Modelling Adaptive Web Applications in OOWS

Doctoral Thesis

Gonzalo Eduardo Rojas Durán

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Department of Information Systems and Computation
Technical University of Valencia

Supervisors: Prof. Dr. Vicente Pelechano Ferragud
Prof. Dr. Óscar Pastor López

March 2008

Members of the Thesis Committee:

Prof. Dr. Antoni Granollers Saltiveri, University of Lleida, Spain
Prof. Dra. Manuela Albert Albiol, Technical University of Valencia, Spain
Prof. Dr. Francisco José García Peñalvo, University of Salamanca, Spain
Prof. Dr. José Antonio Gallud Lázaro, University of Castilla-La Mancha, Spain
Prof. Dr. Juan Carlos Casamayor Ródenas, Technical University of Valencia, Spain

Acknowledgments

I would like to acknowledge many people for helping me during my doctoral work. I would especially like to thank my advisors, Vicente Pelechano and Oscar Pastor, for their assistance, generosity and commitment. Throughout these years, they encouraged me to develop independent thinking and research skills, along with giving me their valuable assistance in communicating the results of this thesis and in writing the final dissertation.

I am very grateful for the generous time and advice I received from Dr. Geert-Jan Houben and his staff at the Web & Information System Engineering (WISE) Laboratory, in Brussels, Belgium, and at the Architecture of Information Systems Group, in Eindhoven, The Netherlands.

I extend many thanks to my colleagues and friends at the OO-Method Research Group, especially Pedro Valderas, Marta Ruiz and Isabel Díaz, for sharing their expertise and friendship with me.

I owe a special note of gratitude to all my friends at the Department of Information Systems and Computation, of the Technical University of Valencia, for all the great moments we had.

Last, but very specially, I would like to thank my family, for all their support, kindness, care, and unconditional love, and my girlfriend Melissa, for her love, wisdom, patience and encouragement.

This research would not have been possible without the financial support from the Government of Chile, through the "Gobierno de Chile - BID" Doctoral Scholarship, and from the Faculty of Engineering of the University of Concepción, Chile.

Contents

Acknowledgments **i**

Abstract **xv**

Resumen **xvii**

Resum **xix**

1 Introduction **1**
1.1 Motivation of the Thesis 3
1.2 Problem Statement 7
1.3 Description of the proposal 9
1.3.1 Objectives of the thesis 10
1.4 Hypothesis of work 13
1.5 Research Methodology 14
1.6 Structure of the Dissertation 15

2 State of the Art **19**
2.1 Introduction 19
2.2 Non-Personalized Interaction in the Web 20
2.3 Adaptive Hypermedia Systems 22
2.3.1 Adaptive Web Applications 23
2.3.2 The notion of Context 24
2.3.3 Adaptive Methods 27
2.3.4 Adaptive Techniques 30
2.4 Adaptivity from a Model-Driven Perspective 33
2.4.1 WebML 34
2.4.2 OO-HDM 35

2.4.3 UWE 37
2.4.4 WSDM 38
2.4.5 OO-H 39
2.4.6 Hera 40
2.5 Analysis of the State of the Art 41
2.5.1 User-related and Adaptivity Requirements 42
2.5.2 User Modelling 43
2.5.3 Adaptivity Modelling 43
2.6 Conclusions 44

3 The OOWS Modelling Process for Adaptive Web Applications 47
3.1 Introduction 47
3.2 OOWS Modelling Process for Non-Adaptive Web Applications 48
3.2.1 Problem Specification in OOWS 49
3.2.2 The OOWS Navigational Model 52
3.3 Presentation of the Proposal 53
3.3.1 Conceptual Modelling Approach for Adaptive Web Applications 55
3.3.2 Requirements Specification for Adaptive Web Applications 57
3.3.3 Transforming Adaptivity Requirements into Adaptivity Models 58
3.4 Scope of the Proposal 58
3.5 Conclusions 60

4 Modelling Adaptive Web Applications in OOWS 63
4.1 Introduction 63
4.2 The OOWS Navigational Model for Non-Adaptive Web Applications 65
4.3 User Modelling approach 71
4.3.1 Previous considerations 71
4.3.2 User Modelling in OOWS 73
4.4 Extended OOWS Navigational Metamodel 80
4.4.1 User description in the OOWS Navigational Metamodel 82

4.5 Conceptual Primitives to model Adaptive Web Applications . . . 84
4.5.1 Reachability of Navigational Contexts . . . 86
4.5.2 Adaptive Population Filter . . . 91
4.5.3 Adaptive Ordering Pattern . . . 98
4.5.4 Mixed Navigational Relationships . . . 105
4.5.5 Visibility of Navigational Relationships . . . 113
4.5.6 Adaptive Ranking of Navigational Relationships . . . 119
4.5.7 Visibility of Attributes and Operations . . . 127
4.6 Conclusions . . . 133

5 Supporting the modelling of Adaptive Methods in OOWS 137
5.1 Introduction . . . 137
5.2 Adaptive Presentation Methods in OOWS . . . 138
5.2.1 Additional Explanations . . . 140
5.2.2 Prerequisite Explanations . . . 151
5.2.3 Comparative Explanations . . . 154
5.2.4 Explanation Variants . . . 157
5.2.5 Sorting . . . 165
5.3 Adaptive Navigation Methods in OOWS . . . 169
5.3.1 Global Guidance . . . 170
5.3.2 Local Guidance . . . 180
5.3.3 Global Orientation Support . . . 185
5.3.4 Local Orientation Support . . . 189
5.4 Conclusions . . . 193

6 Requirements Specification for Adaptive Web Applications 195
6.1 Introduction . . . 195
6.2 OOWS Requirements Specification Phase . . . 196
6.2.1 Specification of Functional Requirements . . . 197
6.2.2 Specification of Navigational Requirements . . . 201
6.2.3 Adaptivity requirements in OOWS . . . 209
6.3 Specification of User-related Requirements . . . 211
6.3.1 User Stereotypes Diagram . . . 212
6.3.2 Users Specification . . . 215

6.4 Identification and Description of Adaptive Tasks . . . 218
6.4.1 Identification of Adaptive Tasks . . . 218
6.4.2 Description of Adaptive Tasks . . . 219
6.5 Specification of Adaptive Task Achievement . . . 222
6.5.1 Extended Data Description . . . 230
6.6 Conclusions . . . 232

7 From Adaptivity Requirements to Navigational Schemas 235
7.1 Introduction . . . 235
7.2 Fixing a Model Transformation strategy . . . 236
7.2.1 Transformations associated to the User Stereotypes Diagram . . . 236
7.2.2 Transformations associated to the User Specifications . . . 237
7.2.3 Transformations associated to the Task Diagram . . . 239
7.2.4 Transformations associated to the Use Case Diagrams . . . 239
7.2.5 Transformations associated to the Activity Diagrams . . . 240
7.2.6 Transformations associated to the Data Descriptions . . . 240
7.3 Modelling Rules: From Adaptivity Requirements to Adaptive Navigational Schemas . . . 241
7.3.1 Modelling Rule 1: Single-Object AIU's . . . 244
7.3.2 Modelling Rule 2: List AIU's . . . 245
7.3.3 Modelling Rule 3: Context Relationships . . . 248
7.3.4 Modelling Rule 4: Navigational Operations . . . 250
7.3.5 Modelling Rule 5: Navigational Attributes . . . 252
7.3.6 Modelling Rule 6: Context-Dependency Relationships . 256
7.3.7 Modelling Rule 7: Adaptive Population Filters . . . 259
7.3.8 Modelling Rule 8: Adaptive Ordering Patterns . . . 261
7.3.9 Modelling Rule 9: Adaptive Visibility of Navigational Relationships . . . 264
7.3.10 Modelling Rule 10: Multiple Adaptive Filtering and Ordering . . . 267

7.3.11 Modelling Rule 11: Adaptive Visibility of Navigational Operations 272
7.3.12 Modelling Rule 12: Adaptive Visibility of Navigational Attributes - Mixed Relationships 274
7.4 Conclusions 278

8 Conclusions 279
8.1 Summary 279
8.2 Main Contributions 281
8.3 Discussion about Implementation Issues 283
8.4 Publications 287
8.4.1 International Conferences 287
8.4.2 International Workshops 287
8.4.3 National Workshops 288
8.5 Future research 288

A The OOWS Navigational Model for Non-Adaptive Web Applications 291
A.1 Authoring-in-the-Large: Global Structure of the Navigation 292
A.1.1 Navigational Map 292
A.1.2 Navigational Contexts 292
A.1.3 Navigational Links 293
A.2 Authoring-in-the-Small: Inner structure of Navigational Contexts 294
A.2.1 Abstract Information Units (AIU) 295
A.2.2 List AIU and Single-Object AIU 295
A.2.3 Navigational Classes 297
A.2.4 Navigational Relationships 297
A.2.5 Population Filters 301
A.2.6 Ordering Pattern 302

List of Figures

2.1 Traditional “black box” versus adaptive applications 22
2.2 Context information processing in a User-Adaptive System . . 26
2.3 Taxonomy of adaptive techniques 31
2.4 Support of Model-Driven proposals to the development of Adaptive Web Applications 41

3.1 OOWS (extended OO-Method) development process 49
3.2 Current OOWS Problem Specification stage 50
3.3 Proposed enhancements to OOWS Problem Specification stage 54

4.1 Example of an OOWS Navigational Map 66
4.2 OOWS Navigational Context and its AIUs 68
4.3 Implementation of `BOOKS` context 69
4.4 Extended OOWS Class Diagram, including user characteristics 74
4.5 Class Diagram for an online bookstore, including its User Schema 78
4.6 OOWS Navigational Metamodel 81
4.7 Adaptive Primitives of the OOWS Navigational Model 85
4.8 Reachability of Navigational Contexts in the OOWS Navigational Metamodel 88
4.9 Abstract modelling example of *Reachability of Navigational Contexts* primitive 89
4.10 Concrete modelling example of *Reachability of Navigational Contexts* primitive 90
4.11 Concrete modelling example of *Reachability of Navigational Contexts* primitive 91
4.12 Adaptive Population Filter in the OOWS Navigational Metamodel 93
4.13 Abstract modelling example of *Adaptive Population Filter* primitive 94

4.14 Abstract implementation of *Adaptive Population Filter* primitive 95
4.15 Modelling example of Adaptive Population Filter 96
4.16 Implementation example of Adaptive Population Filter 97
4.17 Adaptive Ordering Pattern in the OOWS Navigational Metamodel 99
4.18 Adaptive Ordering Pattern 101
4.19 Example of Adaptive Ordering Pattern 102
4.20 Modelling example of Adaptive Ordering Pattern 103
4.21 Implementation example of Adaptive Ordering Pattern 104
4.22 Navigational Relationship Taxonomy in OOWS Navigational Metamodel 108
4.23 The three types of OOWS navigational relationships 109
4.24 Example of adaptive behaviour of mixed relationships 110
4.25 Modelling example of the three types of navigational relationships 111
4.26 Implementation example of Adaptive Mixed Relationship . . . 112
4.27 Visibility of navigational relationships in the OOWS Navigational Metamodel 114
4.28 Visibility of Navigational Relationships 116
4.29 Examples of adaptive visibility of navigational relationships . . 117
4.30 Modelling example of Visibility of Navigational Relationships . 118
4.31 Implementation example of Visibility of Navigational Relationships 119
4.32 Adaptive ranking of navigational relationships in the OOWS Navigational Metamodel 121
4.33 Adaptive Ranking of Navigational Relationships 123
4.34 Example of Adaptive Ranking of Navigational Relationships . 124
4.35 Modelling example of Adaptive Ranking of Navigational Relationships 125
4.36 Implementation example of Adaptive Ranking of Navigational Relationships 126
4.37 Visibility of Navigational Attributes and Operations in the OOWS navigational metamodel 128
4.38 Abstract modelling of Visibility of Navigational Attributes and Operations 129
4.39 Abstract implementation of Visibility of Navigational Attributes and Operations 130

4.40 Modelling example of Adaptive Visibility of Navigational Attributes and Operations 131
4.41 Implementation example of Adaptive Visibility of Navigational Attributes and Operations 132

5.1 Adaptive Presentation Methods and Techniques, with the OOWS Adaptive Primitives that support their modelling 139
5.2 Modelling example of *Additional Explanations* method, Option 1 142
5.3 Implementation example of *Additional Explanations* method, Option 1 143
5.4 Modelling example of *Additional Explanations* method, Option 2 146
5.5 Implementation example of *Additional Explanations* method, Option 2 147
5.6 Modelling example of *Additional Explanations* method, Option 3 149
5.7 Implementation example of *Additional Explanations* method, Option 3 150
5.8 Modelling example of *Prerequisite Explanations* method 152
5.9 Implementation example of *Prerequisite Explanations* method 153
5.10 Modelling example of *Comparative Explanations* method . . . 156
5.11 Implementation example of *Comparative Explanations* method 157
5.12 Modelling example of *Explanation Variants* method, Option 1 159
5.13 Implementation example of *Explanation Variants* method, Option 1 159
5.14 Modelling example of *Explanation Variants* method, Option 2 161
5.15 Implementation example of *Explanation Variants* method, Option 2 162
5.16 Modelling example of *Explanation Variants* method, Option 3 164
5.17 Implementation example of *Explanation Variants* method, Option 3 165
5.18 Modelling example of *Sorting* method 167
5.19 Implementation example of *Sorting* method 168
5.20 Adaptive Navigation Methods and Techniques, with OOWS structures that support their modelling 169
5.21 Modelling example of Global Guidance method, Option 1 . . . 172
5.22 Implementation example of Global Guidance method, Option 1173

5.23 Modelling example of Global Guidance method, Option 2 . . . 175
5.24 Implementation example of Global Guidance method, Option 2 176
5.25 Implementation of *Global Guidance* method, options 1 and 2 . 177
5.26 Modelling of *Global Guidance* method, option 3 179
5.27 Implementation of *Global Guidance* method, option 3 179
5.28 Modelling example of *Local Guidance* method 183
5.29 Implementation example of *Local Guidance* method 184
5.30 Modelling example of *Global Orientation* method 187
5.31 Modelling of *Global Orientation* method 187
5.32 Implementation examples of *Global Orientation* method 188
5.33 Modelling of *Local Orientation* method 191
5.34 Implementation of *Local Orientation* method 192

6.1 Example of a Function Refinement Tree 199
6.2 Example of a Use Case Diagram 200
6.3 Tasks Diagram 203
6.4 Documentation of elementary tasks 204
6.5 Specification of an elementary task achievement 206
6.6 Data template for *Book* entity 207
6.7 OOWS User Diagram 208
6.8 User Stereotypes Diagram for the case study 214
6.9 `CLIENT` stereotype specification 216
6.10 Tasks Diagram with Adaptive Elementary Tasks 219
6.11 Use Cases Diagram of the `List Books` Adaptive Task 220
6.12 Description of an Adaptive Elementary Task 221
6.13 Description of Use Cases from Adaptive Elementary Task . . . 221
6.14 Access Condition for Interaction Points 225
6.15 Access Condition for System Actions 225
6.16 Selection Precondition for Interaction Points 226
6.17 Sorting Precondition for Interaction Points 227
6.18 Activity Diagram of a Use Case of the `List Books` task 228
6.19 Activity Diagram of *Add Book* task 229
6.20 Data description of *Book* entity 231
6.21 Data description of the `Review` entity 232

7.1 Graphical definition of Modelling Rule 1 244
7.2 Definition of AIU's through Modelling Rule 1 245
7.3 Graphical definition of Modelling Rule 2 246

7.4 Definition of AIU's through Modelling Rule 2 247
7.5 Graphical definition of Modelling Rule 3 249
7.6 Definition of Context Relationships through Modelling Rule 3 250
7.7 Graphical definition of Modelling Rule 4 251
7.8 Definition of Navigational Operations through Modelling Rule 4 251
7.9 Graphical definition of Modelling Rule 5 253
7.10 Definition of Navigational Attributes through Modelling Rule 5 255
7.11 Graphical definition of Modelling Rule 6 257
7.12 Definition of Context-Dependency Relationships through Modelling Rule 6 258
7.13 Graphical definition of Modelling Rule 7 259
7.14 Definition of Adaptive Population Filters through Modelling Rule 7 260
7.15 Graphical definition of Modelling Rule 8 262
7.16 Definition of Adaptive Ordering Patterns through Modelling Rule 8 263
7.17 Graphical definition of Modelling Rule 9 265
7.18 Definition of Adaptive Visibility of Navigational Relationships, through Modelling Rule 9 266
7.19 Graphical definition of Modelling Rule 10, case (a) 268
7.20 Graphical definition of Modelling Rule 10, case (b) 270
7.21 Multiple Adaptive Ordering and Filtering criteria through Modelling Rule 10 271
7.22 Graphical definition of Modelling Rule 11 272
7.23 Adaptive Visibility of Navigational Operations through Modelling Rule 11 273
7.24 Graphical definition of Modelling Rule 12 275
7.25 Adaptive Visibility of Navigational Attributes and Mixed Relationships through Modelling Rule 12 277

A.1 Example of OOWS navigational map and the two types of navigational contexts 294
A.2 Example of Navigational Context, composed of a Single-Object AIU and a List AIU 299
A.3 Implementation of *Books* context 303

Abstract

During the last decades, the World Wide Web has rapidly grown in size and complexity. A huge amount of complex Web Applications is available to support the fulfilment of the most diverse tasks, by providing large amounts of information and functionality. However, the high diversity of Web users makes difficult to identify and provide an easy access to data and services that best fit their individual interaction goals.

Adaptive Web Applications is a type of Web-based systems that faces this problem, by automatically adapting the access to information pieces, services and hyperlinks to the characteristics of users and their interaction context. The development of these systems demands the adoption of an engineering approach that facilitates the specification of the required adaptive features and the user characteristics on which the adaptation is based.

The present thesis introduces a Model-Driven approach to the development of Adaptive Web Applications. This proposal integrates traditional development practices of Web Engineering with proved concepts from the Adaptive Hypermedia community. Taking as a basis the OOWS (Object Oriented Web Solutions) development process of Web Applications, this proposal defines a set of conceptual primitives (called *Adaptive Primitives*) that allow expressing well-known adaptive techniques at a high abstraction level. The definition of these primitives is supported by a User Modelling proposal. Furthermore, a set of modelling strategies offers multiple alternatives to incorporate Adaptive Methods in OOWS navigational schemas, based on the introduced Adaptive Primitives.

In order to support the modelling of adaptive features, a Requirements Model is defined, allowing the specification of Adaptivity Requirements, along with the information requirements related to the application users. Finally, a set of transformation rules is introduced, with the goal of facilitating the traceability of these requirements to the conceptual modelling phase.

Resumen

Durante las últimas dos décadas, la Web ha crecido a una gran velocidad, en tamaño y complejidad. Una gran cantidad de Aplicaciones Web soporta el cumplimiento de una amplia gama de tareas, por medio de grandes volúmenes de información y funcionalidad. Sin embargo, la gran diversidad de los usuarios de la Web dificulta la tarea de identificar y acceder a aquellos datos y servicios que satisfacen de mejor forma las necesidades particulares de cada usuario.

Las Aplicaciones Web Adaptativas son sistemas Web que plantean una solución a esta problemática, mediante la adaptación automtica del acceso a ítems de información, servicios e hiperlinks, en base a las características de los usuarios. El desarrollo de estos sistemas exige adoptar una aproximación ingenieril que facilite la especificación de las funcionalidades adaptativas a proveer, junto con las características de los usuarios en las cuales se basan dichas funcionalidades.

La presente tesis introduce una aproximación al desarrollo de Aplicaciones Web Adaptativas desde una perspectiva dirigida por modelos. Esta aproximación integra prácticas tradicionales de desarrollo de Aplicaciones Web con conceptos definidos y probados por la comunidad de Hipermedia Adaptativa. Tomando como base el proceso de desarrollo de aplicaciones Web OOWS (Object Oriented Web Solutions), se define un conjunto de primitivas conceptuales que permiten expresar técnicas adaptativas a un alto nivel de abstracción. La definición de estas primitivas es respaldada por una propuesta de Modelado de Usuarios. Además, un conjunto de estrategias de modelado permite incorporar Métodos Adaptativos a los esquemas navegacionales de OOWS, en base a dichas primitivas.

Como apoyo al modelado de características adaptativas, se ha definido un Modelo de Requerimientos para especificar Requerimientos de Adaptatividad y aquellos relacionados con la información requerida sobre los usuarios. Finalmente, se ha definido un conjunto de reglas de transformación, que facilita la trazabilidad de dichos requerimientos.

Resum

Durant les últimes dues dècades, la Web ha crescut a una gran velocitat, en grandària i complexitat. Una gran quantitat d'Aplicacions Web suporta el compliment d'una àmplia gamma de tasques, per mitjà de grans volums d'informació i funcionalitat. No obstant això, la gran diversitat dels usuaris de la Web dificulta la tasca d'identificar i accedir a aquelles dades i serveis que satisfan de millor forma les necessitats particulars de cada usuari.

Les Aplicacions Web Adaptatives són sistemes Web que plantegen una solució a aquesta problemàtica, mitjanant l'adaptació automàtica de l'accés a ítems d'informació, serveis i hiperlinks, sobre la base de les característiques dels usuaris. El desenvolupament d'aquests sistemes exigeix adoptar una aproximació ingenieril que faciliti l'especificació de les funcionalitats adaptatives a proveir, juntament amb les característiques dels usuaris en les quals es basen aquestes funcionalitats.

La present tesi introduïx una aproximació al desenvolupament d'Aplicacions Web Adaptatives des d'una perspectiva dirigida per models. Aquesta aproximació integra pràctiques tradicionals de desenvolupament d'Aplicacions Web amb conceptes definits i provats per la comunitat de Hipermedia Adaptativa. Prenent com base el procés de desenvolupament d'aplicacions Web OOWS (Object Oriented Web Solutions), es defineix un conjunt de primitives conceptuals que permeten expressar tècniques adaptatives a un alt nivell d'abstracció. La definició d'aquestes primitives és suportada per una proposta de Modelatge d'Usuaris. A més, un conjunt d'estratègies de modelatge permet incorporar Mètodes Adaptatius als esquemes navegacionales de OOWS, sobre la base d'aquestes primitives.

Com suport al modelatge de característiques adaptatives, s'ha definit un Model de Requeriments per a especificar Requeriments d'Adaptativitat i aquells relacionats amb la informació requerida sobre els usuaris. Finalment, s'ha definit un conjunt de regles de transformació, que facilita la trazabilitat d'aquests requeriments.

Chapter 1

Introduction

The adaptation of navigational and presentational characteristics is becoming a high priority requirement in current projects of Web applications, aimed at facing the multiplicity and diversity of the intended users. By integrating knowledge from Adaptive Hypermedia [1] and Web Engineering [2] research fields, the present thesis seeks to promote the systematic development of Adaptive Web Applications, through the introduction of a Model-Driven approach that incorporates adaptive methods and techniques that are being successfully implemented in traditional hypermedia systems.

The World Wide Web is one of the most revolutionary technologies introduced in the past few years. Through the Web, a huge amount of information can be accessed, manipulated and shared by millions of users, with no regard to their geographic location.

During the last two decades, the Web has rapidly grown in size and complexity. Former Web sites have evolved into complex application software, called *Web Applications*, which offer access to a large set of data through complex functionality.

However, the huge and heterogeneous set of Web users makes difficult to provide a personalized interaction, which facilitates the access to the data and services that best support the fulfilment of the particular goals of each user. In front of the diversity of similar Web applications, users rapidly quit browsing those that do not satisfy their needs quickly and efficiently, looking for other alternatives.

Facing this problem, *Adaptive Web Applications* arose as a family of Web-based systems that adapt their interaction to the characteristics of different users. Aimed at providing information and functionality that give answers to the individual needs of different users, these systems automatically adapt the presentation of information pieces, services and hyperlinks to the characteristics of the context in which the user-system interaction occurs, with special emphasis on user characteristics.

The complexity of these systems demands the adoption of an engineering approach to their development, in order to facilitate the specification of the required adaptive features and the user characteristics on which the adaptation is based.

Computer Science has faced this challenge from two main perspectives: (a) by implementating of adaptive methods and techniques that adapt the presentation of contents and links in traditional hypermedia systems, and (b) by enhancing conceptual models of non-adaptive Web applications, through the definition of navigational and presentation properties in terms of user characteristics.

Under the hypothesis that the incorporation of adaptive methods and techniques in a systematic Web development process may help to (a) transfer the successful experience of Adaptive Hypermedia Systems to the Web; (b) augment the number and quality of the developed adaptive Web applications; and (c) promote the use of methodological approaches to develop this kind of systems; we propose a Model-Driven approach based on the conceptual support of well-known adaptive methods and techniques. The characteristics of the intended users and their context are described in a User Model, which is permanently updated through the actions that user performs during a browsing session.

Taking as a basis the OOWS (Object Oriented Web Solutions) development process of Web Applications, this thesis defines a set of conceptual primitives (called *Adaptive Primitives*) that allow expressing well-known adaptive techniques at a high, domain-independent level of abstraction. From the introduced Adaptive Primitives, a set of modelling strategies is built. They offer multiple alternatives to incorporate Adaptive Methods in the conceptual schema of the navigation.

To support the adaptivity modelling process, a Requirements Model is defined, allowing the systematic specification of Adaptivity Requirements, along with the definition of the information requirements related to the application users. To support the traceability of these requirements to the conceptual modelling phase, a set of transformation rules is introduced.

In the following sections, we present the problem that motivates this thesis, by describing the context of the problem and the specific issues that it intends to solve. Afterwards, we describe the main goals of the thesis. Finally, we specify the structure of this dissertation, by describing each of its chapters.

1.1 Motivation of the Thesis

The *World Wide Web Consortium (W3C)*, which is the international organization that works on the development of protocols and standards for Web applications, conceives the World Wide Web as *"the universe of network-accessible information, the embodiment of human knowledge"* [3].

Some years ago, this definition could be considered as an overstatement. However, the increasing presence and influence of the World Wide Web on our lives is nowadays an undeniable fact. Through millions of Web sites, information from the most diverse areas is available to a growing number of people, supporting their daily activities.

As the number of available Web sites is growing at a high rate, people that access the Web may choose among a growing number of alternatives to fulfil their information goals. A recent survey of the Internet company *Netcraft* estimated the existence of 135.166.473 Web sites in September 2007, with an increase of 5.5% since the previous month [4]. If the Web maintains the current growth rate (25% per year in the last years), it will reach 200 million sites by 2010 [5].

Although high, this rate is much lower than the explosive growth rate of the first years of the Web, which reached an estimated value of 850% per year, from 1991 to 1997 [5]. During this period, Web sites not only grew in number, but also in complexity. Multiple and innovative interaction features were introduced in a very short time. Initial Web sites, composed of plain pages that offered static content and very limited functionality, evolved to complex software products: the so-called ***Web Applications***. These systems provide users with rich interaction possibilities and multiple functionalities, by means of pages that are automatically generated from large data repositories.

As a consequence of this evolution, Web users have also gained experience in their interaction with the Web. Nowadays, users are more and more conscious of their own expectations of what a Web application should provide, in the way that best fits their own needs and preferences. Jakob Nielsen, one of the world's foremost experts in Web usability [6], called the Web developers' attention to this fact, when he expressed that "*the Web is no longer a marvel of innovation, it's an everyday tool, and you differentiate yourself by providing both better content and better solutions to users' problems*" [5]. Among the myriad of Web applications that support the fulfilment of the same information goal, a user probably prefers to interact with those that allow reaching that goal in an effective and simple manner.

From different perspectives, Computer Scientists are working on improving the quality of the browsing experience in the Web. The creation of the *Quality Assurance Interest Group* at W3C [7] shows that this aspect is a top concern of Web Engineering. One of the adopted approaches is the automatic adaptation of the user-application interaction to the particular characteristics of each user.

In the 1990s, *Adaptive Hypermedia* [8] arises as a research line that concentrates the efforts on providing this adaptation, through the development of hypermedia applications that *"(...) build a model of the goals, preferences and knowledge of the individual user and use this throughout the interaction for adaptation of the hypertext to the needs of that user"* [9]. As a fruit of this research, a number of *Adaptive Hypermedia Systems* have been developed [8], incorporating *Adaptive Methods* that facilitate the access to contents and links that best fit the goals of each user. These methods are implemented through a set of *Adaptive Techniques*, which allow adapting the presentation of data items, functions and links [10].

Adaptive applications have proved to be helpful in locating users themselves into the application structure, in guiding them to choose the next best navigational alternatives [11], and in reducing the information overload, by facilitating the access to contents that are more relevant to them at that moment [12].

The development of this kind of systems, however, is a very complex process. Their intended users generally present many differences among them: they have different preferences for information, functionality and interactional features; they have a different knowledge level of the concepts of the application domain; their computer skills are diverse; and even may have different information and interaction goals from one browsing session to another. Adaptive Hypermedia Systems must be able to capture this diversity in a *User Model*, to adapt their interaction to different users, and to capture the feedback that they provide through their navigational actions in order to keep this user model permanently updated.

Most of current implementations of Adaptive Hypermedia correspond to traditional, *"non-Web"* hypermedia systems (several examples are described in [8]). This kind of products generally supports a restricted group of users in the fulfilment of very specific tasks. In the case of Web applications, however, the set of potential users is much more numerous (1.244.449.601 users worldwide in Sept 30, 2007, according to *Internet World Stats*, with a growth rate of 244,7 % between 2000 and 2007 [13]), and so are the differences between them, what increases the difficulty in elaborating a comprehensive User Model.

This complexity is augmented by the special importance that some aspects of software development gain in the case of Web applications [14], such as privacy policies, support to distributed execution and complex interfaces, among others. As a consequence, adaptivity has little presence in current Web applications, whose development is more frequently oriented to a very general and ambiguous description of users.

Although some adaptive methods and techniques have been progressively incorporated to the Web [15], there are still few well-known examples of Adaptive Web Applications. One of them is the *Amazon.com* online store

[16]. This application provides users with recommendations of products, based on the items they have purchased or positively evaluated. Furthermore, it supports users in their browsing experience, by presenting items that are related to the recently visited pages and submitted searches [17]. These characteristics provide customers with valuable help in finding the right product to purchase.

Adaptive recommendations provided by *Amazon.com* evidence the benefits of incorporating adaptive methods and techniques in Web applications. The successful case of Amazon.com is just an example of how well adaptive interaction is perceived by users. However, the low number of Web applications that enjoy similar benefits by incorporating adaptive features shows that the adaptivity problem is not solved at all.

Traditionally, Software Engineering has dealt with the complexity of software development by investing efforts on the specification of the tackled problem to the detriment of implementation aspects, by means of an intensive use of conceptual models. The adoption of this approach in the development of Adaptive Web Applications would allow developers to separate the different aspects of the application, focusing their attention on the relevant characteristics of users and on the adaptive features that the system must provide.

Currently, there are several proposals that face the development of Web applications from a Model-Driven perspective [18]. Each of them introduces a conceptual model to describe the navigational and presentation aspects of the system. In order to support the specification of adaptive features, those models have been enhanced, through the incorporation of conceptual structures with adaptive properties. This approach allows obtaining different implementations from a unique structure, for different users of the application. The final implementation is given by the fulfilment of these constraints by each user, according to her characteristics specified in a User Model. All these proposals highly contribute to the goal of providing a systematic development process of Adaptive Web Applications (a State of the Art analysis of Model-Driven approaches to Web adaptivity is presented in Chapter 2 of this dissertation).

From those Model-Driven proposals for Web applications development that still do not support the incorporation of adaptive features, we can distinguish the OOWS method. *OOWS* (Object Oriented Web Solutions) [19] is the extension of an object-oriented software production method (*OO-Method* [20, 21]) that introduces conceptual primitives to model the navigational and presentation requirements of web applications. This method faces the web application development process as a full model transformation process, where the conceptual schema becomes the program of the web application, through a model compilation-based process. To have a true model compiler for web applications, the incorporation of the quoted Adaptive Web perspective is a strong need. The importance of providing this method with the needed expressiveness to support the development of Adaptive Web Applications is the main motivation of this thesis.

1.2 Problem Statement

The described importance of Adaptive Web Applications demands the adoption of a systematic approach to their development, by means of engineering methods that deal with the complexity of such systems, in a systematic and efficient way.

As mentioned above, some Model-Driven methods of Web Application development been extended to support adaptive features. However, there is little evidence of the actual use of these proposals in real projects of Adaptive Web Applications. A common deficiency of these proposals is the lack of explicit consideration of concepts defined by the Adaptive Hypermedia research field. These concepts correspond to Adaptive Methods that have been successfully implemented in traditional hypermedia applications, by means of different Adaptive Techniques [10, 8].

The adaptive features included in a navigational schema are derived from a set of requirements whose fulfilment demands the implementation of those characteristics. However, there are several aspects that are not considered by the"traditional" requirements specifications or, at least, not with the needed level of detail. As non-adaptive Web applications do not consider more user characteristics than basic identification data, we need to provide a way to describe the required user data to perform the adaptation and the operations that allow keeping the user descriptions constantly updated on the run-time user actions. Furthermore, we need to describe the needed adaptivity characteristics of the application, i.e., how the application adapts itself to the user.

This deficiency, along with a lack of proposals to describe the specific requirements of this kind of systems, makes difficult the adoption of a systematic approach to the development of Adaptive Web Applications.

This problem may be detailed through the following points:

1. Model-Driven approaches to Adaptive Web Applications have augmented the expressiveness of their conceptual models, by means of enhanced primitives and definitions of user-based rules. However, these enhancements do not explicitly support the modelling of well-defined adaptive techniques, which have been successfully implemented in hypermedia systems.

2. Model-Driven approaches to Adaptive Web Applications do not provide developers with clear strategies that guide the conceptual modelling process. Adaptive primitives are provided, but not the way to use them in order to fulfil adaptivity requirements. Adaptive methods, which provide high level descriptions of multiple adaptive characteristics, are hardly considered.

3. Besides the requirements of traditional Web-based systems, the development of Adaptive Web Applications demands the specification of the adaptive features to be considered, and the user characteristics on which the adaptation is based. Unfortunately, there is a lack of a sound proposal to specify this kind of requirements, frequently stated through informal, textual descriptions.

4. There exists a gap between textual specifications of adaptivity requirements and their conceptual descriptions, what makes difficult to trace them along the development process.

1.3 Description of the proposal

With the purpose of replicating the successful experience of Adaptive Hypermedia Systems in the Web, Model-Driven proposals should facilitate the specification of *Adaptive Methods*, whose advantages have been proved in such systems. This can be achieved through the definition of modelling strategies or patterns, based on conceptual descriptions of the *Adaptive Techniques* that implement those methods. In this way, the specification of consolidated concepts from Adaptive Hypermedia at early stages of the development process may encourage the development of Adaptive Web Applications that incorporates these concepts, also prompting the introduction of richer possibilities of adaptive interaction.

Adopting this strategy, the present work extends an existing development process of Web applications, with the goal of providing a Model-Driven alternative to the development of Adaptive Web Applications, which is based on the specification of consolidated Adaptive Methods. A set of *Adaptive Primitives* is introduced into the Navigational Model of the OOWS method [19]. From these extensions, a set of modelling strategies has been defined, which provides different alternatives of implementation by means of respective adaptive techniques.

Adaptive primitives are introduced into an already defined navigational model, because Adaptive Methods affect the presentation of the contents and hyperlinks of a page, which are described in the navigational schema of the application. Conceiving this schema as a view of the structural schema of the application domain, characterization of the intended users of the application is incorporated into this structural definition.

Furthermore, in order to support the modelling decisions that can be made from the new conceptual structures, a Requirements Model for Adaptive Web Applications has been introduced, which comprises the specification of the requirements of adaptivity and the information of the intended users on which the adaptivity is based. Aimed at supporting the traceability of these requirements, a set of transformation rules has been defined, which allows systematically obtaining the correspondent conceptual description of each adaptivity requirement in terms of the introduced Adaptive Primitives.

1.3.1 Objectives of the thesis

With the intention of providing the required solutions to the previous problems, the main goals of this work are the following:

1. **Enhancement of a Model-Driven method of Web Applications development, by introducing conceptual primitives that support the modelling of Adaptive Web Applications, in terms of the high-level specification of Adaptive Techniques.**

   From the analysis of how the non-adaptive Navigational Model of the OOWS method could be enhanced to support the description of adaptive techniques, we have defined a set of *Adaptive Primitives*. Each of these conceptual structures allows expressing one or more technique of adaptive navigation (such as *link-hiding* or *link-ordering*) and adaptive presentation (e.g., *conditional fragments*, *strechtext*), lifting their traditional consideration in the Solution Space (implementation level) to the Problem Space (conceptual modelling level).

   In order to completely specify the adaptive features of the modelled application, an *object-oriented User Model* is introduced. This model is based on the description of intended users of the application, in terms of three main views, which respectively describe: (a) the individual information of the user that is relevant to the adaptation goals; (b) the relevance (interest, preferences) that domain objects have for the user, under the definition of relevance as a measure of how closely a data instance or an operation matches the user's needs; and (c) the navigational behaviour of the user, i.e., the interactions that involve navigation between Web pages. Among these three views, a given User Schema also incorporates the operations that keep this description updated, according to the navigational actions that user performs during a browsing session. We have adopted an object-oriented modelling approach [22] attending the need to define operations that capture user information over domain concepts, providing a higher expressiveness in terms of the functional dimension of the Web application.

   The constraints that determine the behaviour of the Adaptive Primitives are defined in terms of the structures of the User Model. The final implementation of a given primitive is determined by the fulfilment of these constraints by a given user, according to her particular instantiation of the User Schema. Reflecting the first order importance of user specification in Adaptive Systems, the User Model as been defined as part of the Structural Schema (Class Diagram in OOWS) of the application.

2. **Definition of a set of modelling strategies to specify Adaptive Methods, in terms of the introduced Adaptive Primitives.**

   Adaptive Methods are high level strategies to support users in their browsing experience, by means of the implementation of different Adaptive Techniques. To incorporate Adaptive Navigation and Presentation methods in OOWS navigational schemas, we introduce a set of guidelines for the use of the introduced Adaptive Primitives in the modelling of these methods. These traceability rules are described in terms of the adaptive techniques that are traditionally used to implement each method.

   This approach stresses the relevance that this thesis gives to Adaptive Hypermedia concepts. Along with introducing tools to model adaptive features, it also permit informing Web developers about the best way to use them for describing adaptive methods whose efficacy has been proved in adaptive hypermedia systems.

3. **Definition of a *Requirements Model* for Adaptive Web Applications, which supports the specification of the requirements related to the needed information about users and the adaptive features that the system must provide.**

   The introduced Requirements Model provides the capabilities to specify:

   - *User-related Requirements*, i.e., those requirements that describe the potential users of the application. Supporting a coarse-grained adaptivity, groups of similar users are identified and hierarchically ordered, by adopting a stereotype-based approach [23]; to support a fine-grained adaptivity, specific user characteristics are defined, comprising personal, application-related and navigational behaviour features, along with the functionality of the system that each user is enabled to access.

- *Adaptivity Requirements*, i.e., those requirements that describe the adaptive functionalities that the application should provide to the user. By adopting a task-based approach, tasks that are available to different user groups are identified, classified and detailed. Furthermore, the fulfilment of those tasks that incorporate adaptive features is described through the involved user-application interactions.

To describe the navigational requirements of non-adaptive web applications, OOWS considers a hierarchical task classification and specific descriptions through UML-compliant Activity Diagrams [24]. In response to the enhancements introduced in the conceptual level, *User-related Requirements* are described in base of the definition, classification and refinement of *User Stereotypes* [23], while *Adaptivity Requirements* are incorporated in the navigational task descriptions, by enhancing the non-adaptive Activity Diagrams.

4. **Definition of a *set of transformation rules* from specifications of user-related and adaptivity requirements to Adaptive Primitives.** The lack of a sound proposal for the specification of Adaptivity and User-related requirements also hinders their correct traceability to later stages of the development. The definition of a Requirements Model and a extended Navigational Model that support Adaptive Web Applications permits to fill this gap.

   Each possible occurrence of the introduced Requirements Model is incorporated as the input of a transformation rule, whose output is the correspondent navigational description, in terms of the introduced Adaptive Primitives.

   Complementing the obtained set of transformation rules, guidelines to derive the User Schema of the application from the set of User-related requirements are also presented.

1.4 Hypothesis of work

The main hypothesis of this work is the following:

> A major confluence between the research areas of *Web Engineering*, which provides a solid rationale in the development of traditional Web applications, and *Adaptive Hypermedia*, which provides well-defined concepts of adaptive methods and techniques successfully implemented in hypermedia systems, may promote the adoption of a systematic approach to the development of Adaptive Web Applications.

This hypothesis discards the need of reformulating the development process of traditional, non-adaptive systems, in favor of incorporating a "layer" of Adaptivity in that process, which comprises the specification of users and the adaptive behaviour of the application, in terms of solid Adaptive Hypermedia concepts. This hypothesis derives the following considerations on which this thesis is based:

- Adaptive techniques that are being incorporated in Adaptive Hypermedia Systems, usually at late stages of their development, can be succesfully described at a higher abstraction level, in the conceptual modelling of the application, in terms of Adaptive Methods.
- Conceptual models that are used to specify non-adaptive Web applications and which allow describing the domain concepts and the navigational structure of the application, can be enhanced with new structures to support the modelling of Adaptive Web Applications. These structures are intended to model the characteristics of the application's users and the adaptive techniques based on these characteristics.
- The specification of Adaptivity Requirements requires an enhancement, and not a big reformulation, of already existing Requirements Model for non-adaptive Web applications. Mechanisms to specify the requirements related to the users and the needed adaptivity characteristics must be provided.

1.5 Research Methodology

The adopted research methodology for developing this thesis is the *action-research* method. This approach is very suitable for research on Information Systems. It allows producing relevant research results based on practical action, with the goal of solving an immediate problem situation, by applying theoretical results.

The methodological approach of this thesis can be described through the following stages:

1. Study and analysis of the State of the Art;
2. Introduction and development of the proposal to solve the problems detected in the previous stage;
3. Validation of the results through their application in evolving prototypes; and
4. Consolidation and description of the final proposal.

The results of these activities were documented, by means of the generation of reports and publications. Phases 2 and 3 are performed in iterative, incremental loops, capturing the feedback from the validation of empirical results. From this, new elements to the proposal were incorporated and already proposed ones were corrected.

For the concrete definitions of each phase of the proposal, a Bottom-Up approach was adopted, taking into account the traditional cascade order of software development. At first, in terms of the deficiencies detected in the State of the Art, a Navigational Model was enhanced to support the modelling of adaptive techniques, from those classified by the Adaptive Hypermedia community in [10]. User characteristics arisen in this process, a User Model proposal was proposed. Afterwards, an enhancement to a Requirements Model was introduced, to give support to the modelling decisions that are possible to made in base of the new conceptual tools. Finally, the derivation process between the requirements specification and the conceptual modelling of Adaptivity was tackled.

1.6 Structure of the Dissertation

The present dissertation is structured as follows:

- Chapter 2 presents a review of the theoretical rationale that supports the development of Adaptive Web Applications, emphasizing the importance of providing an adaptive interaction in the Web.

  An analysis of problems that users face when interacting with non-adaptive Web applications is presented, and the solutions that Adaptive Hypermedia Systems provide. The main concepts of these applications are reviewed: Adaptive Techniques, which implement a set of Adaptive Methods, considering navigational and presentation aspects. Finally, a brief review of the way that Model-Driven proposals of Web development have faced the challenge of give support to Adaptive Web Applications.

- In Chapter 3, we analyze the most relevant approaches to the development of Adaptive Web Applications, from a Model-Driven perspective.

  Each proposal is analyzed in terms of the way that adaptive features are modelled and the development phases on which these aspects are considered, concluding with a summary of the most remarkable aspects of this analysis.

- Chapter 4 introduces enhancements to the Conceptual Modelling phase of OOWS Method, in order to support the modelling of Adaptive Web Applications. These enhancements correspond to the definition of a User Modelling strategy and the introduction of Adaptive Primitives into the OOWS Navigational Model.

  At first, the proposed User Modelling strategy is introduced. A User Model is defined as an extension of the OO-Method Class Diagram, comprising three views that describe the personal characteristics of users, their navigational behaviour and the domain concepts.

  Afterwards, a set of Adaptive Primitives that enhances the OOWS Navigational Model is introduced. These conceptual primitives give support to the modelling of well-known adaptive techniques. Each primitive is introduced according to this structure:

  - a brief description of the type of adaptation that the primitive supports;
  - the description of the adaptive primitive, including its definition in the OOWS Navigational Metamodel and a textual specification in terms of the structures of this metamodel; and
  - an abstract example of how the primitive should be included in an OOWS Navigational Schema.

- Chapter 5 describes strategies to model Adaptive Navigation and Presentation Methods, in terms of the Adaptive Primitives introduced in the previous chapter.

  For each considered method, one or several modelling options are detailed and exemplified with a case study, corresponding to an online bookstore.

- Chapter 6 introduces the proposed Requirements Model of Adaptive Web Applications. Firstly, the current proposal for Requirement Specification of OOWS is briefly described. Afterwards, the enhancements provided to support User-related requirements are introduced: a global specification of users, through the User Stereotype Diagram, and a detailed specification, through a tabular User Specification. Finally, the proposal to describe adaptive requirements is introduced, through an extended Activity Diagram and a Data description, both incorporating user-centered constraints.

- Chapter 7 presents a set of traceability rules that describe how the proposed requirements specifications of Adaptive Web Applications may be modelled in an OOWS Navigational Schema, enhanced with the Adaptive Primitives introduced in Chapter 4.

  In the case of Activity Diagrams and Task Descriptions, a set of traceability rules is described in detail. Each rule is defined in terms of a Requirement Specification occurrence and the corresponding modelling option.

- Chapter 8 presents the conclusions of this thesis, summarizing and discussing its main contributions. Finally, a set of future research works that complement this thesis is presented.

- Appendix A describes the main aspects of the Navigational Model of OOWS Web development process, on which this work is based.

Chapter 2

State of the Art

2.1 Introduction

The introduction of adaptive features into Web applications is a response to multiple and increasingly important problems that Web users face everyday. These problems are mainly caused by the huge amount of information available through the Web and the difficulty in characterizing the high diversity of users. They have been faced from two main research perspectives: Adaptive Hypermedia and Model-Driven Development.

Several adaptive features have been introduced in traditional hypermedia systems, with encouraging results. As a fruit of this experience, a set of concepts has been defined and reused in this kind of systems. However, there are very few examples of the incorporation of these concepts into Web applications. To transfer the successful experience of Adaptive Hypermedia Systems to the Web environment, these concepts should be adopted by traditional methods of Web application development.

This chapter presents a review of the main concepts that have been defined and implemented by the Adaptive Hypermedia community, and a review of the main Model-Driven approaches to Web Adaptivity. The first section describes the main problems of the interaction experience provided by non-adaptive Web applications. The second section describes the solution proposed by hypermedia researchers: *Adaptive Hypermedia Systems*. In this section, we describe: the main characteristics of these systems; the concept of *context*, on which adaptive functionality is based; and the concepts that describe the conceptual strategies to provide adaptivity (*Adaptive Methods*) and the techniques used to implement these strategies (*Adaptive Techniques*). Finally, the third section presents a review of current Model-Driven approaches to Web Adaptivity, along with a comparative analysis of them in order to detect shared characteristics and their main contributions to the development of Adaptive Web Applications.

2.2 Non-Personalized Interaction in the Web

The important role that the World Wide Web plays in our life is an evident fact. An important part of our daily tasks is supported, and sometimes ruled, by the interaction with Web Applications from the most diverse areas.

As the huge set of data and services that the Web provides continues to grow in size and diversity, users of Web applications (Web users) have improved their ability to quickly and easily access information. However, the rapid evolution of the Web has exceeded the human ability to detect and retrieve the information that best fits the individual needs. As early as 1995, Ginige et al. recognized this problem, by declaring that *"our unprecedented ability to rapidly obtain and use information from a growing range of sources has reached the stage where the time required to absorb and analyze it is becoming prohibitive. Many people find it increasingly difficult to keep up. We now have access to so much information that it is difficult, if not impossible, to sift through it and identify those areas relevant to us. This problem will not likely be easily resolved"* [25]. Some of the alluded troubles may be described through the following items:

- **Information overload**. From the complete set of data and services that a Web application provides, a given user may be interested in or have the needed knowledge to access only a small subset. Finding this relevant information typically involves browsing through a huge amount of irrelevant information, what increases the probability of getting disappointed and leaving the application. This phenomenon corresponds to what the sociologist Alvin Toffler called *"Information Overload"* in his book *Future Shock* [26], referring to the state of having too much information to make a decision or remain informed about a topic.

- **Huge and heterogenous set of potential users**. The group of people that potentially access a Web application is unmeasurable and very difficult to describe, due to the high diversity of their characteristics and interaction goals. This diversity is hardly considered by most of the current Web applications, whose interaction possibilities are oriented to a very global description of their users.

- **Mismatch between mental models of users and developers**. A Web developer has a mental model of the proper way that contents must be organized in a hypertext structure. This model rules the development of the application. However, users generally have a different notion of the best organization of the information, being compelled to follow a non-intuitive navigational path to find the information that they are looking for. For instance, users may visit pages that they do not understand because the developer expected them to visit some other page first.

- **Lost-in-Hyperspace**. In their search for relevant information, users may get easily disoriented, when the hyperspace is reasonably large. This fact has been called the *"Lost-in-Hyperspace syndrome"* [27], and it is usually caused by a poor supply of clues about the current position of users and a little guidance about the best next navigational paths to follow. An overview of the application's topology (traditional site maps) is only a partial solution, because the hyperspace is usually much too large to be displayed.

