P. (2008). Marketing operations efficiency and the Internet: An organizing framework. *Journal of Business Research*, 61(4), 300–308. <https://doi.org/10.1016/j.jbusres.2007.06.019>
- Karat, C. M., Brodie, C., Karat, J., Vergo, J., & Alpert, S. R. (2003). Personalizing the user experience on ibm.com. *IBM Systems Journal*. <https://doi.org/10.1147/sj.424.0686>
- Kazman, R., & McGregor, J. (2012). A Mashup of Techniques to Create Reference Architectures (p. 22). Pittsburgh, PA 15213: Software Engineering Institute - Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/Presentation/2012_017_001_23352.pdf
- Kęsik, J., & Żyła, K. (2010). Usability comparison of WebRatio and symfony for educational

- purposes. *Prace Instytutu Elektrotechniki, Z. 247*, 223–240.
- Khambati, A., Grundy, J., Warren, J., & Hosking, J. (2008). Model-Driven Development of Mobile Personal Health Care Applications. In *Proceedings of the 2008 23rd IEEEACM International Conference on Automated Software Engineering* (Vol. 1, pp. 467–470). <https://doi.org/10.1109/ASE.2008.75>
- Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3. *Engineering*, 45(4ve), 1051. <https://doi.org/10.1145/1134285.1134500>
- Kwon, K., & Kim, C. (2012). How to design personalization in a context of customer retention: Who personalizes what and to what extent? *Electronic Commerce Research and Applications*, 11(2), 101–116.
- Martinenghi, D. (2014). How to Model User and Group Management. Retrieved May 13, 2015, from <http://my.webratio.com/learn/learningobject/how-to-model-user-and-group-management-v-72?link=oln72ae.redirect&pcp1x=how-to-model-user-and-group-management-v-72&nav=14&cbck=wrReq58593>
- Matook, S., & Indulska, M. (2009). Improving the quality of process reference models: A quality function deployment-based approach. *Decision Support Systems*, 47(1), 60–71. <https://doi.org/10.1016/j.dss.2008.12.006>
- Molinarioli, A. (2012). WebRatio Web Application Architecture. Retrieved October 8, 2014, from <http://www.webratio.com/learn/learningobject/WebRatio-Web-Application-Architecture?link=oln72ae.redirect&nav=14&cbck=wrReq12879>
- Montes Garcia, A., De Bra, P., Fletcher, G. H. L., & Pechenizkiy, M. (2014). A DSL Based on CSS for Hypertext Adaptation. In *Proceedings of the 25th ACM Conference on Hypertext and Social Media* (pp. 313–315). New York, NY, USA: ACM. <https://doi.org/10.1145/2631775.2631782>
- Moody, D. L., & Shanks, G. G. (2003). Improving the quality of data models: empirical validation of a quality management framework. *Information Systems*, 28(6), 619–650. [https://doi.org/http://dx.doi.org/10.1016/S0306-4379\(02\)00043-1](https://doi.org/http://dx.doi.org/10.1016/S0306-4379(02)00043-1)
- Mukhtar, M. A. O., Hassan, M. F. B., Jaafar, J. Bin, & Rahim, L. a. (2013). Enhanced approach for developing web applications using model driven architecture. *2013 International Conference on Research and Innovation in Information Systems (ICRIIS), 2013*, 145–150.

- <https://doi.org/10.1109/ICRIIS.2013.6716699>
- Oman, P., & Hagemeister, J. (1992). Metrics for assessing a software system's maintainability. In *Proceedings Conference on Software Maintenance 1992* (pp. 337–344). IEEE Comput. Soc. Press. Retrieved from <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=242525>
- Perez, B., & Correal, D. (2011). MENTA: A model-driven architecture to enable self-adaptive SOA systems. In *Computing Congress (CCC), 2011 6th Colombian* (pp. 1–6). <https://doi.org/10.1109/COLOMCC.2011.5936328>
- Politecnico di Milano. (n.d.). The Web Modeling Language. Webml.org. Retrieved June 11, 2017, from <http://webml.elet.polimi.it/webml/page1.do>
- Reed, P. (2002). Reference Architecture: The best of best practices.
- Sadat-Mohtasham, S. H., & Ghorbani, A. A. (2008). A language for high-level description of adaptive web systems. *Journal of Systems and Software*, 81(7), 1196–1217. <https://doi.org/10.1016/j.jss.2007.08.033>
- Salonen, V., & Karjaluoto, H. (2016). Web personalization: The state of the art and future avenues for research and practice. *Telematics and Informatics*, 33(4), 1088–1104. <https://doi.org/10.1016/j.tele.2016.03.004>
- Schauerhuber, A., Schwinger, W., Retschitzegger, W., Wimmer, M., & Kappel, G. (2008). A survey on web modeling approaches for ubiquitous web applications. *International Journal of Web Information Systems*, 4(3), 234–305. <https://doi.org/10.1108/17440080810901089>
- Scotton, J. D. (2013). *Supporting delivery of adaptive hypermedia*. Retrieved from <http://webcat.warwick.ac.uk/record=b2690179~S1>
- SEI. (n.d.). Software Engineering Institute. Glossary. Retrieved September 1, 2015, from <http://www.sei.cmu.edu/architecture/start/glossary/>
- Smits, D., & De Bra, P. (2011). GALE: A Highly Extensible Adaptive Hypermedia Engine. *Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia (HT 2011)*, 63–72. <https://doi.org/http://dx.doi.org/10.1145/1995966.1995978>
- Smits, D., & De Bra, P. (2012). GALE: Generic adaptation language and engine. In *CEUR Workshop Proceedings* (Vol. 872). <https://doi.org/10.1.1.416.5688>
- Stafford, J. A., & Wolf, A. L. (2001). Architecture-level dependence analysis for software systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(4),

431–451. <https://doi.org/10.1142/S021819400100061X>

- Stahl, T., & Voelter, M. (2006). *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley and Sons ISBN9780470025703.
- Stella, L. F. F., Jarzabek, S., & Wadhwa, B. (2008). A comparative study of maintainability of web applications on J2EE, .NET and ruby on rails. In *Proceedings - 10th IEEE International Symposium on Web Site Evolution, WSE 2008* (pp. 93–99).
- Tekinerdogan, B. (2004). ASAAM: aspectual software architecture analysis method. In *Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on* (pp. 5–14). <https://doi.org/10.1109/WICSA.2004.1310685>
- Trias, F. (2012). Building CMS-based web applications using a model-driven approach. In *Proceedings - International Conference on Research Challenges in Information Science* (pp. 1–6). <https://doi.org/10.1109/RCIS.2012.6240465>
- Van Der Sluijs, K., Hidders, J., Leonardi, E., & Houben, G. J. (2009). GAL: A generic adaptation language for describing adaptive hypermedia. In *CEUR Workshop Proceedings* (Vol. 473, pp. 13–24).
- Vera, P. M., Pons, C., Giulianelli, D. A., & Rodríguez, R. A. (2012). Utilizando el enfoque MDA para la construcción de aplicaciones web móviles centradas en los datos. In *XIV Workshop de Investigadores en Ciencias de la Computación* (pp. 488–492). Retrieved from <http://sedici.unlp.edu.ar/handle/10915/18937>
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering. Experimentation in Software Engineering* (Vol. 9783642290). Berlin, Heidelberg: Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-642-29044-2>