These problems may be reasonably solved if individual characteristics of the potential users of the application receive more attention throughout the development process. However, most of Web applications act regardless of who, when and where the user is, whether she is a beginner or an expert user, etc. The *independence from the context of use* has traditionally ruled the software development process. Computer Science has usually considered software systems as *"black boxes"* (as pointed in [28]), where the output is completely determined by the input (as shown in Fig. 2.1, left). Although that vision of software systems is clearly narrow, in the trade-off between the desire of abstraction and the need for context-sensitivity, the trend in software development has been clearly oriented towards the context-abstraction.

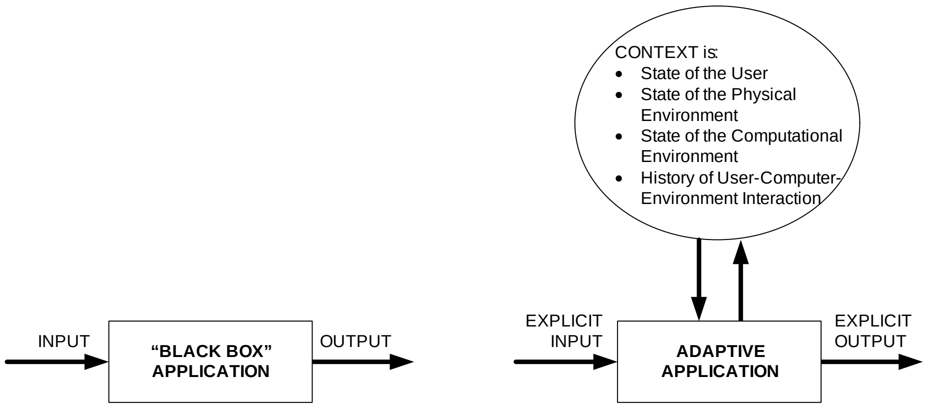

CONTEXT is:
State of the User
State of the Physical Environment
State of the Computational Environment
History of User-Computer-Environment Interaction
INPUT
"BLACK BOX" APPLICATION
OUTPUT
EXPLICIT INPUT
ADAPTIVE APPLICATION
EXPLICIT OUTPUT

Figure 2.1: Traditional “black box” versus adaptive applications

2.3 Adaptive Hypermedia Systems

In front of the problems described above, users increasingly need to interact with applications that consider their particular interests, preferences and goals. This need of personalizing the user-application interaction is present in all hypermedia systems with a reasonably large, highly interconnected information space. In the case of Web applications, as a subset of those systems, this problem becomes even more important, due to the number and heterogeneity of their potential users and interaction contexts.

In order to overcome the described limitations, **Adaptive Hypermedia Systems** (AHS) [9] arise as a new family of software, which goes beyond the classic “black-box” view. According to the definition given by the Adaptive Hypermedia community [10], Adaptive Hypermedia Systems are “*all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user*”.

In AHS, the behaviour of the system is not only determined by explicit inputs, but also by the *context* in which the user accesses the application (Fig. 2.1, right). This context comprises everything that affects the computation, except the explicit input and output. The result of a computation affects the output and the context itself. To describe this context, AHS build a model of the goals, preferences and knowledge of individual users, along with environmental features. This model is usually called **User Model** [29].

The aspects of the application that can be adapted vary from *presentation aspects*, like the media used, the length of the presentation, the difficulty, the style, etc., to *navigational aspects*, such as the availability of hyperlinks offered to the user, or the presentation of these hyperlinks.

2.3.1 Adaptive Web Applications

As a special type of hypermedia systems, Web applications can also provide adaptive features to their intended users. In *Adaptive Web Applications* [30], users are even more numerous and heterogeneous than users of AHS: any person that accesses the Web is a potential user of any Web application, and the same user may interact with a given application seeking to fulfil distinct interaction goals in different moments.

Adaptive Web Applications that provide an effective adaptive interaction, based in a comprehensive description of the intended users, answer the mentioned problems as follows:

- **Information Overload**. An Adaptive Web Application privileges the access to the most relevant contents for a given user or to those that she is able to understand, to the detriment of those less suitable to her needs.
- **Huge and heterogenous set of potential users**. By means of a highly descriptive User Model, an Adaptive Web Application can provide a customized interaction to very different users.
- **Mismatch between mental models of users and developers**. Navigational actions that users perform in an Adaptive Web Application keep the User Model permanently updated, changing the interac-

tion possibilities that are offered to the user. In this way, conceptions of developers and users about the best distribution of contents among the hyperspace tend to converge, facilitating the browsing experience.

- **Lost-in-Hyperspace**. An Adaptive Web Application can inform users about their current position in the hyperspace and suggest to them the best navigational alternatives to follow in order to fulfil their interaction goals.

The implementation of these solutions depends on the specification of the characteristics that affect the user-application interaction, specially those related to the user herself. The notion of context is discussed in the following section.

2.3.2 The notion of Context

The concept of context rules the description of adaptive features in hypermedia systems. Its specification strongly determines the quality of the adaptive functionality that the system provides.

The context on which the user-application interaction occurs may consider not only characteristics of the user (e.g., identification, preferences, knowledge level about a given topic, previous navigational actions), but also physical and environmental aspects that influence this interaction, such as spatiotemporal features, browsing device, weather conditions, recent news, etc.

In spite of its importance to the development of adaptive applications, the concept of *Context* has just recently received some attention by researchers. Lieberman and Selker expressed in [28] that "*a considerable portion of what we call intelligence in artificial intelligence or good design in human-computer interaction actually amounts to being sensitive to the context in which the artifacts are used. Doing the 'right thing' entails that it be right given the user's current context*". Due to this little attention, the boundaries of the definition of context are still blurry and highly dependent on particular researchers and/or developers. Most people tacitly understand what context is, but they find it hard to elucidate, and thus proposed definitions are based on examples or synonyms of context.

In some of the first works that introduce the term "context-aware", context is defined as *location, identities of nearby people and objects, and changes to those objects* [31], whereas [32] also incorporates *environmental features* such as time of day, season and temperature. Dey [33] enumerates context as the *user's emotional state, focus of attention, location and orientation, date and time, objects, and people in the users environment.* By using similar terms, some works define context as the user's environment [32], while others consider it to be the application's environment [34].

The point is that all the features listed above may be part of the context of an adaptive application. As Dey states, *"context is all about the whole situation relevant to an application and its set of users"* [35], and what is relevant to an application may be irrelevant to another. Because it summarizes the main ideas exposed above, in this work we adopt the definition of context agreed by *Human-Computer Interaction* research community [36], which says that *"context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"* [37].

For the adaptivity purposes of a particular system, the relevance of these features is determined by their relationship with the user interaction. If the access to a given content or functionality is adaptive to different users, and the values of a given feature are considered as an input of the involved adaptation, then this feature is part of the relevant context. For instance, some educational contents may be presented to users of different ages in different writing styles; different products on sale may be recommended to a user, according to her previously visited products. In these cases, the user age and the history of visited items are included in the context of the respective adaptive applications.

The context can be modified by the actions that a user performs during a browsing session. For instance, the knowledge of a student may be updated by visiting a given educational application, or the preferences of a customer may change according to a purchase that she makes. An adaptive application can gradually change its behavior based on the changing user knowledge or costumer preferences. If the values of a given feature can be modified by the actions that a user performs, then this feature must be part of the context of the adaptive application.

Concerning to the distinction between the terms *Context* and *User Model*, we can state that the User Model is an abstraction of the context of the application that only includes those characteristics that are directly related to the users of the application. We define a ***User-Adaptive Application*** to those Adaptive Systems whose context is completely defined by a User Model, i.e., the context information do not change if the user is the same. In [38], Jameson defines a User-Adaptive System as *"an interactive system that adapts its behaviour to individual users on the basis of processes of user model acquisition and application that involve some form of learning, inference or decision making."*

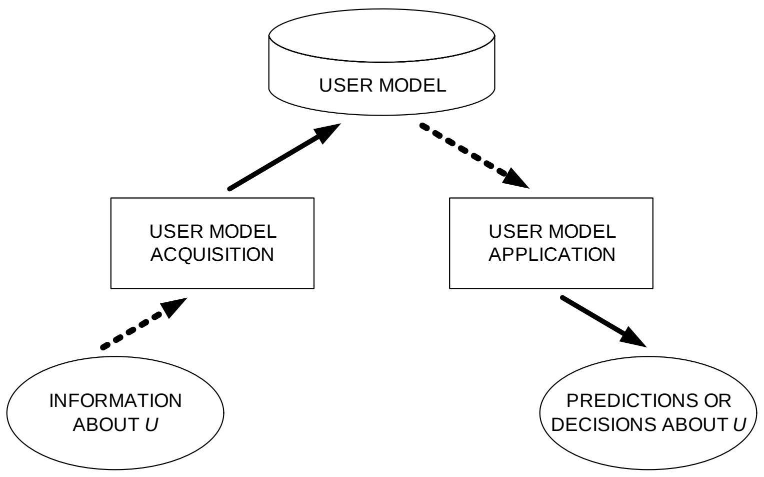


Figure 2.2: Context information processing in a User-Adaptive System

Figure 2.2 shows how the context information in a User Adaptive System (UAS) is processed, according to the mentioned definition of Jameson [38]. In the figure, ovals represent input or output; rectangles: processing methods; cylinder: stored information; dashed arrows: use of information; solid arrows: production of results. A UAS makes use of some type of information about the current individual user *U*, such as the choices *U* has made when filing a personal questionnaire or the links that has activated. In the process of user model acquisition, the system S performs some type of learning and/or inference on the basis of the information about U in order to arrive at some user model, which at first concerns only limited aspects of U. In the process of user model application, S applies the user model to the relevant features of the current situation in order to determine how to adapt its behavior to U.

2.3.3 Adaptive Methods

The examples presented in the previous sections show that the context of an adaptive application and its adaptive features are mutually dependent. Context is defined in terms of the required adaptive functionalities of the system, which are specified in terms of the characteristics that describe the context.

With the goal of distinguishing adaptive functionalities that have been implemented in Adaptive Hypermedia Systems, a set of ***Adaptive Methods*** have been defined. At a high level of abstraction, these methods describe strategies that have been adopted to fulfil diverse adaptation goals.

Adaptive Methods are classified into two groups: ***Adaptive Presentation Methods***, which are based on the adaptation of the presentation of contents, and ***Adaptive Navigation Methods***, which adapt the navigational paths available to users.

2.3.3.1 Adaptive Presentation Methods

Adaptive Presentation methods provide users with different options to access the same contents in an adaptive way. This can be achieved, for instance, by defining adaptive prerequisites to those contents; by establishing comparisons with already known concepts; or by sorting the information according to user preferences.

Some works define the adaptivity type provided by these methods as *Adaptive Content* (an important example of this is the PhD Thesis of Nora

Koch [39]), defining Adaptive Presentation as the adaptation of visual aspects, such as the graphical presentation or the Web page layout. However, the present work adheres to the definition provided by the Adaptive Hypermedia community, considering the term “Adaptive Presentation” referred to the presentation of the contents of the Web application.

The methods of Adaptive Presentation are:

1. ***Additional Explanations***

   It is the most popular method of adaptive presentation [10]. Its goal is to match the level of knowledge and/or expertise of the different users about a certain concept, by providing explanations that they are able to understand or that are interesting to them. In addition to a common presentation of the contents, some category of users can access some additional information, which is specially prepared to them and not shown to other kind of users. For example, low level details can be hidden from users with little knowledge of a given concept, which can be provided with basic explanations instead.

2. ***Prerequisite Explanations***

   This method adapts the presentation of the information of a concept according to the knowledge level that a user has of related concepts. An explanation about a concept is provided only once the information of prerequisite concepts has been checked by the user.

3. ***Comparative Explanations***

   This method allows presenting comparative information between the currently visited concept and others about which the user has enough knowledge.

4. ***Explanation Variants***

   This method proposes to manage different explanations of the same concept for different users, according to their specific characteristics. Variants in the language or the writing style can be provided.

5. ***Sorting***

   This method allows sorting the fragments of information about a concept according to their relevance to the user. The most relevant pieces are placed close to the top of the presentation unit.

2.3.3.2 Adaptive Navigation Methods

Adaptive Navigation methods are implemented to help users to navigate through the most relevant paths, according to their characteristics. Their main goal is providing orientation to users into the hyperstructure, and guidance for choosing the best navigational alternative to fulfil their particular goals. These methods are the following:

1. ***Global Guidance***

   In some hypermedia systems, users have some information goals whose fulfilment requires to access information from several nodes across the hyperspace. The goal of *Global Guidance* is to help the user to find the shortest way to access the information that she requires. It can be provided by suggesting the best following link from those included in the currently visited node, at each step of the browsing process, or by sorting the links available in a node according to their relevance to the global goal.

2. ***Local Guidance***

   Similarly to Global Guidance, *Local Guidance* method helps the user to choose the most relevant links from the currently visited node. However, Local Guidance is not based on the fulfilment of a global information goal. Instead, this method suggests relevant links basing on the preferences, knowledge and/or background of the user. The obtained recommendation is intended to support only the very next navigation step, with no concern for further pages to visit.

3. ***Local Orientation Support***

   The goal of this method is to help users to know their current position in the hyperspace, in terms of information of the surrounding pages. Local Orientation Support has been implemented by providing additional information about the nodes reachable from the current one, and by limiting the navigation opportunities to give priority to the most important links.

4. ***Global Orientation Support***

   The goal of this method is to help the user to understand the structure of the entire hyperspace and her absolute position in it. This is usually achieved by providing a navigational map and guided tours to familiarize the user with the hyperstructure.

5. ***Personalized Views***

   This method lets users organize personalized goal-oriented views of the hyperspace, in order to reduce its complexity. This is made by selecting the links to all hyperdocuments that are relevant to a particular goal. The selection process can be made directly by the user or by the system.

2.3.4 Adaptive Techniques

Adaptive methods described above are available to users by means of different implementation techniques. Adaptive Hypermedia community [10] has classified these techniques according to the features of the application that they help to adapt: (a) *Adaptive Presentation Techniques*, which allow adapting the displaying of contents, and (b) *Adaptive Navigation Techniques*, to adapt the displaying of links. Figure 2.3 shows the adaptation techniques of each group.

2.3.4.1 Adaptive Presentation Techniques

These techniques implement the described Adaptive Presentation methods. In order to do this, they can adapt the multimedia material, the modality of the medium used for presenting the information (sound, text, image, etc.), or reconstruct the text by inserting, removing, altering, or highlighting fragments of information [40].

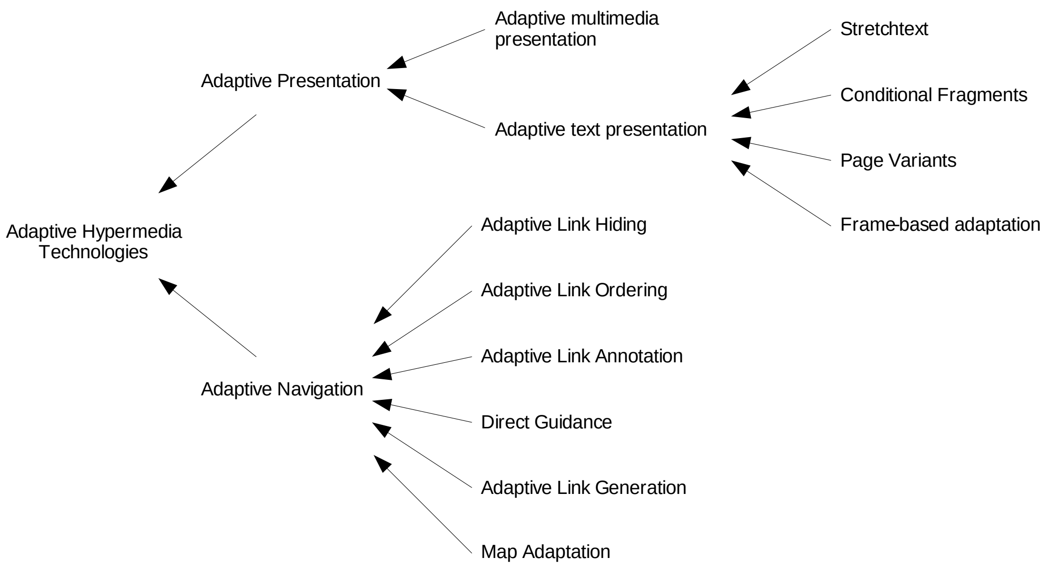


Figure 2.3: Taxonomy of adaptive techniques

As a hypermedia application usually includes contents in textual and multimedia formats, this group of techniques is classified into two groups: *adaptive text presentation* and *adaptive multimedia presentation*, respectively. For example, by applying an adaptive text technique, an educational text can be summarized for users that have enough knowledge of the related topic, while a detailed text can be presented to novice users.

The following groups of techniques are mainly oriented to textual data, but most of them can also be applied to multimedia content in general:

1. ***Stretchtext***: The content is presented as a placeholder. When activated, the placeholder is replaced by an extended version of the related text. The adaptive system determines which fragments are "stretched" (expanded) and which are "shrunk" (collapsed) for the initial presentation, based on user characteristics. Afterwards and based on the user-application interaction, the system decides which placeholder will be stretched, and which shrunk.

2. ***Conditional Fragments***: Some information items can be irrelevant or not suited for certain users. The system determines which piece of

information should be presented to the user, in base of her characteristics.

3. ***Page Variants***: The system provides different user groups with alternative versions of a whole Web page, according to their characteristics.

4. ***Frame-based adaptation***: These techniques adapt the presentation of frames or clusters of information about related concepts. Frames can be shown, hidden, alternatively presented or ordered. The frameset includes rules to decide which frames must be presented to a given user.

2.3.4.2 Adaptive Navigation Techniques

Adaptive navigation techniques are focused on adapting the paths that the user can follow into the application hyperspace. As Fig. 2.3 shows, they are grouped into the following six groups:

1. ***Adaptive Link Hiding***: these techniques restrict the navigation space by hiding the links to pages that are irrelevant or forbidden to a certain kind of users. The "hiding" may be implemented, for instance, by deactivating or not showing the irrelevant link(s).

2. ***Adaptive Link Ordering***: these techniques sort the links of a given page, according to the user characteristics and adopting some criteria to emphasize some links from the others, e.g., the closer to the top of the page the link is displayed, the more important the target information is.

3. ***Adaptive Link Annotation***: this kind of techniques allows complementing links with comments about the pages that are accessible through those links. These annotations can be textual (comments) or visual (icons, different colours or text fonts) and give information about the relevance of the target pages or their current state (visited, unavailable, etc.).

4. ***Direct Guidance***: the goal of these techniques is to suggest the next best node, that is to say, the most relevant page to be accessed immediately after the current one.

5. ***Adaptive Link Generation***: these techniques consist on the generation of new links, which have not been considered in the application authoring phase.

6. ***Map Adaptation***: It refers to the distinct ways of adapting the navigational maps, influencing the structure or topology of the map.

Adaptive Methods have been implemented in traditional hypermedia systems, with successful results. By means of different Adaptive Techniques, these methods provide valuable help to users in order to face the interaction problems described above. However, these concepts have been hardly incorporated by engineering methods for Web applications development. In the following section, the main Model-Driven approaches to Web Adaptivity are analyzed.

2.4 Adaptivity from a Model-Driven Perspective

The conceptual specification of Data-intensive Web applications is a complex task. Developers must consider multiple features to design highly interactive Web pages, which actively communicate with distributed data repositories and integrate heterogeneous pieces of content.

As stated in [37], *"an application doesn't actually determine why a situation is occurring, but the designer does. The designer uses incoming context to determine why a situation is occurring and uses this to encode some action in the application"*. According to this philosophy, methodological proposals for Web development have focused their efforts on providing highly expressive conceptual models, with the goal of systematically obtaining final implementations.

The wide spectrum of possible users of Web applications demands developers to provide software solutions that meet individual information needs and preferences. In particular, the introduction of adaptive features into Web applications requires to provide conceptual tools to model different implementation alternatives for different users, from a unique navigational structure. Following this trend and facing the increasing demand of personalized Web applications, Model-Driven proposals have augmented the expressiveness of their conceptual models to incorporate adaptive functionality to their resulting applications.

In [41], Schwinger and Koch proposed a categorization of fourteen approaches to model Web applications. From this group, Schwinger distinguishes those proposals that support the modelling of adaptive features [42]. Along with a preliminary version of the work of this thesis [43], this survey identifies the following approaches: *WebML*, *OO-HDM*, *UWE*, *WSDM*, *OO-H* and *Hera*. In this section, we present an analysis of these proposals. There are for sure more relevant works in the literature, but we think that with this set we cover the main aspects we want to introduce in the context of this PhD thesis.

2.4.1 WebML

WebML (Web Modelling Language [44]) is a visual language for specifying the content structure of Web applications and the organization and presentation of contents in one or more hypertexts [45]. Contents and their organization are specified through an Entity-Relationship [46] schema (called *Data Schema*), whereas the *WebML Hypertext Model* allows describing the way that these contents are published into the hypertext.

The navigational specification of a Web application is made by defining a set of views of the data schema, called *site views*. The main sections of a site view are *areas*, which are comprised of *subareas* and *pages*. *Pages* are the main containers of information and are composed of *content units*, which are the elementary pieces of information. Content units and pages are connected through *hyperlinks*. Operations are defined in site views.

Adaptivity Proposal

WebML proposal for Adaptivity [47, 48] incorporates a new model to describe navigational patterns, whose fulfilment determines the adaptation of the hypertext.

In this proposal, *Context* is described into the *Data Schema*, through the following three subschemas: (a) *User profile sub-schema*, which specifies personal characteristics of users, classifying them into *user groups*, each of them associated to a *site view*; (b) *Personalization sub-schema*, consisting of entities and relationships that interconnect users with application objects; and (c) *Context model sub-schema*, which comprises the context information that is not directly related to users, such as temporal, environmental and technological aspects.

Adaptive pages of a Web application are called *context-aware pages*. Each of them is provided with an external *context cloud*, which is the set of adaptivity actions associated to this page. At a broader granularity, context-aware areas and site-views may be also defined. Developed Web applications (so-called *User-Behavior-Aware Web Applications* [49]) perform adaptation according to the fulfilment of predefined navigational patterns, in reaction to link activations.

WebML is used to specify the application's hypertext structure, while a new model, WBM (Web Behaviour Model), provides a finite state automaton notation for describing possible Web navigation patterns. Adaptive features are specified through *Event-Condition-Action* rules [50], where *Events* are normally page requests; *Conditions* are WBM scripts of navigational patterns; and *Actions* are adaptation operations valid on a set of context-aware pages.

2.4.2 OO-HDM

The Object-Oriented Hypermedia Design Method (OO-HDM) [51] is a model-based approach for developing Web applications. It describes a Web Application at a high abstraction level through a *Conceptual*, a *Navigational* and an *Interface* Model. The Navigational description of the application is composed of Object-Oriented views of the conceptual model. As abstractions of actual Web pages, OO-HDM also defines *navigational contexts*, which are sets of objects that may be explored in a certain order.

Adaptivity Proposal

The Adaptivity proposal of OO-HDM [52] permits to specify adaptations of link topology and contents of nodes. Consideration of user characteristics is made from the Requirements Specification stage.

OO-HDM defines the Requirements of the Application through a *Scenario-Based approach* [53]. From the definition of *user roles*, *scenarios* that describe the user tasks are specified. Scenarios associated to the same task are grouped in one UML Use Case. For each Use Case, an *User Interaction Diagram (UID)* graphically represents the exchange of information between the user and the application in the corresponding task.

In the *Conceptual Model*, users are described in terms of *user characteristics*, *user roles* and *user groups*. Algorithms that implement distinct business rules can be assigned to different user groups at design-time.

Adaptive features are specified in the *Navigational Model*, by customizing the composition of information nodes, and the visibility of links and indexes that allow their access. The access to contents is adaptively provided through a function of user attributes. The adaptive behaviour of links and indexes is declared through *adaptive algorithms*. The access to a given context and its object instances can be limited to certain user groups.

OO-HDM introduces some *personalization patterns* [54] that are likely to be detected in adaptive hypermedia design, defining strategies to support this process.

2.4.3 UWE

The UML-based Web Engineering approach (UWE) [55] is conceived as an object-oriented methodology for the development of Adaptive Web applications. It makes use of the graphical representations, modelling techniques and semantic of UML and its language of constraints specification OCL [56].

The basis of the design process is the *Munich Reference Model* [30], an object-oriented model that incorporates user modeling aspects and rule-based adaptation mechanisms. It is an extension of the *Dexter Hypertext Reference Model* [57].

Adaptivity Proposal

Munich extends the functionality of the three layers from the Dexter Model, specially the *Storage Layer*. This layer is divided into three submodels: (a) the *Domain meta-model*, which manages the basic network structure of the system; (b) the *User meta-model*, which manages the set of users in base of their attributes; and (c) the *Adaptation meta-model*, which consists of a set of rules that implement the adaptivity functionality.

Classes of the reference model are the containers of three types of *operations*, which implement the adaptivity functionality: (a) *Authoring Operations* that update components, adaptation rules and user attributes; (b) *Retrieval Operations* that access the domain structure and the user model; and (c) *Adaptation Operations*, which allow automatically adapting the user model to the user behaviour and the presentation to the current state of the user model.

The UWE proposal describes the development process of AHS in base of a set of *workflows*. For each of them, it describes which experts must be involved, which activities they must perform and which specific artifacts they produce. Besides the development process itself, workflows are defined to process management and quality management processes. In this way, the whole process of development are specified, from requirements capture to the maintenance of the system [58].

2.4.4 WSDM

The Web Site Design Method (WSDM) [59] is a development method for Web applications, which follows an audience-driven design approach. This means that the different target audiences (visitors) and their requirements are taken as starting point for the design, from which the main structure of the web site is derived. Concretely, this results in different navigation paths (called audience tracks) offered from the homepage, one for each different kind of visitor.

The method comprises four main phases [60]: (a) *Mission Statement Specification*, where the purpose of the web system, as well as the subject and the target users are described in natural language; (b) *Audience Modelling*, where target users are refined and grouped into so-called *audience classes*. These classes are organized in a hierarchical structure. For each audience class, its relevant characteristics (e.g., age, experience level) are described; (c) *Conceptual Design*, where the information, functionality and structure of the web system are specified at conceptual level, through corresponding subphases *Task and Information Modeling* and *Navigational Design* sub phases; and (d) *Implementation Design* phase, where the conceptual design models are complemented with information required for an actual implementation.

Adaptivity Proposal

In the adaptivity proposal of WSDM [61], adaptive features are defined in terms of four types of basic model transformation, described through a set of formal operations that permit to: (1) add and delete nodes from the *Navigation Model*; (2) connect and disconnect object chunks to or from a node in the *Navigation Model*; (3) add or remove links of the *Navigation Model*; and (4) add or remove nodes from implementation descriptions.

To specify adaptation strategies, this proposal introduces a high level language, called *Adaptation Specification Language (ASL)*. This language is based on ECA (*Event - Condition - Action*) rules [50]. Events correspond to navigational actions performed by users during their browsing sessions. Conditions are specified on stored information, by using arithmetic expressions.

Actions are specified using the described transformations.

2.4.5 OO-H

OO-H Method [62] is a generic model that provides conceptual tools to develop web-based interfaces and connect them with previously existing modules of the application logic. OO-H method extends the automatic software generation method OO-Method [20], through the definition of a *Navigational Access Diagram* (NAD) and an *Abstract Presentation Diagram* (APD), which describes the navigational and presentational aspects of the application, respectively. By compiling these diagrams, the application front-end may be automatically generated. As views of an OO-Method (UML-compliant) class diagram, different NADs are defined, in order to fulfil the navigational requirements of each user type. By following a set of mapping rules, a default web interface is generated. This is the basis of the final interface specification, which is made through the definition of a template-based APD.

Adaptivity Proposal

The approach of OO-H to adaptation of Web sites [63] describes application users by defining *user profiles*, which are sets of relevant user data classified into three categories: (a) *user characteristics*, which comprise the personal, domain-independent user characteristics; (b) *user requirements*, which describe the information and functionality that users expect to find when interacting with the system; and (c) *user context*, which contains characteristics of the browsing device, spatial and temporal data of the interaction and technological constraints.

Along with the user description, OO-H also defines a *Personalization Framework*, which can be connected to OO-H based applications to provide it with personalization capabilities. This framework comprises: (a) a *User Model*, which is a class diagram that describes the user information specified in the previous step and data about the user behaviour; and (b) a *Personalization Model*, which is a set of *Event - Condition - Action* rules [50] that define the strategy to acquire data about a user, classify her into a given profile and perform the required adaptation.

In a browsing session, a user is classified into one of the defined user profiles; her interaction is driven by the rules attached to her profile, which are triggered by user events (link activations, operation executions). The specifications of these rules are made in a high-level, textual language called PRML (*Personalization Rules Modeling Language* [64]), and they can eventually be mapped to distinct Web development environments.

2.4.6 Hera

Hera is a model-driven methodology that supports the design and engineering of Web Information Systems [65]. It proposes a system architecture composed of three layers: (a) *Semantic Layer*, which describes the domain data through a *conceptual model*, and defines the process of integration of data from different sources through an *integration model*; (b) *Application Layer*, which defines the navigation view of the data in terms of an *application model*, along with the adaptation of the hypermedia generation to a user model; and (c) *Presentation Layer*, which describes the presentation details of the generated hypermedia presentations through a *presentation model.*

The generation of a hypermedia presentation comprises two main phases: (a) *Integration and Data Retrieval*, which allows making the data available from different sources. In response to a *user query*, a *conceptual model instance* is generated, containing the data to be presented to the user; and (b) *Presentation Generation*, where these data are firstly transformed into an *application model instance*, considering adaptation characteristics, and finally transformed into a presentation suited to the user's platform.

Adaptivity Proposal

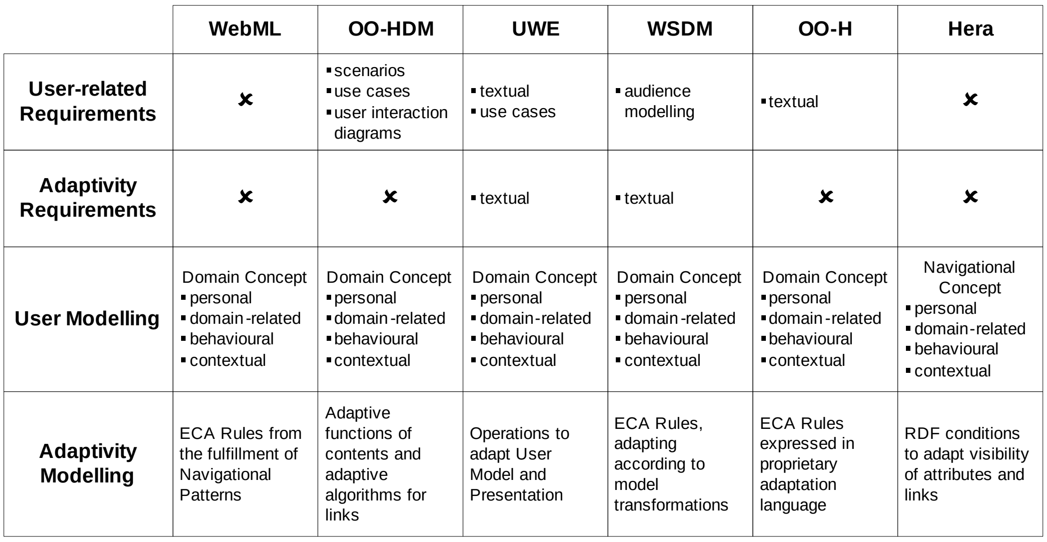
Unlike other model-driven approaches to Adaptivity, Hera specifies the context (*User Model*) into the navigational schema, instead of the domain description. According to the authors [66], this decision has been made by considering that the perceived state of the user influences the navigation over the data content and typically this does not belong to the content.

The *application model* is composed of *slices*, which are data presentation units associated to a domain concept, joined through *navigational relationships*. The adaptation of this model is described through *conditions* defined in terms of features of the user model. These conditions rule the visibility of attributes and relationships of the slices [67]. In analogous way, adaptivity conditions are defined over conceptual and presentational models.

All models in Hera and their adaptivity conditions are specified in RDF Schema [68], for interoperability purposes. Hera also supports reuse of its models, mainly through extensions of existing models [69].

2.5 Analysis of the State of the Art

The table of the following figure shows a summary of the most relevant characteristics of each of the reviewed proposals, concerning to the development of Adaptive Web Applications.



	WebML	**OO-HDM**	**UWE**	**WSDM**	**OO-H**	**Hera**
User-related Requirements	✗	▪ scenarios ▪ use cases ▪ user interaction diagrams	▪ textual ▪ use cases	▪ audience modelling	▪ textual	✗
Adaptivity Requirements	✗	✗	▪ textual	▪ textual	✗	✗
User Modelling	Domain Concept ▪ personal ▪ domain-related ▪ behavioural ▪ contextual	Domain Concept ▪ personal ▪ domain-related ▪ behavioural ▪ contextual	Domain Concept ▪ personal ▪ domain-related ▪ behavioural ▪ contextual	Domain Concept ▪ personal ▪ domain-related ▪ behavioural ▪ contextual	Domain Concept ▪ personal ▪ domain-related ▪ behavioural ▪ contextual	Navigational Concept ▪ personal ▪ domain-related ▪ behavioural ▪ contextual
Adaptivity Modelling	ECA Rules from the fulfillment of Navigational Patterns	Adaptive functions of contents and adaptive algorithms for links	Operations to adapt User Model and Presentation	ECA Rules, adapting according to model transformations	ECA Rules expressed in proprietary adaptation language	RDF conditions to adapt visibility of attributes and links

Figure 2.4: Support of Model-Driven proposals to the development of Adaptive Web Applications

In the next subsections, we present an analysis of these characteristics, focused on the main contributions of each proposal and their detected deficiencies.

2.5.1 User-related and Adaptivity Requirements

As the table of Fig. 2.4 shows, the characteristics that less attention have received from the Model-Driven approaches are the specification of User-related and Adaptivity Requirements.

WSDM provides the only example of an actual concern about the specification of User-related Requirements, by introducing the concept of *Audience*: a set of users with similar characteristics. Different audiences are organized in a hierarchical structure. WebML and Hera proposals do not give details about the way that User Requirements are specified. In the rest of the proposals, they are mainly described by means of textual specifications, with the exception of UWE and OO-HDM, which allow making a detailed description of tasks that distinct users perform, through the definition of UML Use Cases Diagrams. In the case of OO-HDM, the information exchange between user and application for each task may also be specified through the above mentioned UIDs.

As the comparative table shows, efforts of the Web Engineering community have been mainly focused on augmenting the expressiveness of their conceptual models to support Adaptive Web Applications. In most of the proposals, User-related and Adaptivity Requirements are informally described, and usually considered directly in the conceptual modelling phase. This scenario makes difficult to trace those requirements along the development process and hence visualize whether the resulting application will satisfy them. Described proposals fail in providing a systematic way to derive the conceptual descriptions of the adaptivity from the specified requirements. The proper enhancement of the UID definition in OO-HDM, by including adaptive requirements, would be an important contribution in the solution of this deficiency.

2.5.2 User Modelling

In general, the conceptual modelling of the user characteristics has been well covered by the reviewed proposals, describing users in terms of the following sets of characteristics:

- *Personal Characteristics*, which refer to those features that are intrinsic to the user and do not depend on specific application domains.
- *Domain-Related Characteristics*, which are those user characteristics that are related to concepts of the application domain.
- *Behavioural Characteristics*, which describe the navigational actions (link activations, operation executions) that user performs when interacting with the application.
- *Contextual Characteristics*, which describe the rest of the context-of-use's characteristics that are not directly related to the user, such as spatiotemporal information and technological limitations.

Most of the proposals model these characteristics through a *User Model*, which is incorporated as part of the domain specification. As an exception, Hera defines user characteristics in the navigational modelling stage.

The consideration of the Contextual characteristics evidences the attempts to provide a conceptual solution to a broader definition of adaptivity, by considering characteristics of the browsing devices and the environment where the interaction occurs. However, the adaptation to characteristics that are directly related to the user (personalization) is a challenging problem to which Web Engineering community has not yet provided a robust proposal.

2.5.3 Adaptivity Modelling

All the reviewed approaches describe the Adaptivity in their respective Navigational Models. In order to describe adaptive characteristics, the definition of *adaptive rules* is the most used strategy, and they are mainly based on

Event - Condition - Action (ECA) paradigm. Events correspond to the access to a given page or the fulfilment of predefined navigational patterns (WebML, OO-H). When the *Event* is produced, the system checks whether the current user fulfills the *Condition* of the rule: if she does, then the corresponding adaptive Action is performed. Other mechanisms to specify adaptivity follow a similar strategy.

Adaptive actions mainly restrict the access to contents and links, expressed as navigational views of attributes and associations, respectively. Furthermore, the adaptive access to an entire Web page may also be defined. None of the proposals, however, provide conceptual mechanisms to express well-known Adaptive Techniques at a high abstraction level. This could be a valuable help in order to define strategies to incorporate Adaptive Methods into the modelling process. Along with reducing of modelling efforts, this possibility may help to obtain adaptive descriptions that are closer to the adaptivity requirements, more intuitive for the developer and with a more direct mapping to the final implementation.

2.6 Conclusions

The rapid growing of the Web in size and complexity, along with the evolution of its users, stresses the need to develop Adaptive Web Applications. However, the complexity of these systems demands the adoption of a systematic development method.

Adaptive Hypermedia community has defined and implemented a set of adaptive methods through different adaptive techniques. Although Web applications have a wider and more diverse set of target users, the consideration of these concepts from a high abstraction perspective may help to reduce this complexity.

Model-Driven proposals to develop Adaptive Web Applications present many similarities in the modelling of users and adaptive features, providing a valuable basis to the achievement of a robust development method to this kind of systems. However, little evidences of their use in real software projects give the impression that something is still to be done to have a clear, precise framework to specify correctly and completely Adaptive properties of Web Applications.

Furthermore, in terms of Model Transformations Technologies [70] to go from requirements to final software products, these approaches do not provide a common, whole, unified working software development environment where the Adaptive Web characteristics are clearly embedded in the expressiveness of each level, and are adequately transformed into the corresponding software artefact of the next level.

Our goal is to demonstrate that Adaptive Hypermedia concepts and Model-Driven approaches can (and must) be integrated, supporting a full software process, from requirements to conceptual models and to the final web applications, where Adaptivity is managed according to its importance. Being OOWS the Web extension of an MDA-based software production process (OO-Method) and having that work being done with a precise formal support, the final contribution of this thesis is to prove that all these ideas and experience can work well for managing and solving the problems associated with the Adaptive Web challenge, well-fitted with the more advanced ideas on web application generation from models.

Chapter 3

The OOWS Modelling Process for Adaptive Web Applications

3.1 Introduction

In this chapter, we introduce a Model-Driven development method for Adaptive Web Applications, based on the specification of Adaptive Hypermedia concepts at early stages of the development cycle.

In the previous chapter, the main concepts involved in the development of Adaptive Hypermedia Systems have been introduced. Furthermore, an analysis of Model-Driven proposals for the development of Adaptive Web Applications has been presented. In this chapter, concepts from both fields are integrated. The OOWS development process of Web Applications is extended with conceptual tools that describe Adaptive Methods and Techniques in the Problem Specification stage, abstracting implementation concerns.

The first section of this chapter presents the current OOWS proposal for the development of non-adaptive Web applications; the second section describes the proposal of this thesis, by introducing new conceptual diagrams and primitives to specify and model Adaptive Web Applications in OOWS; finally, the last section describes the characteristics of the Adaptive Web Applications that are supported by this proposal.

3.2 OOWS Modelling Process for Non-Adaptive Web Applications

OOWS (Object Oriented Web Solutions) [19] is an extension of the object-oriented software production method called *OO-Method* [20, 21], which introduces the required expressivity to capture the navigational and presentation requirements of Web applications.

OO-Method defines a development process that integrates conventional object-oriented modelling notations with formal specification techniques, in order to obtain a final implementation by following a model-based code generation strategy. This development process comprises two major stages: *Problem Specification* and *Solution Development*.

In the ***Problem Specification*** stage, the problem that is faced by the software project is completely described. Requirements from different aspects of the application are specified and modelled, by means of object-oriented conceptual models. Following the Model-Driven development philosophy, most of the efforts of the development process are focused on this stage.

The ***Solution Development*** stage proposes a code generation strategy, in order to integrate the proposed solution in Web environments. In this stage, a Web application is obtained, providing a functionality that is equivalent to the specifications built in the Problem Specification. In a completely automatic way through the use of a *model compiler*, a software system is build by applying architectural patterns for different execution platforms.

Figure 3.1 illustrates the development process of OOWS, comprising the two described phases. As shown in the upper part of the figure, the original OO-Method proposal considers two phases in the Problem Specification stage: *Requirements Specification* and *Conceptual Modelling*. OOWS incorporates two models to specify the navigational and presentation aspects of the application. In the Solution Development stage, conceptual specifications are transformed into implementation components, which conform the three typical tiers of a Web application. Conceptual schemas from OO-Method allow generating the Persistence and Business Logic tiers, while OOWS schemas are transformed into Web interface components.

The Problem Specification stage of OOWS is the main focus of attention of the present work. A precise and complete specification of adaptive characteristics at early stages allows abstracting the high complexity of their implementation. In the next subsections, we describe the current OOWS Problem Specification stage.

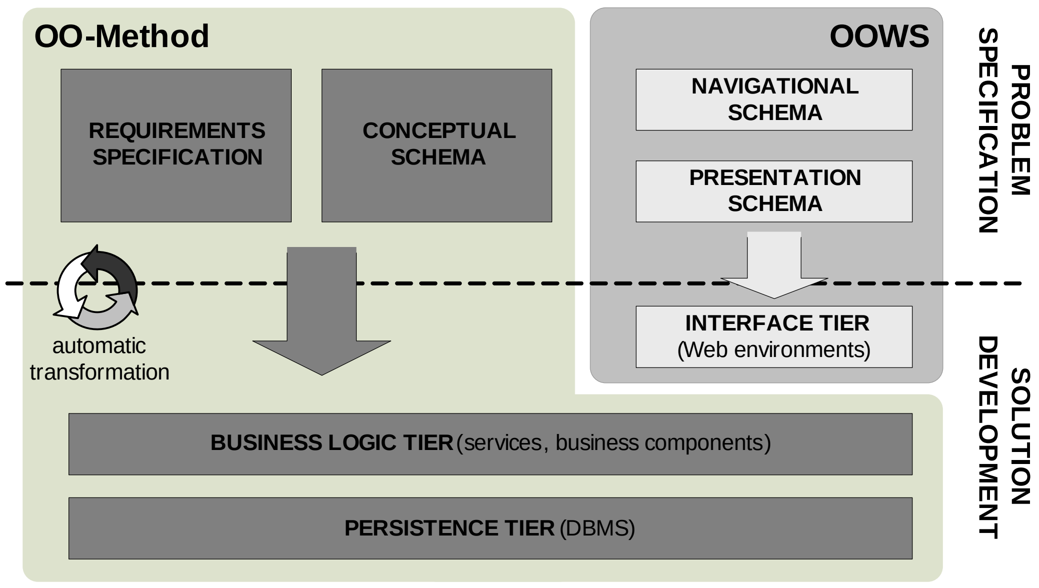


Figure 3.1: OOWS (extended OO-Method) development process

3.2.1 Problem Specification in OOWS

In the *Problem Specification* stage, requirements of Web applications are specified and modelled at a high abstraction level, by means of a set of highly expressive conceptual models. This permits to concentrate efforts on detailing the problems whose solution the system must support, leaving the consideration of implementation aspects after the complete definition of the system.

As mentioned above, the Problem Specification is made through two main phases: *Requirements Specification* and *Conceptual Modelling*. Figure 3.2 shows a detailed schema of this stage.

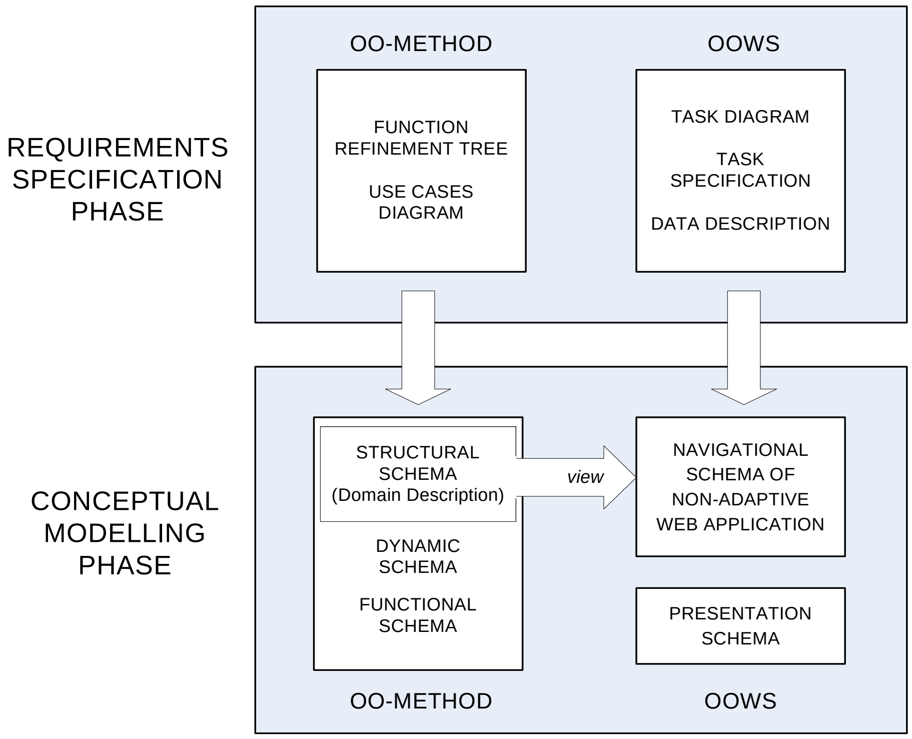


Figure 3.2: Current OOWS Problem Specification stage

In the ***Requirements Specification*** phase, we must distinguish two complementary approaches: (a) the specification of functional requirements, through different proposals of OO-Method; and (b) the specification of navigational requirements, through the extensions provided by OOWS for Web applications.

OO-Method proposes different strategies to model the requirements of a system: classification and specification of functions [71]; business process modelling; natural language processing of Use Cases [72]; organization modelling [73]. All these strategies share the use of a ***Function Refinement Tree (FRT)*** [74], where all the relevant system functions are hierarchically structured. It acts as an ordered, high-level representation of the system functionality. As this concept appears in all the requirements-oriented proposals of OO-Method, we will use it as the functional connection with the solution that results from extending OO-Method for the Web at the requirements level. As a complement of the FRT, we also consider the Use Case Diagrams, which describe the main user-system interactions involved in the fulfilment of each elementary task.

In order to describe the navigational requirements of a Web application, OOWS proposes the use of three diagrams:

1. ***Task Diagram***, which is a hierarchical tree of the navigational tasks that users may perform during a browsing session. It also includes the specification of temporal dependencies between pairs of tasks.
2. ***Task Specification***, which is a detailed description of the navigational steps that a user must follow in order to complete each elementary task from those included in the previous diagram. This description is made through a UML-like *Activity Diagram*.
3. ***Data Description***, which specifies the data items that can be accessed by the users in each navigational step.

The second phase of the *Problem Specification* stage is the ***Conceptual Modelling*** of the application, where the requirements of the system are modelled from different perspectives. The OO-Method *Conceptual Schema* allows specifying the structural and behavioural aspects of traditional applications by means of three models. These models are the following:

1. ***Structural Model***, which defines the structure of the system, in terms of classes, operations and attributes, along with the relationships between classes (specialization, association, and aggregation), by means of a UML-like *Class Diagram*;
2. ***Dynamic Model***, which describes the valid object-life sequences for each class of the system, by using *State Transition Diagrams.* Object interactions (communications between objects) are also represented by UML-like *Sequence Diagrams*; and

3. ***Functional Model***, which captures the semantics of state changes to define service effects, by using a textual formal specification.

OOWS extends the OO-Method modelling proposal by introducing two new models, which describe the navigational and presentation aspects of Web applications, respectively. These models are:

1. ***Navigational Model***, which captures the navigational semantics of web applications. This model permits to define the navigational paths that users are allowed to follow inside the application hyperspace, along with the inner structure of each of its nodes (pages), in terms of their contents and hyperlinks.

2. ***Presentation Model***, which captures the presentation properties that rule that way in which the information and functionality are shown in the final Web application.

3.2.2 The OOWS Navigational Model

The OOWS Navigational Model is defined as a view of the Structural Model of the application. Once the structural schema is obtained, comprising the classes, associations, attributes and operations that describe the application domain, a Navigational Schema is defined. At first, a set of *Navigational Maps* shows the global navigational structure of the application, in terms of nodes and links. Furthermore, a detailed description of each node (*Navigational Context*, in OOWS) is also made. Each node includes views of the different structures of the Class Diagram, describing the data, services and hyperlinks that are available in each page of the application.

All the conceptual descriptions made through this model lead to the same implementation, with no regard to the user that is accessing the system. The only consideration of user characteristics is made in the definition of navigational maps, where different groups of users have access to different navigational possibilities. However, this consideration is completely made at design-time, and more specific characteristics of users are totally ignored. Therefore, the applications developed through OOWS are non-adaptive.

3.3 Presentation of the Proposal

In order to support the specification of Adaptive Web Applications, the present thesis proposes to enhance the Problem Specification stage of OOWS method. This is made through the following three steps:

1. Definition of conceptual models that support the specification of Adaptive Web Applications:
    - a set of Adaptive Primitives in the OOWS Navigational Model to describe well-known Adaptive Techniques;
    - a User Model to specify the needed characteristics of application users and their context; and
    - a set of modelling strategies to support the specification of consolidated Adaptive Methods.
2. Definition of a Requirements Model that supports the specification of Adaptivity and User-related requirements;
3. Definition of a set of rules to model the specified adaptivity and user-related requirements through the enhanced OOWS Conceptual Model.

Figure 3.3 illustrates the Problem Specification stage that this work proposes. The three introduced steps are illustrated in grey squares, and numbered according to the list presented above. In the following subsections, we describe each of these steps in detail.

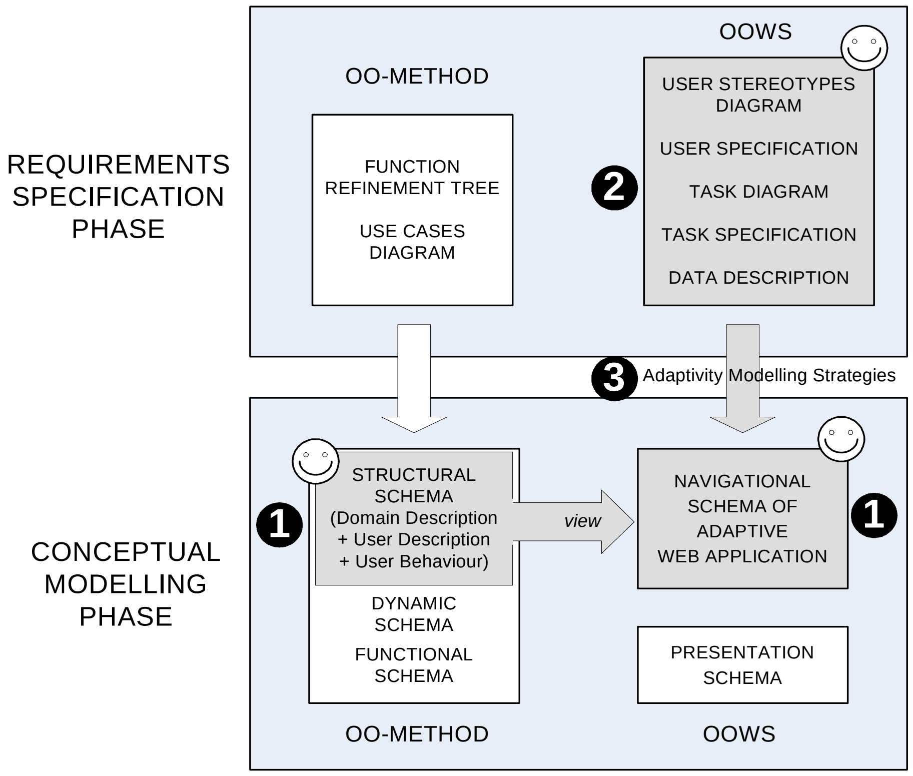


Figure 3.3: Proposed enhancements to OOWS Problem Specification stage

3.3.1 Conceptual Modelling Approach for Adaptive Web Applications

This step is the center of interest of the present work. The Structural Model (Class Diagram) and the OOWS Navigational Model are extended to support the modelling of user characteristics and adaptive functionality, respectively. The main contributions of this step are:

1. A ***User Model***, which supports the description of the users of the application and their context. These characteristics are the basis on which the adaptive features are defined. As an enhancement of the OO-Method *Class Diagram*, this model adds two views to the traditional domain description: a *User Description View*, which models the characteristics that describe the user as an individual; and a *User Behaviour View*, which characterizes the actions that users perform in the application during a browsing session. Along with these characteristics, operations that are needed to keep the user description permanently updated are also included. In this way, the provided adaptivity always matches the current state of the user.

   The consideration of users and their context as part of the application domain adheres to the strategy for user modelling that is adopted by the majority of the Model-Driven approaches to Web Adaptivity.

2. Introduction of ***Adaptive Primitives*** into the *OOWS Navigational Model*. Adaptive Hypermedia community has defined, classified and implemented [10, 8] a set of Adaptive Techniques. However, they have been not explicitly considered by most of the Model-Driven proposals of Adaptive Web Applications. The present work introduces conceptual primitives that give support to the specification of Adaptive Navigation and Presentation techniques at a high abstraction level. This enhancement allows modelling different options of presentation of contents and links through a unique navigational schema, by adopting techniques that have been successfully incorporated in traditional hypermedia systems. In this way, the visibility and the presentation of navigational attributes, operations and hyperlinks vary from one user to another, depending on their particular characteristics.

To describe the adaptive features of the application, an *Event - Condition - Action (ECA)* approach [50] is adopted. Each adaptive primitive has an *adaptivity constraint*, which determines its final implementation for a given user. Adaptivity constraints are ECA-like rules, where the *Event* is the access to the page that contains the constrained primitive by a given user; *Condition* corresponds to requisites imposed to the user, defined in terms of domain concepts and structures from the User Model; and the *Action* is the assignment of one of the possible variants of the primitive to the user, according to her fulfilment of the condition.

The proposed definition of Adaptive Primitives is compliant with the broad use of *Event - Condition - Action (ECA)* rules to define adaptive features (see Section 2.5.3 of Chapter 2, *State of the Art*). However, as an innovation in the Model-Driven development of Adaptive Web Applications, this definition explicitly supports the adaptive techniques proposed by the Adaptive Hypermedia community.

3. Definition of strategies to model ***Adaptive Methods***. Adaptive Methods [10, 8] describe different strategies to support the browsing experience of the user from an adaptive perspective. Each of these methods can be implemented through one or more Adaptive Techniques. From the introduced Adaptive Primitives, the present work introduces a set of modelling strategies to incorporate Adaptive Methods in OOWS.

   Different alternatives to model each of the considered adaptive methods are introduced, taking into account the adaptive techniques that are typically used to implement it. These modelling strategies are defined in terms of the adaptive primitive corresponding to those techniques.

   The resulting set of modelling strategies is intended to provide developers with guidelines to incorporate Adaptive Methods in Web applications at early stages of the development process. In this way, we expect to augment the acceptance of Model-Driven proposals in the development of this kind of systems, providing not only “yet another conceptual model”, but also the best way to use their primitives to specify well-proven methods from a high abstraction level.

3.3.2 Requirements Specification for Adaptive Web Applications

To support the modelling decisions that can be made from the use of the new Adaptive Primitives, this work extends the current OOWS Requirements Model with tools to specify the requirements that are specifically related to the development of Adaptive Web Applications. This phase proposes an alternative to one of the main deficiencies detected in Model-Driven proposals to Web Adaptivity: the lack of conceptual tools to specify adaptivity requirements in a systematic way. For this, two mechanisms to specify the required characterization of users are introduced, while the existing diagrams of OOWS to specify navigational requirements [24] are enhanced:

1. ***User Stereotypes Diagram***, which permits to characterize the application users in terms of their similar characteristics, classifying them into groups of users or *stereotypes* [23], which are hierarchically organized. Similar approach to characterize the user of the application is adopted by the WSDM proposal in [61].

2. ***User Specification***, which details the characteristics that differentiate users from each stereotype in a tabular description.

3. ***Enhancement of Task Diagram***, by identifying those elementary tasks whose fulfilment involves the incorporation of adaptive features. These tasks are called *adaptive tasks*.

4. ***Enhancement of Task Specification***, by introducing new structures to the Activity Diagrams that characterize the navigational requirements of each adaptive task. These structures permit to specify the required adaptive constraints that rule the access to navigational nodes, operations and links by different users.

5. ***Enhancement of Data Description***, by introducing the required constraints that limit the access to the contents included in each node.

3.3.3 Transforming Adaptivity Requirements into Adaptivity Models

In order to support the traceability of Adaptivity Requirements along the rest of the development cycle, a set of traceability rules is defined. These rules describe the way in which User-related and Adaptivity Requirements can be modelled through the introduced User Model and the enhanced Navigational Model. The two steps of this phase cover the corresponding two mapping processes:

1. Definition of transformation rules from ***User-related requirements to User Model***, by determining the correspondence between the classes, attributes, operations and associations that conform the User Model, and the primitives of the Diagram of User Stereotypes and User Specification descriptions.
2. Definition of transformation rules from ***Adaptivity Requirements to the OOWS Navigational Model***, focused on the modelling of the inner structure of navigational nodes, along with the corresponding adaptive primitives, from the specification of the adaptive tasks that the system must provide.

These phases are illustrated through the analysis of a case study, corresponding to an adaptive on-line bookstore (like Amazon.com).

3.4 Scope of the Proposal

Adaptive Web Applications that this development proposal supports have the following characteristics:

- *Provided adaptation is mainly based on user characteristics.*

  Adaptive features of hypermedia systems are based on a permanently updated description of the context in which the user-system interaction occurs. This description may include personal characteristics of the user, specifications of her navigational actions during a browsing session, and environmental and technological features.

The present work adopts a definition of context that considers characteristics that are directly related to the particular users of the application, i.e., their personal characteristics and navigational behaviour. Although recognizing the importance of spatiotemporal and technological features in a wide domain of Web applications, we think that the adaptation based exclusively on user characteristics proposes numerous challenges to the Model-Driven community. These challenges are only partially addressed by existing Web development proposals.

Considering this notion of context, this work is focused on the development of *User-Based Adaptive Systems* in the World Wide Web. However, the modelling of the context of use in the domain specification (Class Diagram) permits to adopt a wider concept of context with little additional considerations. Therefore, the generic term of *Adaptive Web Applications* is adopted to refer the applications whose development this work tackles. In future works, environmental and technological features may be increasingly considered to provide richer adaptation features.

- *Adaptation of the Navigation of Adaptive Web Applications*

  This work is focused on enhancing the conceptual specifications of the navigational aspects of Adaptive Web Applications. This decision is motivated by considering the big impact that adaptations of the access to contents and hyperlinks have on the browsing experience of users and on the way they understand the visited contents. Along with this, the decision is supported by the high expressivity and flexibility of the navigational model on which this work is based, which allows adopting a Model-Driven approach to the development of this kind of applications, by enhancing previously defined conceptual structures with adaptive properties, or by defining new primitives that are compliant with the existing ones.

- *Adaptation of Data-Intensive Web Applications*

  In spite of the possibility to extend the scope of this proposal to a broader domain of Adaptive Web applications, this thesis is specially focused on the so-called *Data-intensive Web applications*, in which users have to deal with a great amount of information. The adaptation proposed is focused on managing the access to that information and the functionality that system provides, not in graphical or multimedia aspects. Due to this, graphic-intensive Web applications are out of our immediate interest.

3.5 Conclusions

This chapter has presented an overview of the proposal of this thesis. Problem Specification stage of OOWS is enhanced with conceptual tools to support the specification of Adaptive Web Applications. Conceptual Modelling phase is extended: conceptual primitives are added to the OOWS Navigational Model, in order to specify the adaptive access and presentation of contents and links; furthermore, OO-Method Class Diagram is also enhanced, by incorporating two views that describe the characteristics of the application users and their navigational behaviour.

These extensions integrate knowledge from the two main research fields that have tackled the development of Adaptive Web Applications: Adaptive Techniques and Methods defined by Adaptive Hypermedia community have been incorporated by adopting strategies traditionally used by Model-Driven proposals, like the definition of Event - Condition - Action rules and the incorporation of the user as part of the application domain.

Aimed at covering one of the main deficiencies of similar previous works, this work also extends the OOWS Requirements Model, in order to support the specification of the new kind of requirements related to Adaptive Applications. Traceability of these requirements are supported by the definition of traceability rules.

In the next chapters of this dissertation, every part of this proposal is detailed. Firstly, the new Adaptive Primitives are introduced, along with the User Modelling approach that supports their definition. Afterwards, the set of traceability rules to specify Adaptive Methods is presented, providing guidelines about how the introduced primitives must be used in order to fulfil typical adaptivity goals. Next, the Requirements Specification proposals for

Adaptive Web Applications is introduced, with the goal of supporting the modelling decisions that can be made through the new Adaptive Primitives. Finally, a set of traceability rules is presented, to support the conceptual modelling of the specified requirements.

Chapter 4

Modelling Adaptive Web Applications in OOWS

4.1 Introduction

The present chapter describes the enhancements proposed to the Conceptual Modelling phase of OOWS Method, in order to support the development of Adaptive Web Applications. These enhancements are:

1. a *User Modelling* proposal, which allows describing the characteristics of the users of the application, on which the adaptive features are defined.
2. a set of *Adaptive Primitives*, which permit to model well-known adaptive techniques in the OOWS Navigational Schema of a Web application.

In Adaptive Systems, characteristics of users and their context are as important as concepts from the application domain. For this reason, this proposal integrates the modelling of users into the Structural Schema (OO-Method Class Diagram) of the application. As an OOWS Navigational Schema is defined from views of the Class Diagram, this approach permits to incorporate the needed user characteristics into the navigational modelling of adaptive features, through the use of Adaptive Primitives.

Adaptive Primitives are conceptual primitives that allow specifying different Adaptive Techniques at a high abstraction level. In this proposal, they are incorporated into the OOWS Navigational Model, which allows describing the global structure of the application hyperspace, along with the data, operations and hyperlinks that are available in each Web page. As Adaptive Techniques permit to adapt the way that contents and hyperlinks are accessed by different users, their correspondent high level descriptions must be incorporated into this Navigational Model.

The rest of this chapter is structured as follows:

- Section 4.2 presents a review of the current OOWS Navigational Model, which is the basis for the specification of adaptive features. The primitives of this model are briefly described, in order to facilitate the understanding of the Adaptive Primitives. For a detailed description of this model, the reader is referred to the Appendix A of this dissertation.
- Section 4.3 presents the User Modelling approach of this proposal. Relevant characteristics of users and their context are modelled into the OO-Method Class Diagram, by means of two views that describe personal characteristics of users and their navigational actions during different browsing sessions, respectively.
- Section 4.4 presents the complete OOWS Navigational Metamodel, which describes the already defined primitives of this model and the new Adaptive Primitives that are introduced in this proposal.
- Section 4.5 introduces the set of Adaptive Primitives that are incorporated into the OOWS Navigational Model. For each primitive, we describe the adaptive goal that supports, its relationship with the already defined OOWS primitives, the supported Adaptive Technique(s), its specification in the OOWS Navigational Metamodel complemented with an OCL description, and an abstract example of its use in an OOWS Navigational Schema.
- Finally, Section 4.6 presents some conclusions of this chapter.

4.2 The OOWS Navigational Model for Non-Adaptive Web Applications

OOWS (Object Oriented Web Solutions) method [19] is an extension of the OO-Method proposal for automatic code generation [20, 21], which supports the development of Data-intensive Web Applications. In this section, we provide a brief review of the OOWS Navigational Model, which is the basis of the proposal of this PhD thesis. A detailed description of this model is provided in *Appendix A* of this dissertation.

The ***OOWS Navigational Model*** provides developers with conceptual primitives to model the navigational characteristics of a Web application. This is made from two perspectives: (a) the modelling of the global structure of the application, by describing the navigational topology of the application in terms of its nodes and links; and (b) the modelling of the inner navigational structure, in terms of the data, operations and hyperlinks that can be accessed in each of the defined nodes.

To model the global structure of the application, a set of ***Navigational Maps*** is defined. Each map corresponds to a global view of the application, which is provided to a given group of users. It is represented by a directed graph, whose nodes are called ***Navigational Contexts*** and its arcs are ***Navigational Links***.

Navigational Contexts represent interaction points that provide users with access to data and operations. There are two types of navigational contexts: (a) ***Exploration Contexts***, which are reachable from any node; and (b) ***Sequence Contexts***, which are reachable only via some predefined navigational paths. These two types of contexts are reachable through respective types of Navigational Links, which define the valid navigation paths of the application: ***Exploration Links*** and ***Sequence Links***.

Figure 4.1 shows the graphical notation of an OOWS Navigational Map, corresponding to a `Client` user type of an online bookstore. In the example, users that belong to the `Client` group can access five *Exploration Contexts* (depicted with an "E" label). Users can access these contexts from any page of the application at any moment of their browsing session. This access is

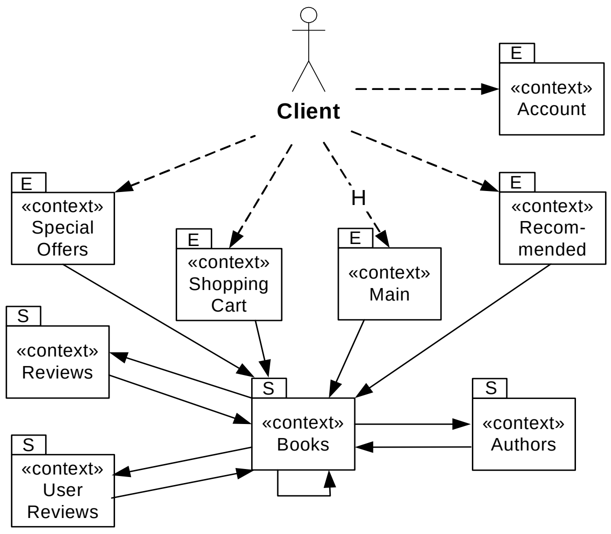


Figure 4.1: Example of an OOWS Navigational Map

modelled through corresponding *Exploration Links* (dashed arrows). One of these links has been marked with an "H", which denotes that the `Main` context represents the page from which users start browsing the application (i.e., the *Home Page*). `Client` users can also access to four *Sequential Contexts* (depicted with "S"), but only following the predefined paths indicated by the included *Sequential Links* (solid arrows).

Navigational Contexts aggregates one or more navigational views of the Class Diagram. These views are called ***Abstract Information Units (AIU)***. Each *AIU* is composed of one or more ***Navigational Classes***, which are related through ***Navigational Relationships***. Both structures are navigational views of classes and associations from the Class Diagram of the application, respectively. The navigational class that provides the main information of the AIU is called ***manager class***, while those that provide complementary information are called ***complementary classes***. *Navigational Relationships* are also classified into two types: (a) ***Context Dependency Relationship***, which represents a basic information recovery; and (b) ***Context Relationship***, which represents an information recovery along with a navigation to a target context.

When a context relationship is crossed, the information of the selected object is carried to a target context. According to their dependence of this information, OOWS distinguishes two types of AIUs: (a) ***Contextual AIU*** which instantiates its manager class to the object received as contextual information; and (b) ***Non-Contextual AIU*** (labelled with a circled "NC"), which does not depend on the contextual information received by its containing context. Furthermore, AIUs can be classified according to the number of instances that can be retrieved, into the following two types: (a) ***Single-Object AIU***, which allows retrieving information of only one object from the manager class; and (b) ***List AIU***, which allows retrieving information of several objects.

Finally, two structures can be defined in any navigational class: ***Population Filters***, which constrains the objects that are shown to users; and ***Ordering Pattern*** to define an order of presentation of the retrieved objects. Both filter constraints and ordering criteria are expressed through OCL [56] expressions.

Figure 4.2 shows the inner structure of the `BOOKS` *Navigational Context*, previously shown in the Navigational Map of Fig. 4.1. This context represents a Web page that shows detailed information of a book from the online bookstore, along with a list of books that have been recently included in the bookstore stock. Each of these sections is modelled through an AIU, respectively:

1. `BOOK DETAILS` AIU: The `Book` *manager class* shows basic data of a book, along with two operations. Six *complementary classes* are also included, related through Context and Context-Dependency relationships (complete and dashed arrows, respectively). This AIU is *contextual* (what is represented through the "C" tag), receiving the selected `Book` instance as the contextual information. Furthermore, it is a *Single-Object* AIU (represented through the ≪AIU≫ label), retrieving information of one `Book` instance only; and

2. `NEW RELEASES` AIU: a *Population Filter* (Fig. 4.1, bottom of the class) constrains the retrieved `Book` instances to those whose `date` attribute value belongs to the last week. An *Ordering Pattern* (Fig. 4.1, connected to the top of the class) establishes that these instances are ordered by their `title` attribute, in ascendant order. This AIU is

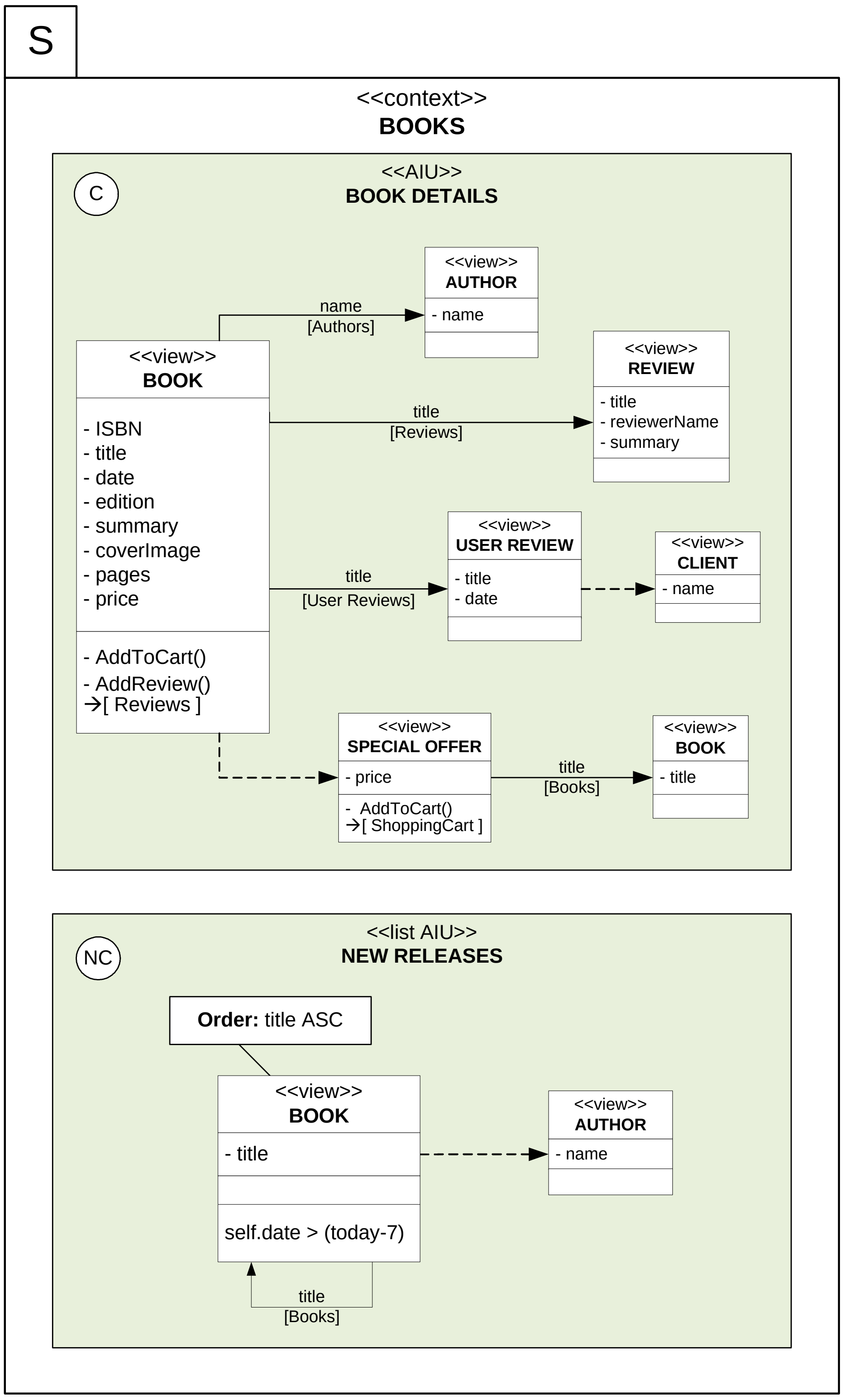
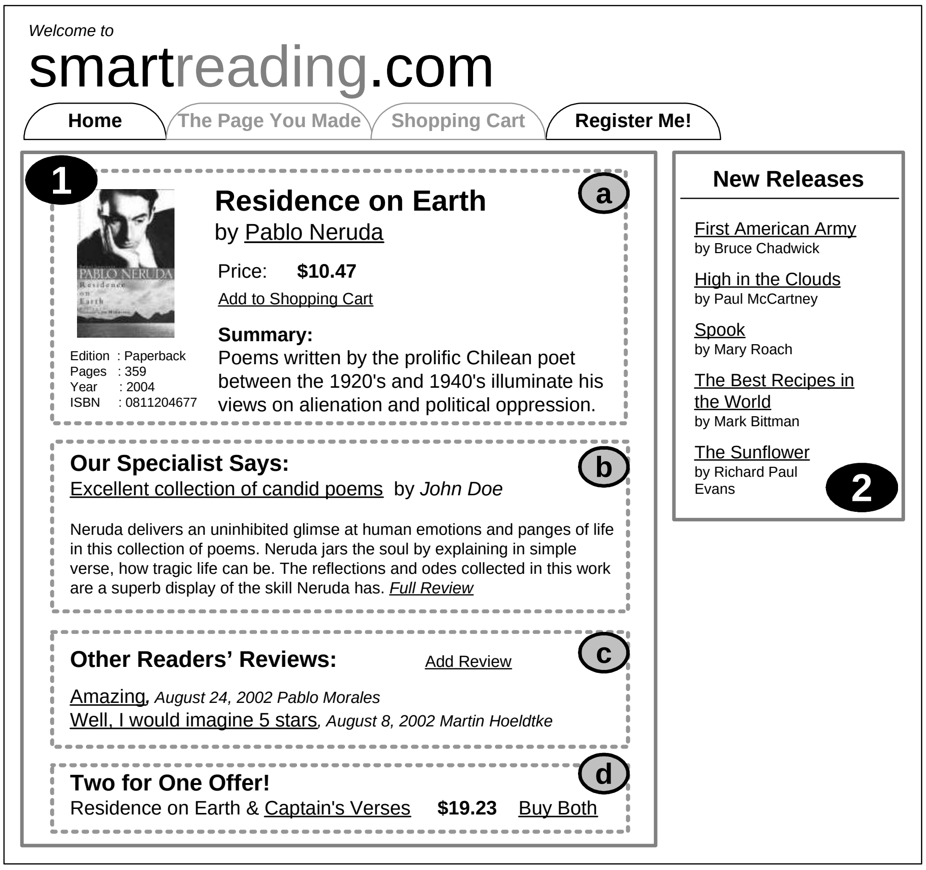


Figure 4.2: OOWS Navigational Context and its AIUs

non-contextual (what is represented through the “NC” tag), i.e., the displayed Book instances are not related to the contextual information that is received by the context. Furthermore, it is a *List* AIU (represented through the ≪List AIU≫ label), allowing the retrieval of several Book instances.


Welcome to

smartreading.com

Residence on Earth
by Pablo Neruda

Price: **$10.47**
Add to Shopping Cart

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Our Specialist Says:
Excellent collection of candid poems by *John Doe*

Neruda delivers an uninhibited glimse at human emotions and panges of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has. *Full Review*

Other Readers’ Reviews: Add Review

Amazing, *August 24, 2002 Pablo Morales*
Well, I would imagine 5 stars, *August 8, 2002 Martin Hoeldtke*

Two for One Offer!
Residence on Earth & Captain's Verses **$19.23** Buy Both

New Releases

First American Army
by Bruce Chadwick

High in the Clouds
by Paul McCartney

Spook
by Mary Roach

The Best Recipes in the World
by Mark Bittman

The Sunflower
by Richard Paul Evans

Figure 4.3: Implementation of BOOKS context

Figure 4.3 shows an implementation of the BOOKS Navigational Context. Frames *(1)* and *(2)* implement both AIUs, respectively. In Frame *1*, the structure of the BOOK DETAILS AIU can be distinguished through the dashed frames. Frame *(a)* shows instances of attributes from the Book manager class, along with information of the Author complementary class, while access to operations from Book is provided in frames *(a)* and *(c)*, respectively; Frame *(b)* shows information corresponding to the `Review` complementary class, while Frame *(c)* shows information from `User Review`. Finally, Frame *(d)* shows information from `Special Offer`. Access to hyperlinks (context relationships) and operations is represented through underlined words. On the other side, Frame 2 represents an implementation of the `NEW RELEASES` AIU. Only a small subset of books are shown, as a consequence of the definition of a *Population Filter*. Furthermore, the displayed books are alphabetically ordered, according to the definition of the corresponding *Ordering Pattern*.

The main structures of the OOWS Navigational Model have been presented above. From these structures, it is not possible to specify adaptive features of Web Applications. As a first step, the incorporation of a User Schema as part of the OOWS structural diagram helps to define navigational views that includes user characteristics. However, in order to provide the needed expressiveness to model Adaptive Web Applications, existing OOWS navigational structures need to be enhanced with properties to support the modelling of the different implementations that are provided to different users.

4.3 User Modelling approach

The definition of adaptive features in an adaptive Web application is based on high level descriptions of its intended users. In this section, we introduce a User Model to support the modelling of Adaptive Web Applications in OOWS. This model is defined as an extension of the Structural Schema of the application, which is expressed through the OO-Method Class Diagram.

4.3.1 Previous considerations

From an analysis of existing adaptive hypermedia applications, Kobsa et al. [75] distinguished three main categories of user characteristics that determine the adaptation of an application: *user data*, *usage data* and *environmental data*. *User data* refer to information about personal characteristics of the user, like objective facts (name, sex, income) and assumptions about user's knowledge, skills, goals, interests and preferences related to domain concepts; *Usage data* refers to data that can be directly observed and recorded, or acquired by analyzing observable data: navigational actions, visits, submitted searches, submitted rate of interests, etc., are examples of these characteristics; and *Environmental data*, which refers to the environmental conditions in which the user-application interaction occurs, comprising those related to the used technological platform (software and hardware) and locale's characteristics that may influence the navigational experience (geographical location, weather conditions, etc.).

The limits of these groups of characteristics are not precise, and a given characteristic may be classified into one group or another depending on the particular application and its usage conditions. We should consider this classification as a guideline to describe the intended users of a system rather than as a strict template to do this.

From reviews of approaches to user modelling in adaptive hypermedia applications, Brusilovsky established in [10] and [8] a finer-grained taxonomy of user characteristics, which comprise the following groups: user's knowledge, goals, background, hyperspace experience, preferences, user's interests, individual traits and environmental characteristics.

- *User's knowledge* about a concept is variable along the time, so the incorporation of this characteristic in a user specification should contemplate the ways to update its value. A frequently used strategy to model this characteristic is defining an *overlay model* over the domain schema, that associates user descriptions with each relevant domain concept, assigning a knowledge estimation for each (user,concept) pair. Values for this estimation may be qualitative or quantitative.

- *User's goals* are very related to the particular application with which the user interacts. For instance, an information retrieval system promotes the fulfilment of search goals, while educational applications are oriented to learning goals. Like the case of *User's knowledge*, this concept is usually modelled through an overlay model.

- *Background and Experience* are similar topics related to the familiarity of users with the knowledge domain of the application and with the navigational structure of the application, respectively.

- *Preferences* about certain domain concepts or application characteristics can be described in different ways. We can distinguish absolute preferences and relative or comparative preferences among concepts. Furthermore, these preferences may be explicitly described by the user, implicitly captured by the system or inferred from the navigational behaviour of the user.

- *Interests* are wider and more long-term user's preferences, and are mainly focused on domain areas rather than on specific concepts.

- *Individual traits* correspond to a set of characteristics that together define an individual as a relevant user of the application: psychological characteristics, learning styles, age group, etc.

- *Environmental characteristics*, which describe the platform of interaction and location data.

The concepts of this second taxonomy may be included in one or more groups from the first classification. According to the definition of these groups, we can say that the needed information to determine the values for each characteristic from the second taxonomy (excepting *environmental* *characteristics*) belongs to the *usage data* group, while the captured, calculated or estimated values belong to the *user data* group. These two groups are the focus of the user modelling proposal of this work, which is introduced in the following sections.

4.3.2 User Modelling in OOWS

With the goal of providing OOWS-based applications with highly expressive user specifications, which consider different kinds of features like the presented above, and on which adaptive features may be based, this work also includes a User Modelling proposal for Adaptive Web Applications.

OOWS method describes the structure of the problem domain through the definition of a UML-compliant *Class Diagram* [19]. By means of classes and associations, the relevant concepts of the universe of discourse are specified, including their relevant data and operations.

We propose to integrate the specification of users into the Class Diagram of the application. This decision is justified by the strong relationship between the domain concepts and the user characteristics in the development of Adaptive Web Applications. User is no longer an abstract entity that accesses the running application and receives outputs from the inputs that she provides, and whose particular characteristics are only considered to specify the application requirements or define access security levels. In Adaptive Web Applications, the user is part of the own application, and her characteristics becomes as important as the rest of the domain concepts.

The proposed user-domain integration permits that navigational views of the Class Diagram may incorporate the needed user characteristics to model the adaptive features of the application.

Adopting the taxonomy of user characteristics introduced by Kobsa et al. [75], this proposal considers user characteristics from *user data* and *usage data* groups, i.e., personal characteristics related to the particular application domain, and descriptions of the user's interaction with the application. However, the high abstraction level of the proposed descriptions allows a further inclusion of characteristics from the third group (*environment data*) with few additional considerations.

The current OOWS Class Diagram, which exclusively specifies the application domain, is transformed into a three-view diagram, as shown in Fig. 4.4.

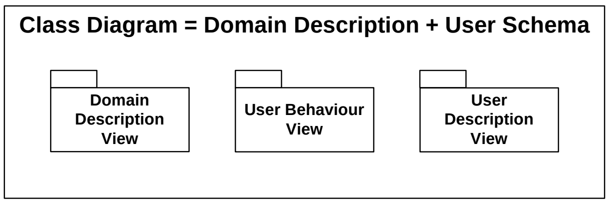


Figure 4.4: Extended OOWS Class Diagram, including user characteristics

Besides the traditional domain description, expressed through the so-called ***Domain Description View***, the extended Class Diagram includes: (a) the ***User Description View***, which specifies the different characteristics of individuals as relevant users of the application; and (b) the ***User Behaviour View*** which describes the different actions that users perform when interacting with the application.

4.3.2.1 User Description View

This view allows modelling users in terms of their individual characteristics and of those features that describe their relationship with the application domain.

From all possible characteristics of the application's users, the modeller selects those that may play some role in the fulfilment of its functional requirements and in the achievement of the desired adaptive functionalities, incorporating corresponding descriptions in terms of classes and associations.

These characteristics are more related to the individual that plays the role of application user than to her interaction with the application, thus values of these characteristics are mostly stable during a browsing session. Due to this reason, the state of represented objects and the associations among them has generally a low frequency of updating. Their modification may be made in the following ways:

- Explicitly by the user, modifying values of personal attributes: name, address, areas of interest, etc.

- Implicitly, as a consequence of performing navigational actions: previous purchases amount (from made purchases), proficiency level (from performed tests), areas of interests (from visited pages of the application).

Considering the taxonomy proposed by Kobsa et al. [75], this view includes the modelling of characteristics from the *user data* group. Characteristics included in this view are, for instance, personal or domain-independent characteristics of the current user; estimations of knowledge level about a concept and, in general, estimations of all the characteristics from the described taxonomy of Brusilovsky, excepting the environmental features.

Users can be grouped according to certain characteristics in common. The most relevant user groups may be incorporated as classes of this view. The consideration of a user group as a class of the diagram may be based on requisites of the application, on modeller's criteria, on the number of relationships that the group of users has with domain concepts, etc.

4.3.2.2 Domain Description View

It corresponds to the traditional Class Diagram, which describes the multiple domain concepts and associations among them, which are relevant to fulfil the requirements of the modelled application.

Although this view may be interpreted as the complete structural schema of a non-adaptive Web application, it may also include some classes and associations that support the modelling of adaptive features. For instance, the inclusion of keywords associated to a given domain class to support an adaptive comparison explanation method; reflexive prerequisite associations to support the modelling of an adaptive prerequisite explanation method; etc.

4.3.2.3 User Behaviour View

It allows describing users in terms of their navigational actions during a browsing session in the Web application. These characteristics are modelled through classes and associations that connect concepts from the two previous views, i.e., concepts that are used to describe the intended users with concepts that specify the structure of the application domain. From the actions that a user may perform during one or more browsing sessions in the application, the modeller selects those that provide explicit or implicit information about preferences, knowledge, goals and relevant characteristics needed to provide adaptivity, and describe them through classes and associations.

The characteristics of this view correspond to the mentioned *usage data* group. Their values are highly inconstant, and may be updated even several times during a single browsing session. Therefore, the state of represented objects and associations among them has generally a high frequency of updating.

To specify different types of navigational actions, users that are already modelled in the *User Description View* must be characterized from a different perspective, through a description according to their role in the browsing process. This can be made by defining a class that represents a browsing session in the application. Let us name this class ***Browsing Session Class***, only for identification purposes. Each instance from this class may be associated to a certain group of known (registered) users. If it is not, the instance corresponds to a traditional "anonymous" user's session.

There are some navigational actions that are independent of particular applications and very likely to be considered as an input for the descriptions of adaptive features. To *rate* or evaluate a given instance of a domain class; to *visit* a page that informs about a certain object; and the submission of a *search* action to find relevant pages associated to a provided keyword, are examples of these frequent actions. Furthermore, this view may include behavioural characteristics that are more related to the specific functionality of the application, such as the purchase of products in an e-commerce Web Site, or the submitted answers of a test in an e-learning application. All these actions are modelled by connecting the concepts that describe individual users (from *User Description View*) and the relevant domain concepts (from

Domain Description View).

This "connection" between user specifications and domain concepts is similar to the *overlay model* approach [10] traditionally adopted to model user's knowledge and goals. However, in the present proposal the user's description is considered as part of the domain structure and not just as a parallel layer over the domain specifications. This permits to model a broader set of relationships between users and domain concepts, although sacrificing modularity in the specification of the structural diagram.

The way of modelling user characteristics that are relevant to the application and its adaptation goals depends on the particular persons in charge of this task. Their expertise in structural modelling is fundamental to obtain conceptual descriptions that are expressive and comprehensive. The quality of the resulting Class Diagram will determine the quality of the final application and, in particular, of the adaptive features provided.

In the rest of the present dissertation, we use the term "Class Diagram" to the three-view diagram of a particular Adaptive Web Application, and the term "User Schema" to refer exclusively to the *User Description* and *User Behaviour* views of a particular Class Diagram.

Figure 4.5 shows an example of an extended Class Diagram, describing the structure of an adaptive electronic bookstore. We can distinguish its three previously described views, in colored frames. The upper frame shows the *User Description View*, which includes a class that represents a relevant user group (`CLIENT`). Its attributes and operations, along with the other two classes, describe characteristics from *Individual Traits*, like the `name`, `email`, `occupation` attributes and the `ADDRESS` class; and some features that describe user's interests, like the `readingHabit` attribute and the `INTEREST AREA` class.

The *Domain Description View* of the Class Diagram is shown at the bottom of the figure. Along with describing the domain of the electronic bookstore, it also includes some structures that support the further modelling of adaptive methods: the `COMPARATIVE` class, supporting an adaptive comparative explanation method between multiple instances of `BOOK`; the reflexive `prerequisiteOf` association, to model an prerequisite explanation method; and the `KEYWORD` class, to establish similarities among `BOOK` instances, for

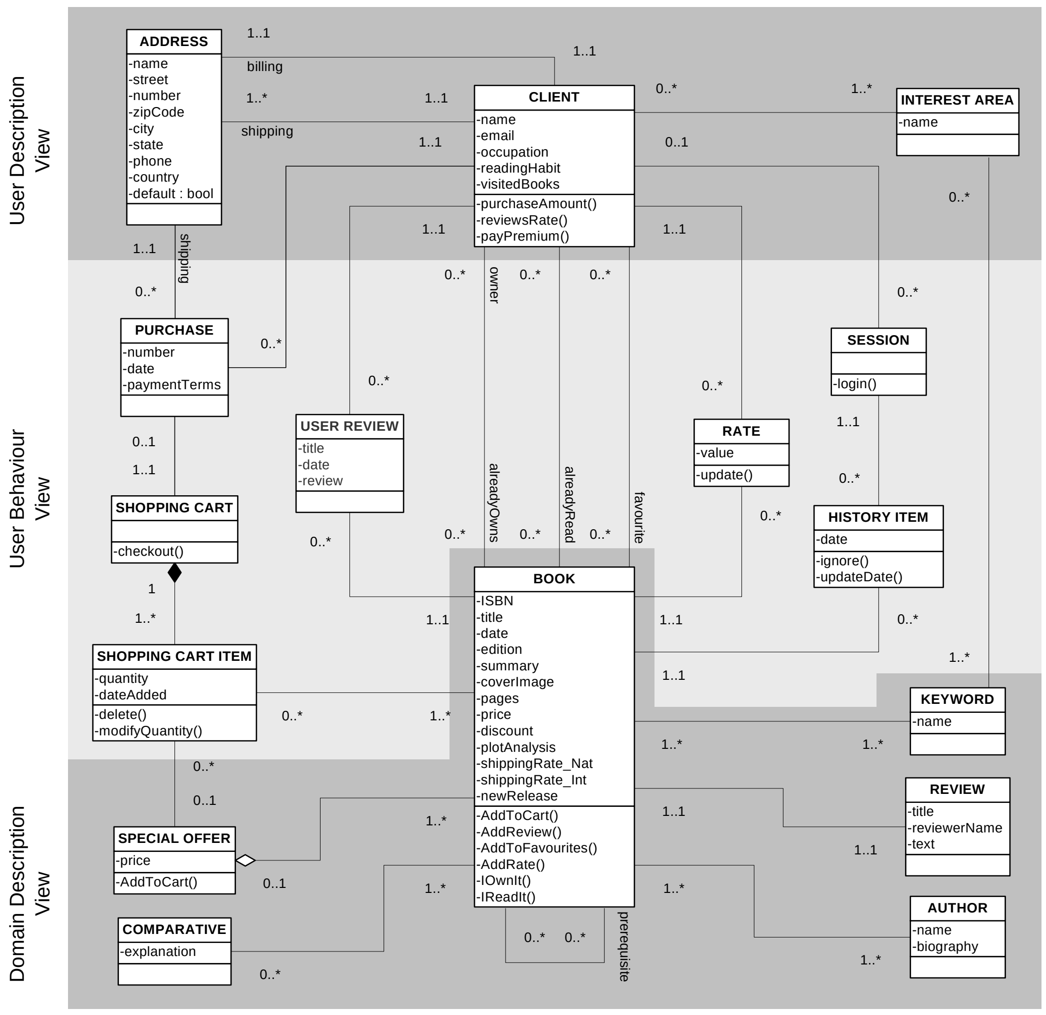


Figure 4.5: Class Diagram for an online bookstore, including its User Schema

possible recommendations to different users.

Finally, the middle frame shows the *User Behaviour View*. It includes the `SESSION` class, which represents the different browsing sessions, and whose instances may be associated to instances of the `CLIENT` class. The visit made to the page related to a given concept is described by means of the `HISTORY ITEM` class; some frequent functionalities to capture the feedback of the user, such as evaluating a concept (in this case, a `BOOK` instance), are also considered by means of the `RATE` class and the `alreadyOwns` and `favourite` associations. Finally, this view also includes the structural modelling of user-centered functionalities that are specifically related to the application, such as the submission of comments about a given product (`USER REVIEW` class), and the structures related to the purchase of products (`PURCHASE`, `SHOPPING CART`, etc).

4.4 Extended OOWS Navigational Metamodel

As a preamble of the introduction of Adaptive Primitives into the OOWS proposal, this section presents the metamodel that defines the conceptual elements of the OOWS Navigational Model.

The OOWS Navigational Metamodel defines the primitives that permit to describe the global navigational structure of a Web application, and the inner structure of each of its navigational nodes. Furthermore, it includes the specification of the Adaptive Primitives that this proposal introduces.

Figure 4.6 shows the complete OOWS Navigational Metamodel, represented through a UML Class Diagram. In the figure, the metaclasses and metaassociations that define the new Adaptive Primitives are marked with grey frames. The rest of elements compose the current OOWS Navigational Model. However, all shown metaattributes correspond to the definition of the new adaptive structures. For visibility purposes, frequently used metaatributes, such as "*name*" of each metaclass, have been excluded from the figure.

As a special case, the definitions of *Population Filter* and *Ordering Pattern* structures remain unmodified when adaptive features are included. However, their specification in the metamodel has been marked as part of the set of Adaptive Primitives, due to they have been provided with a more precise definition and a highly relevant role in the modelling of Adaptive Methods. Furthermore, the specialization of *Abstract Interaction Units (AIU)* in *Single-Object* and *List AIU* has been introduced by the present proposal, in order to facilitate the definition of adaptive primitives that affect the presentation of one object and several objects, respectively.

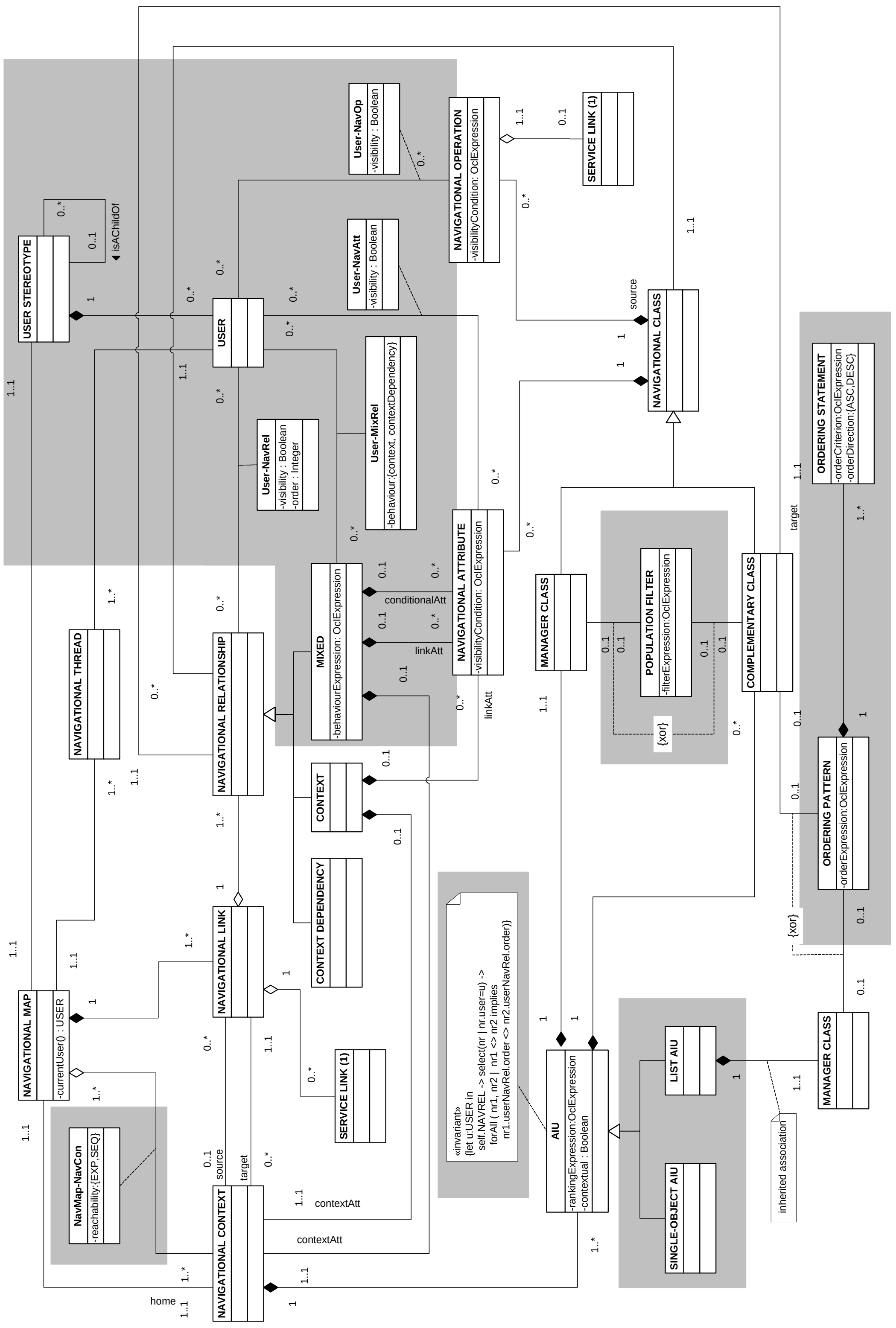


Figure 4.6: OOWS Navigational Metamodel

4.4.1 User description in the OOWS Navigational Metamodel

The specification of Adaptive Primitives demands the consideration of user characteristics in the OOWS Navigational Metamodel. Three elements fulfil this purpose: `USER`, `USER STEREOTYPE` and `NAVIGATIONAL THREAD` classes. These elements do not represent any primitive of an OOWS navigational schema, but describe the application users at a higher level of abstraction than the adopted in the User Schema.

- The `USER` class describe all the users that access an OOWS-based adaptive web application. As seen in the metamodel schema (Fig. 4.6), this class is associated with different elements of the metamodel. These associations describe the availability of the corresponding OOWS navigational structure to a particular user. In the definition of some Adaptive Primitives, the access to a given structure is ruled by adaptive constraints. In these cases, user-structure associations are represented through UML *association classes*, which describe those constraints through metaatributes (in Fig. 4.6, `User-NavOp`, `User-NavAtt`, `User-NavRel`, `User-MixRel` association classes).
- The `USER STEREOTYPE` class describes the relevant groups of users of the application. A user stereotype [23] is a subset of all potential users that share some characteristics and whose definition is relevant to the application. The set of the defined user stereotypes of an application are organized in a hierarchical structure. In adaptive systems, the definition of user stereotypes helps to provide similar adaptive features to similar users, saving development efforts and improving the quality of the adaptation. As seen in Fig. 4.6, a `USER STEREOTYPE` instance is composed of several instances of `USER`. In the definition of Adaptive Primitives, `USER STEREOTYPE` is associated to `NAVIGATIONAL CONTEXT`, through `NAVIGATIONAL MAP`. The `NavMap-NavCon` association class allows defining an adaptive behaviour of a given Navigational Context for a particular user stereotype. The concept of user stereotype is described in detail in Chapter 6 (*Requirements Specification for Adaptive Web Applications*) of this dissertation.

- Any Web application is accessed by multiple users at the same time, and multiple execution (or browsing) threads are assigned to respective users. These threads are described through the `NAVIGATIONAL THREAD` class. This class is directly associated to the `NAVIGATIONAL MAP` metaclass, expressing that a navigational thread is assigned to browse the entire application by following the navigational paths of a given map. A special operation is defined on the `NAVIGATIONAL MAP` metaclass: `currentUser()`. From the thread that is currently running, this operation allows returning the corresponding user. This operation is very important in the definition of the constraints for each adaptive structure, helping to identify the `USER` instance on which the adaptation is based.

These three elements fulfil an important role in the specification of the extended metamodel. Their associations with other metaclasses complete the template that rules the definition of actual Adaptive Primitives. At metamodelling abstraction level and considering this role, these elements may be considered as *classes*. On the other side, and unlike the rest of metaclasses of the OOWS Navigational Metamodel, their instances do not represent occurrences of any navigational primitive, but instead they characterize the actual users of an application, with an identity and attributes: from this perspective, these elements are *objects*. This double level of abstraction in which these elements are defined may be expressed through the concept of ***clabject*** [76]. A *clabject* is an entity that can exhibit, concurrently, a type (or class) facet and an instance (or object) facet [77].

The following section presents the set of Adaptive Primitives that this work introduces into the OOWS Navigational Model. In the description of each of them, its corresponding definition into the metamodel is included and explained.

4.5 Conceptual Primitives to model Adaptive Web Applications

This section presents the extensions proposed to the current OOWS Navigational Model, in order to support the modelling of Adaptive Web Applications. A set of conceptual primitives are incorporated in this model, each of them supporting the high level specification of one Adaptive Technique from the Adaptive Hypermedia's taxonomy [10, 8].

Each of the new primitives, which we called *"Adaptive Primitives"*, is described according to the following characteristics:

a) *Adaptation goal*, which provides a brief description of the type of adaptation that the adaptive primitive supports;

b) *Adaptive primitive*, which introduces the adaptive primitive, describing the way in which it is defined in an OOWS Navigational Schema and the adaptive functionality that it represents.

c) *Affected OOWS primitives*, which indicates which are the conceptual primitives of the current OOWS Navigational Model that are directly affected by the introduction of the adaptive primitive.

d) *Supported adaptive technique*, which indicates the specific Adaptive Technique(s) whose modelling is supported by the introduced adaptive primitive. In this chapter, these techniques are only mentioned. Their modelling is indicated in the next chapter of this dissertation. A more detailed description of these techniques, the reader is referred to the Section 2.3.4.

e) *Specification in the OOWS Navigational Metamodel*, which graphically describes the way that the adaptive primitive is inserted into the OOWS Navigational Model and its relationships with the rest of the primitives.

f) *Textual specification*, which complements the previous specification, through an OCL [56] description of the primitive in terms of the structures of the metamodel.

g) *Abstract example*, which illustrates the graphical notation of the introduced primitive, by means of an abstract OOWS Navigational Schema. This is complemented by an example of an abstract implementation.

h) *Case study example*, which presents a modelling example of the primitive, corresponding to the presented online bookstore case study. Furthermore, an example of the corresponding implementation is included.

The order of presentation of the Adaptive Primitives corresponds to the granularity of the OOWS primitive that is mainly affected by the new structure. For instance, Adaptive Primitives that affect the definition of Navigational Contexts precede those that affect Navigational Classes, while these ones precede those defined over Navigational Attributes.

Adaptive Primitive	**Affected OOWS primitive**	**Supported Adaptive Technique (AP/AN)**
Adaptive Reachability of Navigational Contexts	– Navigational Context	– Link Hiding (AN) – Page Variants (AP)
Adaptive Population Filter	– Navigational Class – Population Filter	– Link Hiding (AN) – Direct Guidance (AN) – Conditional Fragments (AP) – Frame-Based Adaptation (AP)
Adaptive Ordering Pattern	– Navigational Class – Ordering Pattern	– Link Ordering (AN) – Link Annotation (AN)
Mixed Relationship (*extended taxonomy of Nav.Relationships*)	– Navigational Relationship	– Strechtext (AP)
Adaptive Visibility of Navigational Relationships	– Navigational Relationship	– Frame-Based Adaptation (AP) – Link Hiding (AN)
Adaptive Ranking of Navigational Relationships	– Navigational Relationship	– Frame-Based Adaptation (AP) – Link Ordering (AN)
Adaptive Visibility of Navigational Attributes and Operations	– Navigational Attribute – Navigational Operation	– Conditional Fragments (AP)

Figure 4.7: Adaptive Primitives of the OOWS Navigational Model

Figure 4.7 shows the list of the introduced Adaptive Primitives, in the mentioned order. The second column of the table indicates the conceptual primitives that are mainly affected by the definition of the corresponding adaptive primitive; and the third column shows the adaptive technique whose modelling is supported by the primitive, indicating if it corresponds to an Adaptive Presentation (AP) or an Adaptive Navigation (AN) technique.

4.5.1 Reachability of Navigational Contexts

a) Adaptation goal

For different groups of users, the application must provide different access levels to the nodes of the application hyperstructure, i.e., different navigational maps.

b) Adaptive primitive

To model this characteristic, the Navigational Context structure (through which OOWS represents the application nodes) has been extended, incorporating a ***Reachability*** property.

For a given User Stereotype *U*, a Navigational Context can take one of the following values of Reachability:

1. ***Exploration***: the context can be accessed by users from *U* from any context of the hyperstructure, at any moment of the browsing process. A context with an *Exploration* value of Visibility corresponds to what OOWS previously defined as an ***Exploration Context*** [19]. Graphically is depicted as an UML package with an "E" label.
2. ***Sequential***: the context can be accessed by users from *U* only from certain navigational contexts through predefined navigational links. A context with an *Sequential* value of Visibility corresponds to what OOWS previously defined as an ***Sequential Context*** [19]. Graphically is depicted as an UML package with an "S" label.
3. ***Negative***: the context can not be accessed by any user from *U*. Graphically, a context with a *Negative* visibility is omitted from the corresponding navigational map.

The definition of Navigational Maps for different User Stereotypes is supported by this property. The *M* Navigational Map for a *U* User Stereotype is conformed by the set of Navigational Contexts with *Exploration* and *Sequential* values of Reachability for *U*, not including those contexts with *Negative* Reachability. Contexts with *Exploration* Reachability are accessed through *Exploration Links*, whereas those with *Sequential* Reachability are accessed through *Sequential Links*.

c) Affected OOWS primitives

Navigational Context

d) Supported adaptive technique

Although this primitive permits to define variants of the navigational hyperstructure for different users, the involved adaptation is completely defined at design-time, i.e., it corresponds to adaptable characteristics, instead of being based on adaptive techniques. Under this consideration, the implementation of this primitive adopts characteristics of ***Link-Hiding*** and ***Page Variants*** techniques. In the case of *Link-Hiding*, the links to contexts with negative reachability will be hidden or deactivated to users from the corresponding stereotype; in the case of *Page Variants*, access to different versions of the same context can be provided to different stereotypes, by properly managing the corresponding reachability values.

e) Specification in the OOWS Navigational Metamodel

Figure 4.8 shows the fragment of the OOWS enhanced metamodel that specifies the reachability property.

f) Textual specification

From this graphical definition, the Reachability property for Navigational Contexts may be defined as follows:

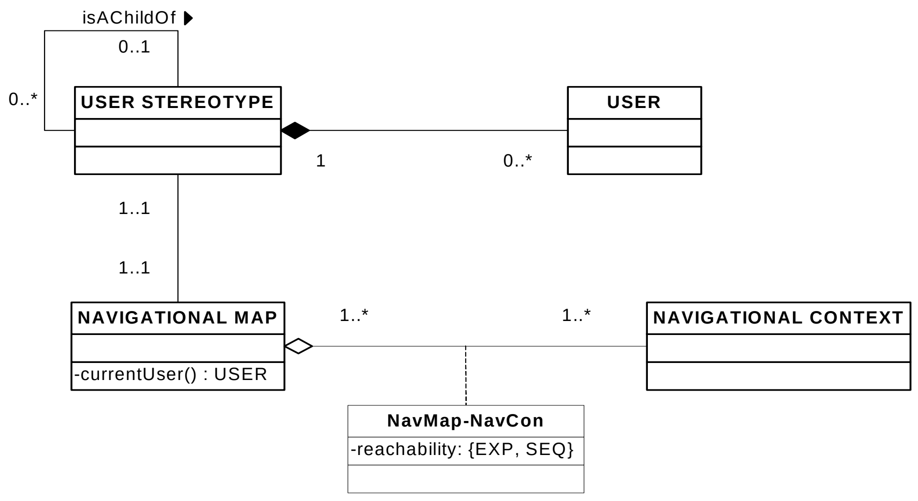


Figure 4.8: Reachability of Navigational Contexts in the OOWS Navigational Metamodel

Let

NS be the OOWS Navigational Schema of an Adaptive Web Application
NC be the set of Navigational Contexts of **NS**
NM be the set of Navigational Maps of **NS**,

then **Reachability property of Navigational Contexts** is a function defined by:

$$\begin{array}{rll} \textbf{Reachability}: & \textbf{NM} \text{ x } \textbf{NC} & \longrightarrow \{\text{exploration, sequential, negative}\} \\ & (\text{nm,nc}) & \longrightarrow \text{reachability(nm,nc)} \end{array}$$

This definition is based on navigational contexts and navigational maps, but without explicit reference to a set of users, as we can expect from an adaptivity property. However, there is a one-to-one relationship between navigational maps of a schema and the set of *User Stereotypes*, as seen in Fig. 4.8, so there is a strong consideration of user characteristics in the proposed definition. In this way, in the OOWS Adaptive Navigation Metamodel a relationship between a given pair (navigational map, navigational context) exists only if the reachability property assumes a value other than *“negative”* for this pair.

g) Abstract example

Figure 4.9 shows an abstract example of two navigational maps, associated to corresponding user stereotypes. Reachability values of navigational contexts are implicitly indicated: the inclusion of a context into a navigational map indicates a positive reachability value of that context for the corresponding user stereotype; a negative value is indicated by not including the context in the corresponding navigational map.

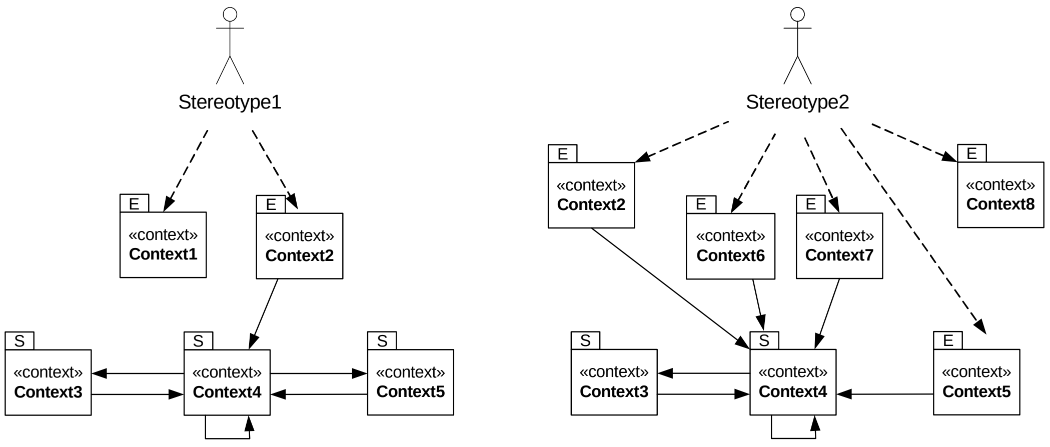


Figure 4.9: Abstract modelling example of *Reachability of Navigational Contexts* primitive

Some of the included navigational contexts have the same Reachability value for the two kinds of users (e.g., *Context2*, with *Exploration* reachability; *Context3*, *Context4*, with *Sequential* reachability). Meanwhile, other contexts have different reachability values: *Context6*, *Context7* and *Context8* contexts have an *Exploration* reachability for users from *Stereotype2*, and a *Negative* reachability for *Stereotype1* users. This means that the pages represented by these contexts are reachable by users from *Stereotype2*, from any page of the application at any time, while users from *Stereotype1* have no access to those pages. In the case of *Context5*, users from *Stereotype2* are allowed to access its corresponding pages from any page at any time (*Exploration* visibility), while users from *Stereotype1* can only access them through a predefined navigational path (*Sequential* visibility).

h) Case study example

Figure 4.10 shows two navigational maps, corresponding to two user stereotypes of the online bookstore: `Non-Registered Users` and `Clients`. In the left-side map, three contexts have an *Exploration* reachability, in contrast with the six contexts of the right-side map with the same value. Both maps have the same four contexts with *Sequential* reachability. Although the `Non-Registered Users` stereotype has less access to exploration contexts, the `Register` context has a negative value only for users from the `Clients` stereotype, probably because they are already registered in the system.

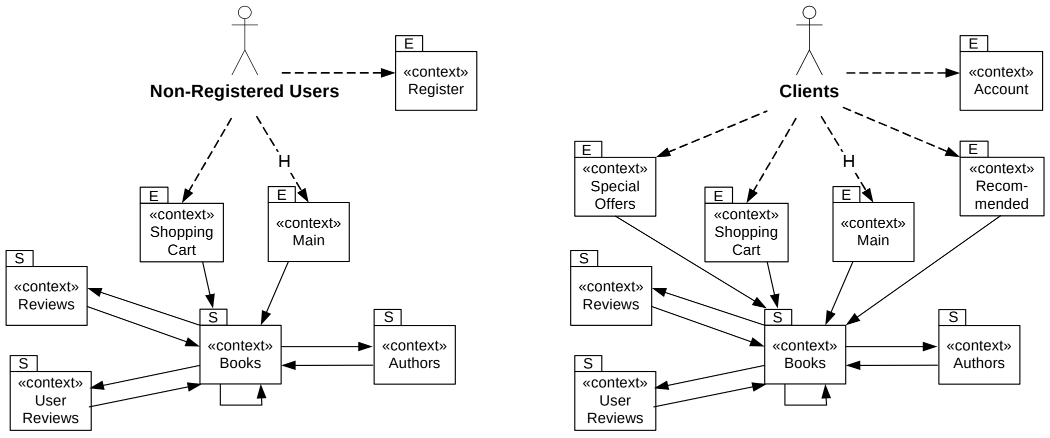
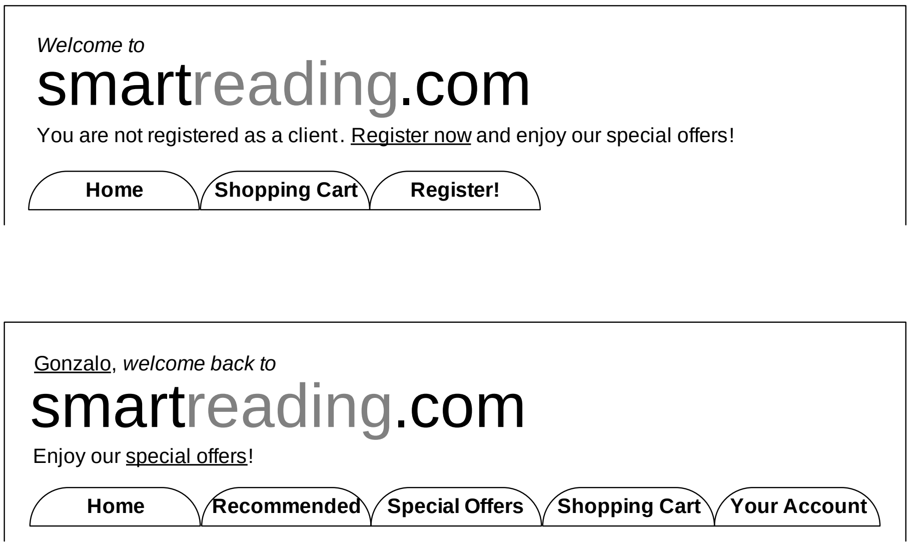


Figure 4.10: Concrete modelling example of *Reachability of Navigational Contexts* primitive

The different reachability values of the contexts included in Fig. 4.10 are illustrated in the implementation example of Fig. 4.11. This figure represent the top of all pages of the application, which include links to exploration contexts. The upper section shows the access provided to `Non-Registered Users`, including three tabs that link to the respective contexts, while the lower section corresponds to `Clients`, where a bigger number of exploration links can be accessed, but excluding the access to the `Register` context.


Welcome to
smartreading.com
You are not registered as a client. Register now and enjoy our special offers!

Gonzalo, *welcome back to*
smartreading.com
Enjoy our special offers!

Figure 4.11: Concrete modelling example of *Reachability of Navigational Contexts* primitive

4.5.2 Adaptive Population Filter

a) Adaptation goal

From a set of multiple information items, the system must adaptively select those that best fit the characteristics and goals of different users, discarding the less relevant ones.

b) Adaptive primitive

The OOWS ***Population Filter*** structure defines conditions that permit to filter the objects that are retrieved from a given navigational class. The inclusion of user characteristics into these conditions permits to select those objects that best fit the needs of a particular user, discarding the rest. However, the access to some objects may be suitable for some users, but not for others. By defining different subsets of users, it is also possible to retrieve instances under distinct selection criteria, adapting the filtering to the particular needs of users from each subset.

c) Affected OOWS primitives

Population Filter, Navigational Class.

d) Supported Adaptive Technique

Link Hiding, ***Direct Guidance***, ***Conditional Fragments***, ***Frame-Based Adaptivity*** . From a set of multiple information instances, an *Adaptive Population Filter* allows hiding the access to those that do not fulfil the adaptive conditions, selecting those that best fit the user's needs (*Link-Hiding*); if only the most relevant item is selected, then this primitive describes a *Direct Guidance* technique. When the adaptive filtering affects not only hyperlink anchors, but also data items and operations, *Conditional Fragments* and *Frame-Based Adaptivity* are implemented.

e) Specification in the OOWS Navigational Metamodel

Figure 4.12 shows the definition of the *Population Filter* structure in the OOWS Navigational Metamodel. The adaptive feature of this structure is given by the definition of *filterExpression* attribute of the *POPULATION FILTER* metaclass. This attribute belongs to the *OclExpression* domain, which corresponds to the set of all valid expressions in OCL language that can be evaluated in a given environment [56]. When this expression includes user characteristics, the defined structure corresponds to an *Adaptive Population Filter*; otherwise, it defines a non-adaptive Population Filter. As shown in Fig. 4.12, a Population Filter can be associated to the *Manager Class* or to one of the *Complementary Classes* of an AIU.

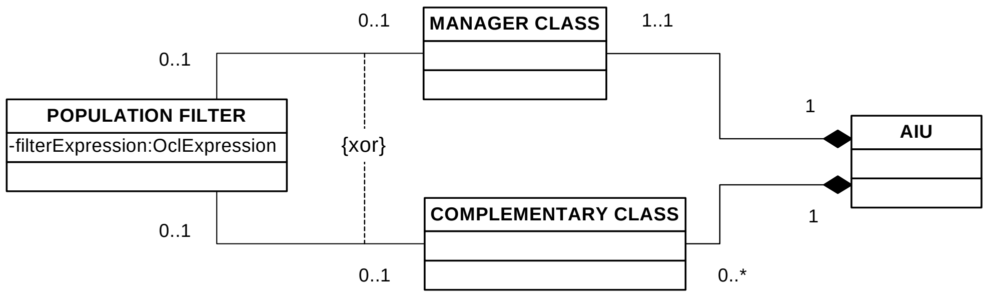


Figure 4.12: Adaptive Population Filter in the OOWS Navigational Metamodel

f) Textual specification

The textual specification of the *Adaptive Population Filter* primitive is the following:

Let

NS be the OOWS Navigational Schema of an Adaptive Web Application
USER be the whole set of users that access an instance of **NS**
AIU be the set of AIUs of **NS**
NCLASS be the set of Navigational Classes of **NS**,

then an **Adaptive Population Filter** on a given navigational class for a given user is defined as follows:

adPopFilter: **USER** x **AIU** x **NCLASS** $\longrightarrow$ {***OclExpression***}
(u,aiu,ncl) $\longrightarrow$ adPopFilter(u,aiu,ncl) =

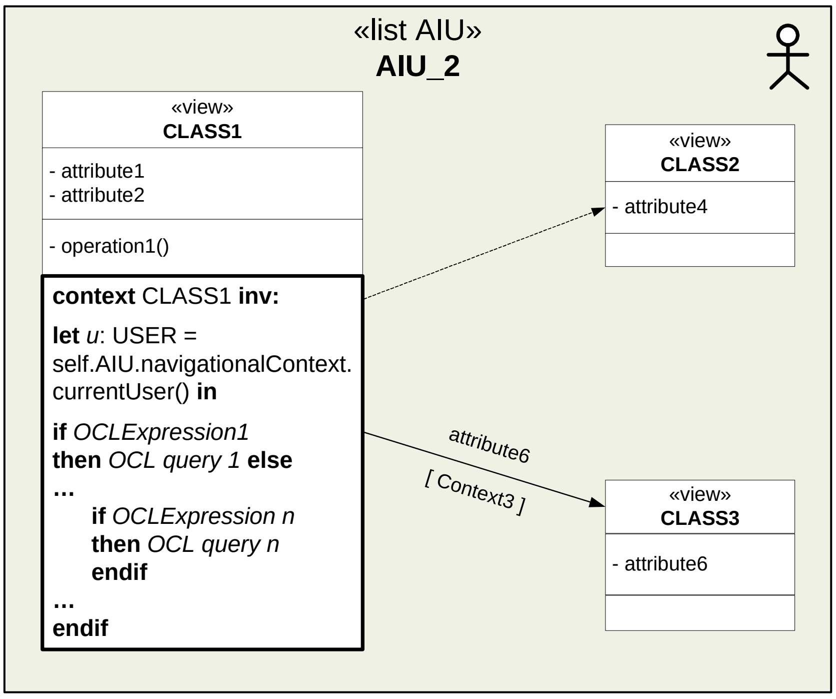
aiu.managerClass.populationFilter.filterExpression(u)
; if aiu = AIU$\rightarrow$select(managerClass=ncl)

aiu.complementaryClass.populationFilter.filterExpression(u)
; if aiu = AIU$\rightarrow$select(complementaryClass=ncl)

In this definition, the filtering expression of the structure (*filterExpression* attribute) is evaluated according to the characteristics of the (u) user. The result of this evaluation is an *OclExpression* instance, corresponding to the specific query that filters the objects of the *ncl* class for the *u* user. The presented definition varies according to the type of the navigational class in which the filtering is defined (manager or complementary class).

g) Abstract example

Figure 4.13 shows the definition of an Adaptive Population Filter into the manager class of an abstract *List AIU*. Its *filterExpression* attribute is located at the bottom of the class. It includes the conditions that are evaluated for the current user (*OclExpression1*, ..., *OclExpression n*) to decide which filtering query must be applied (*OCL Query 1*, ..., *OCL Query n*).



```
context CLASS1 inv:
let u: USER =
self.AIU.navigationalContext.
currentUser() in
if OCLExpression1
then OCL query 1 else
...
    if OCLExpression n
    then OCL query n
    endif
...
endif
```

Figure 4.13: Abstract modelling example of *Adaptive Population Filter* primitive

Figure 4.14 shows two pages, modelled through the List AIU of Fig. 4.13. According to the filter expression of the defined population filter, the user accessing the left-side version has access only to three objects (*obj1*, *obj2* and *obj3*) from *CLASS1*, while the user of the right-side version has access to more objects (as the scroll-bar seems to indicate), but no access to *obj1* and *obj3*.

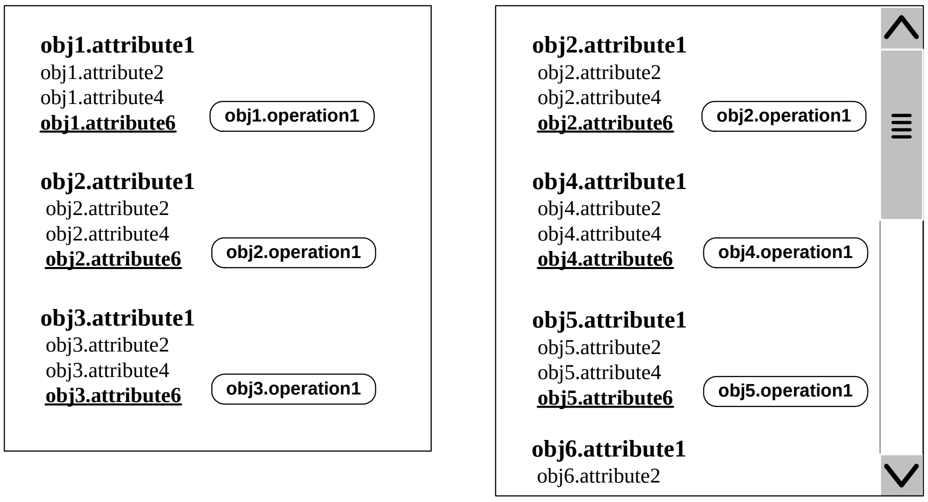
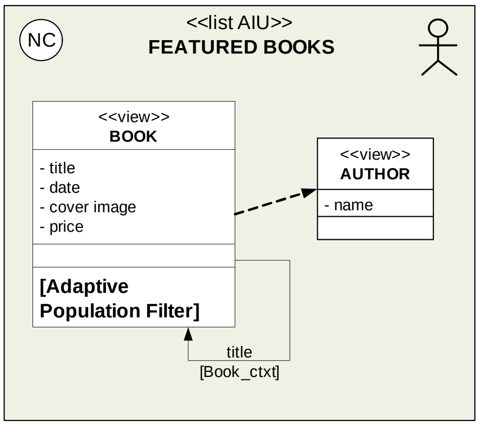


Figure 4.14: Abstract implementation of *Adaptive Population Filter* primitive

h) Case Study example

Figure 4.15 shows an example of the modelling of an *Adaptive Population Filter*. `Featured Books` is a Non-Contextual, List AIU, that shows a list of instances of the `Book` manager class. The definition of a population filter avoids to show all the `Book` instances in a single page. In this definition, u variable represents the current user of the application, while the `preferred` variable is a set of those books that u has favorably evaluated. If u user has been logged into the application as a `Client` and she has well evaluated at least one book, then the filter selects those books that have at least three keywords in common with those preferred by u, discarding those that u has marked as already owned. If u does not fulfil that condition, then the filter selects those `Book` instances whose `newRelease` boolean attribute has a positive value.

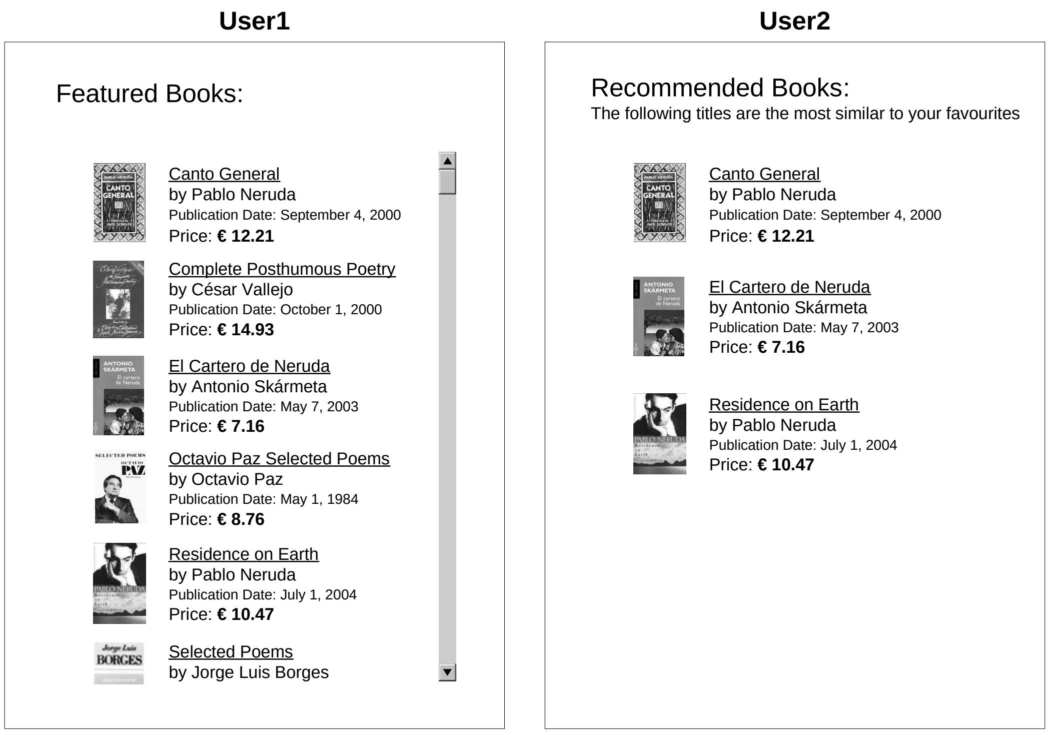


Adaptive Population Filter

```
context BOOK : NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in
let preferred:  Set(BOOK) = u.client.favourite -> union
                (u.client.rate-> select(value>2).book) in
if (u.login( ) = 'client') and (preferred.notEmpty( )) then
  (u.client.alreadyOwns-> excludes(self)) and
  (preferred.keyword.asSet( ) ->
  intersection (self.keyword) -> size( ) >= 3)
else  self.newRelease = 'true' endif
```

Figure 4.15: Modelling example of Adaptive Population Filter

Figure 4.16 shows an example of a concrete implementation of the Adaptive Population Filter modelled in Fig. 4.15. It represents two versions of a Web page section, corresponding to the `Featured Books` AIU. In the left-side section, a set of new released books is presented to `User1`, which means that this user does not fulfil the described condition. The right-side section presents a reduced set of books, that correspond to those that are most similar to the books preferred by `User2`. In the first case, the adopted filtering criterion is non-adaptive, because it does not depend on any user characteristic. As a result, the same set of books is presented to any user that does not fulfil the mentioned condition. On the contrary, the filtering criterion of the second case is adaptive, and the displayed books are likely to vary from one user to another.


User1

Featured Books:

Canto General
by Pablo Neruda
Publication Date: September 4, 2000
Price: **€ 12.21**

Complete Posthumous Poetry
by César Vallejo
Publication Date: October 1, 2000
Price: **€ 14.93**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

Octavio Paz Selected Poems
by Octavio Paz
Publication Date: May 1, 1984
Price: **€ 8.76**

Residence on Earth
by Pablo Neruda
Publication Date: July 1, 2004
Price: **€ 10.47**

Selected Poems
by Jorge Luis Borges

User2

Recommended Books:
The following titles are the most similar to your favourites

Canto General
by Pablo Neruda
Publication Date: September 4, 2000
Price: **€ 12.21**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

Residence on Earth
by Pablo Neruda
Publication Date: July 1, 2004
Price: **€ 10.47**

Figure 4.16: Implementation example of Adaptive Population Filter

4.5.3 Adaptive Ordering Pattern

a) Adaptivity goal

Multiple information items must be adaptively ordered, according to their respective relevance to the goals of different users.

b) Adaptive primitive

The OOWS ***Ordering Pattern*** structure defines different ordering criteria for displaying the objects that are retrieved from a given navigational class. Similarly to the Population Filter, this primitive can be enhanced with an adaptive behaviour. As an *Adaptive Ordering Pattern* considers user characteristics in the definition of the ordering criteria, then the retrieved objects can be ordered according to their relevance to different users. In this way, the most relevant objects are located near to the top of the page, and the less ones closer to the bottom.

c) Affected OOWS primitives

Ordering Pattern, Navigational Class

d) Supported Adaptive Technique

Link Ordering and ***Link Annotation***. From a set of multiple information instances, the *Adaptive Ordering Pattern* allows ordering the presentation of those instances according to some adaptive criteria. These information instances include not only hyperlink anchors, but also data items and operations. When textual or graphical clues about the order of each instance are provided, a *Link-Annotation* technique is implemented.

e) Specification in the OOWS Navigational Metamodel

The definition of an Adaptive Ordering Pattern into the OOWS Navigational Metamodel is shown in Fig. 4.17:

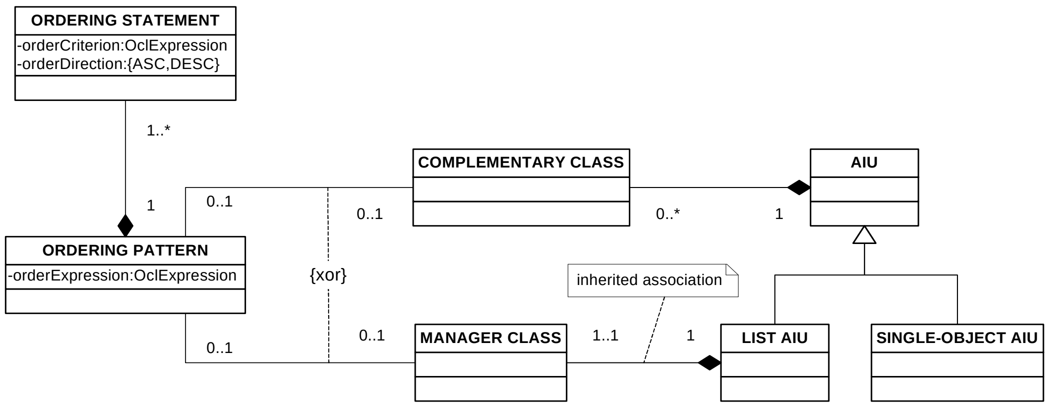


Figure 4.17: Adaptive Ordering Pattern in the OOWS Navigational Metamodel

According to the specification of Fig. 4.17, an ordering pattern is defined as the aggregation of multiple ordering statements. In the metamodel, an *Ordering Statement* metaclass describes this composition, containing an ordering criterion (the *orderCriterion* OCL expression), and an ordering direction (*orderDirection* attribute), which can be ascending or descending. The *Ordering Pattern* metaclass, and specifically its *orderExpression* atribute, defines the correspondence between different groups of users and different sets of ordering statements, along with the precedence among them, i.e., which is the primary order criterion, which is the secondary one, etc. The adaptive nature of the primitive is determined by the inclusion of user characteristics in the expressions of the *orderExpression* and *orderCriterion* attributes.

f) Textual specification

The following textual description supports the metamodel's segment that is shown in Fig. 4.17.

Let

NS be the OOWS Navigational Schema of an Adaptive Web Application
USER be the whole set of users that access an instance of **NS**
AIU be the set of AIUs of **NS**
NCLASS be the set of Navigational Classes of **NS**,

then an **Adaptive Ordering Pattern** on a given navigational class for a given user is defined as follows:

```
adOrdPattern: USER x AIU x NCLASS ⟶ {OclExpression}
              (u,aiu,ncl) ⟶ adOrdPattern(u,aiu,ncl) =

              aiu.complementaryClass.orderingPattern.orderExpression(u)
                  ; if aiu = AIU→select(complementaryClass=ncl)

              aiu.managerClass.orderingPattern.orderExpression(u)
                  ; if (aiu = AIU→select(managerClass=ncl))
                    and (AIU.isTypeOf(LIST_AIU))
```

In this definition, the ordering expression of the structure (*orderExpression* attribute) is evaluated according to the characteristics of the (u) user. The result of this evaluation is an *OclExpression* instance, corresponding to the specific set of ordering statements that define the relevance order of the objects of the *ncl* class for the *u* user. The presented definition varies according to the type of the navigational class in which the filtering is defined (manager or complementary class). Both graphical and textual descriptions of Adaptive Ordering Pattern restrict the navigational classes on which this structure may be incorporated, excluding the *manager class* of a *Single-object AIU*, because it represents the retrieval of information of only one object, and thus there is no need to define any ordering criterion.

g) Modelling example

Figure 4.13 shows the definition of an Adaptive Population Filter in the manager class of an abstract *List AIU*. Its *orderExpression* attribute is depicted as a framed expression linked to the manager class. Users that fulfil the *OclExpression1* condition visualize the retrieved objects according to *OclExpression1.i* (i=1..r) set of ordering criteria, while users that fulfil *OclExpression m* visualize those objects under the *OclExpression m.j* (j=1..s) criteria.

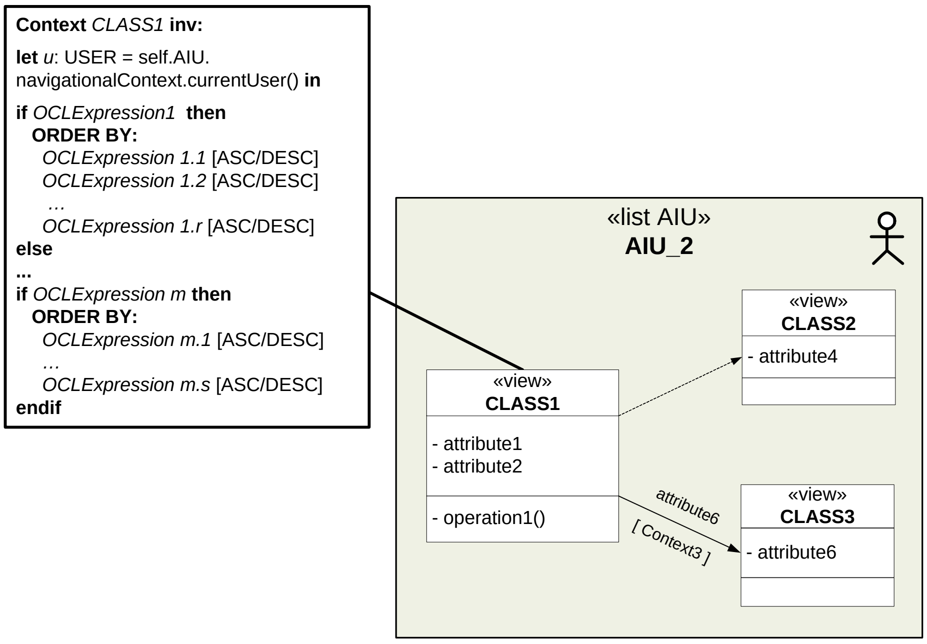

Context *CLASS1* **inv:**
let *u*: USER = self.AIU.
navigationalContext.currentUser() **in**
if *OCLExpression1* **then**
ORDER BY:
OCLExpression 1.1 [ASC/DESC]
OCLExpression 1.2 [ASC/DESC]
...
OCLExpression 1.r [ASC/DESC]
else
...
if *OCLExpression m* **then**
ORDER BY:
OCLExpression m.1 [ASC/DESC]
...
OCLExpression m.s [ASC/DESC]
endif
«list AIU»
AIU_2
«view»
CLASS1
- attribute1
- attribute2
- operation1()
«view»
CLASS2
- attribute4
«view»
CLASS3
- attribute6
attribute6
[Context3]

Figure 4.18: Adaptive Ordering Pattern

Figure 4.19 shows two versions of an abstract implementation of the List AIU shown in Fig. 4.18. Both users have access to the same set of objects, but the whole clusters of data and operations that are retrievable from the AIU are presented in different order, according to the corresponding ordering criterion.

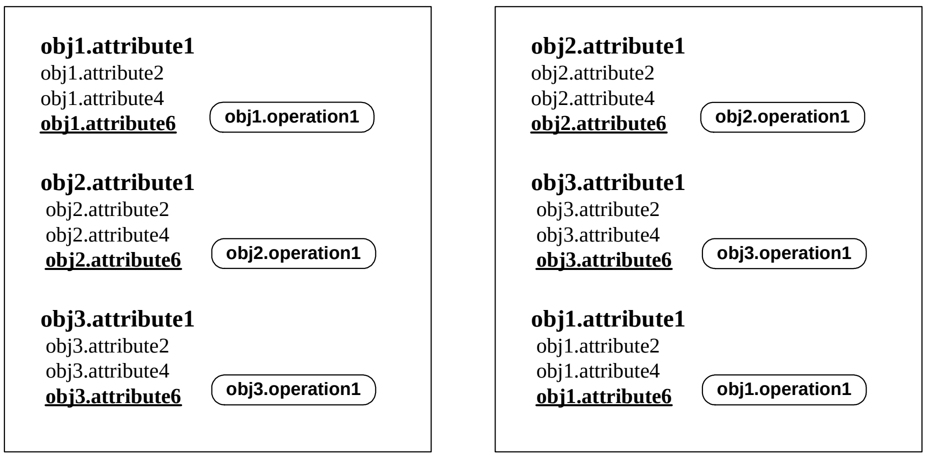
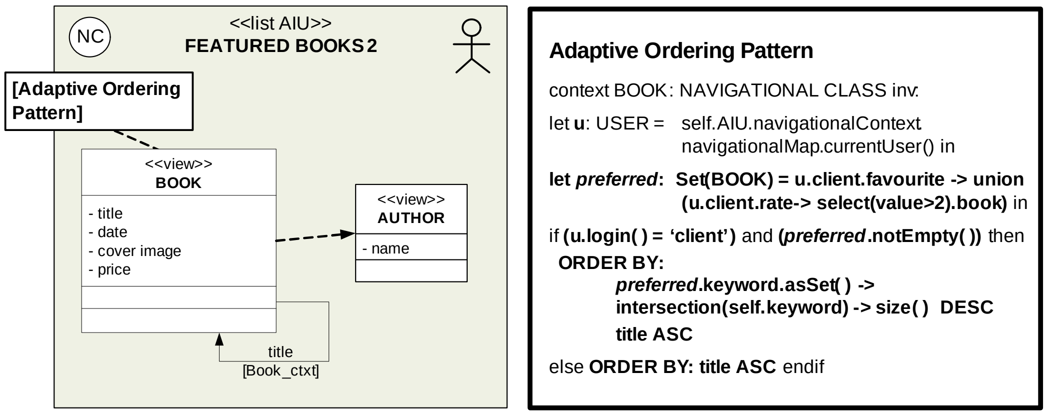


Figure 4.19: Example of Adaptive Ordering Pattern

h) Case study example

Figure 4.20 shows an example of the modelling of an *Adaptive Ordering Pattern* in the already introduced `Featured Books` AIU. For readability purposes, the corresponding ordering expression (*orderExpression* attribute) is shown at the right of the figure. In this expression, the u variable represents the current user of the application, while the `preferred` variable is a set of those books that u has favorably evaluated. If the u user has been logged into the application as a `Client` and she has well evaluated at least one book, then the objects of the `Book` class are ordered according to two ordering statements: (a) according to the number of associated keywords that each object has in common with those preferred by u (adaptive criterion), in descending direction, and (b) according to the corresponding value of `title` attribute (non-adaptive criterion), in an ascendant direction. If u does not fulfil that condition, then the ordering expression considers only one ordering statement, corresponding to the non-adaptive criterion based on the `title` attribute.

Figure 4.21 shows an example of a concrete implementation of the Adaptive Ordering Pattern modelled in Fig. 4.20. It represents two versions of a Web page section, corresponding to the `Featured Books` AIU. In the left-side section, the books presented to `User1` are alphabetically ordered according their title, which means that this user does not fulfil the described condition.

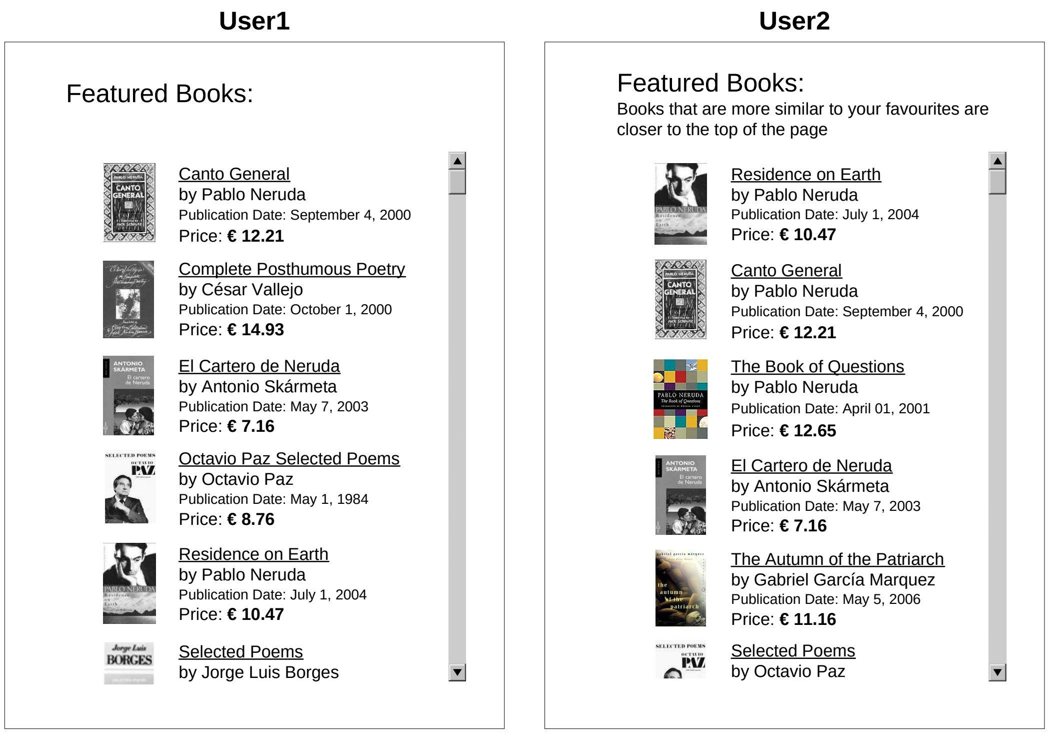


Adaptive Ordering Pattern

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =  self.AIU.navigationalContext.
               navigationalMap.currentUser() in
let preferred:  Set(BOOK) = u.client.favourite -> union
               (u.client.rate-> select(value>2).book) in
if (u.login( ) = 'client') and (preferred.notEmpty( )) then
  ORDER BY:
         preferred.keyword.asSet( ) ->
         intersection(self.keyword) -> size( )  DESC
         title ASC
else ORDER BY: title ASC endif
```

Figure 4.20: Modelling example of Adaptive Ordering Pattern

The right-side section presents the same set of books, but ordered according to their similarity to the books preferred by `User2`. In the first case, the adopted ordering criterion is non-adaptive, because it does not depend on any user characteristic. As a result, the set of books is presented in the same order to any user that does not fulfil the mentioned condition. On the contrary, the ordering expression of the second case is adaptive, and the order of the displayed books is likely to vary from one user to another.



User1

Featured Books:

Canto General
by Pablo Neruda
Publication Date: September 4, 2000
Price: **€ 12.21**

Complete Posthumous Poetry
by César Vallejo
Publication Date: October 1, 2000
Price: **€ 14.93**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

Octavio Paz Selected Poems
by Octavio Paz
Publication Date: May 1, 1984
Price: **€ 8.76**

Residence on Earth
by Pablo Neruda
Publication Date: July 1, 2004
Price: **€ 10.47**

Selected Poems
by Jorge Luis Borges

User2

Featured Books:
Books that are more similar to your favourites are closer to the top of the page

Residence on Earth
by Pablo Neruda
Publication Date: July 1, 2004
Price: **€ 10.47**

Canto General
by Pablo Neruda
Publication Date: September 4, 2000
Price: **€ 12.21**

The Book of Questions
by Pablo Neruda
Publication Date: April 01, 2001
Price: **€ 12.65**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

The Autumn of the Patriarch
by Gabriel García Marquez
Publication Date: May 5, 2006
Price: **€ 11.16**

Selected Poems
by Octavio Paz

Figure 4.21: Implementation example of Adaptive Ordering Pattern

4.5.4 Mixed Navigational Relationships

a) Adaptation goal

According to their particular characteristics, different users can be provided with different access levels to the same content: for one group of users, the content is immediately shown, while for other users it is only reachable through a link activation.

b) Adaptive primitive

Complementary classes of an AIU allow including valuable information which complement the data and operations provided by the manager class. The current two types of Navigational Relationship (*Context* and *Context-Dependency*) allow specifying two different ways to retrieve this information: information retrieval plus navigation and only information retrieval, respectively.

However, a user may require a special behaviour of a navigational relationship. She may need to activate the navigation provided through a Context Relationship, in order to obtain further information; in other cases, she may only need to retrieve the complementary information through the same relationship, but with no need to navigate to another context. For instance, let us think of a customer that uses to activate a link that leads to a specialized review about on-sale products. In base of this recurrent navigational behaviour, the application can relocate the text of the specialized review to the main page of the product, avoiding the need of additional "clicks" to access it.

To model such kind of adaptation, the taxonomy of the OOWS Navigational Relationship has been extended, incorporating the ***mixed*** type. The updated taxonomy is the following:

- **Context-Dependency Relationship** (graphically represented using dashed arrows), which represents a basic information recovery by crossing a structural relationship between classes.

- **Context Relationship** (solid arrows), which represents an information recovery and a navigation to a target Navigational Context.
- **Mixed Relationship** (dash-point arrows), which can behave as a Context-Dependency or as a Context relationship, depending on the characteristics of the user who accesses the AIU in which the relationship is included.

To specify whether a mixed relationship acts as a context-dependency or a context relationship, a ***Behaviour*** property is defined. The values of this property are: *Context-Dependency* and *Context*, respectively. Each mixed relationship has attached a conditional clause, which includes the constraint that must be fulfilled for each value.

The introduced Mixed Relationship structure has the following properties:

- A ***context attribute***, which indicates the target context of the navigation (depicted as [target context]).
- A ***link attribute***, which specifies the attribute from the target class that is used as the *anchor* of the navigation.
- A ***behaviour property***, whose values indicate the two possible behaviours of a mixed relationship: *context* and *context-dependency.*
- A ***behaviour expression***, which specifies the criteria to determine whether the relationship behaves as a context-dependency relationship or as a context one, when a given user accesses the corresponding AIU. It is specified through an *IfExp* OCL expression, which assigns the relationship one of the two possible values of *behaviour property*, depending on the evaluated value of a boolean condition. This condition is imposed to characteristics of the user included in the User Schema of the application. Graphically, the behaviour expression is written at the bottom of the graphical primitive of the corresponding AIU.
- One or more optionally included ***conditional attributes***, which specify attributes that are shown in the modelled page only if the mixed relationship behaves as a context relationship. Graphically, the name of a conditional attribute is enclosed in parenthesis.

c) Affected OOWS primitives

Navigational Relationship.

d) Supported Adaptive Technique

Strechtext. The definition of the Mixed Relationship primitive permits to provide users with a collapsible version of an explanation. The full version of an explanation can be shown to a certain group of users, by retrieving the complete information from the corresponding complementary class, while a summarized version is provided to the rest of users, along with a hyperlink to access the full version. The adaptive character of this modelling alternative is given by the inclusion of user characteristics in the *behaviour expression* of the *mixed relationship.*

e) Specification in the OOWS Navigational Metamodel

Figure 4.22 shows the new taxonomy of navigational relationships in the OOWS Navigational Metamodel. The special components of *context* and *mixed relationships* relationships (*context attribute*, *link attribute* and *conditional attributes*) are described through composition associations between the corresponding metaclasses. The definition of *mixed relationship* describes the adaptive feature of this taxonomy. The evaluation of its *behaviourExpression* attribute for the current user updates the *behaviour* attribute value of the *User-MixRel* class-association. This value determines the way in which the information from the relevant complementary class is shown to that user.

f) Textual specification

From the updated taxonomy of navigational relationships, only the *mixed relationship* supports the definition of adaptive features. The following specification shows the way to specify the behaviour of a mixed relationship for a given user, according to the elements of the OOWS Navigational Metamodel. The specification shows that this behaviour is obtained by evaluating

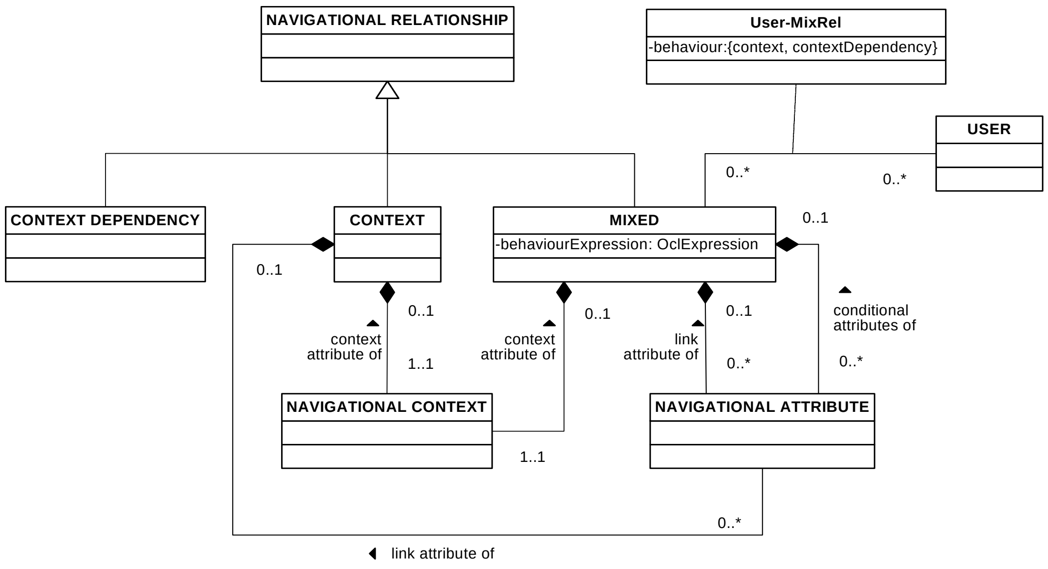


Figure 4.22: Navigational Relationship Taxonomy in OOWS Navigational Metamodel

the *behaviourExpression* attribute of the corresponding *Mixed Relationship* instance, according to the characteristics of the current user. The obtained value is assigned to the *behaviour* attribute of the corresponding (user, relationship) association.

Let

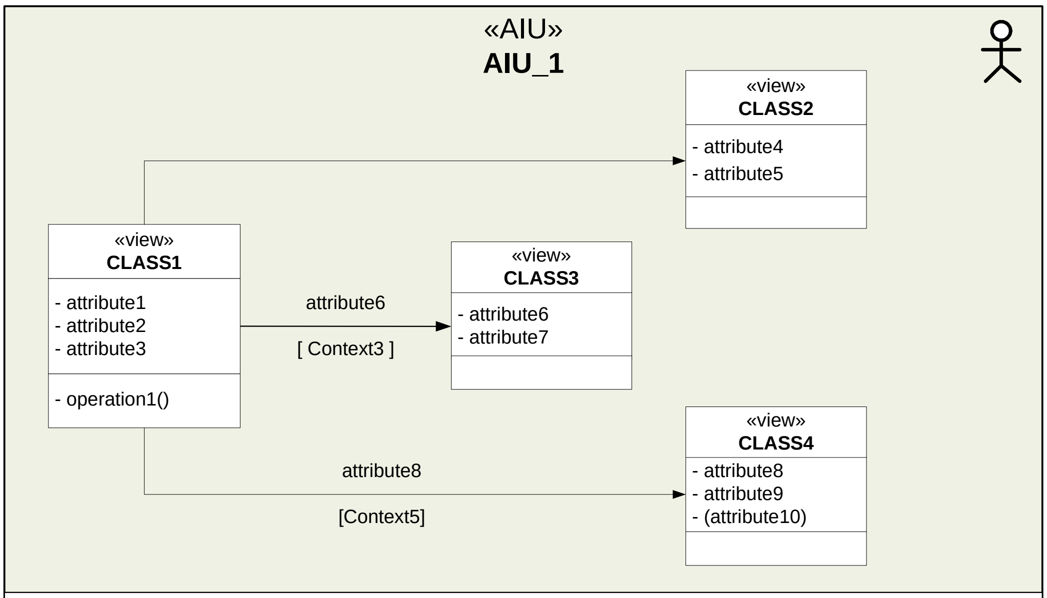
NS be the OOWS Navigational Schema of an Adaptive Web Application
USER be the whole set of users that access an instance of **NS**
MR be the set of Mixed Navigational Relationships of **NS**,

then the **adaptive behaviour** of a mixed relationship is defined by the following function:

$$\textbf{behaviour}: \textbf{USER} \text{ x } \textbf{MR} \longrightarrow \{\text{context, context dependency}\}$$
$$(\text{u,mr}) \longrightarrow \text{behaviour(u,mr)} = \text{mr.behaviourExpression(u)}$$

g) Modelling example

Figure 4.23 shows an abstract specification of an AIU that contains the three types of navigational relationships. A *context dependency relationship*, which associates the *CLASS1* and *CLASS2* navigational classes; a *context relationship*, between *CLASS1* and *CLASS3*, which allows navigating to the *Context3* navigational context by activating the anchor provided through the *attribute6* attribute; and a *mixed relationship*, which associates *CLASS1* and *CLASS4*. Its *behaviour expression* is shown at the bottom of the AIU.



```
context AIU_1 inv:
    let u: USER = self.navigationalContext.currentUser() in
    let navrel14: NAVREL = self.navrel-> select (nr | nr.source=CLASS1 and nr.target=CLASS4) in
    let um14: USER-MIXED = navrel14.userMixed-> select (mr | mr.user = u) in
    if OCLExpression1
        then   um14.behaviour = 'context'
        else   um14.behaviour  = 'contextDependency'
    endif
```

Figure 4.23: The three types of OOWS navigational relationships

Figure 4.24 shows an abstract implementation of the AIU of Fig. 4.23, accessed by two different users. The user accessing the left-side version fulfills the *OclExpression1* condition, so the mixed relationship adopts a *context* behaviour. The value of the *attribute8* attribute plays the role of anchor of the navigation to the *Context5* navigational context, while the value of the *attribute10* conditional attribute is not displayed. On the contrary, the same relationship behaves as a *context-dependency* one towards the user accessing the right-side version, which does not fulfil the *OclExpression1* condition. In this case, values of the whole set of attributes from the *CLASS4* target class are retrieved and displayed in the page, including the conditional *attribute10*, and none of these values provides an anchor to navigate to any context.

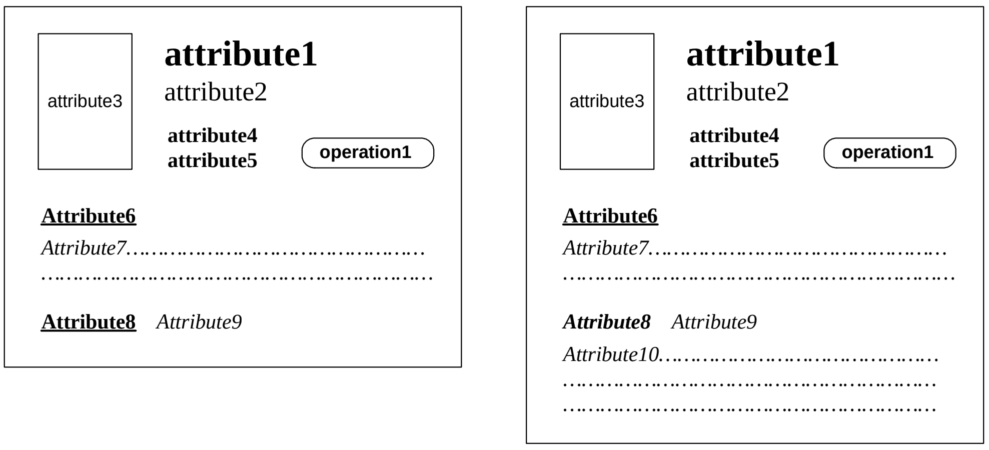
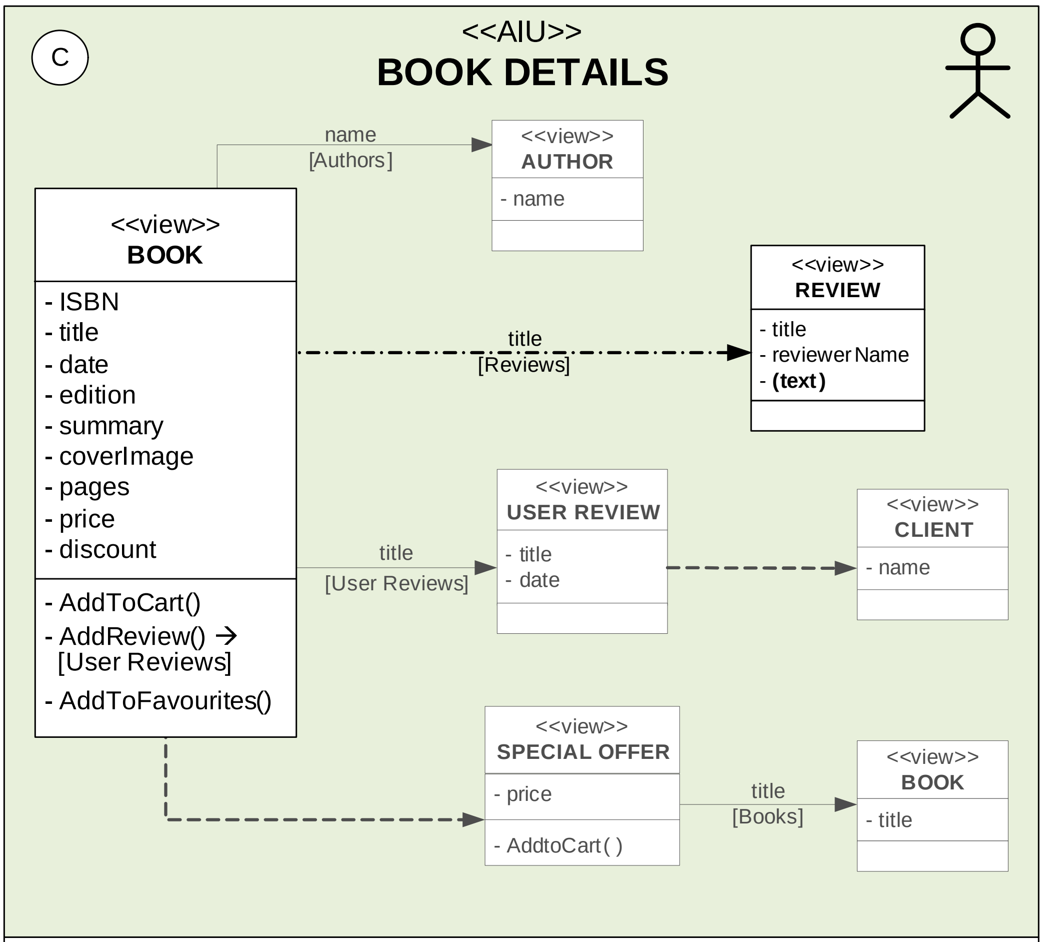


Figure 4.24: Example of adaptive behaviour of mixed relationships

h) Case study example

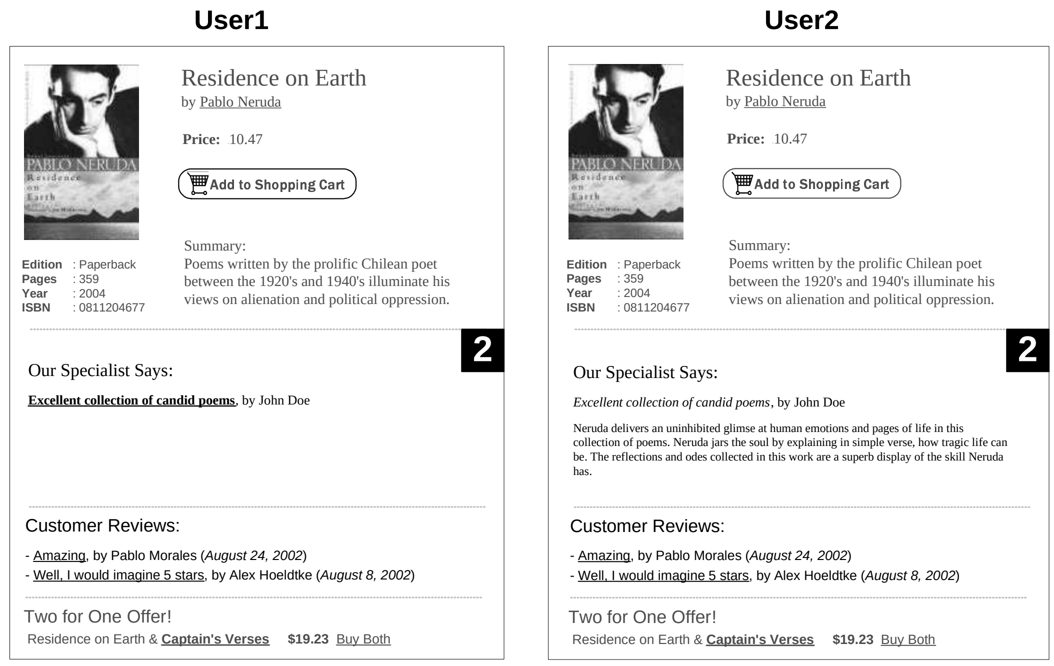
Figure 4.25 shows the modelling of the three types of navigational relationships from the updated taxonomy. A *mixed relationship* rules the access to the specialized `Review` class. According to the *behaviourCondition* expression, users must be identified as member of the `Clients` stereotype and they must have navigated to the full version of the review frequently. In this case, the mixed relationship has as a context-dependency behaviour and the value of the `text` attribute is shown with no need of previous navigation.



```
context BOOK DETAILS : AIU inv:
let u: USER = self.navigationalContext.navigationalMap.currentUser() in

let navrel12: NAVIGATIONAL RELATIONSHIP = self.navrel->
select (nr|nr.source.name = 'BOOK' and nr.target.name = 'REVIEW') in
let um12: USER-MIXED = navrel12.userMixed-> select (mr|mr.user = u) in
  if (u.login( ) = 'client') and (u.reviewsRate( ) >= 80) and (u.visitedBooks >= 20)
  then um12.behaviour = 'contextDependency'
  else um12.behaviour = 'context'
  endif
```

Figure 4.25: Modelling example of the three types of navigational relationships

Figure 4.26 shows a concrete implementation of the *Mixed Relationship* primitive, by means of two pages that can be obtained from the AIU of Fig.4.25. Frame 2 of both pages shows information that is retrieved through the described mixed relationship. The left-side page shows that `User1` is not logged as a `Client` or does not fulfil the condition of having visited at least 20 pages of `Book` instances and navigated to the full specialized `Review` in at least the 80% of those visits. For this reason, the `text` attribute is hidden to this user, and a hyperlink to deeper information is provided. On the contrary, the right-side page shows that `User2` fulfills the *behaviour condition* and value of text attribute is visible immediately, with no need for previous navigation.



User1

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

2

Our Specialist Says:

Excellent collection of candid poems, by John Doe

Customer Reviews:

- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

User2

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

2

Our Specialist Says:

Excellent collection of candid poems, by John Doe

Neruda delivers an uninhibited glimse at human emotions and pages of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has.

Customer Reviews:

- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

Figure 4.26: Implementation example of Adaptive Mixed Relationship

4.5.5 Visibility of Navigational Relationships

a) Adaptation goal

Data and operations that complement the main information of a page may be suitable for a given group of users only. This information may be adaptively hidden from the rest of users.

b) Adaptive primitive

Navigational relationships allow specifying the access to data and operations that complement the information of the manager class, as well as to the navigation to another contexts. Some adaptive methods consider the adaptation of this access, by restricting it to some users and allowing it for others. As an enhancement of the Navigational Relationship structure, we introduce the *Visibility* property to specify the adaptation of the access to navigational relationships for different subsets of users.

Each navigational relationship of an AIU has a ***Visibility expression***. This is an OCL expression, defined in terms of domain and user-oriented structures, which assigns a binary value of *visibility* to the relationship, according to the user that currently accesses the corresponding AIU. If a navigational relationship has a *negative* visibility, the intended users cannot access the attributes and operations from the complementary class that is target of that relationship, nor the eventual navigation to another context in the case of a context or a mixed relationship.

c) Affected OOWS primitives

Navigational Relationship.

d) Supported Adaptive Technique

Frame-Based Adaptation, ***Link-Hiding***. The information that is retrieved through a navigational relationship is shown as a frame of data, operations and hyperlinks. For this reason, adapting the access to a navigational relationship implies to adapt the access to the entire frame. This characteristic corresponds to the *Frame-Based Adaptation* technique. On the other side, the adaptive access to a context relationship involves to adapt the access to a hyperlink of the page, which corresponds to a *Link-Hiding* technique.

e) Specification in the OOWS Navigational Metamodel

Figure 4.27 shows the definition of the *Visibility* property in the OOWS Navigational Metamodel. The *User-NavRel* association-class (which describes the availability of navigational relationships to the application users) contains the boolean *visibility* attribute, whose value is determined by evaluating the *visibilityExpression* attribute of the corresponding association, defined as an attribute of *Navigational Relationship* metaclass of *OclExpression* type. In case of relationships with no adaptive access, the corresponding visibility expression always assigns a positive value to the *visibility* property, no matter the characteristics of the current user.

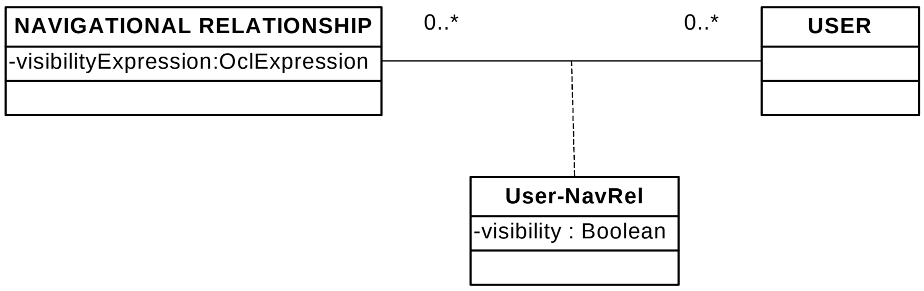

NAVIGATIONAL RELATIONSHIP
-visibilityExpression:OclExpression
0..*
0..*
USER
User-NavRel
-visibility : Boolean

Figure 4.27: Visibility of navigational relationships in the OOWS Navigational Metamodel

f) Textual specification

The textual specification of this property is the following:

Let

NS be the OOWS Navigational Schema of an Adaptive Web Application
USER be the whole set of users that access an instance of **NS**
NR be the set of Navigational Relationships of **NS**,

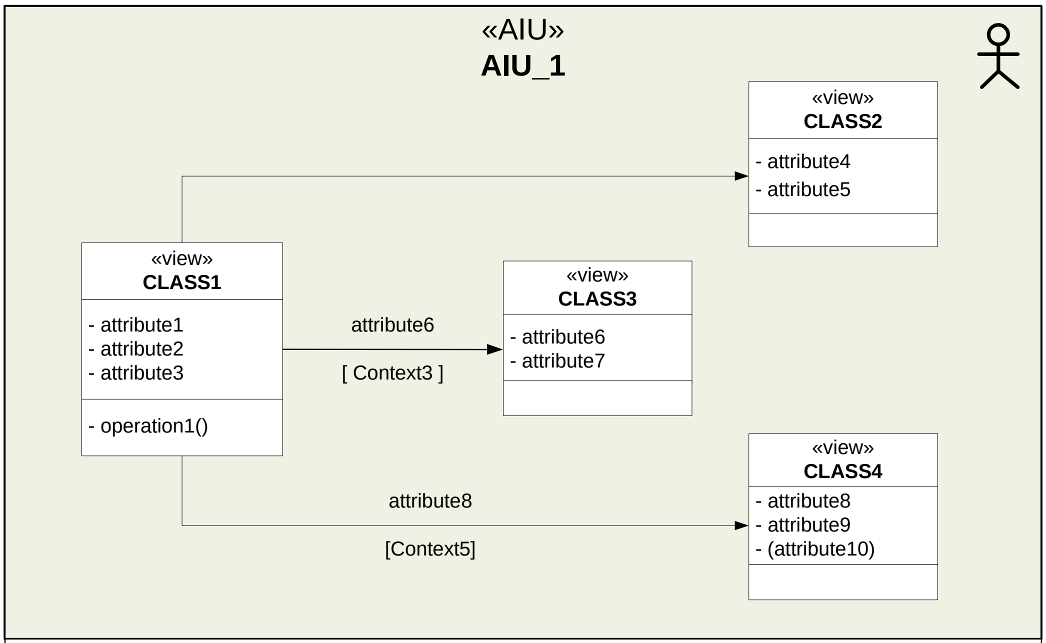
then the adaptive value of **Visibility of a navigational relationship** with respect to a given user, expressed through the *visibility* metaattribute of the *User-NavRel* metaclass, is obtained through the following function:

$$\begin{array}{lll} \textbf{visibility}: & \textbf{USER} \text{ x } \textbf{NR} & \longrightarrow \{\text{positive, negative}\} \\ & (\text{u,nr}) & \longrightarrow \text{visibility(u,nr)} = \text{nr.visibilityExpression(u)} \end{array}$$

g) Modelling example

Figure 4.28 shows an abstract example of an AIU, which incorporates visibility expressions in two of their three navigational relationships (in Fig. 4.28, at the bottom of the AIU). If the current user fulfills the *OclExpression2* condition, then she has access to the context-dependency relationship that retrieves the information from the *CLASS2* complementary class. Analogously, the fulfilment of the *OclExpression3* condition allows the access to the mixed relationship to *CLASS4*. The context relationship to *CLASS3* is always visible.

Figure 4.29 shows an abstract implementation of the AIU of Fig. 4.28 for two different users. In the version of the left side, the user fulfills the *OclExpression2* condition, having access to the context-dependency relationship and to values of the *attribute4* and *attribute5* attributes. However, she does not fulfil the *OclExpression2* condition, and the mixed relationship is not reachable for her. The right-side version shows a version that is presented to those users that fulfil *OclExpression3*, but not *OclExpression2*.



context *AIU_1* inv:

let *u*: USER = self.navigationalContext.currentUser() in

let *navrel12:* NAVREL = self.navrel-> select (*nr* | *nr*.source=CLASS1 and *nr*.target=CLASS2) in
let *unr12:* USER-NAVREL = *navrel12*.userNavrel->select (*unr* | *unr*.user = *u*) in
if ***OCLExpression2*** then ***unr12*.visibility = '*positive*'** else ***unr12*.visibility = '*negative*'** endif

let *navrel14:* NAVREL = self.navrel-> select (*nr* | *nr*.source=CLASS1 and *nr*.target=CLASS4) in
let *unr14:* USER-NAVREL = *navrel14*.userNavrel->select (*unr* | *unr*.user = *u*) in
if ***OCLExpression3*** then ***unr14*.visibility = '*positive*'** else ***unr14*.visibility = '*negative*'** endif

Figure 4.28: Visibility of Navigational Relationships

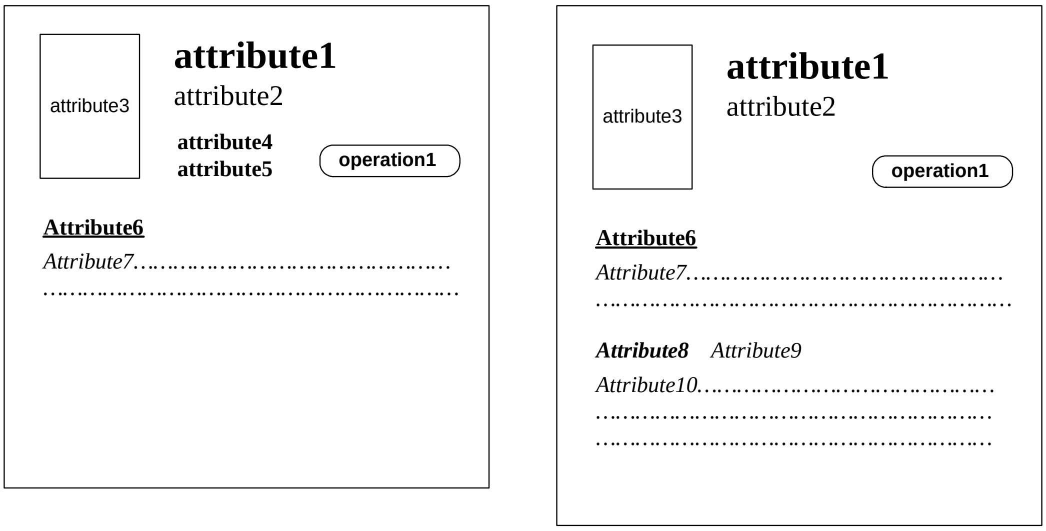
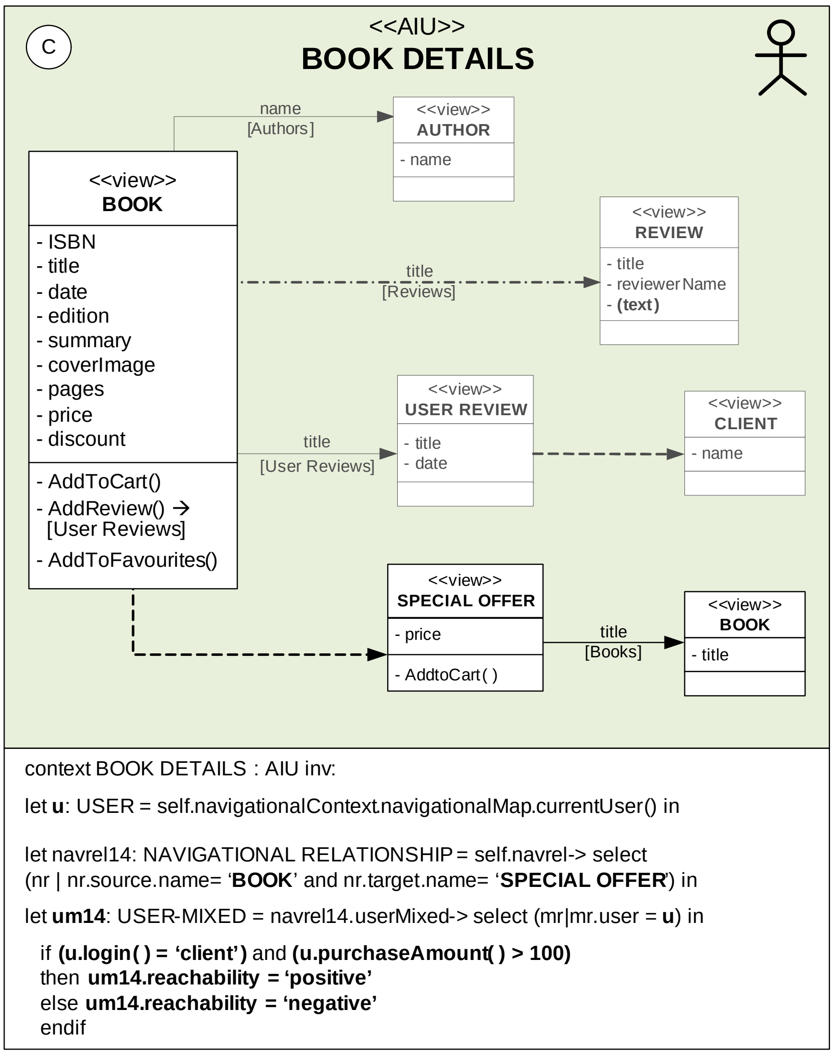


Figure 4.29: Examples of adaptive visibility of navigational relationships

h) Case study example

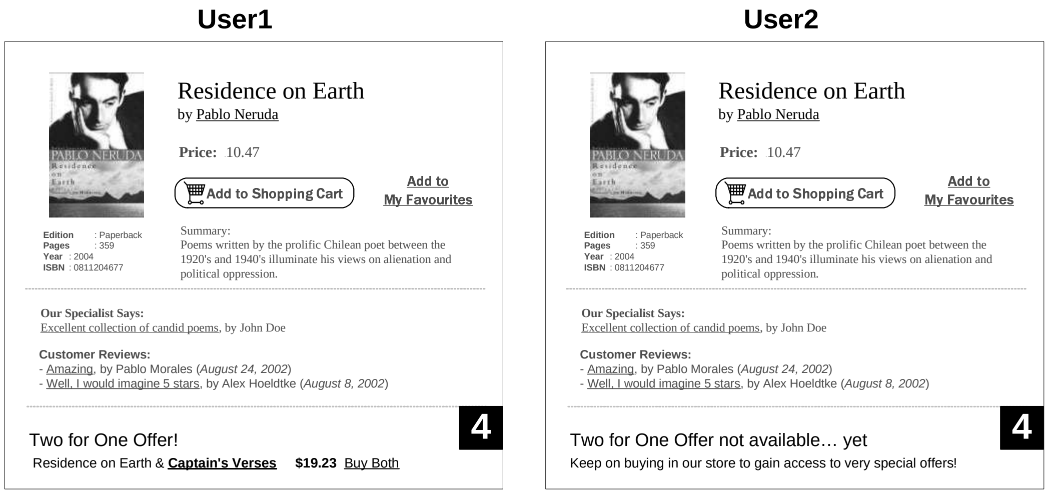
Figure 4.30 shows a modelling example of this Adaptive Primitive. In this case, the *context-dependency relationship* between the `Book` and `Special Offer` classes has an adaptive visibility expression. This constraint makes the associated information only visible to users identified as `Clients` and with a certain amount of previous purchases.

Figure 4.31 shows the implementation of the AIU of Fig. 4.30. Left-side page includes the information about an `Special Offer` associated to the currently detailed `book`, which indicates that `User1` fulfills the visibilitty expression of the relationship. On the other side, right-side page only includes a textual message that informs that `User2` has no access to associated `Special Offer` instances.



```
context BOOK DETAILS : AIU inv:
let u: USER = self.navigationalContext.navigationalMap.currentUser() in

let navrel14: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= 'BOOK' and nr.target.name= 'SPECIAL OFFER') in
let um14: USER-MIXED = navrel14.userMixed-> select (mr|mr.user = u) in
  if (u.login( ) = 'client') and (u.purchaseAmount( ) > 100)
  then um14.reachability = 'positive'
  else um14.reachability = 'negative'
  endif
```

Figure 4.30: Modelling example of Visibility of Navigational Relationships



User1

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart | Add to My Favourites

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

4

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

User2

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart | Add to My Favourites

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

4

Two for One Offer not available... yet
Keep on buying in our store to gain access to very special offers!

Figure 4.31: Implementation example of Visibility of Navigational Relationships

4.5.6 Adaptive Ranking of Navigational Relationships

a) Adaptation goal

Data and operations that complement the main information of a page may be presented in different order to different users.

b) Adaptive primitive

By means of each navigational relationship of an AIU, the system retrieves a collection of data, operations and hyperlinks with some cohesive semantics among them. These “frames of interaction” complement the information provided through the manager class. However, these frames may have different importance for different users. The order of their displaying into the Web page may be adapted, prioritizing the more important frames over the less important ones.

In order to support the modelling of this feature, we propose the assignment of ***rankings of precedence*** to the set of navigational relationships of each AIU. A ranking is a set of ordinal numbers that are assigned to each of the navigational relationships whose source is the manager class of the AIU (and through which the main fragments of the Web page are retrieved).

AIUs are provided with a *rankingExpression* characteristic, which assigns different rankings according to the fulfilment of user-based constraints. In this way, different users may visualize the different frames of information in different order.

c) Affected OOWS primitives

Navigational Relationship.

d) Supported Adaptive Technique

Frame-Based Adaptation, ***Link-Ordering***. The adaptation of the presentation of the whole frame of information that are retrieved through a navigational relationship corresponds to a *Frame-Based Adaptation* technique. Furthermore, as these frames include hyperlinks, the adaptive ordering of this frames involves an adaptive ordering of their hyperlinks, which corresponds to a *Link-Ordering technique.*

e) Specification in the OOWS Navigational Metamodel

Figure 4.32 shows the structures from the OOWS Navigational Metamodel that support the definition of adaptive ranking of navigational relationships. *User-NavRel* association-class (described in Section 4.5.5) is augmented with an integer value of an *order* attribute. The lower value of *order* a relationship has, the more important it is for a given user. A *ranking expression* is associated to each AIU. This is an OCL expression that is evaluated for each user accessing the AIU, determining the *order* values of its navigational relationships for this user. An additional constraint is defined, which indicates that different values of *order* must be assigned to each navigational relationship of an AIU for a given user.

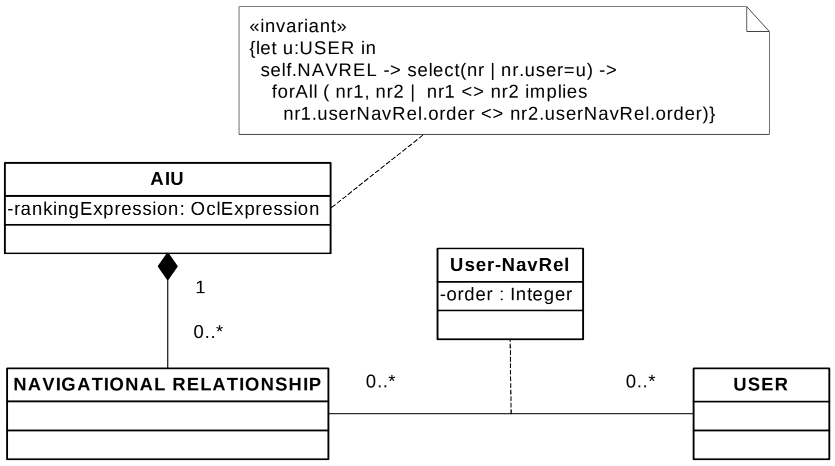

«invariant»
{let u:USER in
self.NAVREL -> select(nr | nr.user=u) ->
forAll (nr1, nr2 | nr1 <> nr2 implies
nr1.userNavRel.order <> nr2.userNavRel.order)}
AIU
-rankingExpression: OclExpression
1
0..*
User-NavRel
-order : Integer
NAVIGATIONAL RELATIONSHIP
0..*
0..*
USER

Figure 4.32: Adaptive ranking of navigational relationships in the OOWS Navigational Metamodel

f) Textual specification

The textual definition of the Adaptive Ranking of Navigational Relationships is the following:

Let

NS be the OOWS Navigational Schema of an Adaptive Web Application
USER be the whole set of users that access an instance of **NS**
NR be the set of Navigational Relationships of **NS**
$\mathcal{N}$ be the set of Natural Numbers,

then the **Adaptive Ranking of Navigational Relationships** is a function that assigns an ordinal value to each navigational relationship of an AIU according to the following definition:

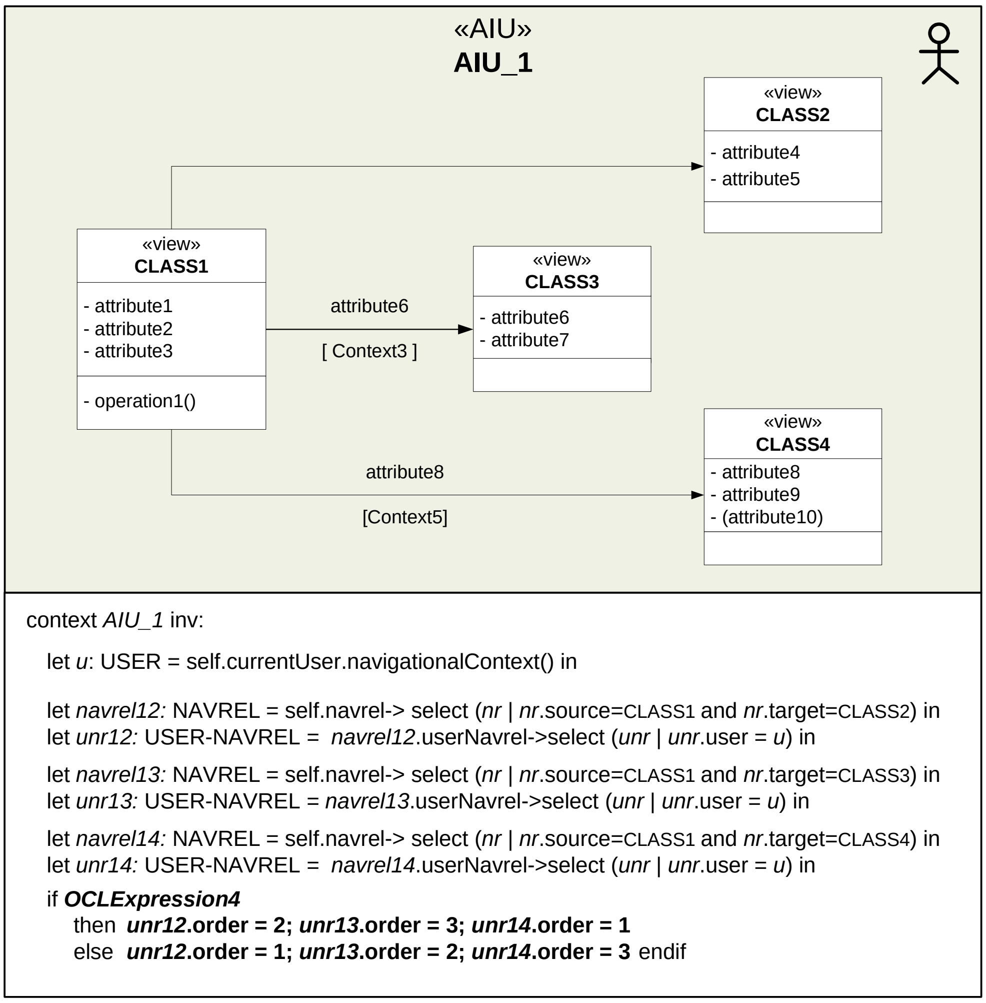
$$\begin{array}{lll} \textbf{ranking}: & \textbf{USER} \text{ x } \textbf{NR} & \longrightarrow \mathcal{N} \\ & (\text{u,nr}) & \longrightarrow \text{ranking(u,nr)} = \text{nr.AIU.orderExpression(u,nr)} \end{array}$$

This function returns a numerical value that corresponds to the ordinal number assigned to *nr* navigational relationship for *u* user. This value is obtained by evaluating the user-based constraints of the *orderExpression* element of the corresponding AIU, according to the characteristics of the *u* user.

g) Modelling example

Figure 4.33 shows an example of an AIU that includes an *orderExpression* element that affects its navigational relationships. According to the fulfilment of a user-based condition (*OclExpression4* in the figure), different *rankings of precedence* are assigned to those users, i.e., different values of the *order* attribute.

Figure 4.34 shows an abstract implementation of adaptive ranking of navigational relationships. Just like the previous examples, two versions are presented. The user that accesses the left-side version of the page fulfills the *OclExpression4* condition, so data retrieved from the relationship with a highest place in the ranking (*navrel14*, mixed relationship of the figure) are placed closer to the top of the page, while attributes from the "least important" relationship (*navrel13*) are closer to the bottom. The user that accesses the right-side version does not fulfil *OclExpression4* and the same data are displayed in different order.



```
context AIU_1 inv:
  let u: USER = self.currentUser.navigationalContext() in

  let navrel12: NAVREL = self.navrel-> select (nr | nr.source=CLASS1 and nr.target=CLASS2) in
  let unr12: USER-NAVREL =  navrel12.userNavrel->select (unr | unr.user = u) in
  let navrel13: NAVREL = self.navrel-> select (nr | nr.source=CLASS1 and nr.target=CLASS3) in
  let unr13: USER-NAVREL = navrel13.userNavrel->select (unr | unr.user = u) in
  let navrel14: NAVREL = self.navrel-> select (nr | nr.source=CLASS1 and nr.target=CLASS4) in
  let unr14: USER-NAVREL =  navrel14.userNavrel->select (unr | unr.user = u) in
  if OCLExpression4
     then  unr12.order = 2; unr13.order = 3; unr14.order = 1
     else  unr12.order = 1; unr13.order = 2; unr14.order = 3 endif
```

Figure 4.33: Adaptive Ranking of Navigational Relationships

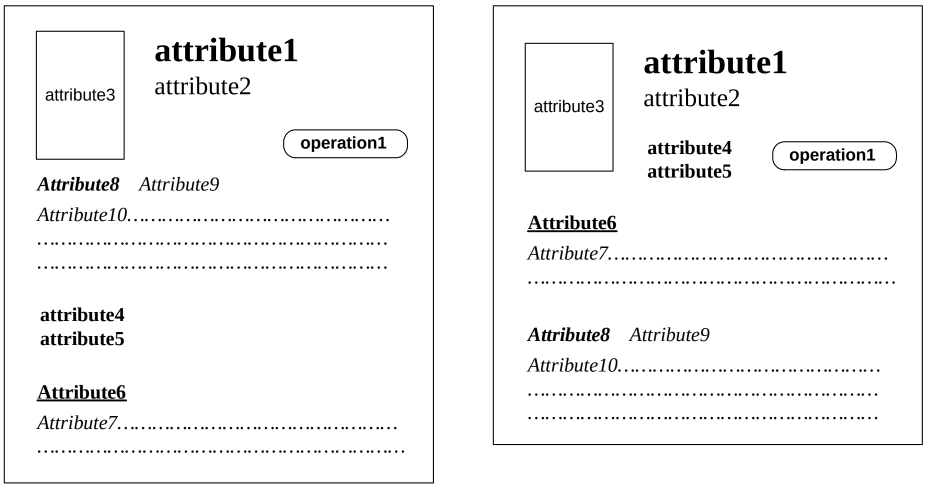
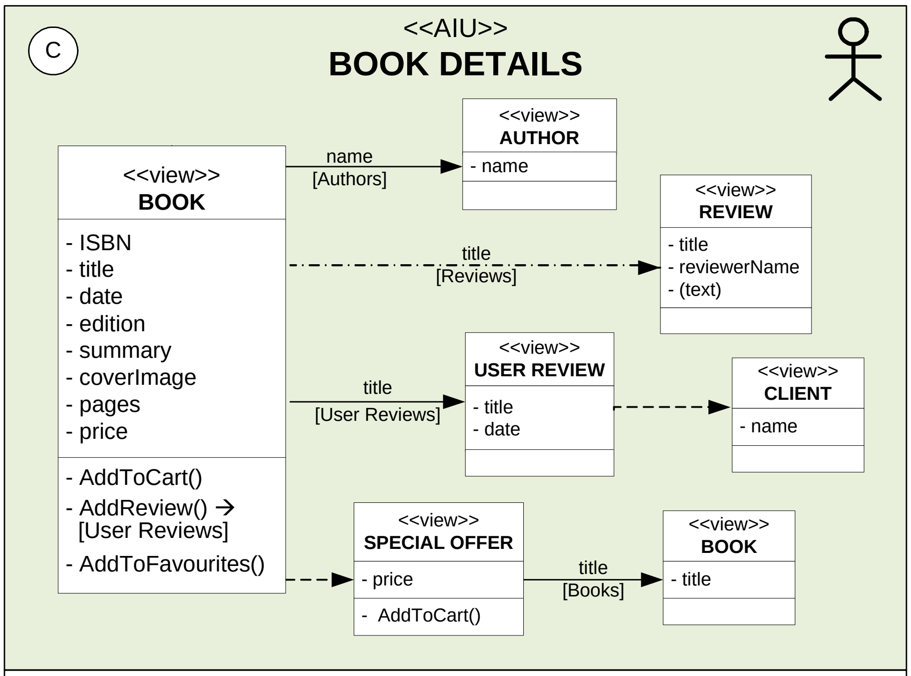


Figure 4.34: Example of Adaptive Ranking of Navigational Relationships

h) Case study example

Figure 4.35 shows the definition of two rankings of precedence for the navigational relationships that are connected to the manager class (right-side OCL expression). At the bottom of the definition, it can be seen that a given ranking is associated to `CLIENT` users with "“`habitual`” value of their corresponding `readingHabit` attribute (`Client` class, from Class Diagram of Fig. 4.5). For the rest of users, another ranking is defined.

Figure 4.36 shows the corresponding implementation. `User1` fulfills the condition of being a `CLIENT` and a `habitual` reader, so when visiting a book probably prefer to visualize a `specialized review` than information about `special offers`. On the other side, `User2` does not fulfil one of these conditions, and shopping information is privileged, locating the `Special Offer` frame closer to the top of the page.



context BOOK DETAILS : AIU inv:

let **u**: USER = self.navigationalContext.navigationalMap.currentUser() in

let navrel11: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= ‘**BOOK**’ and nr.target.name= ‘**AUTHOR**’) in

let navrel12: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= ‘**BOOK**’ and nr.target.name= ‘**REVIEW**’) in

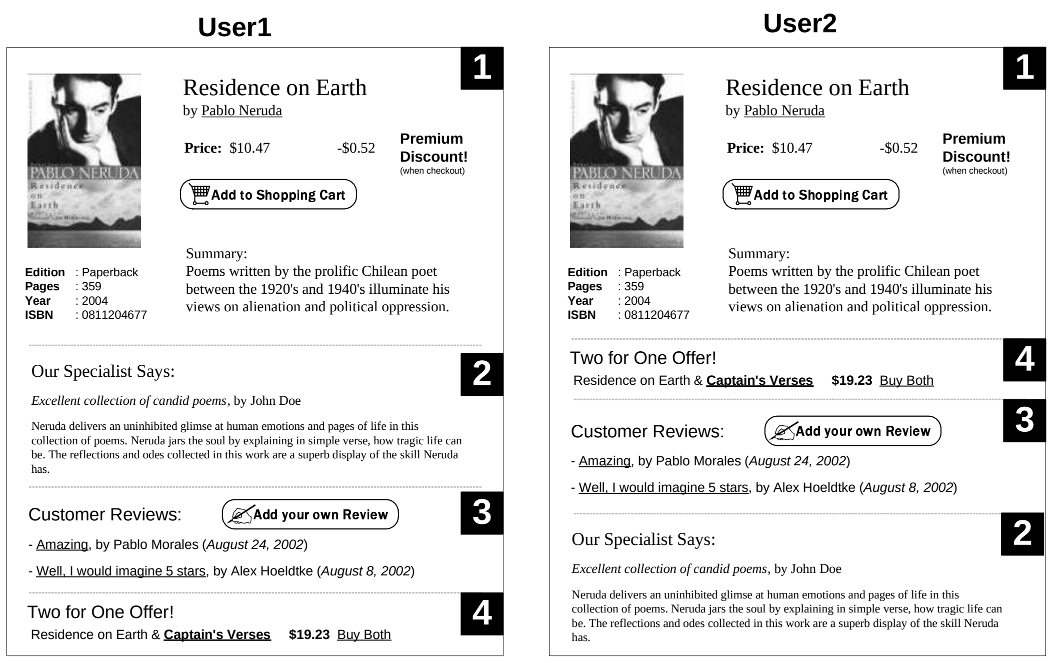
let navrel13: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= ‘**BOOK**’ and nr.target.name= ‘**USER REVIEW**’) in

let navrel14: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= ‘**BOOK**’ and nr.target.name= ‘**SPECIAL OFFER**’) in

let **unr11**: USER-NAVREL = navrel11.userNavRel-> select (nr|nr.user = **u**) in
let **unr12**: USER-NAVREL = navrel12.userNavRel-> select (nr|nr.user = **u**) in
let **unr13**: USER-NAVREL = navrel13.userNavRel-> select (nr|nr.user = **u**) in
let **unr14**: USER-NAVREL = navrel14.userNavRel-> select (nr|nr.user = **u**) in

if **(u.login( ) = ‘client’)** and **(u.client.readingHabit = ‘habitual’)**
then **unr11.order = 1; unr12.order = 2; unr13.order =3; unr14.order = 4;**
else **unr11.order = 1; unr12.order = 4; unr13.order =3; unr14.order = 2;**
endif

Figure 4.35: Modelling example of Adaptive Ranking of Navigational Relationships



User1

1

Residence on Earth
by Pablo Neruda

Price: $10.47 -$0.52 **Premium Discount!** (when checkout)

Add to Shopping Cart

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

2

Our Specialist Says:

Excellent collection of candid poems, by John Doe

Neruda delivers an uninhibited glimse at human emotions and pages of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has.

3

Customer Reviews: Add your own Review

- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

4

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

User2

1

Residence on Earth
by Pablo Neruda

Price: $10.47 -$0.52 **Premium Discount!** (when checkout)

Add to Shopping Cart

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

4

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

3

Customer Reviews: Add your own Review

- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

2

Our Specialist Says:

Excellent collection of candid poems, by John Doe

Neruda delivers an uninhibited glimse at human emotions and pages of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has.

Figure 4.36: Implementation example of Adaptive Ranking of Navigational Relationships

4.5.7 Visibility of Attributes and Operations

a) Adaptation goal

The availability of data and operations that a Web page includes may be adapted to different users.

b) Adaptive primitive

Navigational Classes are the ultimate containers of data and operations in OOWS. The information and functionality provided through these structures may be adaptively accessed by different users. To model this kind of adaptation, a ***Visibility*** property has been defined for navigational attributes and operations.

From a given navigational schema, some of these structures have a constraint that defines their visibility. This constraint, called ***visibility condition***, is expressed in terms of domain concepts and user characteristics. In this way, when a user accesses a certain context, the system checks whether she fulfills the conditions to access the corresponding attributes and operations. The visibility conditions are instantiated, giving as a result a binary value. The user is allowed to access the corresponding attribute or operation only if the resulting visibility value is *positive.* If no visibility condition has been defined, the attribute or operation is visible (it has a positive value of visibility) for all users.

c) Affected OOWS primitives

Navigational Attribute, Navigational Operation.

d) Supported Adaptive Technique

Conditional Fragments. The adaptive visibility of data items implies that the inclusion of fragments of textual and multimedia information depends on user-centered conditions or constraints. This characteristic corresponds to a *Conditional Fragments* technique. In the case of the present primitive, the notion of "information fragment" also includes the access to operations.

e) Specification in the OOWS Navigational Metamodel

Figure 4.37 shows the fragment of the OOWS Navigational Metamodel that describes the *Visibility of Navigational Attributes and Operations* primitive. Relationships between the set of application users and Navigational Attributes and Operations describe which instances from these structures are available to a given user when she accesses a given page. Each relationship includes the visibility value for the corresponding (user, attribute/operation) pair, which is the result of evaluating the OCL expressions that the *Navigational Attribute* and *Navigational Operation* metaclasses include for the corresponding user.

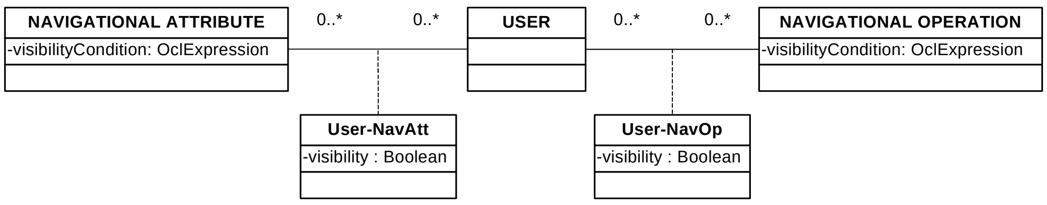


Figure 4.37: Visibility of Navigational Attributes and Operations in the OOWS navigational metamodel

f) Textual specification

The textual specification of this primitive is the following:

Let

NS be the OOWS Navigational Schema of an Adaptive Web Application
USER be the whole set of users that access an instance of **NS**
NA be the set of Navigational Attributes of **NS**
NO be the set of Navigational Operations of **NS**,

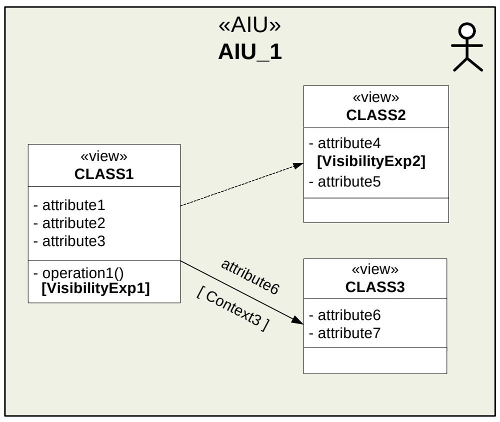
then the **Visibility of Navigational Attributes and Operations** is a function that assigns a value to *visibility* attribute from *User-NavAtt* and *User-NavOp* metaclasses, according to the following definition:

$$\begin{aligned}\textbf{visibility}\text{: } & \textbf{USER} \text{ x } (\textbf{NA} \cup \textbf{NO}) \longrightarrow \{\text{positive, negative}\}\\ & \text{(u,m)} \longrightarrow \text{visibility(nm,nc)} = \text{m.visibilityCondition(u)}\end{aligned}$$

As seen in the definition, a binary value of visibility is obtained from the evaluation of the *visibilityCondition* expression of a given navigational attribute or operation, for a given *u* user.

g) Modelling example

The abstract example of Fig. 4.38 shows how this primitive is included in an OOWS Navigational Schema. Two visibility expressions are defined: *VisibilityExp1* and *VisibilityExp2*, corresponding to the *operation1()* operation and to the *attribute4* attribute, respectively. If users fulfil the respective user-centered constraints (*OclExpression1* and *OclExpression2*), they gain access to the mentioned structures.



VisibilityExp1

```
context CLASS1: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.navigationalMap.currentUser() in
let  u_op1: USER-NAVOP = self.userNavOp-> select
(unop | unop.user = u and unop.navigationalOperation.name = ‘operation1’) in
  if  OCLExpression1  then u_op1.visibility = ‘positive’
  else u_op1.visibility = ‘negative’   endif
```

VisibilityExp2

```
context CLASS2: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.navigationalMap.currentUser() in
let  u_att4: USER-NAVATT = self.userNavAtt-> select
(unatt | unatt.user = u and unatt.navigationalAttribute.name = ‘attribute4’) in
  if  OCLExpression2 then u_att4.visibility = ‘positive’
  else u_att4.visibility = ‘negative’ endif
```

Figure 4.38: Abstract modelling of Visibility of Navigational Attributes and Operations

Figure 4.39 shows an abstract implementation of the AIU of Fig. 4.38. The user that accesses the left-side page only fulfils the *OclExpression1* constraint, so she can execute the *operation1* operation, but she can not visualize the value of *attribute4*. In the right-side page, the user only fulfils *OclExpression2*, and only has access to the attribute value.

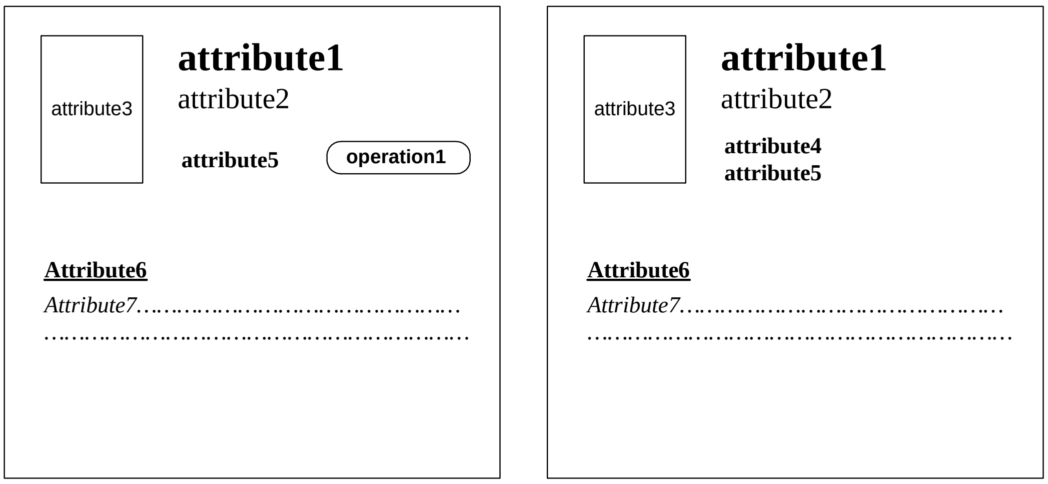
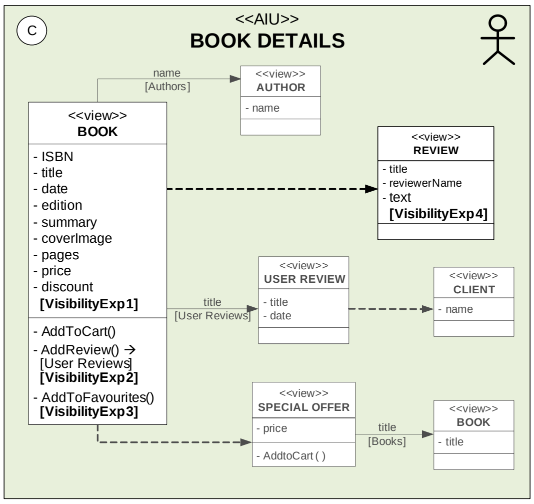


Figure 4.39: Abstract implementation of Visibility of Navigational Attributes and Operations

h) Case study example

Figure 4.40 shows a modelling example of this primitive in the described case study. In this case, visibility expressions have been defined for four elements: `Book.discount` attribute is only visible to `Client` users that have been paid a `premium` fee; the `Book.AddReview()` operation is visible to `Clients` that have made at least one `purchase` through the application; the `Book.AddToFavourites()` operation and the `Review.text` attribute are visible only for `Clients`. All these constraints are included in their respective *visibility* expressions (at the right of Fig. 4.40, bold expressions).



VisibilityExp 1

```
context BOOK: NAVIGATIONAL CLASS inv:

let u: USER =   self.AIU.navigationalContext.navigationalMap.currentUser() in
let u_ds: USER-NAVATT = self.userNavAtt-> select (unatt | unatt.user = u and
          unatt .navigationalAttribute.name = 'discount') in

if ((u.login( ) = 'client') and (u.client.payPremium( )))
then u_ds.visibility = 'positive' else u_ds.visibility = 'negative' endif
```

VisibilityExp 2

```
context BOOK: NAVIGATIONAL CLASS inv:

let u: USER =   self.AIU.navigationalContext.navigationalMap.currentUser() in
let u_ar: USER-NAVOP = self.userNavOp-> select (unop | unop.user = u and
          unop .navigationalOperation.name = 'AddReview') in

if  ((u.login( ) = 'client') and (u.client.purchase.notEmpty( ))
then u_ar.visibility = 'positive' else u_ar.visibility = 'negative' endif
```

VisibilityExp 3

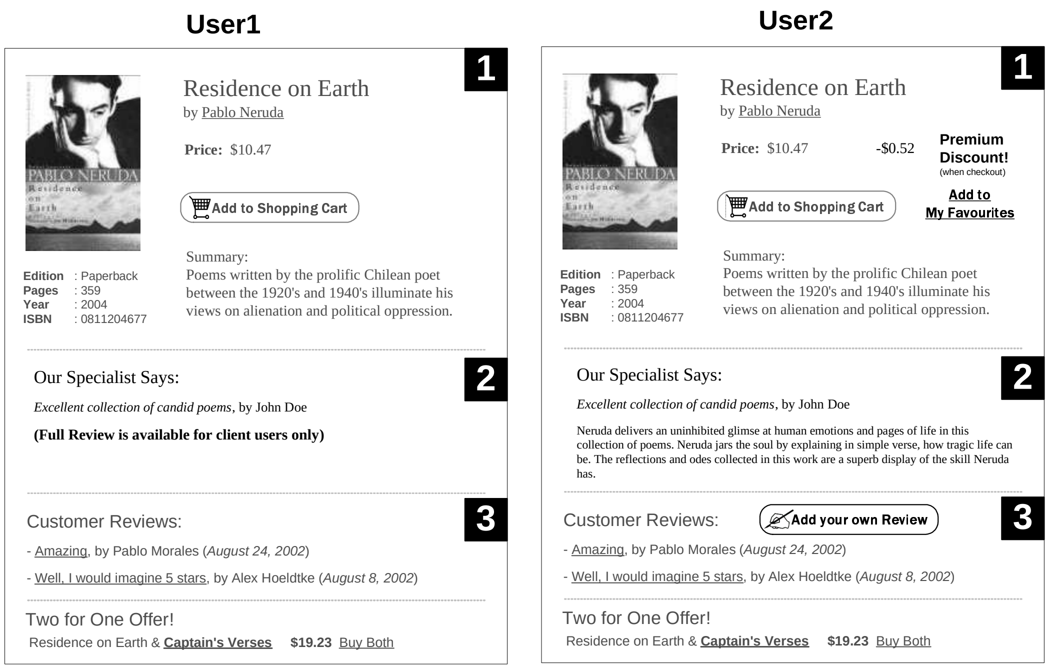
```
context BOOK: NAVIGATIONAL CLASS inv:

let u: USER =   self.AIU.navigationalContext.navigationalMap.currentUser() in
let u_atf: USER-NAVOP = self.userNavOp-> select (unop | unop.user = u and
          unop .navigationalOperation.name = 'AddToFavourites') in

if (u.login( ) = 'client')
then u_atf.visibility = 'positive' else u_atf.visibility = 'negative' endif
```

VisibilityExp 4

```
context REVIEW: NAVIGATIONAL CLASS inv:

let u: USER =   self.AIU.navigationalContext.navigationalMap.currentUser() in
let u_tx: USER-NAVATT = self.userNavAtt-> select (unatt | unatt.user = u and
          unatt .navigationalAttribute.name = 'text') in

if (u.login( ) = 'client')
then u_tx.visibility = 'positive' else u_tx.visibility = 'negative' endif
```

Figure 4.40: Modelling example of Adaptive Visibility of Navigational Attributes and Operations

Figure 4.41 shows a concrete implementation of this primitive, by means of two page-segments corresponding to the AIU of Fig. 4.40. Frames 1 and 3 shows the adaptive visibility of the elements from `Book` class, while Frame 2 shows the case of the `text` attribute from the `Review` class. `User1` does not fulfil any of the visibility expressions, so corresponding attributes and operations are not visible in the left-side segment. In the case of text attribute, the system informs of this situation through a textual message; the rest of elements are simply hidden to this user. On the other side, `User2` fulfills all the cited conditions, so values of the corresponding attributes and the anchors to execute the operations are visible.



User1

Residence on Earth
by Pablo Neruda

Price: $10.47

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Our Specialist Says:
Excellent collection of candid poems, by John Doe
(Full Review is available for client users only)

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

User2

Residence on Earth
by Pablo Neruda

Price: $10.47 -$0.52 **Premium Discount!** (when checkout)

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Our Specialist Says:
Excellent collection of candid poems, by John Doe
Neruda delivers an uninhibited glimse at human emotions and pages of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has.

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

Figure 4.41: Implementation example of Adaptive Visibility of Navigational Attributes and Operations

4.6 Conclusions

This chapter presented the extensions proposed to the Conceptual Modelling phase of the OOWS method, in order to support the modelling of Adaptive Web Applications: (a) the introduction of Adaptive Primitives into the OOWS Navigational Model; and (b) the definition of a User Modelling strategy, by extending the OO-Method Class Diagram. Due to the high expressivity and flexibility of the conceptual models of OOWS, the introduction of new primitives did not involve a major modification of the previously existing ones, but the domain of supported Web applications is importantly increased.

Each Adaptive Primitive is associated to the implementation of an adaptive technique. However, some of the techniques proposed by the Adaptive Hypermedia community [10] present difficulties in the definition of a corresponding conceptual primitive that preserves the essence of the OOWS Navigational Model. In particular, the following two techniques are not considered in the present proposal:

- ***Link Generation***: techniques of Link Generation allows incorporating new links to the application, not previously considered in the application authoring phase, according to the navigational behaviour of the user. In the *Authoring-in-the-Large* phase of navigational modelling (Section A.1), OOWS distinguishes two kinds of navigational links: *sequential links* and *exploration links*. *Sequential Links* represent those navigational paths with a semantic associated to the linked pages. These links are specified as views of structural relationships between conceptual objects. This means that every instance of a sequential link included in the application has been defined because there is a corresponding and already existing instance of a semantic relationship defined in the Class Diagram. For this reason, the generation of new sequential links requires the previous creation of the corresponding new instances of those structural relationships, through the execution of operations. This may be triggered by conditions defined in terms of characteristics and navigational actions of the user. However, the definition of the corresponding links in the OOWS navigational schema remains unmodified. In consequence, this proposal does support the generation of new links, not as an explicit modelling strategy in the navigational schema, but providing conceptual mechanisms to describe navigational views of new structural relationships as new application links.

In the other side, *Exploration Links* describe intentional changes of tasks made by the user. Generation of this kind of links could be justified by a reiterative access to a given task that is not reachable from any page of the application. However, we consider that such an important modification in the navigational structure should be a consequence of high level modifications in the application's requirements instead of a being a result of a given navigational pattern of a specific user. This is the reason that the generation of this kind of links is excluded from the present proposal.

- ***Map Adaptation***: techniques for Map Adaptation provides ways to adapt the structure or topology of the navigational map of the application, according to the user characteristics.

  As described in previous sections, OOWS defines a set of navigational maps, each one corresponding to a different user group. A navigational map describes the possible navigational paths that respective users are allowed to follow, and *it can not be modified*.

  Some adaptive structures introduced in the previous section allow restricting the access to a given navigational page, for instance, reachability of navigational contexts (Section 4.5.1), visibility of navigational relationships (Section 4.5.5), or adaptive population filter (Section 4.5.2). In this way, the actual navigational possibilities for a given user may correspond to a subset of the possibilities described in her corresponding navigational map. From the perspective of the present proposal, however, this fact *does not entail an adaptation of the hyperstructure*, but only an adaptation of instances of represented navigational links. The maps remain unmodified and valid for any user of the corresponding stereotype.

The analysis of new ways to incorporate -either totally or partially- these unsupported adaptive techniques in the extended OOWS method, will be a source of further work for the next future.

The introduced Adaptive Primitives permit to model a set of Adaptive Methods, traditionally considered by researchers on Adaptive Hypermedia [10] in the construction of adaptive systems. With the present proposal, these methods may be specified at a high abstraction level, regardless of implementation aspects In the next chapter, the use of the new primitives in the modelling of Adaptive Methods is described.

Chapter 5

Supporting the modelling of Adaptive Methods in OOWS

5.1 Introduction

Adaptive Methods are high level strategies that support the browsing experience of users, by means of the implementation of adaptive features, that is to say, by adapting the presentation of contents and hyperlinks to the individual characteristics of the user.

This chapter shows how the present proposal gives support to the modelling of Adaptive Methods, in the context of the OOWS approach. Based on the Adaptive Primitives introduced in the previous chapter, a set of modelling strategies has been defined to model different Adaptive Methods through the OOWS Navigational Model. Each strategy is defined in terms of one or more Adaptive Primitives. In this way, each Adaptive Method is implemented through one or more Adaptive Techniques.

These modelling strategies complement the introduction of Adaptive Primitives into the OOWS Navigational Model. They are intended to provide developers with guidelines to facilitate the use of these primitives in the modelling of Adaptive Web Applications, by means of high level descriptions of adaptive methods that are frequently used in hypermedia systems.

Adaptive Hypermedia community has distinguished two categories of Adaptive Methods [10], according to the type of adaptation that they support: ***Adaptive Presentation Methods***, which facilitate the access to the most relevant information, by adapting the presentation of the available data items and operations to the characteristics of the current user; and ***Adaptive Navigation Methods***, which help users to locate themselves into the application hyperspace and to guide them in finding the best alternatives to navigate, in order to fulfil their particular interaction goals.

In the following two sections of this chapter, we present the modelling strategies that are defined for both types of Adaptive Methods. The first section introduces the strategies to model *Adaptive Presentation* methods, while the second section introduces those corresponding to *Adaptive Navigation Methods.* In both sections, modelling strategies are described through the following characteristics: (a) the structures that must be included in the Class Diagram to support the navigational description of the corresponding method; (b) the navigational modelling strategy by using one of the introduced OOWS Adaptive Primitives; and (c) the Adaptive Technique that implements the strategy. Furthermore, they are illustrated by means of a case study, corresponding to an adaptive online bookstore.

The third section of this chapter describes the methods that are not supported by this proposal, along with some guidelines to their further incorporation. Finally, the last section presents some conclusions of this chapter.

5.2 Adaptive Presentation Methods in OOWS

Adaptive Presentation Methods support the user-system interaction through the adaptation of the information that is shown in each page of the application. These methods face the *information overload* problem by privileging the access to the data and operations that best fit the characteristics of different users, filtering those items that do not contribute to their needs. For instance, information may be added to a page to compensate for knowledge the user does not have yet; a given description may be provided through different versions; a technical term may be replaced by a known non-technical term; etc.

Table of Fig. 5.1 summarizes the modelling proposal of Adaptive Presentation in OOWS. Rows of the table show the *Adaptive Presentation Methods* that this proposal supports, while columns contain the different *Adaptive Presentation Techniques* that are used to fulfil the adaptation goals of each method. Each cell contains the Adaptive Primitives introduced in the OOWS Navigational Model that support to modelling of the corresponding method, leading to the implementation of the corresponding adaptive technique.

		Adaptive Presentation Techniques			
		Conditional Fragments	*Strechtext*	*Page Variants*	*Frame-Based Adaptation*
Methods for Adaptive Presentation	*Additional Explanations*	Adaptive Visibility of Nav. Attributes and Operations	Mixed Navigational Relationship		Adaptive Visibility of Navigational Relationships
	Prerequisite Explanations	Adaptive Visibility of Nav. Attributes and Operations	Mixed Navigational Relationship		Adaptive Visibility of Navigational Relationships
	Comparative Explanations	Adaptive Population Filter			
	Explanation Variants	Adaptive Visibility of Nav. Attributes and Operations		Reachability of Navigational Contexts	Adaptive Population Filter
	Sorting				Adaptive Ranking of Navigational Relationships

Figure 5.1: Adaptive Presentation Methods and Techniques, with the OOWS Adaptive Primitives that support their modelling

The following subsections describe the modelling of each method through the Adaptive Primitives included in its corresponding cells.

5.2.1 Additional Explanations

This method provides users with adaptive access to additional explanations about an object, complementing a common presentation of related contents. We propose three options to model this method in OOWS, based on different primitives from those introduced in Chapter 4. In each case, the used navigational primitive contains a condition imposed to the application users, which allows classifying them into one of two subsets: users enabled to access the additional explanations, and those that are not allowed to do it. According to their interactions with the system, users may be moved from one subset to another at run-time, changing their access level to the additional explanations. The three proposed modelling options specify the additional explanations through: (1) *navigational attributes*, by adapting their visibility; (2) attributes that are reachable through *mixed relationships*, by adapting the behaviour of the corresponding relationship; and (3) *complementary classes*, by adapting the visibility of navigational relationships that lead to them.

Option 1: Additional Explanations as Navigational Attributes

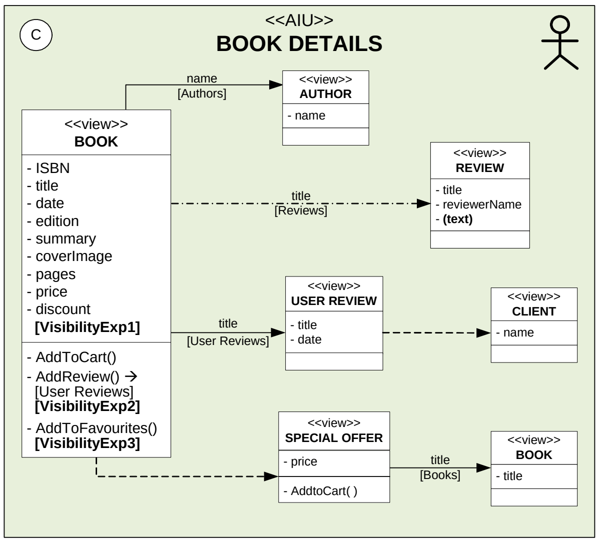
a) **Class Diagram:** In the *Class Diagram*, additional explanations about a concept are modelled through one or more attributes and/or operations of the class that represents that concept.

b) **OOWS Navigational Schema:** In the AIU where the explanation is provided, a navigational class shows information about the concept to be explained. The adaptive access to the corresponding navigational attributes/operations is described by means of the ***Visibility of Navigational Attributes/Operations*** primitive (see Section 4.5.7). The corresponding *visibility expressions* contain the restrictions that users must fulfil in order to gain access to the additional explanations.

c) **Adaptive Technique**: The *Adaptive Technique* that is used in this case is the ***Conditional Fragments*** technique. The possibility of including the access to operations as part of the explanations extends the traditional implementations of this technique, which are mostly focused on adapting the access to data items only.

d) **Modelling example:** Figure 5.2 shows an example of an AIU, which considers the three modelling strategies for *Additional Explanations* method. *Option 1* is illustrated through the definition of *Adaptive Visibility* expressions for the `discount` attribute and the `AddReview()` and `AddToFavourites()` operations of the `Book` manager class. These OCL expressions are detailed in *Frame 1* of Fig. 5.2. In all of them, the first sentence identifies the user that currently accesses the context of the corresponding AIU. The following sentence identifies the association instance between the current user (called u) and the adaptive attribute or operation, in order to save the visibility value of the structure for the u user. The last (bold) sentence describes the visibility constraint. In the first case, only users that are logged in as `CLIENT` users and that have paid a `premium` fee can visualize the value of the `discount` attribute; only `CLIENT` users that have made a previous purchase in the store can execute the `AddReview()` operation; and `CLIENT` users are the only allowed to execute `AddToFavourites()`.

e) **Implementation example:** Figure 5.3 shows an implementation of the `Book Details` AIU, accessed by two different users: `User1`, which accesses the left-side version of the page, and `User2`, accessing the right-side page. Comparing both versions, we can infer that `User1` does not fulfil any of the described *visibility* conditions, while `User2` fulfills all of them. Therefore, `User2` can visualize the `discount` associated to the currently presented `Book`, as seen in *Frame 1* of the right-side page. Furthermore, `User2` can mark this `Book` as favourite, by executing the `AddToFavourite()` operation (*Frame 1*) and submit her own review of it (`AddReview()`, *Frame 3*).



1

VisibilityExp1

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in
let  u_ds: USER-NAVATT = self.userNavAtt-> select
(unatt | unatt.user = u and unatt.navigationalAttribute.
 name = 'discount') in
 if (u.login( ) = 'client' and u.client.payPremium( ))
 then u_ds.visibility = 'positive'
 else u_ds.visibility = 'negative'
 endif
```

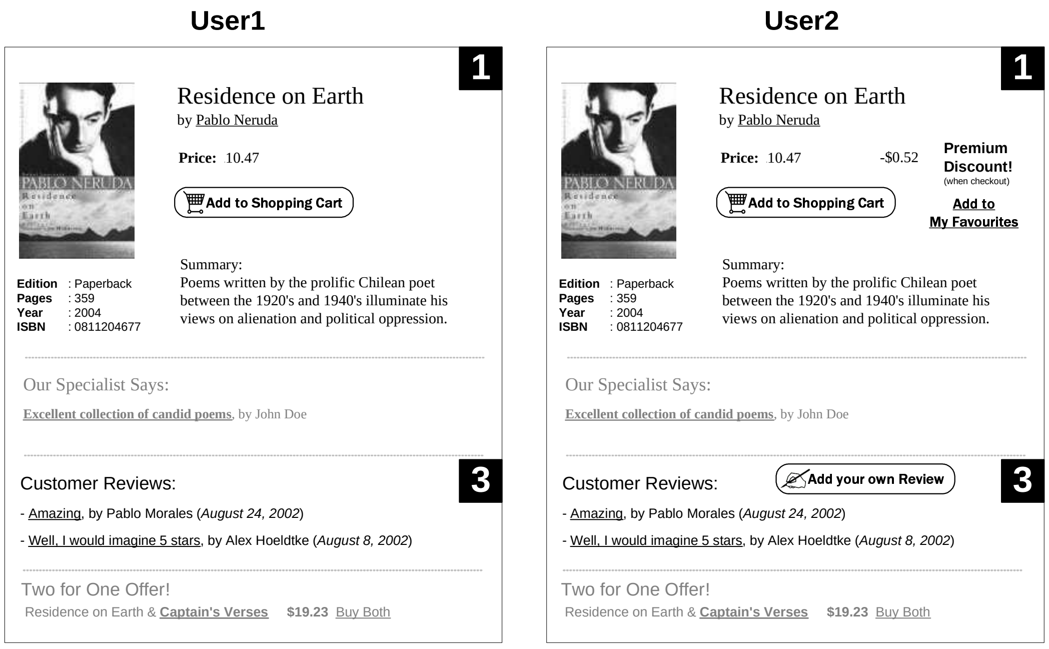
VisibilityExp2

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in
let  u_ar: USER-NAVOP = self.userNavop-> select
(unop | unop.user = u and unop.navigationalOperation.
 name = 'AddReview') in
 if  (u.login( ) = 'client' and  u.client.purchase.notEmpty( ))
     then u_ar.visibility = 'positive'
     else u_ar.visibility = 'negative'
     endif
```

VisibilityExp3

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in
let  u_atf: USER-NAVOP = self.userNavop-> select
(unop | unop.user = u and unop.navigationalOperation.
 name = 'AddToFavourites') in
if (u.login( ) = 'client')
    then u_atf.visibility = 'positive'
    else u_atf.visibility = 'negative'
    endif
```

Figure 5.2: Modelling example of *Additional Explanations* method, Option 1



User1

1

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

3

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

User2

1

Residence on Earth
by Pablo Neruda

Price: 10.47 -$0.52 **Premium Discount!** (when checkout)

Add to Shopping Cart

Add to My Favourites

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

3

Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

Figure 5.3: Implementation example of *Additional Explanations* method, Option 1

Option 2: Additional Explanations through a Mixed Relationship

a) **Class Diagram:** Explanations about a concept are modelled through attributes of a class that is associated to the class corresponding to that concept. Two versions of an explanation are included: one that contains the common version of the explanation, which any user can access; and the other one containing the additional explanation, whose access is adaptive. Only the additional version is mandatory.

b) **OOWS Navigational Schema:** The association between the two mentioned classes is modelled through a ***Mixed Navigational Relationship*** (Section 4.5.4), between the corresponding navigational classes. The common explanation is modelled as a normal navigational attribute of its target class, while the adaptive, additional explanation is expressed through the *conditional attribute* of that class.

   The *behaviour condition* of the relationship distinguishes two relevant groups of users, according to the way that they can access the additional explanation.

   Users for which the relationship has a *“context”* value of the *behaviour* property only can visualize the common explanation, with no access to the *conditional attribute*. To access the additional explanation, these users must navigate to another page, by activating the *link attribute* (anchor) of the relationship.

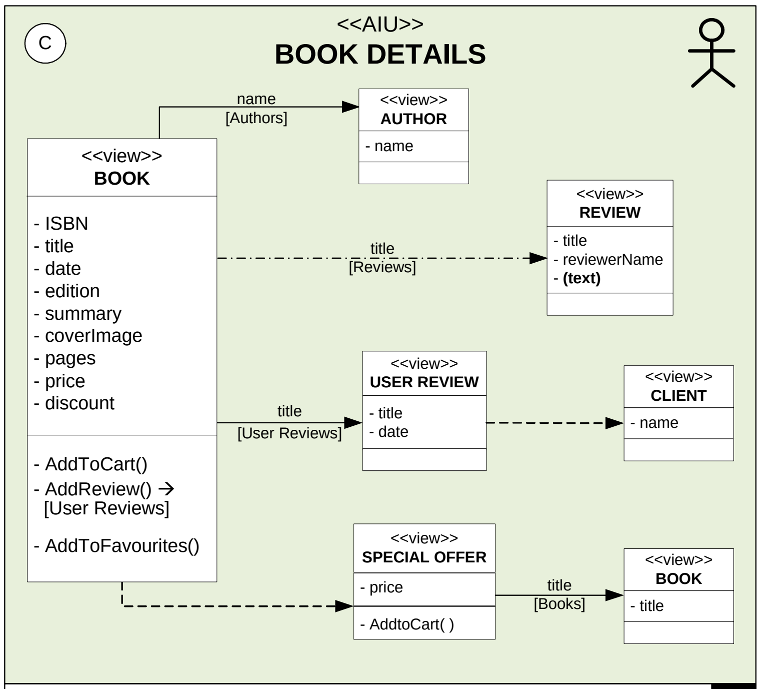
   On the contrary, those users for which the relationship has a *“context-dependency”* behaviour have immediate access to the common and to the additional explanations in the same page.

c) **Adaptive Technique:** This strategy implements a ***Strechtext*** technique. The conditions that define which users have immediate access to the explanation may be only based on static user characteristics, or also consider their navigational behaviour. In this last case, for instance, the number of *clicks* to access additional explanations (*“context”* behaviour) may be used to determine the need to provide that user with the full explanation in the same context (*“context-dependency”* behaviour).

d) **Modelling example:** The modelling example of Fig. 5.4 includes a *mixed relationship* between the `Book` and `Review` navigational classes.

An *Additional Explanation* about the specialized `Review` is represented through the `text` attribute of the `Review` class. In the OCL expression at the bottom of the AIU, the adaptive behaviour of this relationship is detailed in *Frame 2*. The first sentence identifies the mixed relationship through the navigational classes that it associates. The following sentence identifies the association instance between the current *u* user and the mixed relationship (called *navrel12*), in order to store the behaviour of the relationship for that user. In bold, the last sentence describes the behaviour expression of *navrel12*: if the current user has been logged in as a `CLIENT`, visited information of at least 20 `books` and navigated to the full `review` in at least an 80% of them, then the mixed relationship has a *context dependency* behaviour, and the value of the `text` attribute is shown together with the rest of the data of the corresponding `book`; on the contrary, the mixed relationship has a context behaviour, the `book` attribute is not visible and it is replaced by a link to another context, which provides detailed information of the corresponding specialized `review`.

e) **Implementation example:** In Fig. 5.5, the implementation of this mixed relationship is illustrated in *Frame 2* of both pages. In the left-side page, `User1` does not fulfil the imposed condition, so the additional explanation is not visible, and a link to the full version of the review is provided. On the contrary, `User2` has been identified as a `CLIENT`, and in their multiple visits she has activates the link to the full version of the review in at least the 80% of the times. For this reason, the system infers that she is interested in this kind of information and then it provides her with immediate access to the explanation, as seen in *Frame 2* of the right-side page of Fig. 5.5.



2

context BOOK DETAILS : AIU inv:

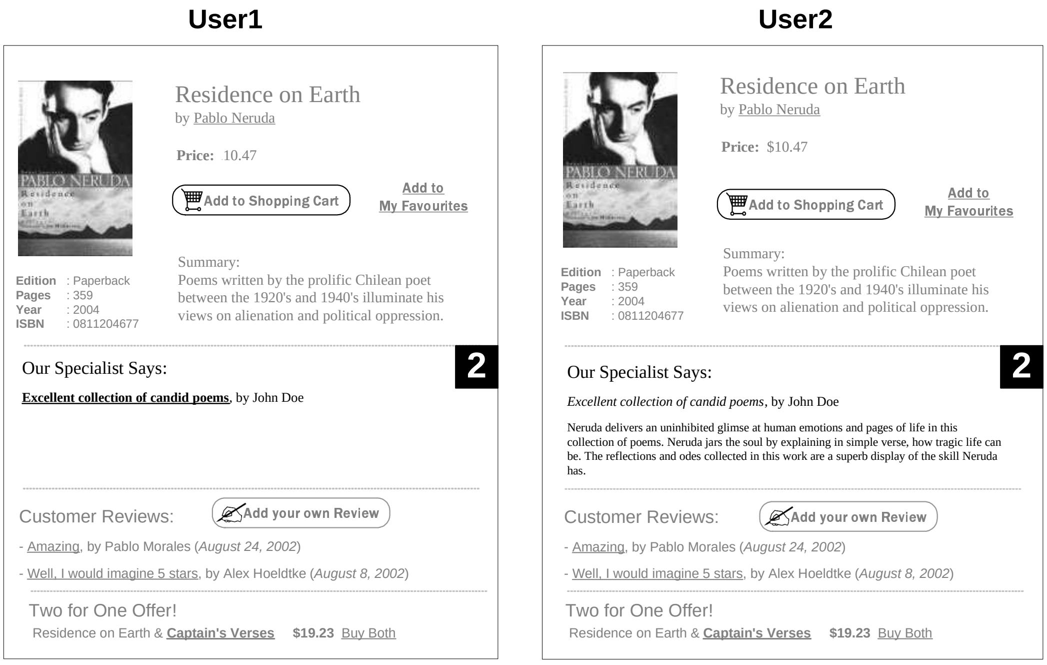
let **u**: USER = self.navigationalContext.navigationalMap.currentUser() in

let navrel12: NAVIGATIONAL RELATIONSHIP = self.navrel->
select (nr|nr.source.name = '**BOOK**' and nr.target.name = '**REVIEW**') in

let **um12**: USER-MIXED = navrel12.userMixed-> select (mr|mr.user = **u**) in

if **(u.login( ) = 'client') and (u.reviewsRate( ) >= 80) and (u.visitedBooks >= 20)**
then **um12.behaviour = 'contextDependency'**
else **um12.behaviour = 'context'**
endif

Figure 5.4: Modelling example of *Additional Explanations* method, Option 2



User1

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart

Add to
My Favourites

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

2

Our Specialist Says:

Excellent collection of candid poems, by John Doe

Customer Reviews: Add your own Review

- Amazing, by Pablo Morales (*August 24, 2002*)

- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

User2

Residence on Earth
by Pablo Neruda

Price: $10.47

Add to Shopping Cart

Add to
My Favourites

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

2

Our Specialist Says:

Excellent collection of candid poems, by John Doe

Neruda delivers an uninhibited glimse at human emotions and pages of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has.

Customer Reviews: Add your own Review

- Amazing, by Pablo Morales (*August 24, 2002*)

- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

Figure 5.5: Implementation example of *Additional Explanations* method, Option 2

Option 3: Additional Explanations as a cluster of information

a) **Class Diagram:** Complex additional explanations about a concept are modelled through a set of attributes and operations of a class, which is associated to the class corresponding to that concept.

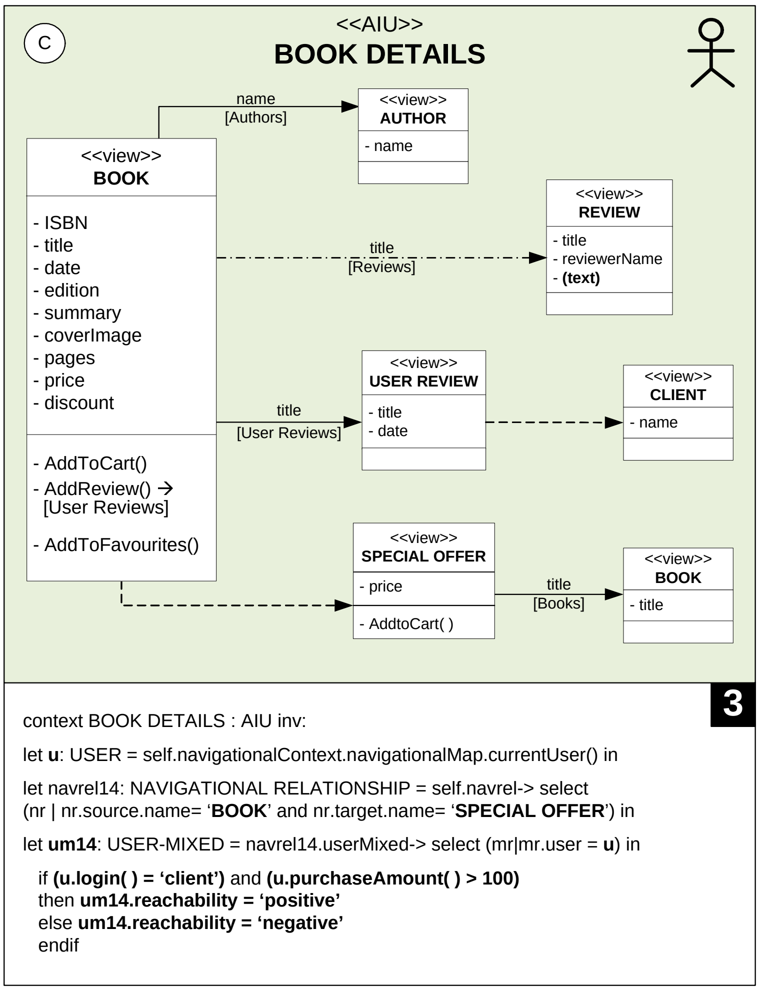
b) **OOWS Navigational Schema:** This schema includes a navigational class that corresponds to the concept to be explained, and an associated complementary class, which contains the additional explanations. These explanations comprise the complete frame of data, operations and hyperlinks that are defined through that complementary class.

   The adaptive access to the explanations is achieved by adapting the access to the navigational relationship that connects the mentioned classes, through the ***Visibility of Navigational Relationship*** primitive (see Section 4.5.5). This relationship will have a *positive* visibility for users that fulfil the corresponding *visibility expression*. For the rest of users, these explanations are not provided, so the navigational relationship will have a *negative* value of visibility.

c) **Adaptive Technique:** This strategy implements an ***Frame-Based Adaptation*** technique, in which a set of cohesive information is hidden to certain users and made available to others.

d) **Modelling example:** In Fig. 5.6, the modelled AIU illustrates this option through the context-dependency relationship between the `Book` and `Special Offer` classes. In this case, the *Additional Explanation* corresponds to the entire complementary information that can be retrieved through this relationship. In the OCL expression at the bottom of the AIU, the adaptive visibility of this relationship is detailed in *Frame 3*. After the identification of the relationship (*navrel14*) and its association with the current *u* user, the last sentence assigns a *positive* visibility value to those users that have been logged in as `CLIENT` users, visited information of at least 20 `books` and navigated to the full `review` in at least an 80% of them; on the contrary, the relationship has a *negative* visibility.

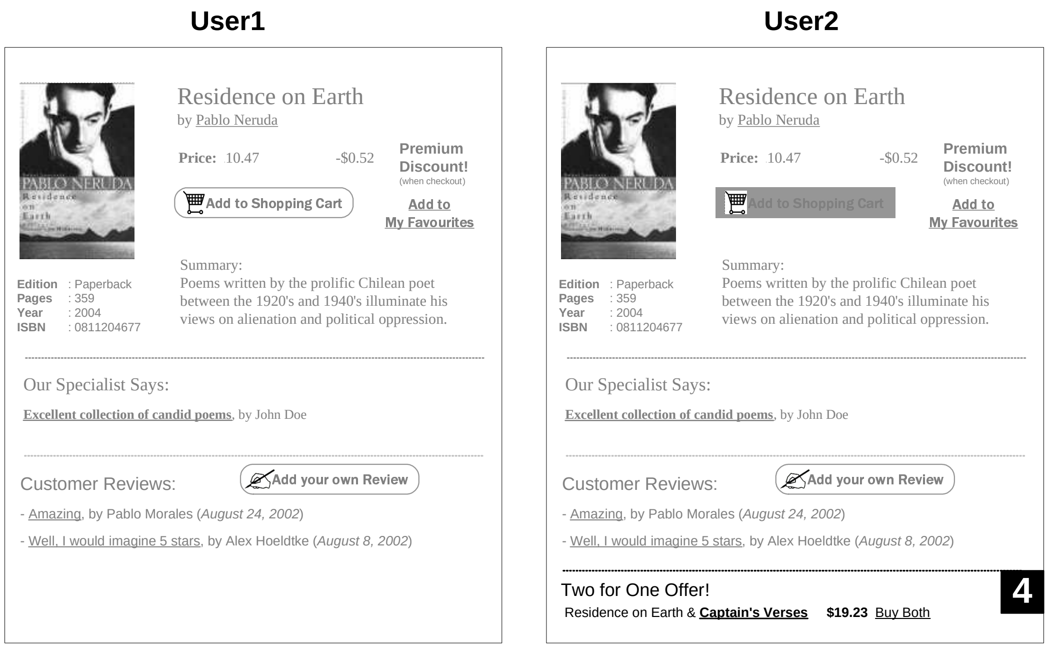
e) **Implementation example:** In the implementation of Fig. 5.7, `User1` does not fulfil the mentioned condition, so she can not access any information of `special offers` associated to the shown `book`; `User2`



```
context BOOK DETAILS : AIU inv:
let u: USER = self.navigationalContext.navigationalMap.currentUser() in
let navrel14: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= 'BOOK' and nr.target.name= 'SPECIAL OFFER') in
let um14: USER-MIXED = navrel14.userMixed-> select (mr|mr.user = u) in
  if (u.login( ) = 'client') and (u.purchaseAmount( ) > 100)
  then um14.reachability = 'positive'
  else um14.reachability = 'negative'
  endif
```

Figure 5.6: Modelling example of *Additional Explanations* method, Option 3

does fulfil the visibility condition and thus has access to information of `special offers`, as shown in *Frame 4* of the right-side page. This information contains textual data (name of the offered books), operations (add to cart operation through the `Buy Both` anchor) and hyperlinks (the name of the second book).


User1

Residence on Earth
by Pablo Neruda
Price: 10.47 -$0.52 Premium Discount! (when checkout)
Add to Shopping Cart
Add to My Favourites

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

User2

Residence on Earth
by Pablo Neruda
Price: 10.47 -$0.52 Premium Discount! (when checkout)
Add to Shopping Cart
Add to My Favourites

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer! 4
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

Figure 5.7: Implementation example of *Additional Explanations* method, Option 3

5.2.2 Prerequisite Explanations

This method adapts the presentation of explanations about a concept, enabling its access only to users that have previously reviewed the information of a set of other concepts, which are prerequisites of the first one. This method promotes a complete understanding of explanations about a concept, when a previous knowledge level of other related concepts is needed. The adaptive system checks if users reach this knowledge level before providing them with access to the explanation.

This method is a variant of the *Additional Explanation* method, so we propose to adopt the same three modelling options of that case, implementing identical adaptive techniques. For all of these cases, however, we must incorporate some additional concepts and modelling considerations in the corresponding conceptual schemas:

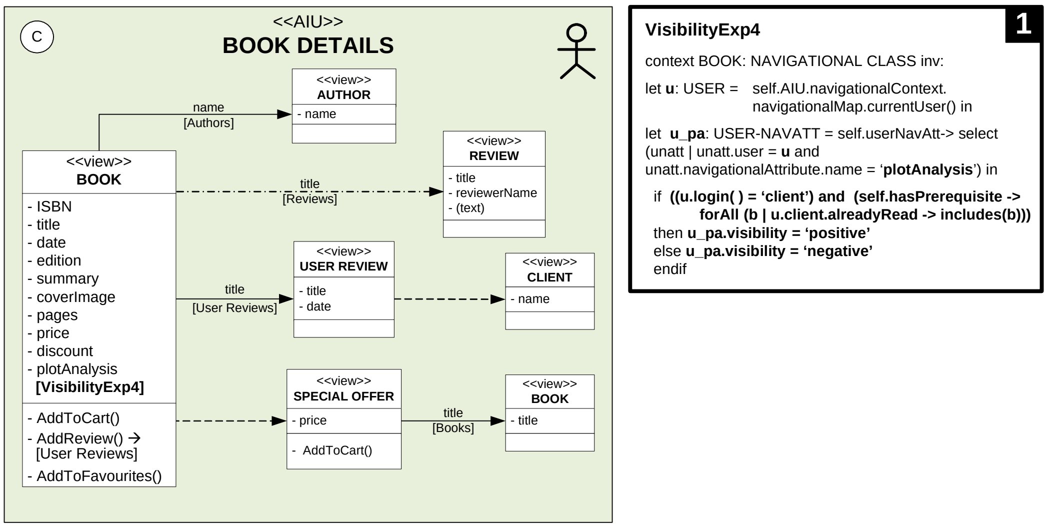
a) **Class Diagram:** The special characteristics of Prerequisites Explanations must be considered in the Class Diagram of the application, through the modelling of the following concepts:

   - The *prerequisite relationship* between the relevant concepts. The adaptive access to a prerequisite explanation about a concept requires to know which concepts a user must previously review, i.e., which are those concepts that are prerequisite of the first one. In order to provide this expressiveness, the class to whose objects the explanations refer must be complemented with classes and associations that specify the set of concepts that are prerequisites of other ones.
   - The *review* of a concept by a given user. Besides the specification of the prerequisites of a given concept, the Class Diagram must include the classes and associations that represent the fulfilment of those prerequisites by a certain user. This may be expressed through the execution of an operation (e.g., a knowledge test), or just through visiting a page that details the prerequisite concept.

b) **OOWS Navigational Model:** As a variant of *Additional Explanations*, the modelling of the *Prerequisite Explanations* method is made by adopting similar strategies. The only difference is that the condition to decide which users may access the explanations must include a statement that verifies whether the current user has already reviewed all the prerequisite concepts of the current one.

c) **Adaptive Techniques:** The same techniques of the previous method are used to implement the Prerequisite Explanations method, i.e., ***Conditional Fragments***, ***Strechtext*** and ***Frame-Based Adaptation*** technique.

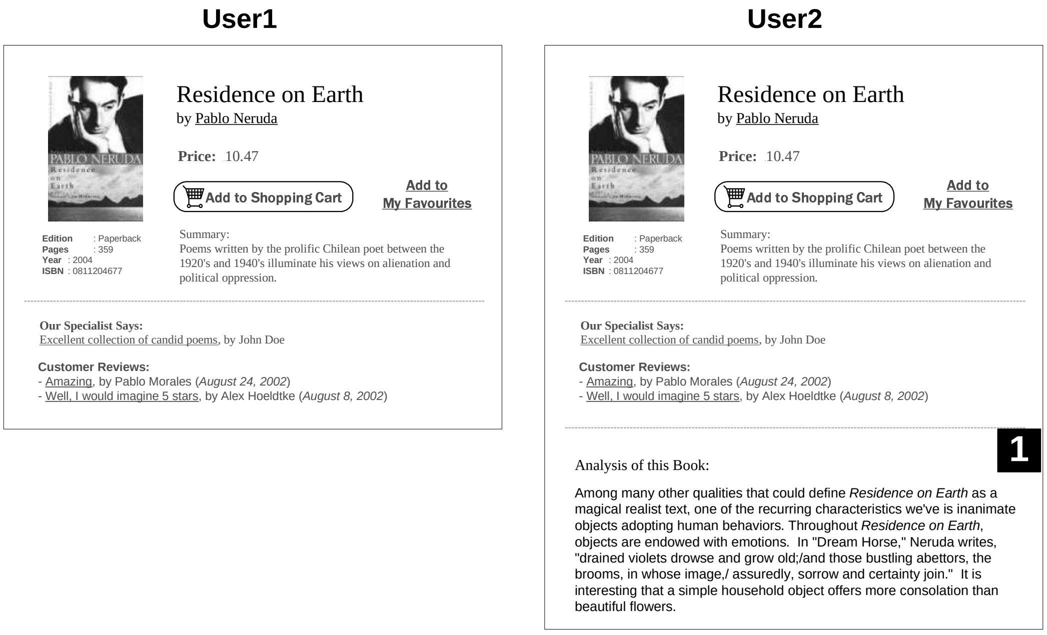
d) **Modelling example:** Figure 5.8 shows the modelling of this method in the `Book Details` AIU. As a *Prerequisite Explanation*, the `plotAnalysis` attribute is included in the manager class, representing a deep analysis of the plot of the `book`. Using the reflexive `prerequisite` association from the *Class Diagram* (Fig. 4.5) and describing the fulfilment of a prerequisite by marking a given book as "already read" (what is modelled in the Class Diagram through the "`alreadyRead`" association), the plot analysis of a book is only shown to users that have already read its prerequisite books, because the analysis may include some references to the plot of those books. The fulfilment of this condition is checked through the *Visibility* expression of `plotAnalysis`, included in *Frame 1* of Fig. 5.8.



```
VisibilityExp4

context BOOK: NAVIGATIONAL CLASS inv:

let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in

let  u_pa: USER-NAVATT = self.userNavAtt-> select
(unatt | unatt.user = u and
unatt.navigationalAttribute.name = 'plotAnalysis') in

  if  ((u.login( ) = 'client') and  (self.hasPrerequisite ->
          forAll (b | u.client.alreadyRead -> includes(b)))
  then u_pa.visibility = 'positive'
  else u_pa.visibility = 'negative'
  endif
```

Figure 5.8: Modelling example of *Prerequisite Explanations* method

e) **Implementation example:** Figure 5.9 illustrates the implementation of the modelled method, with two possible implementations of the modelled AIU. By comparing both versions of the page, we may infer that `User2` has marked all the prerequisites of the currently shown `book` as already read, while `User1` has not. According to this, value of `plotAnalysis` is only shown in the page accessed by `User2` (see *Frame 1* of the right-side page).



User1

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart | Add to My Favourites

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

User2

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart | Add to My Favourites

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

1

Analysis of this Book:

Among many other qualities that could define *Residence on Earth* as a magical realist text, one of the recurring characteristics we've is inanimate objects adopting human behaviors. Throughout *Residence on Earth*, objects are endowed with emotions. In "Dream Horse," Neruda writes, "drained violets drowse and grow old;/and those bustling abettors, the brooms, in whose image,/ assuredly, sorrow and certainty join." It is interesting that a simple household object offers more consolation than beautiful flowers.

Figure 5.9: Implementation example of *Prerequisite Explanations* method

5.2.3 Comparative Explanations

This method allows introducing textual comparisons between the currently visited concept and other related concepts that the user already knows. This method supports the complete understanding of an explanation about a concept, by comparing it with information that is familiar to the user. From a set of comparative explanations associated to the shown object, the system adaptively selects those related to other concepts that the user already knows. The modelling strategy of this method is described as follows:

a) **Class Diagram:** Two concepts must be considered in the Class Diagram:

   - The modelling of the *comparative explanations* between different pairs of concepts. This kind of explanations suggests the incorporation of a conceptual structure that relates both concepts. A *"comparative explanation"* class, which stores the explanation through textual attributes, is a suitable modelling option for this case.
   - The modelling of the *knowledge level* of users about a given concept. Depending on the specific requirements of the application, this characteristic can be either expressed through attributes of a class, measured through a process triggered by operations, or inferred from the visit to its corresponding page. As part of its *Navigational Behaviour View*, the Class Diagram must include the structures that allow checking whether a given user has enough knowledge of a concept.

b) **OOWS Navigational Model:** In the navigational representation of this method, the AIU(s) where comparative explanations are included must have the following characteristics:

   - Its manager class is a view of the class whose objects are being compared;
   - A complementary class is included as a view of the *"comparative explanation"* class.

- Both navigational classes are connected by a navigational relationship of any kind, which is the navigational view of the structural association among them.

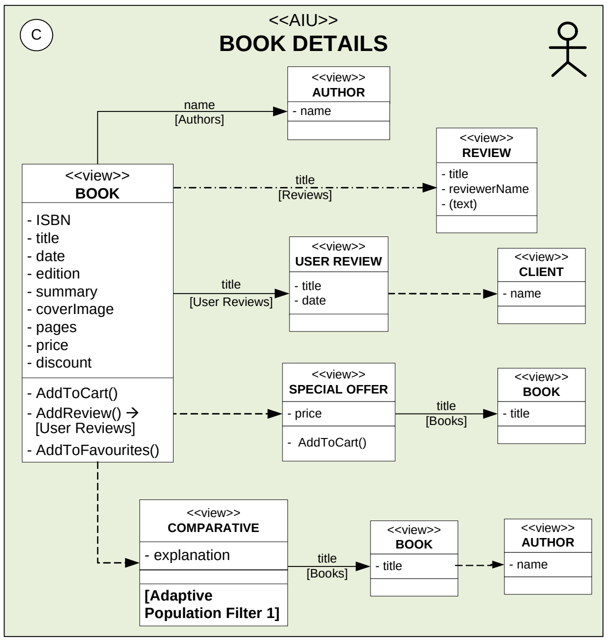
A Web page that is modelled by such an AIU would include all the existing comparative explanations with the currently visited object. A constraint that filters these results must be defined, verifying the current user's knowledge about the candidate objects.

The adaptive selection of Comparative Explanations is specified by means of an ***Adaptive Population Filter*** (see Section 4.5.2), defined on that complementary class. This filter includes a constraint that checks the knowledge level of the current user about each of the candidate objects, in base of the structures of the User Schema. In this way, only those comparative explanations that are relevant to the current user are retrieved and displayed.

c) **Adaptive Technique:** The implementation of this method is made through a ***Conditional Fragments*** technique, by adaptively displaying pieces of information (in this case, the comparative explanations) under conditions on current characteristics of the user.

d) **Modelling example:** Figure 5.10 illustrates the modelling of *Comparative Explanation* method, in the `Book Details` AIU. `Comparative` complementary class includes an `explanation` attribute. The adaptive access is controlled through the definition of an *Adaptive Population Filter* in this class, which measures the knowledge about the complementary book by checking whether a given user has visited its page (*Frame 1* of Fig. 5.10). From the set of existing `Comparative` objects for which the user fulfills that condition, the system randomly selects and displays one of them.

e) **Implementation example:** Fig. 5.11 shows the implementation of the modelled *Comparative Explanation* method. Right-side version of the page shows the information retrieved from the filtered `Comparative` class (*Frame 1* of Fig. 5.11). This means that, among other possible instances, a comparative explanation exists between the book entitled "`General Song`" and the currently visited one ("`Residence on Earth`"), and since `User2` has previously visited the page of the first book, this comparative explanation is displayed. On the other hand,



Adaptive Population Filter 1 **1**

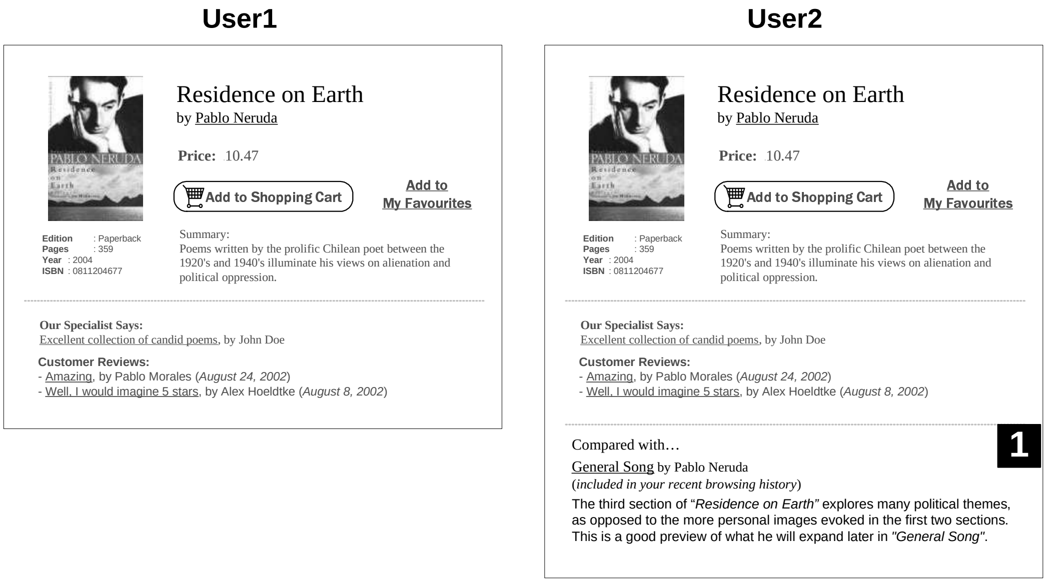
context **COMPARATIVE**: NAVIGATIONAL CLASS inv:

let **u**: USER = self.AIU.navigationalContext.navigationalMap.currentUser() in

self -> any (*comp* | u.historyItem.book -> includesAll (*comp*.book))

Figure 5.10: Modelling example of *Comparative Explanations* method

left-side version does not include any comparative explanation. This means that `User1` has not previously visited any `book` or there is not any comparative explanation between the currently visited one and some of the books she has visited.



User1

Residence on Earth
by Pablo Neruda

Price: 10.47

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

User2

Residence on Earth
by Pablo Neruda

Price: 10.47

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Compared with…
General Song by Pablo Neruda
(*included in your recent browsing history*)
The third section of "*Residence on Earth*" explores many political themes, as opposed to the more personal images evoked in the first two sections. This is a good preview of what he will expand later in *"General Song"*.

Figure 5.11: Implementation example of *Comparative Explanations* method

5.2.4 Explanation Variants

This method provides different users with different explanations of the same concept, according to their individual characteristics, knowledge level or information goals. The modelling of this method requires the definition of the different groups of users to which respective explanations are provided. As in the previous cases, this definition is made through expressions based on user characteristics.

We propose three options to incorporate this method in OOWS Navigational Schemas, according to the complexity of the referred explanation:

Option 1: Alternative versions of navigational attributes

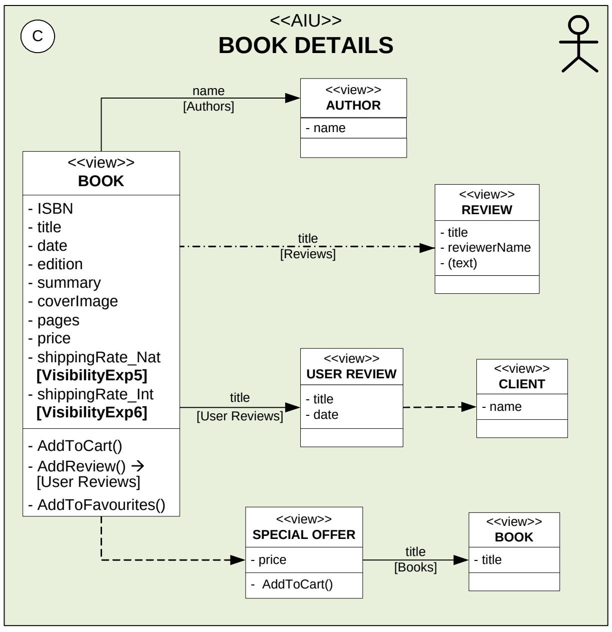
a) **Class Diagram:** In the *Class Diagram*, variants of non-complex, text-only explanations are modelled through alternative *attributes* of a class.

b) **OOWS Navigational Schema:** The ***Visibility*** property of the corresponding navigational attributes (see Section 4.5.7) is adapted to different users: for each considered group of users, only one of the alternative attributes has a *positive* value of visibility, while the rest of the alternatives will have a *negative* value. In this way, users from different groups can visualize different variants of the same attribute.

c) **Adaptive Technique:** This strategy allows implementing a ***Conditional Fragments*** technique (see Section 2.3.4), providing different users with different versions of a piece of information.

d) **Modelling example:** Figure 5.12 shows the modelling of this option in `Book Details` AIU. The `Book` manager class includes two alternative attributes: `shippingRate_Nac` and `shippingRate_Int`. Both inform of the shipping rate associated to the currently visited `book`, but the first one describes the value for its national shipping and the second one for international shipping. By means of respective *Visibility* expressions (*Frame 1* of Fig. 5.12), the value of only one of these attributes is shown, depending on the `default shipping address` chosen by the current `CLIENT` user. If this address belongs to the `defaultCountry` country, the system shows the value of `shippingRate_Nac`; elsewhere, `shippingRate_Int` variant is presented.

e) **Implementation example:** Figure 5.13 shows a possible implementation of the two modelling options of *Explanation Variants* method described in Fig. 5.12. As shown in the left-side page (*Frame 1*), `User1` has chosen a foreign location as her default shipping address, so the international variant of the shipping rate is shown, while `User2` has access to the national variant (right-side page, *Frame 1*).



1

VisibilityExp5

context BOOK: NAVIGATIONAL CLASS inv:

let **u**: USER = self.AIU.navigationalContext.
navigationalMap.currentUser() in

let **u_srn**: USER-NAVATT = self.userNavAtt-> select
(unatt | unatt.user = **u** and
unatt.navigationalAttribute.name = '**shippingRate_Nat**') in

if **((u.login( ) = 'client') and (u.client.shipping ->**
select (default = 'yes').country = 'defaultCountry')
then **u_srn.visibility = 'positive'**
else **u_srn.visibility = 'negative'**
endif

VisibilityExp6

context BOOK: NAVIGATIONAL CLASS inv:

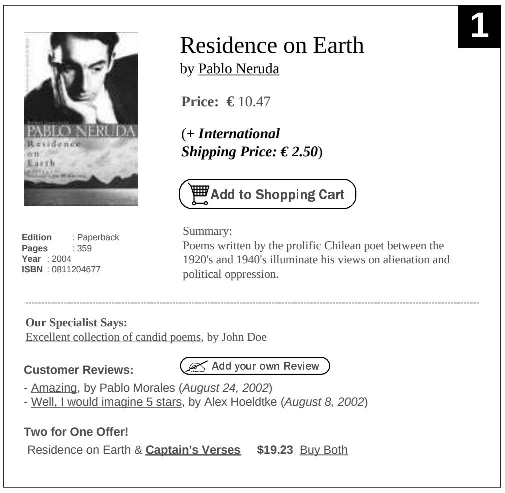
let **u**: USER = self.AIU.navigationalContext.
navigationalMap.currentUser() in

let **u_sri**: USER-NAVATT = self.userNavAtt-> select
(unatt | unatt.user = **u** and
unatt.navigationalAttribute.name = '**shippingRate_Int**') in

if **((u.login( ) = 'client') and (u.client.shipping ->**
select (default = 'yes').country <> 'defaultCountry')
then **u_sri.visibility = 'positive'**
else **u_sri.visibility = 'negative'**
endif

Figure 5.12: Modelling example of *Explanation Variants* method, Option 1

User1



1

Residence on Earth
by Pablo Neruda

Price: € 10.47

(+ ***International***
Shipping Price: € 2.50)

Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

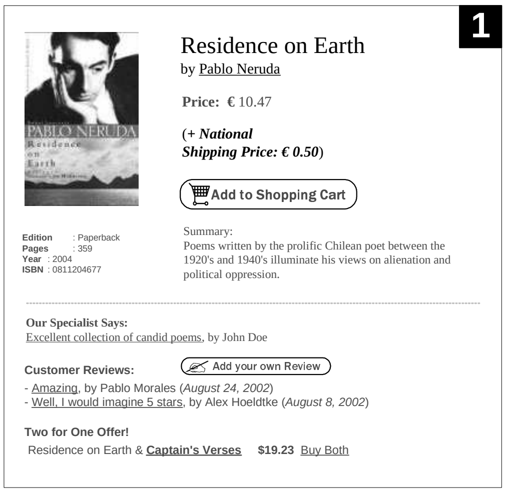
Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

User2



1

Residence on Earth
by Pablo Neruda

Price: € 10.47

(+ ***National***
Shipping Price: € 0.50)

Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

Figure 5.13: Implementation example of *Explanation Variants* method, Option 1

Option 2: Alternative version of complementary objects

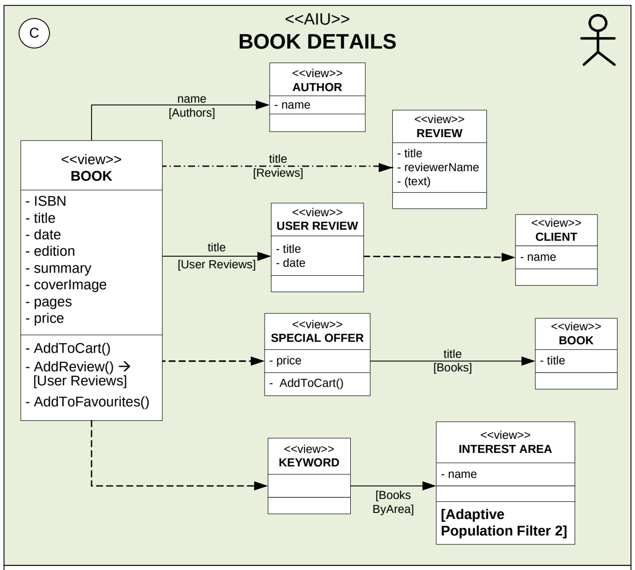
a) **Class Diagram:** Explanations about objects of a given class may be modelled as other associated class. Variants of the explanation are different objects of this class.

b) **OOWS Navigational Schema:** In the navigational schema, the access to the associated objects is represented through a *complementary class*. The adaptive access to the different explanation variants is specified by defining an ***Adaptive Population Filter*** (see Section 4.5.2) in this class. This primitive allows selecting the objects that represent the proper explanation variant to different subsets of users.

c) **Adaptive Techniques:** In this modelling option, the presentation of the possible variants is made through different clusters of information and functionality. This can be implemented through an ***Frame-Based Adaptation*** technique (see Section 2.3.4).

d) **Modelling example:** Figure 5.14 shows this modelling option in the `Book Details` AIU. The `Interest Area` complementary class represents the subjects in which a `CLIENT` user has explicitly indicated her interest, and that are related to the currently visited `book`. This information may help users to know how much this `book` fits their own preferences. The *Adaptive Population Filter* defined in the `Interest Area` class (and detailed in (*Frame 2* of Fig. 5.14) describes this adaptive selection of variants. Let us note that this complementary class is connected to the manager class through a `Keyword` class, which does not contain visible information and only serves to choose the `Interest Area` instances related to the current `book`.

e) **Implementation example:** In Fig. 5.15, the implementation of the *modelling option 2* is shown in *Frame 1* of both pages: while left-side page shows an index of subjects associated to the shown book that have interest to `User1`, page of `User2` shows a void variant of the same information segment. This adaptive presentation of contents may mean that the shown book has interest to `User1`, but not to `User2`. Additionally, and according to the modelled AIU of Fig. 5.14, `User1` can also navigate to obtain detailed information of one of the shown `Interest Area` instances.



context BOOK DETAILS : AIU inv:

let **u**: USER = self.navigationalContext.navigationalMap.currentUser() in

let navrel15: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= '**BOOK**' and nr.target.name= '**KEYWORD**') in

let **unr15**: USER-NAVREL = navrel15.userNavRel-> select (nr|nr.user = **u**) in

if **(u.login( ) = 'client')**
then **unr15.reachability = 'positive'** else **unr15.reachability = 'negative'** endif

1

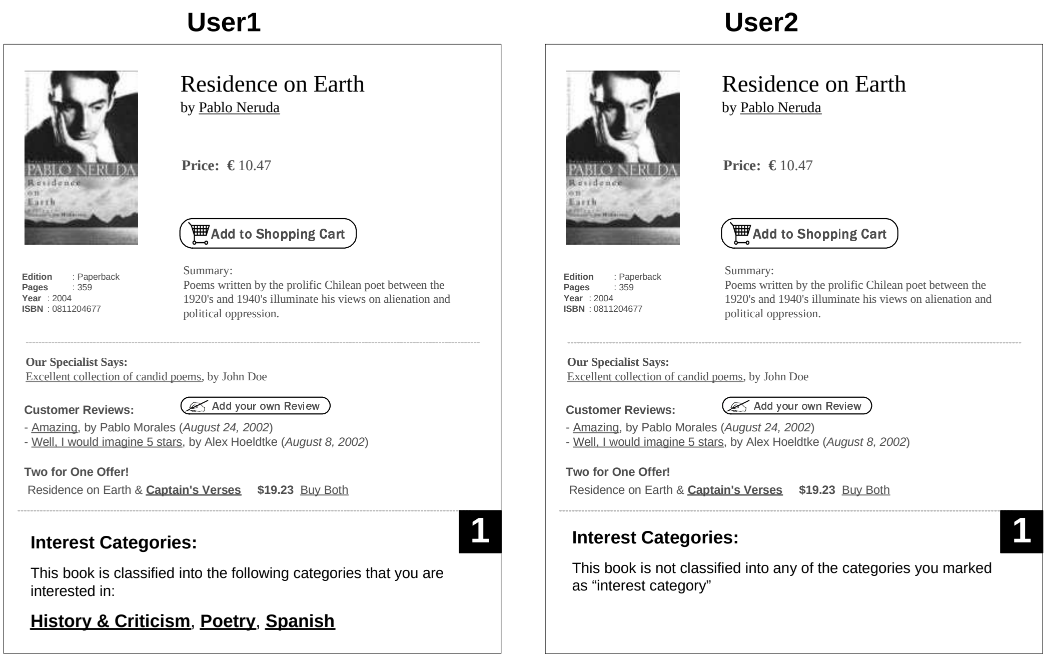
Adaptive Population Filter 2

context **INTEREST AREA**: NAVIGATIONAL CLASS inv:

let **u**: USER = self.AIU.navigationalContext.
navigationalMap.currentUser() in

u.client.interestArea -> includes(self)

Figure 5.14: Modelling example of *Explanation Variants* method, Option 2


User1

Residence on Earth
by Pablo Neruda

Price: € 10.47

Add to Shopping Cart

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

1

Interest Categories:

This book is classified into the following categories that you are interested in:

History & Criticism, **Poetry**, **Spanish**

User2

Residence on Earth
by Pablo Neruda

Price: € 10.47

Add to Shopping Cart

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

1

Interest Categories:

This book is not classified into any of the categories you marked as "interest category"

Figure 5.15: Implementation example of *Explanation Variants* method, Option 2

Option 3: Alternative version of navigational contexts

a) **Class Diagram:** When different versions of navigational contexts need to be defined, corresponding versions of all the attributes and operations whose navigational views are included must be defined as well.

b) **OOWS Navigational Schema:** Complex explanations may be modelled as a whole navigational context. A subset of users from a given *User Stereotype* may need to access some variants of the contexts from their corresponding Navigational Map. In this case, values of the ***Reachability*** property of these contexts (see Section 4.5.1) are adapted. The described subset of users defines a sub-stereotype of the original one. However, this sub-stereotype does not directly inherits the navigational map from its ancestor (as usual stereotypes). Instead, variants of a given navigational context are defined and included in the new navigational map. Users have access to only one variant of the same context, so this context preserves its reachability value (*sequential* or *exploration*), while its corresponding variants become unreachable to those users, changing their *reachability* value to *negative*.

c) **Adaptive Technique:** The adoption of this strategy leads to implement a ***Page Variants*** technique, where different versions of the same page are presented to different users.

d) **Modelling example:** Figure 5.16 shows an example of the *modelling option 3* for the *Explanation Variants* method. A variant of the navigational map previously defined for the `CLIENTS` stereotype is defined for a new `SPANISH CLIENTS` sub-stereotype, which refers to clients with Spanish as their mainly used language. As seen in the figure, both kinds of users have access to the same contexts and links, but the right-side map includes an "`(Esp)`" tag in some of the contexts, which indicates that corresponding pages are presented in Spanish language. In this way, for instance, when a `SPANISH CLIENT` user gains access to a page from the `Books` context, ***reachability*** property of the `Books` context changes to *negative* , while reachability of `Books(Esp)` remains unmodified (sequential) and the user can only access the Spanish version of the visited page.

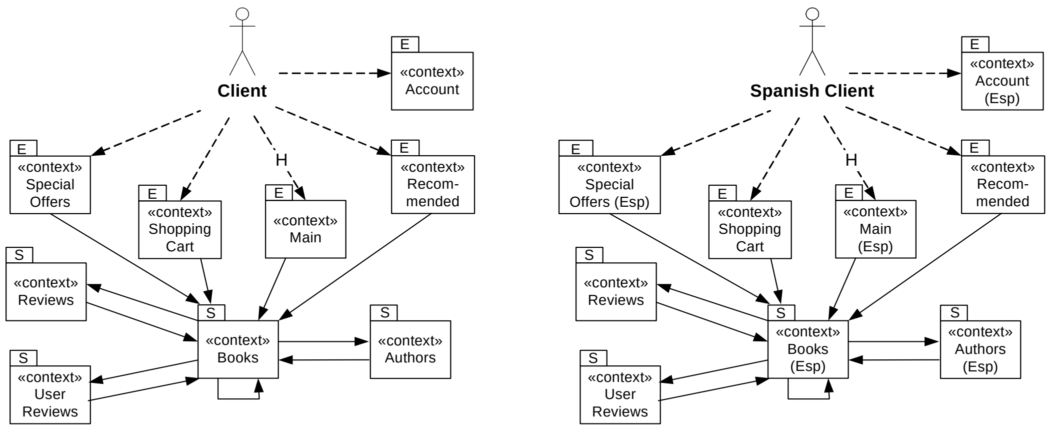
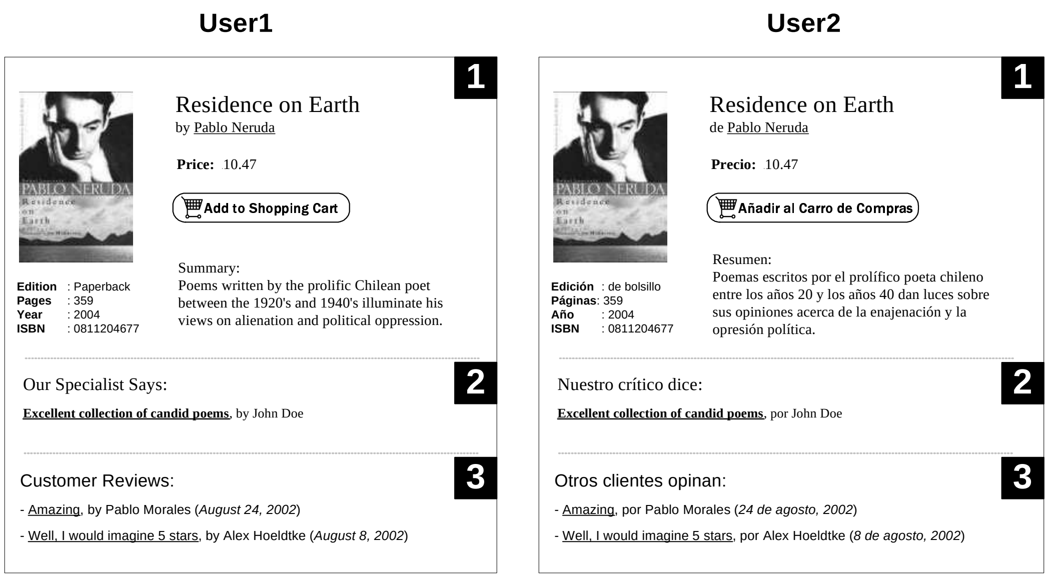


Figure 5.16: Modelling example of *Explanation Variants* method, Option 3

e) **Implementation example:** Figure 5.17 shows how two variants of the same page, each of them modelled through the `Books` and `Books(Esp)` contexts, respectively, should look like. `CLIENT User1` accesses the default English variant of the page, where all the textual contents are shown in that language. In contrast, `SPANISH CLIENT User2` accesses the Spanish variant of the page, where most of the contents are shown in Spanish. Note that information of `Specialized Review` and `User Reviews` (*Frames 2* and *3* of the right-side page) are shown in English in both pages. This occurs because corresponding navigational contexts (`Reviews` and `User Reviews`, respectively) have only one defined version, with no variants, as shown in navigational maps of Fig. 5.16.



User1

1

Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

2

Our Specialist Says:

Excellent collection of candid poems, by John Doe

3

Customer Reviews:

- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

User2

1

Residence on Earth
de Pablo Neruda

Precio: 10.47

Añadir al Carro de Compras

Edición : de bolsillo
Páginas: 359
Año : 2004
ISBN : 0811204677

Resumen:
Poemas escritos por el prolífico poeta chileno entre los años 20 y los años 40 dan luces sobre sus opiniones acerca de la enajenación y la opresión política.

2

Nuestro crítico dice:

Excellent collection of candid poems, por John Doe

3

Otros clientes opinan:

- Amazing, por Pablo Morales (*24 de agosto, 2002*)
- Well, I would imagine 5 stars, por Alex Hoeldtke (*8 de agosto, 2002*)

Figure 5.17: Implementation example of *Explanation Variants* method, Option 3

5.2.5 Sorting

In Web applications, the presentation of detailed information about a concept is usually made through multiple frames, each of them composed of data, operations and hyperlinks with semantic cohesion among them. However, a given fragment may have different levels of interest to different users. The goal of the *Sorting* method is to provide an immediate access to the most relevant frames, avoiding to move along a big amount of less relevant information to reach them. In order to achieve this, information fragments that are associated to a concept may be adaptively ordered, in such a way that the frame that is most relevant to the current user is placed toward the top of the page, while the less relevant is located closer to the bottom.

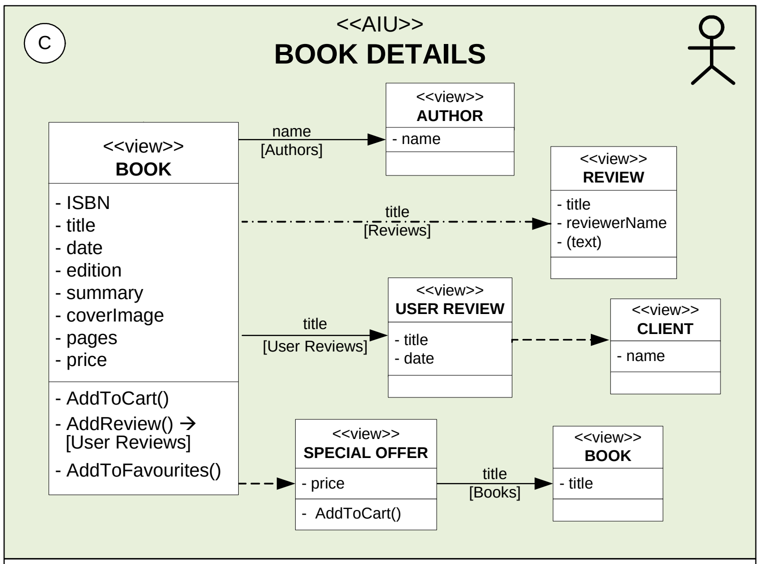
In OOWS, the modelling of this method is made through the *Adaptive Ranking of Navigational Relationships* primitive, which defines an order for the clusters of information that are retrieved from each complementary class of a given AIU. The description of this modelling strategy is the following:

a) **Class Diagram:** The modelling of this method requires the definition of the class that represents the main concept to be detailed and the classes that provide information about that concept, along with the corresponding associations.

b) **OOWS Navigational Schema:** This method is modelled through the use of ***Adaptive Ranking of Navigational Relationships*** primitive (see Section 4.5.6). An AIU that models the page to be adapted is defined. Its manager class corresponds to the class of the concept to be detailed, while different complementary classes represent the information frames that are adaptively ordered. Different *rankings of precedence* are assigned to the set of navigational relationships that associate the manager class with the complementary classes. User-centered constraints define different subsets of users: each of them is provided with a different ranking. The final implementation makes use of these rankings to determine the order in which the defined frames of information are presented to the current user.

c) **Adaptive Technique:** This implementation corresponds to a ***Frame Based Adaptation*** (see Section 2.3.4) technique, which adapts the presentation of different information frames associated to a concept.

d) **Modelling example:** The modelling strategy of *Sorting* method directly uses the introduced *Adaptive Ranking of Navigational Relationships* primitive (see Section 4.5.6). Therefore, in order to illustrate this modelling strategy, we reuse the case study example presented in the description of this primitive. Figure 5.18 shows the definition of two rankings of precedence for the navigational relationships connected to the manager class. The OCL expression of this primitive shows that a given ranking is associated to `CLIENT` users with "“`habitual`" value of their corresponding `readingHabit` attribute. For the rest of users, another ranking is defined.



context BOOK DETAILS : AIU inv:

let **u**: USER = self.navigationalContext.navigationalMap.currentUser() in

let navrel11: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= '**BOOK**' and nr.target.name= '**AUTHOR**') in

let navrel12: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= '**BOOK**' and nr.target.name= '**REVIEW**') in

let navrel13: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= '**BOOK**' and nr.target.name= '**USER REVIEW**') in

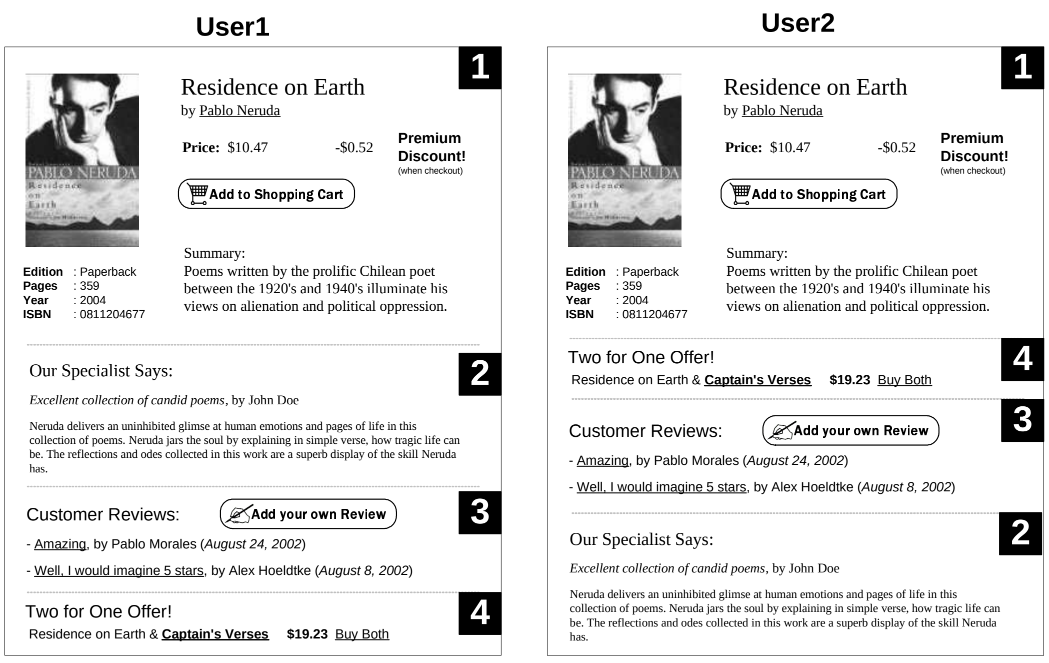
let navrel14: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= '**BOOK**' and nr.target.name= '**SPECIAL OFFER**') in

let **unr11**: USER-NAVREL = navrel11.userNavRel-> select (nr|nr.user = **u**) in
let **unr12**: USER-NAVREL = navrel12.userNavRel-> select (nr|nr.user = **u**) in
let **unr13**: USER-NAVREL = navrel13.userNavRel-> select (nr|nr.user = **u**) in
let **unr14**: USER-NAVREL = navrel14.userNavRel-> select (nr|nr.user = **u**) in

if **(u.login( ) = 'client')** and **(u.client.readingHabit = 'habitual')**
then **unr11.order = 1; unr12.order = 2; unr13.order =3; unr14.order = 4;**
else **unr11.order = 1; unr12.order = 4; unr13.order =3; unr14.order = 2;**
endif

Figure 5.18: Modelling example of *Sorting* method

e) **Implementation example:** Figure 5.19 shows the implementation of this method. Information frames that are retrieved through the ranked navigational relationships are displayed in different order. `User1` fulfills the condition of being a `CLIENT` and a `habitual` reader, so the adaptive system infers a primary interest in the `specialized review`, locating the `Review` frame closer to the top of the page. On the other side, `User2` does not fulfil one of these conditions, and the access to the `Special Offer` frame is privileged.


User1

1
Residence on Earth
by Pablo Neruda
Price: $10.47 -$0.52 **Premium Discount!** (when checkout)
Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

2
Our Specialist Says:
Excellent collection of candid poems, by John Doe
Neruda delivers an uninhibited glimse at human emotions and pages of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has.

3
Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

4
Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

User2

1
Residence on Earth
by Pablo Neruda
Price: $10.47 -$0.52 **Premium Discount!** (when checkout)
Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

4
Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

3
Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

2
Our Specialist Says:
Excellent collection of candid poems, by John Doe
Neruda delivers an uninhibited glimse at human emotions and pages of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has.

Figure 5.19: Implementation example of *Sorting* method

5.3 Adaptive Navigation Methods in OOWS

Different methods have been designed to achieve adaptation in the presentation of links. These methods allow developers to specify different topologies of the hyperstructure from a unique conceptual schema of the navigation, depending on the particular characteristics of who accesses the application. This is made by privileging the presentation of links to those objects that are most relevant to a given user, hiding or demoting links to the irrelevant ones.

Adaptive Primitives that have been introduced in the previous chapter permit to incorporate this kind of adaptation into the OOWS Navigational Schema of an Adaptive Web Application. Table of Fig. 5.20 shows the list of *Methods of Adaptive Navigation* that this proposal supports.

		Adaptive Navigation Techniques			
		Link Hiding	*Link Ordering*	*Link Annotation*	*Direct Guidance*
Methods for Adaptive Navigation	*Global Guidance*	Adaptive Population Filter	Adaptive Ordering Pattern	Adaptive Ordering Pattern	Adaptive Population Filter
	Local Guidance	Adaptive Population Filter	Adaptive Ordering Pattern	Adaptive Ordering Pattern	
	Global Orientation	Adaptive Population Filter	Adaptive Ordering Pattern	Adaptive Ordering Pattern	
	Local Orientation	Adaptive Population Filter Adaptive Visibility of Navigational Relationships	Adaptive Ordering Pattern	Adaptive Ordering Pattern	

Figure 5.20: Adaptive Navigation Methods and Techniques, with OOWS structures that support their modelling

Similarly to the table of Adaptive Presentation Methods (Fig. 5.1), the columns of the table of Fig. 5.20 show different *Adaptive Navigation techniques* that support the adaptation goals of each method. The cells of the table contain the Adaptive Primitives that we propose to use in the modelling of each method, implementing the corresponding adaptive technique.

From the considered adaptive navigation methods, we may distinguish two groups: the one composed of *Global Guidance* and *Local Guidance* methods and the one composed of *Global Orientation* and *Local Orientation* methods. The first group helps users to decide which are the best navigational choices from the set of available links, and the second one orients users about their current position into the navigational structure of the application.

The goals of these two groups of methods, however, are not very different and modelling proposals for one group may also help to fulfil the goal of the other group. For instance, let us consider a list of links suggested from previous navigational actions, and a list of links to pages that a user is allowed to access from her current knowledge level. Each list supports the fulfilment of two different goals: guidance and orientation, respectively. However, both can be modelled in OOWS through the same conceptual structures. What may be seen as a disadvantageous ambiguity in the definition of Adaptive Navigation Methods, becomes an advantage in the context of our proposal, by allowing us to suggest similar modelling strategies in both cases.

In the following subsections, the modelling of each considered Adaptive Navigation Method is described.

5.3.1 Global Guidance

From the set of available links in a page, the user may need to know which navigational possibilities lead to the contents that are more relevant to her, in order to fulfil some global information goal. *Global Guidance* method supports the user in this task, helping her to find the shortest way to reach those pages.

The present proposal provides three different alternatives to include this method into OOWS Navigational Schemas. The first two are oriented to facilitate the selection of one link among a list of multiple choices, all of them related to the same class of objects (e.g., an index of multiple educational topics or on-sale products), by selecting and sorting the links available in a node according to their relevance to the global goal; the third option directly proposes the best alternative from such a list, i.e., the link that leads to the next most relevant page to the current user.

Option 1: Selection of the most relevant links

From a list of multiple links, all of them leading to pages of different objects from the same class, some of them may have little or no relevance to a given user. In this case, their availability as an option to consider for future navigation may distract the user from the fulfilment of a global information goal. An overloaded list of topics may hinder the goal of learning a given content, as well as a non-filtered index of products may distract the user from buying a given product. Providing a list of links that includes only those items that are relevant to the current user is a possible solution for this problem.

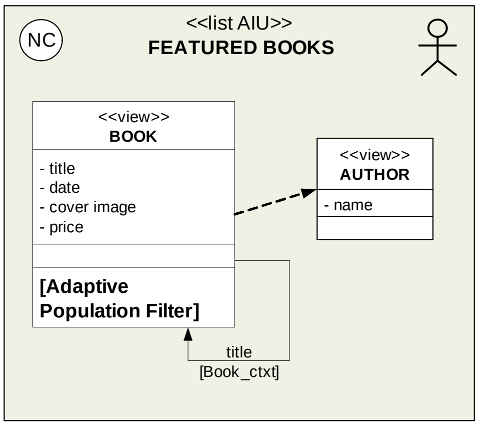
a) **Class Diagram:** It must include those classes and associations that describe the information items to be listed.

b) **OOWS Navigational Schema:** In OOWS, we propose a conceptual solution by means of the ***Adaptive Population Filter*** primitive (see Section 4.5.2). The referred list of multiple links may be modelled through a *List AIU*. The inclusion of an Adaptive Population Filter on its manager class allows defining different criteria for selecting instances from that class and assigning them to different subsets of users. In this way, a user that accesses the described list of links will visualize links to relevant information only.

c) **Adaptive Technique:** This strategy leads to implement an adaptive ***Link-Hiding*** technique, by denying the access to the instances of the List AIU (set of data and links) that lead to irrelevant pages.

d) **Modelling example:** Figure 5.21 shows a modelling example of the first option of *Global Guidance* method. It consists on a *List AIU*, which provides users with an index of `Recommended Books`. The manager class of this AIU includes an *Adaptive Population Filter*, which selects those books that best fit the current user's preferences. The OCL description of this primitive is shown at the right side of Fig. 5.21. It defines a variable set (`preferred`) that contains those books that the current user has marked as favourite (through the `AddToFavourites()` operation) and those books which this user has evaluated with a rate higher than *2* (expressed through the `RATE` class). If the user is a

CLIENT and the preferred set is not empty, then the system selects those books that the user does not already own and that have at least three keywords into the set of keywords associated to the books from the preferred set. If the user does not fulfil the described condition, the system provides her with a non-adaptive recommendation, by showing those books that have been recently released.

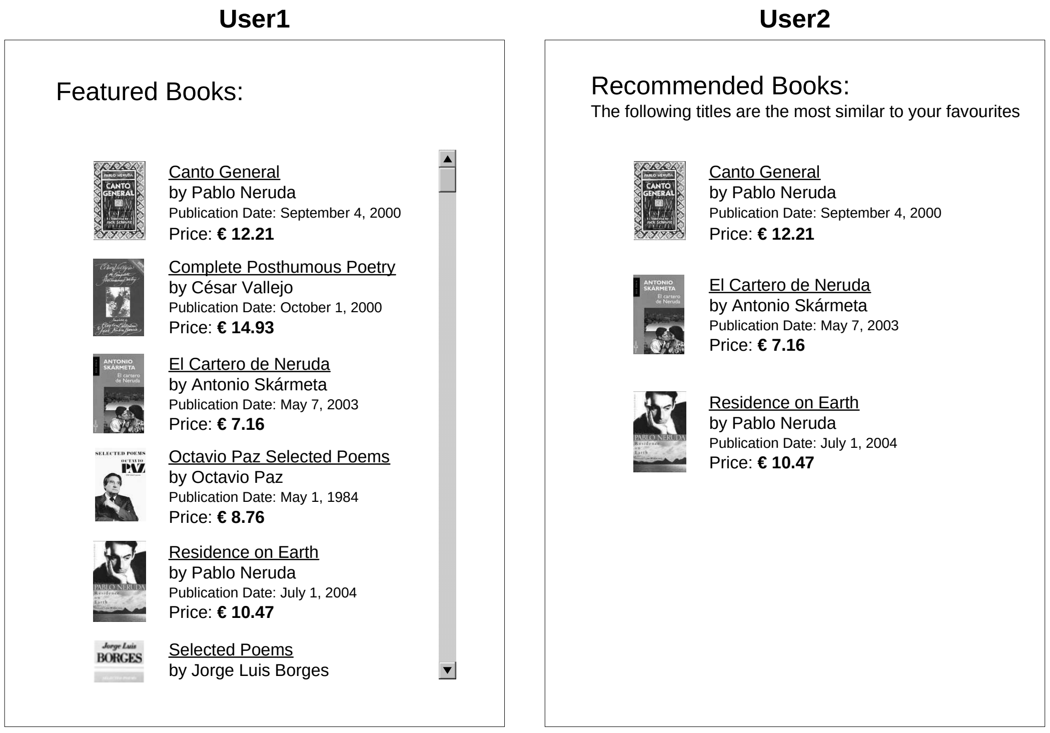


Adaptive Population Filter

```
context BOOK : NAVIGATIONAL CLASS inv:
let u: USER =  self.AIU.navigationalContext.
               navigationalMap.currentUser() in
let preferred: Set(BOOK) = u.client.favourite -> union
               (u.client.rate-> select(value>2).book) in
if (u.login( ) = 'client') and (preferred.notEmpty( )) then
  (u.client.alreadyOwns-> excludes(self)) and
  (preferred.keyword.asSet( ) ->
  intersection (self.keyword) -> size( ) >= 3)
else  self.newRelease = 'true' endif
```

Figure 5.21: Modelling example of Global Guidance method, Option 1

e) **Implementation example:** Figure 5.22 shows the implementation of the modelled AIU. In the left-side page, a set of new released books is presented to User1, which indicates that this user does not fulfil the described condition. The right-side section presents a reduced set of books, which correspond to those that are most similar to the books preferred by User2. In the first case, the adopted filtering criterion is non-adaptive, because it does not depend on any user characteristic. As a result, the same set of books is presented to any user that does not fulfil the mentioned condition. On the contrary, the filtering criterion of the second case is adaptive, and the displayed books are likely to vary from one user to another.



User1

Featured Books:

Canto General
by Pablo Neruda
Publication Date: September 4, 2000
Price: **€ 12.21**

Complete Posthumous Poetry
by César Vallejo
Publication Date: October 1, 2000
Price: **€ 14.93**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

Octavio Paz Selected Poems
by Octavio Paz
Publication Date: May 1, 1984
Price: **€ 8.76**

Residence on Earth
by Pablo Neruda
Publication Date: July 1, 2004
Price: **€ 10.47**

Selected Poems
by Jorge Luis Borges

User2

Recommended Books:
The following titles are the most similar to your favourites

Canto General
by Pablo Neruda
Publication Date: September 4, 2000
Price: **€ 12.21**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

Residence on Earth
by Pablo Neruda
Publication Date: July 1, 2004
Price: **€ 10.47**

Figure 5.22: Implementation example of Global Guidance method, Option 1

Option 2: Ordering of the available links

From a list of links to pages of different objects from the same class, some of them may have more relevance than others to different users. Displaying these links according to a fixed ordering criterion gives little clues about that relevance. The consecution of a global information goal may be delayed when all the options of future navigation seem to be equally relevant. An unsorted list of topics hardly helps to distinguish those most relevant to fulfil a given learning goal; an alphabetical order for a list of products gives the user little information about how much those products match her interests in the buying process. Instead, ordering the presentation of these links according to their importance to the current user is a valuable help to shorten the navigational path to the relevant information items.

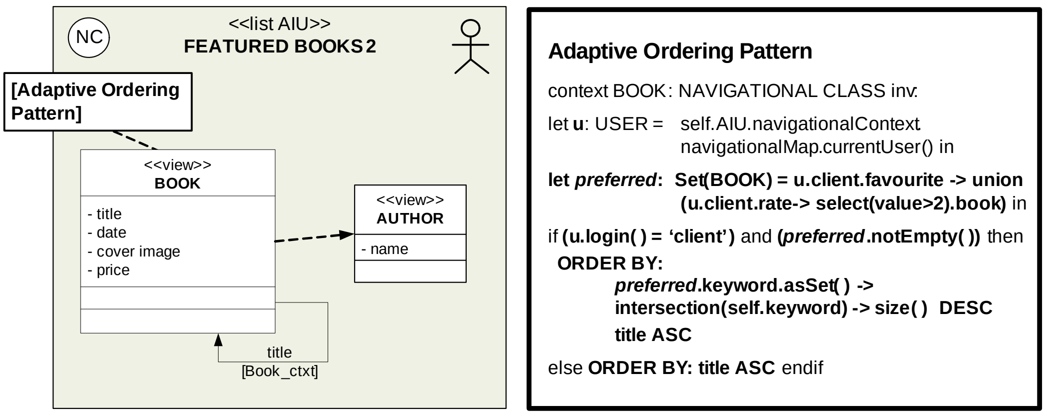
a) **Class Diagram:** It must include those classes and associations that describe the information items to be listed.

b) **OOWS Navigational Schema:** This strategy is based on the definition of an ***Adaptive Ordering Pattern*** (see Section 4.5.3) on the manager class of the List AIU that represents the mentioned list. Each *ordering statement* specifies a criterion to sort the instances of the List AIU. The *order expression* assigns different set of ordering statements to different groups of users.

c) **Adaptive Technique:** According to further presentational specifications, the order of relevance can be used in different implementations of this method:

    - The order of presentation in the page can be directly guided by the order of relevance, in such a way that, for instance, the most relevant links are displayed closer to the top of the page. This implementation corresponds to an Adaptive ***Link-Ordering*** technique.
    - The inclusion of graphical or textual annotations that complement the presentation of the link anchors, to indicate the relevance of the corresponding pages. This implementation corresponds to an Adaptive ***Link-Annotation*** technique.

d) **Modelling example:** Figure 5.23 illustrates this modelling option, through the *Adaptive Ordering Pattern* defined in the `Book` manager class of the `Featured Books` AIU. The OCL description of this primitive is shown at the right side of Fig. 5.23. This primitive adopts a similar criterion than the adopted in the modelling example of the previous option, but in this case it is used to sort the retrieved instances. For a given user that is a `CLIENT` and whose `preferred` set is not empty, the order of a selected book is defined by the number of associated keywords in common with the books from her `preferred` set. The higher this number is, the more important place is assigned to the book. Alphabetical order of `titles` of books is adopted as a secondary ordering criterion, which is also the only criterion provided to users that do not fulfil the previous condition.

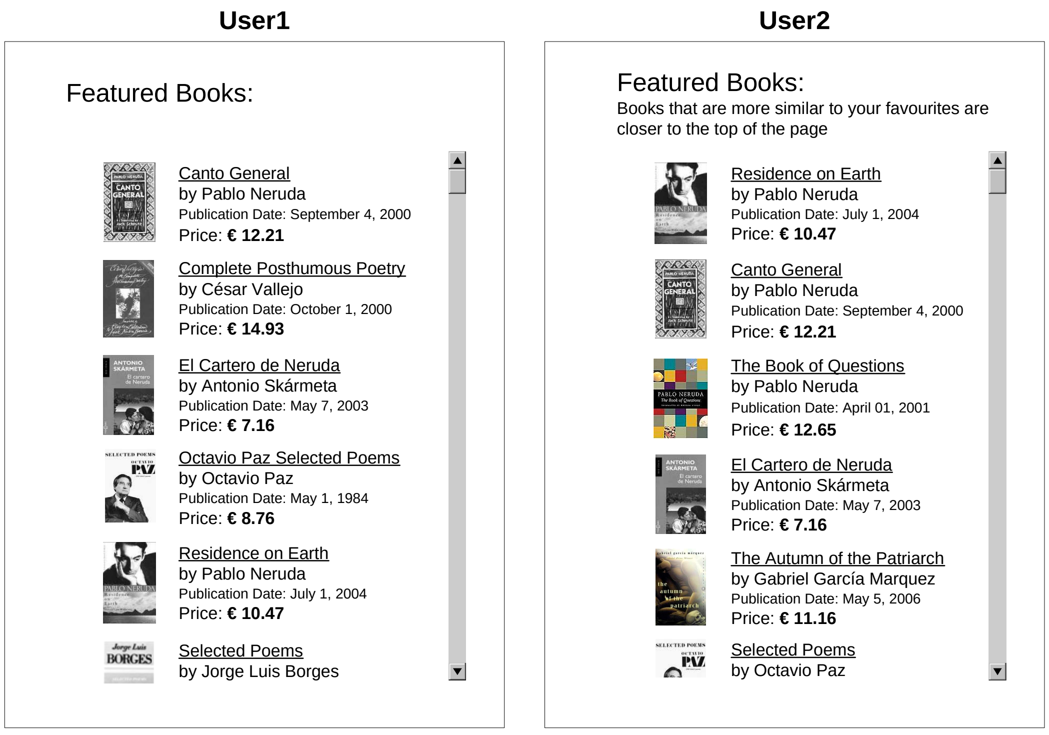


Adaptive Ordering Pattern

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =  self.AIU.navigationalContext.
               navigationalMap.currentUser() in
let preferred:  Set(BOOK) = u.client.favourite -> union
                (u.client.rate-> select(value>2).book) in
if (u.login( ) = 'client') and (preferred.notEmpty( )) then
  ORDER BY:
        preferred.keyword.asSet( ) ->
        intersection(self.keyword) -> size( )  DESC
        title ASC
else ORDER BY: title ASC endif
```

Figure 5.23: Modelling example of Global Guidance method, Option 2

e) **Implementation example:** Figure 5.24 shows the implementation of this modelling option. In both page sections, the list of featured books are ordered according to different criteria. In the left page section, books are alphabetically ordered by their titles, which indicates that `User1` either is not a `CLIENT` user or has not marked any book as `favourite` or rated it positively. On the other side, `User2` fulfills the mentioned condition, so the right page section shows the same set of books, but ordered according to the described adaptive criteria.


User1

Featured Books:

Canto General
by Pablo Neruda
Publication Date: September 4, 2000
Price: **€ 12.21**

Complete Posthumous Poetry
by César Vallejo
Publication Date: October 1, 2000
Price: **€ 14.93**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

Octavio Paz Selected Poems
by Octavio Paz
Publication Date: May 1, 1984
Price: **€ 8.76**

Residence on Earth
by Pablo Neruda
Publication Date: July 1, 2004
Price: **€ 10.47**

Selected Poems
by Jorge Luis Borges

User2

Featured Books:
Books that are more similar to your favourites are closer to the top of the page

Residence on Earth
by Pablo Neruda
Publication Date: July 1, 2004
Price: **€ 10.47**

Canto General
by Pablo Neruda
Publication Date: September 4, 2000
Price: **€ 12.21**

The Book of Questions
by Pablo Neruda
Publication Date: April 01, 2001
Price: **€ 12.65**

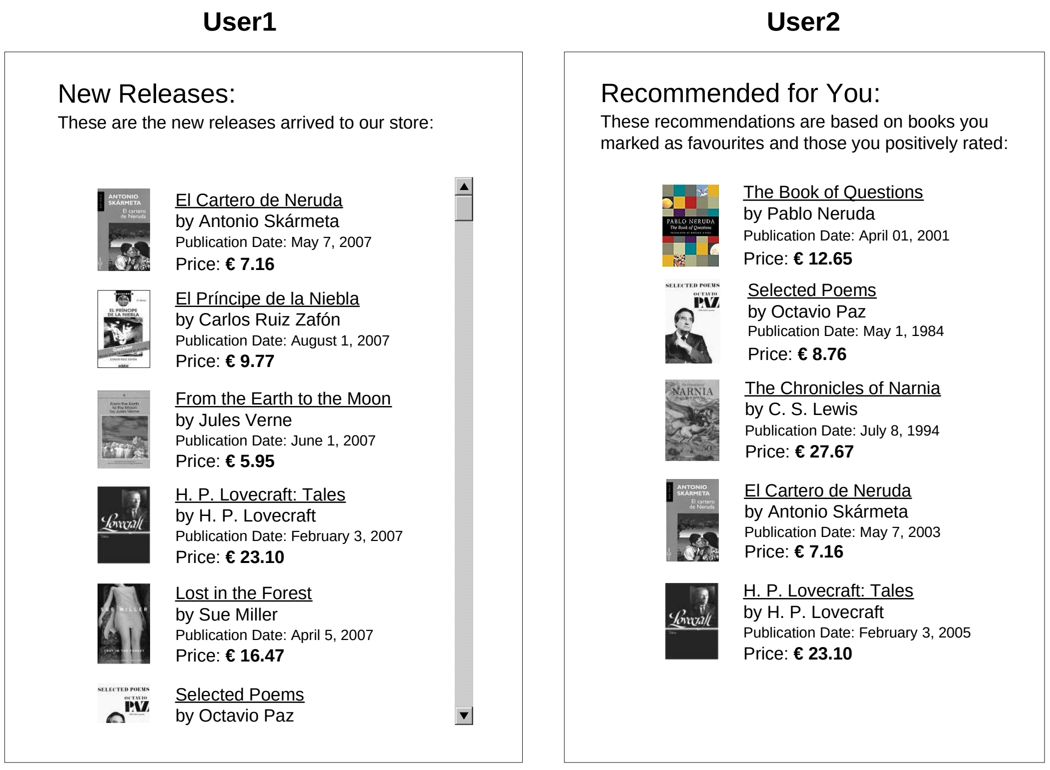
El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

The Autumn of the Patriarch
by Gabriel García Marquez
Publication Date: May 5, 2006
Price: **€ 11.16**

Selected Poems
by Octavio Paz

Figure 5.24: Implementation example of Global Guidance method, Option 2

Figure 5.25 shows the implementation of the `Featured Books` AIU, but considering the two previous modelling strategies, that is to say, incorporating the described *Adaptive Population Filter* and *Adaptive Ordering Pattern.* In the left page section, `User1` does not receive adaptive recommendations from the system: she is not a `CLIENT` user or she has not marked any book as `favourite` or rated it positively. As a result, only an alphabetically ordered index of `new released books` are displayed. On the contrary, `User2` is a `CLIENT` and has expressed her preferences, so the system provides her with an adaptive index of recommended books. Furthermore, we may infer that the first listed book (entitled `‘‘The Book of Questions’’`) best fits the preferences of `User2` than the last one (`‘‘H. P. Lovecraft: Tales’’`). This adaptive implementation supports the *global orientation* of the user into the application hyperstructure. The *non-contextual* character of the modelled *List AIU* permits to include these recommendations in *exploration contexts*, so they may be accessed from any point of the browsing session. The consideration of updateable user preferences to select and order the recommended instances allows providing a guidance to choose the next best alternative of navigation.


User1

New Releases:
These are the new releases arrived to our store:

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2007
Price: **€ 7.16**

El Príncipe de la Niebla
by Carlos Ruiz Zafón
Publication Date: August 1, 2007
Price: **€ 9.77**

From the Earth to the Moon
by Jules Verne
Publication Date: June 1, 2007
Price: **€ 5.95**

H. P. Lovecraft: Tales
by H. P. Lovecraft
Publication Date: February 3, 2007
Price: **€ 23.10**

Lost in the Forest
by Sue Miller
Publication Date: April 5, 2007
Price: **€ 16.47**

Selected Poems
by Octavio Paz

User2

Recommended for You:
These recommendations are based on books you marked as favourites and those you positively rated:

The Book of Questions
by Pablo Neruda
Publication Date: April 01, 2001
Price: **€ 12.65**

Selected Poems
by Octavio Paz
Publication Date: May 1, 1984
Price: **€ 8.76**

The Chronicles of Narnia
by C. S. Lewis
Publication Date: July 8, 1994
Price: **€ 27.67**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

H. P. Lovecraft: Tales
by H. P. Lovecraft
Publication Date: February 3, 2005
Price: **€ 23.10**

Figure 5.25: Implementation of *Global Guidance* method, options 1 and 2

Option 3: Selection of the next best link

This conceptual alternative for the Global Guidance method allows specifying the link to the most relevant page among multiple alternatives. From the whole set of instances of a given concept, this method selects the one that best fits the information goals of a given user, according to a certain relevance criterion.

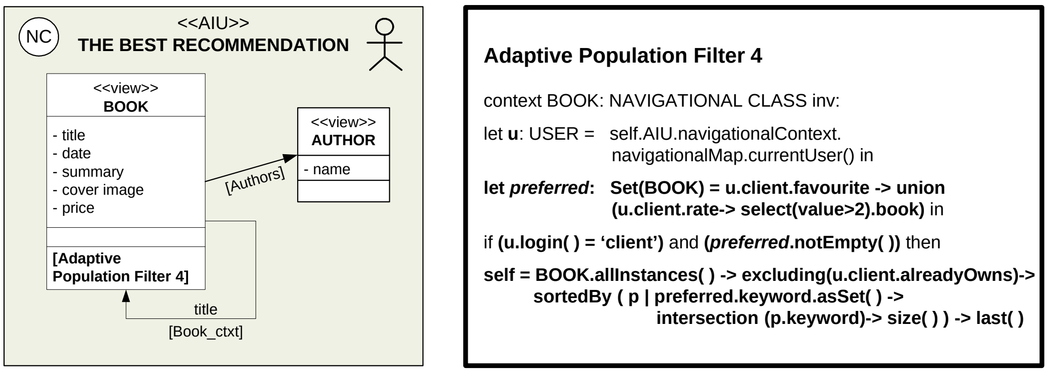
a) **Class Diagram:** It must include those classes and associations that describe the information item to be selected.

b) **OOWS Navigational Schema:** By means of an ***Adaptive Population Filter***, the most relevant object from that ordered list is chosen. The navigational description only allows the access to the link corresponding to that object.

c) **Adaptive Technique:** The implementation of this strategy uses a ***Direct Guidance*** technique (see Section 2.3.4), suggesting the "next best link" to follow, from multiple alternatives of navigation that are provided in a page.

d) **Modelling example:** A modelling example of this option is shown in Fig. 5.26. Through a *non-contextual Single-Object AIU*, the system shows an adaptive recommendation of a single `Book` instance, which best fits the user's preferences according to a given selection criterion. An *Adaptive Population Filter* selects the `Book` instance that the user does not already own and that have the highest number of associated keywords included into the set of keywords associated to books from the`preferred` set. Conditions imposed to the user in order to access this recommendation are the same than the previous case.

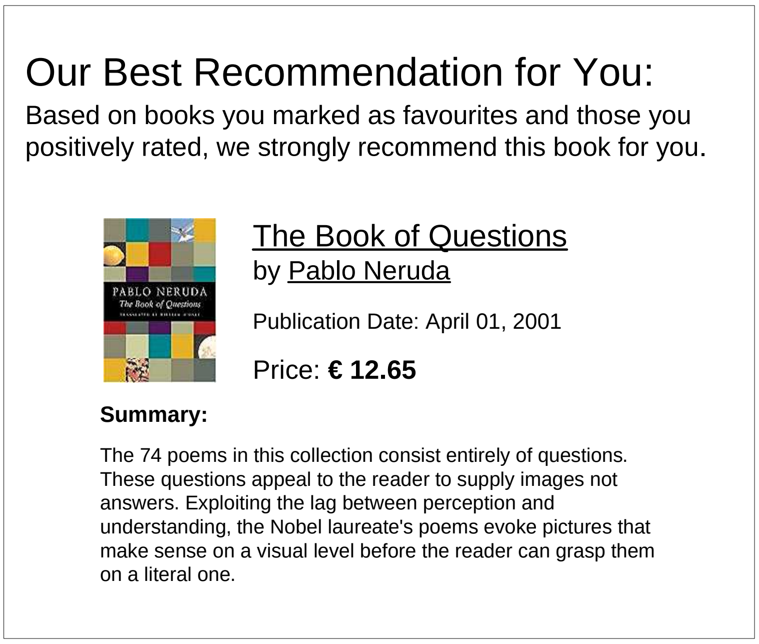
e) **Implementation example:** Figure 5.27 shows an implementation of this method. The system recommends the current user to visit the page of the book that is most similar to her `preferred books`. This is a more direct *global guide* to choose the next navigational step than the provided by the list of recommendations. The non-contextual character of the AIU may also permit to include this recommendation in exploration contexts, and in this way to be reachable from any point of the hyperspace.



Adaptive Population Filter 4

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in
let preferred:  Set(BOOK) = u.client.favourite -> union
                (u.client.rate-> select(value>2).book) in
if (u.login( ) = 'client') and (preferred.notEmpty( )) then
self = BOOK.allInstances( ) -> excluding(u.client.alreadyOwns)->
       sortedBy ( p | preferred.keyword.asSet( ) ->
                      intersection (p.keyword)-> size( ) ) -> last( )
```

Figure 5.26: Modelling of *Global Guidance* method, option 3



Our Best Recommendation for You:

Based on books you marked as favourites and those you positively rated, we strongly recommend this book for you.

The Book of Questions
by Pablo Neruda

Publication Date: April 01, 2001

Price: **€ 12.65**

Summary:

The 74 poems in this collection consist entirely of questions. These questions appeal to the reader to supply images not answers. Exploiting the lag between perception and understanding, the Nobel laureate's poems evoke pictures that make sense on a visual level before the reader can grasp them on a literal one.

Figure 5.27: Implementation of *Global Guidance* method, option 3

Note: Most of the strategies defined for the following methods propose several modelling options, based on the use of *Adaptive Population Filter* and *Adaptive Ordering Pattern* primitives, respectively. Just like the example of Fig. 5.25, the inclusion of both primitives into the same example illustrates the adaptation goal of the corresponding Adaptive Navigation Method in a better way that the separate presentation of each modelling option. For this reason, in such cases, modelling strategies are illustrated through the same example, which includes all the proposed modelling options, and its respective implementation.

5.3.2 Local Guidance

From a page that provides a detailed description of a given concept, information about related concepts may help to decide which is the next page to visit. More than pursuing a global information goal, this navigational step complements the understanding of the currently accessed object. The adaptation of the links to this complementary information can be specified by incorporating the *Local Guidance* method into the navigational specifications of the application.

The ***Local Guidance*** method helps the user to choose the most relevant links from the currently visited node, but oriented to support the very next navigational step instead of fulfilling a global information goal. The strategies to include this method into navigational specifications of OOWS are very similar to those adopted for the *Global Guidance* method. In both cases, the adaptivity is defined on a list of links corresponding to different objects from the same class. However, the present proposal specifies the Global Guidance method in terms of the main objects of the AIU, while it focuses the Local Guidance method on objects that complement the information provided about the main objects. Therefore, the needed adaptive navigational structures are defined on complementary classes instead of on the manager class of the involved AIU. Two options of modelling are provided: the first option adaptively restricts the set of complementary objects to be shown, while the second one provides these objects with adaptive ordering criteria.

The modelling and implementation of these two options are illustrated through the same graphical example.

Option 1: Selection of the most relevant links

a) **Class Diagram:** Once defined the class that models the main concept of the page, a class that represents a concept that complements the information of the main one must be included. Furthermore, both classes must be related through an association, in which one object of the main class is associated to several objects of the complementary class.

b) **OOWS Navigational Schema:** In a page that gives information about a certain object, some related objects of a class may have little or no relevance to a given user. To restrict the access to the irrelevant objects, an ***Adaptive Population Filter*** (see Section 4.5.2) is defined on the complementary class from where those objects can be retrieved. The corresponding OCL constraint allows defining different selection criteria for different subsets of users.

c) **Adaptive Technique:** This strategy is implemented by means of an Adaptive ***Link-Hiding*** technique, by hiding or deactivating the links to the irrelevant objects, according to presentational specifications.

Option 2: Ordering of the available links

a) **Class Diagram:** Just like the previous option, this modelling strategy demands the definition of a class that complements the information of the class of the main concept of the page, along with the corresponding association between them.

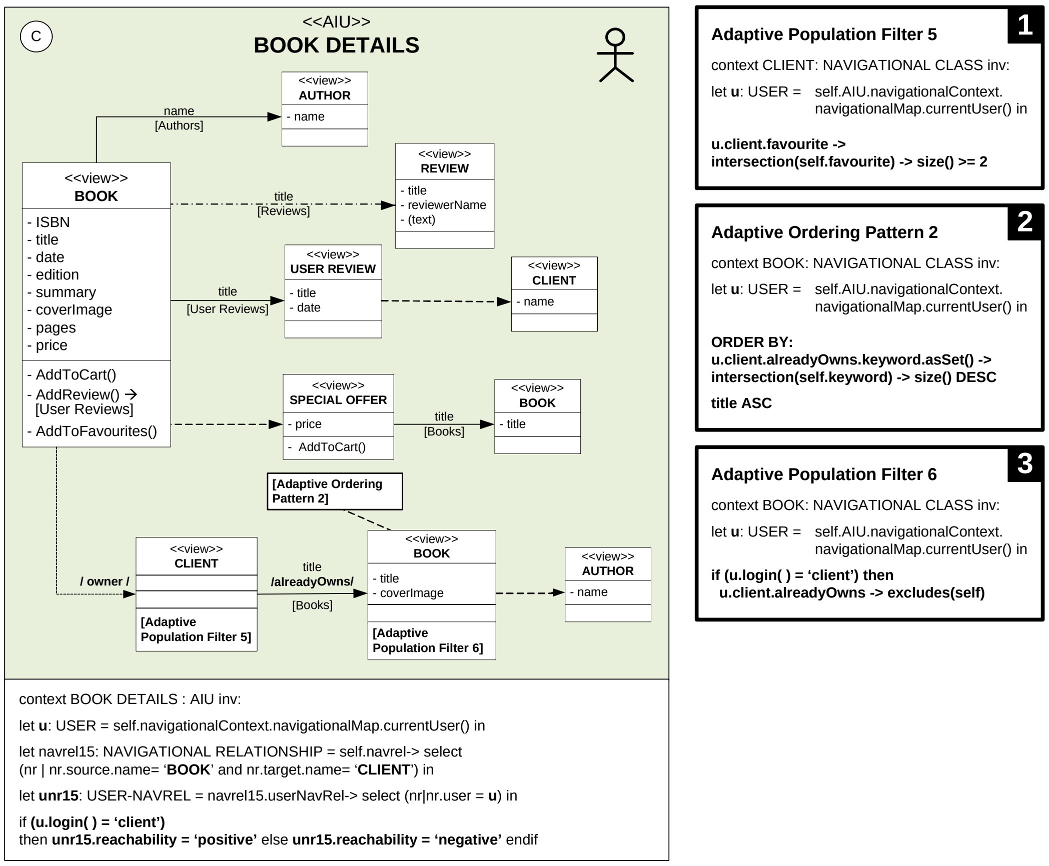
b) **OOWS Navigational Schema:** Objects from a complementary class may have difference relevance to different users. As in the Global Guidance case, their corresponding links can be ordered according to that relevance. By defining an ***Adaptive Ordering Pattern*** (see Section 4.5.3) over the relevant complementary class, different ordering criteria can be defined for different subsets of users.

c) **Adaptive Technique:** This strategy is implemented by means of a ***Link-Ordering*** or a ***Link-Annotation*** technique.

Example:

Figure 5.28 shows a modelling example that comprises the two described options: *selection* and *ordering* of links. Modelled AIU comprises the detailed information about a single instance of `Book`. Navigational relationship to `Client` complementary class allows accessing the definition of a list of books, related to the currently visited one. `Client` instances correspond to `CLIENTS` that have marked the current `book` through the `alreadyOwns` association. At first, from `Client` instances, `Adaptive Population Filter 5` (*Frame 1* of Fig. 5.28) allows selecting those that have similar preferences to the current user. Afterwards, through `Adaptive Population Filter 6` (*Frame 3* of Fig. 5.28), from the other `books` that selected `clients` already `owns`, the system excludes those `books` that the current user has marked through the `alreadyOwns` association. The filtered selection of `books` are finally ordered through `Adaptive Ordering Pattern 2` (*Frame 2* of Fig. 5.28), according to the number of associated `keywords` shared with those the current user already `owns`.

Figure 5.29 shows an example of implementation of the Local Guidance method. Details of the visited `book` are complemented with a list of related `books`. This list contains books purchased by `CLIENT` users with similar preferences than the current one, and which already own the currently visited book. The similarity of listed `books` with those that the current user already owns are expressed through the following presentation criterion: the closer to the left of the page, the more similar the listed `book` is.



```
context BOOK DETAILS : AIU inv:
let u: USER = self.navigationalContext.navigationalMap.currentUser() in
let navrel15: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= 'BOOK' and nr.target.name= 'CLIENT') in
let unr15: USER-NAVREL = navrel15.userNavRel-> select (nr|nr.user = u) in
if (u.login( ) = 'client')
then unr15.reachability = 'positive' else unr15.reachability = 'negative' endif
```

Adaptive Population Filter 5

```
context CLIENT: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in

u.client.favourite ->
intersection(self.favourite) -> size() >= 2
```

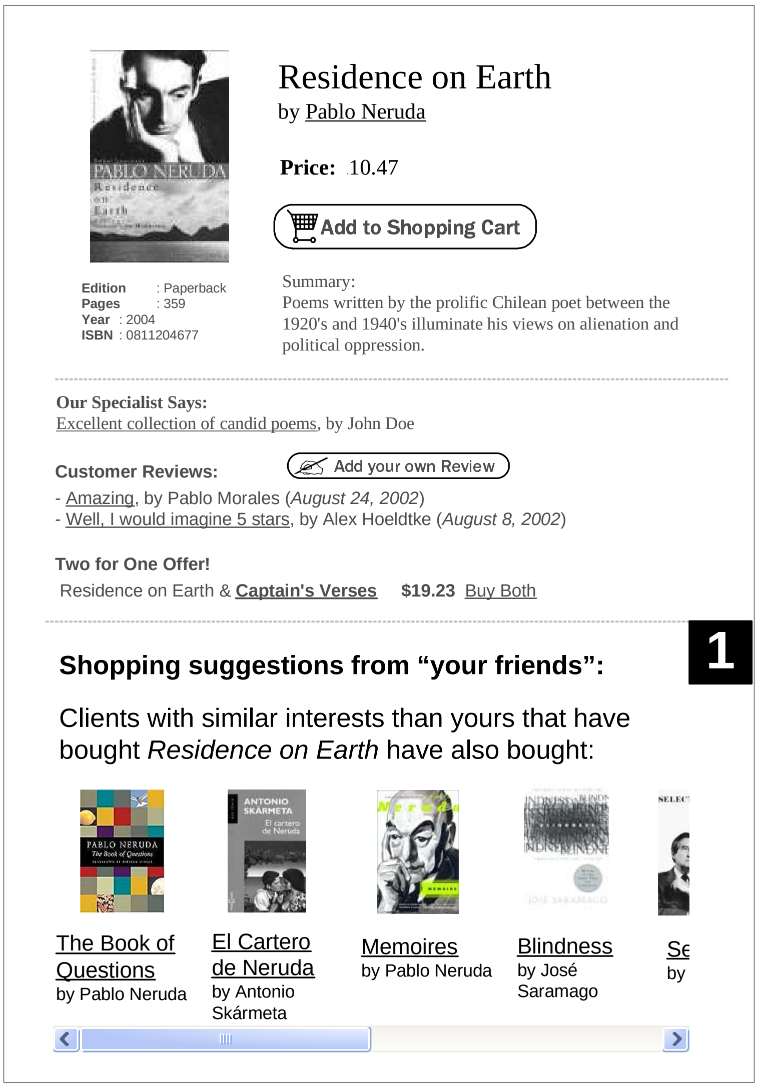
Adaptive Ordering Pattern 2

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in

ORDER BY:
u.client.alreadyOwns.keyword.asSet() ->
intersection(self.keyword) -> size() DESC
title ASC
```

Adaptive Population Filter 6

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in
if (u.login( ) = 'client') then
  u.client.alreadyOwns -> excludes(self)
```

Figure 5.28: Modelling example of *Local Guidance* method



Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews: Add your own Review
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

1

Shopping suggestions from "your friends":

Clients with similar interests than yours that have bought *Residence on Earth* have also bought:

The Book of Questions by Pablo Neruda
El Cartero de Neruda by Antonio Skármeta
Memoires by Pablo Neruda
Blindness by José Saramago
Se by

Figure 5.29: Implementation example of *Local Guidance* method

5.3.3 Global Orientation Support

This method allows orientating users about their absolute position into the hyperstructure of the application. From the information about her previous navigational actions (visited pages, submitted searches, made evaluations, etc.), the user is informed about which pages she is allowed to access and/or which ones are recommended to visit. The proposed modelling strategies for these methods are the following:

Option 1: Selection of links

a) **Class Diagram:** It must include those classes and associations that describe the information items to be listed.

b) **OOWS Navigational Schema:** Given a list of multiple instances from a *List AIU*, each of which provides navigational access to deeper information about the corresponding concept, the definition of an *Adaptive Population Filter* on the manager class allows selecting those links related to the concepts that are most important to the user. Along with suggesting the best alternatives for future navigation (*Global Guidance*), the consequent restriction of the available navigational paths reduces the possible disorientation of the user inside the hyperstructure caused by the access to irrelevant information (*Global Orientation*).

c) **Adaptive Technique:** This strategy is implemented through a Link-Hiding adaptive navigation technique.

Option 2: Sorting and Annotation of links

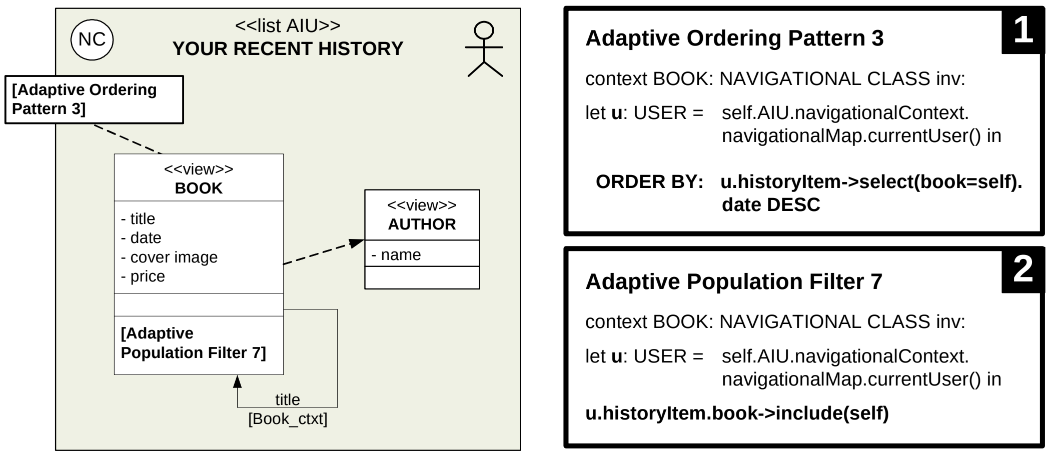
a) **Class Diagram:** It must include those classes and associations that describe the information items to be listed.

b) **OOWS Navigational Schema:** Given a list of multiple instances from a *List AIU*, the definition of an *Adaptive Ordering Pattern* on its manager class allows sorting those instances according to how relevant each one is for a given user.

c) **Adaptive Technique:** When the corresponding order expression is defined in terms of previous navigational actions of the user, the final order gives implicit and/or explicit information about the surrounding pages, giving users valuable information about their current position in the hyperspace, in terms of their previously visited pages, submitted searches, etc., and of the information that may be inferred from those actions. This strategy is implemented through a ***Link-Ordering*** technique. To provide a more effective orientation, values obtained from evaluation of different order criteria for each *(user, concept instance)* pair may be used by presentational specifications to provide textual or graphical clues, which informs about the relative importance of each concept. This variant is implemented through a ***Link-Annotation*** technique.

Figures 5.30 and 5.31 exemplify the modelling of the *Global Orientation* method. In both cases, the recent navigational steps of the current user are described from different perspectives. In Fig. 5.30, an indexed list of books that have been recently viewed by the current user is shown. The inner structure of the corresponding *List AIU* is identical to the modelling example of the index of `Recommended Books`, from the *Global Guidance* method (Fig. 5.23). The differences are in the definition of the respective *Adaptive Population Filter* and *Adaptive Ordering Pattern.* The index of recently viewed `books` selects its instances directly from the `History Item` instances associated to the current user, and sorts them according to the `date` attribute.

Figure 5.31 shows a non-contextual *List AIU* that describes the recent browsing history of the user in terms of the `keywords` associated to the visited `books`. `Keyword` instances are selected by means of an *Adaptive Population Filter* (*Frame 2* of Fig. 5.31), which checks the association of each stored `keyword` to `Book` instances associated to the current user through the `History Item` class. Selected `keywords` are ordered according to how many times each of them is present in the browsing history of the current user.



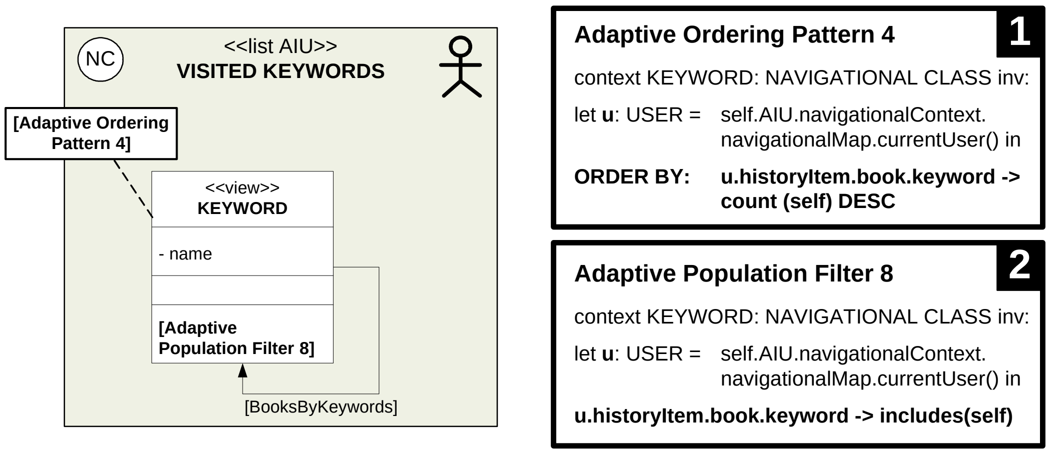
Adaptive Ordering Pattern 3

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =  self.AIU.navigationalContext.
               navigationalMap.currentUser() in
ORDER BY:      u.historyItem->select(book=self).
               date DESC
```

Adaptive Population Filter 7

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =  self.AIU.navigationalContext.
               navigationalMap.currentUser() in
u.historyItem.book->include(self)
```

Figure 5.30: Modelling example of *Global Orientation* method



Adaptive Ordering Pattern 4

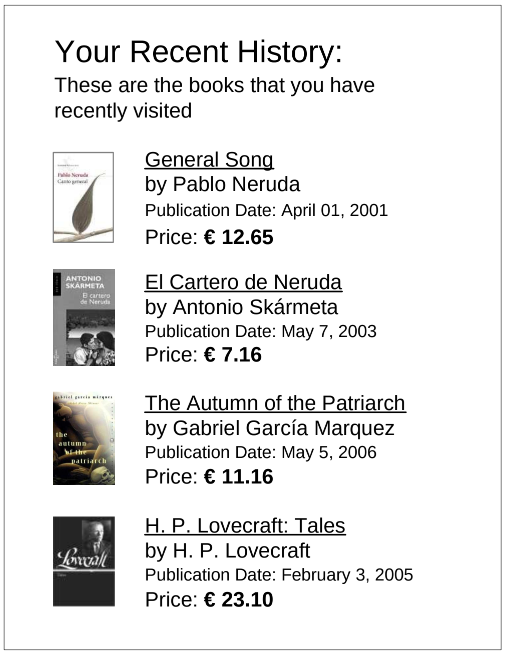
```
context KEYWORD: NAVIGATIONAL CLASS inv:
let u: USER =  self.AIU.navigationalContext.
               navigationalMap.currentUser() in
ORDER BY:      u.historyItem.book.keyword ->
               count (self) DESC
```

Adaptive Population Filter 8

```
context KEYWORD: NAVIGATIONAL CLASS inv:
let u: USER =  self.AIU.navigationalContext.
               navigationalMap.currentUser() in
u.historyItem.book.keyword -> includes(self)
```

Figure 5.31: Modelling of *Global Orientation* method

Figure 5.32 shows two possible implementations of the List AIU's shown in Figs. 5.30 and 5.31, respectively. Left-side page shows an indexed list of books whose pages the current user has recently viewed, in chronological order. More recently visited `books` are closer to the top of the page. The right-side page shows a *cloud of keywords.* In this case, keywords associated to books that have been recently viewed by the current user are shown in different font sizes: those keywords that are more recurrent in the browsing history of the user are shown in bigger size. This is an example of the implementation of a *Global Orientation* method through a *Link-Annotation* technique. Both implementations help users to orientate themselves into the hyperspace, by giving information about their recently taken navigational steps. From their already covered navigational path, users may identify their current position into the application in a more easy way.

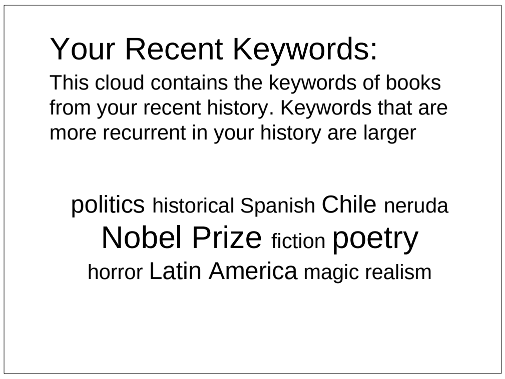

Your Recent History:
These are the books that you have recently visited

General Song
by Pablo Neruda
Publication Date: April 01, 2001
Price: **€ 12.65**

El Cartero de Neruda
by Antonio Skármeta
Publication Date: May 7, 2003
Price: **€ 7.16**

The Autumn of the Patriarch
by Gabriel García Marquez
Publication Date: May 5, 2006
Price: **€ 11.16**

H. P. Lovecraft: Tales
by H. P. Lovecraft
Publication Date: February 3, 2005
Price: **€ 23.10**


Your Recent Keywords:
This cloud contains the keywords of books from your recent history. Keywords that are more recurrent in your history are larger

politics historical Spanish Chile neruda
Nobel Prize fiction poetry
horror Latin America magic realism

Figure 5.32: Implementation examples of *Global Orientation* method

5.3.4 Local Orientation Support

This method permits to orientate users in their relative position into the hyperstructure of the application, by providing them with information about pages that surround the currently visited.

1. **Class Diagram:** Once defined the class corresponding to the main concept of the page, a class that represents a concept that complements the information of the main one must be included. Furthermore, both classes must be related through an association, in which one object of the main class is associated to several objects of the complementary class.

2. **OOWS Navigational Schema:** We propose to adopt identical modelling strategies to select, sort and annotate links than those proposed to the *Global Orientation* method. This decision is founded on the hypothesis that a same adaptive feature can provide either local or global orientation to the user, depending on the domain concepts on which the adaptation is defined. Pages that inform about the most important or global concepts most likely supports the global orientation of the users (e.g., chapters of an e-learning course); while pages that inform about more specific or secondary concepts may help users to know their relative position into the application (e.g., individual contents of the course).

3. **Adaptive Technique:** The adoption of the same modelling strategies than those proposed to Global Orientation involves the use of the same Adaptive Techniques: *Link-Hiding*, *Link-Ordering* and *Link-Annotation*.

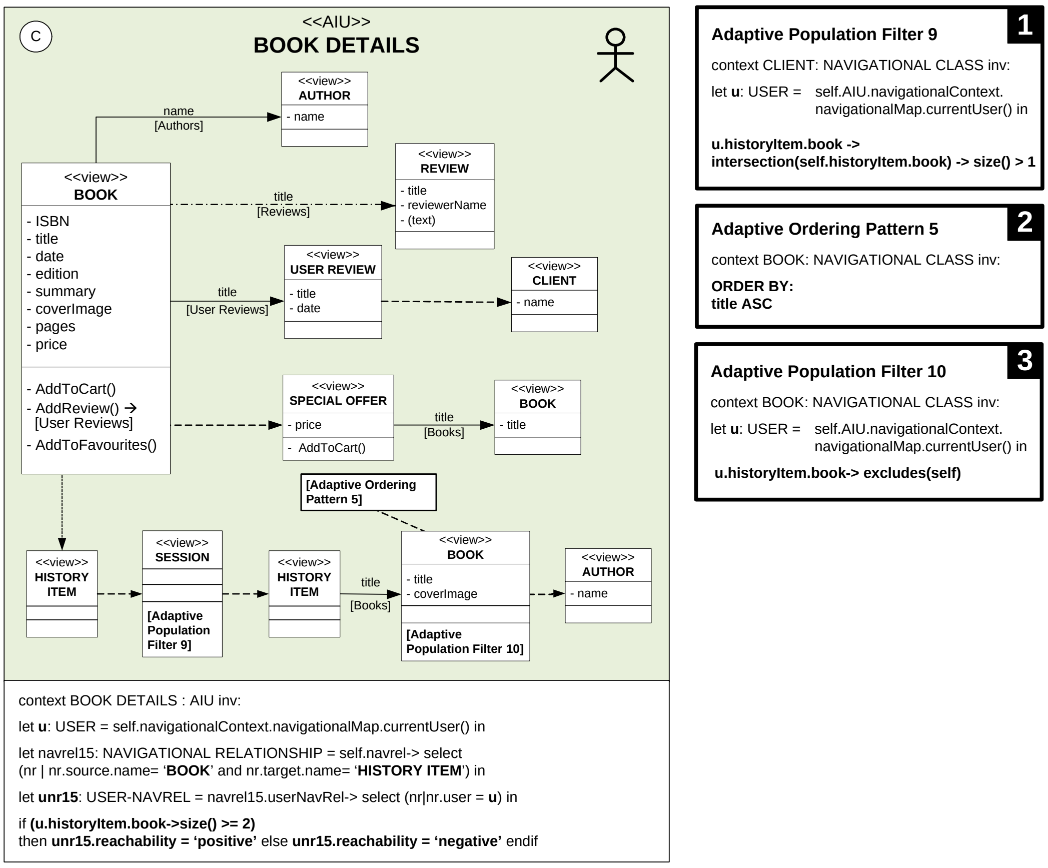
Example:

Figure 5.33 shows the modelling of *Local Orientation* method by means of the selection and sorting of links. The modelled AIU complements the detailed information about a single instance of `Book` with a list of related books, according to their presence in the browsing history of different users (`History Item` complementary class). At first, associated `Session` instances are filtered through the *Adaptive Population Filter 9* (*Frame 1* of Fig. 5.33), selecting those `sessions` whose users has visited at least two `books` that the current `user` has also visited. Afterwards, from the other `books` that selected `users` have already `visited`, the system excludes those `books` that the current user has already `visited`, by means of the *Adaptive Population Filter 10* (*Frame 3* of Fig. 5.33). Finally, the filtered selection of `books` is ordered through the *Adaptive Ordering Pattern 5* (*Frame 2* of Fig. 5.28), according to a non-adaptive criterion (alphabetical order of the corresponding titles).

Figure 5.34 shows an implementation example of the *Local Orientation* method. Like the example of the *Local Guidance* method, details of the visited `books` are complemented with a list of related `books`. In this case, however, this list contains books that have been `visited` by users with a similar `browsing history` than the current user, and which have also visited the currently shown `book`. Listed `books` are alphabetically ordered by their `titles`.

This list of books provides a contextual support to the orientation of users, displaying information of instances that are related to their previous navigational actions during a browsing session. Users can view the navigational history of users that are currently performing similar navigational tasks, what helps to locate theirselves into the hyperspace and also provides alternatives to their next navigational steps.

Additionally, and to illustrate the integration of multiple AIU's in a single context (page), *Frame 2* of Fig. 5.34 offers a simplified version of the *List AIU* displayed in Fig. 5.30, which implements a Global Orientation method. By discarding some of the attributes of the manager class of that AIU, the shown implementation provides a non-contextual, more direct information about the navigational steps taken by the user.



```
context BOOK DETAILS : AIU inv:
let u: USER = self.navigationalContext.navigationalMap.currentUser() in
let navrel15: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr | nr.source.name= 'BOOK' and nr.target.name= 'HISTORY ITEM') in
let unr15: USER-NAVREL = navrel15.userNavRel-> select (nr|nr.user = u) in
if (u.historyItem.book->size() >= 2)
then unr15.reachability = 'positive' else unr15.reachability = 'negative' endif
```

Adaptive Population Filter 9

```
context CLIENT: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in
u.historyItem.book ->
intersection(self.historyItem.book) -> size() > 1
```

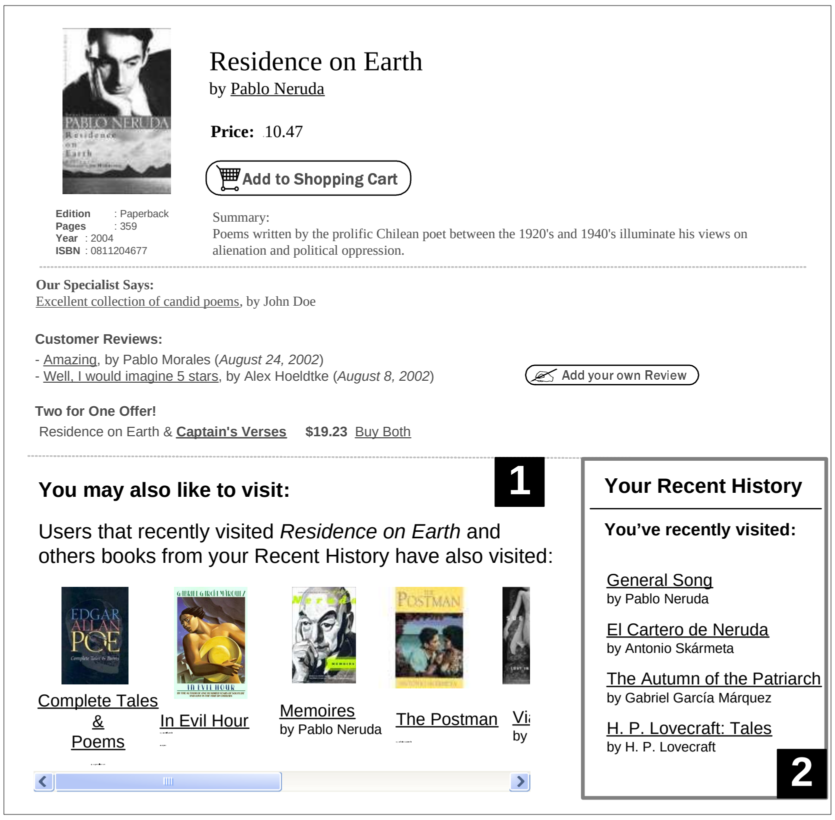
Adaptive Ordering Pattern 5

```
context BOOK: NAVIGATIONAL CLASS inv:
ORDER BY:
title ASC
```

Adaptive Population Filter 10

```
context BOOK: NAVIGATIONAL CLASS inv:
let u: USER =   self.AIU.navigationalContext.
                navigationalMap.currentUser() in
u.historyItem.book-> excludes(self)
```

Figure 5.33: Modelling of *Local Orientation* method



Residence on Earth
by Pablo Neruda

Price: 10.47

Add to Shopping Cart

Edition	: Paperback
Pages	: 359
Year	: 2004
ISBN	: 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems, by John Doe

Customer Reviews:
- Amazing, by Pablo Morales (*August 24, 2002*)
- Well, I would imagine 5 stars, by Alex Hoeldtke (*August 8, 2002*)

Add your own Review

Two for One Offer!
Residence on Earth & **Captain's Verses** **$19.23** Buy Both

You may also like to visit:

1

Users that recently visited *Residence on Earth* and others books from your Recent History have also visited:

Complete Tales & Poems

In Evil Hour

Memoires
by Pablo Neruda

The Postman

Vi
by

Your Recent History

You've recently visited:

General Song
by Pablo Neruda

El Cartero de Neruda
by Antonio Skármeta

The Autumn of the Patriarch
by Gabriel García Márquez

H. P. Lovecraft: Tales
by H. P. Lovecraft

2

Figure 5.34: Implementation of *Local Orientation* method

5.4 Conclusions

This chapter has introduced a set of modelling strategies to specify Adaptive Methods at a high abstraction level. This approach complements the introduction of Adaptive Primitives, aimed at providing not only conceptual tools with adaptive behaviour, but also some guidelines in the use of these tools to specify adaptive aspects from a broader perspective, which allows giving answers to concrete interaction problems. The definition of these strategies intends to make a step ahead in the adoption of a Model-Driven proposal to Web Adaptation, making an additional contribution to developers in comparison with the approaches reviewed in the State of the Art of this dissertation (Chapter 2).

One of the main problems in the definition of these strategies was to leverage the definition of Adaptive Methods. In Adaptive Hypermedia works, these methods have been mostly implemented in educational software, what strongly influences their definition. This is specially evident in Adaptive Presentation methods, which conceives the adaptation of the contents in terms of concepts like “explanations”, “knowledge level”, etc.

The definition of Adaptive Methods has been made from the observation of already implemented adaptive hypermedia systems [10]. As a consequence of the low abstraction perspective from which adaptivity has been traditionally considered, the specification of different adaptive methods are very similar from a higher abstraction level. For instance, when comparing the introduced modelling examples of Global and Local Guidance with those of Global and Local Orientation, we may visualize that the inner navigational structure is almost identical. The differences are in the adaptivity primitives defined in each case and on the user characteristics on which the definition of these primitives are based. While *Guidance* methods are mainly oriented to support the next navigational actions of the current user, *Orientation* methods are intended to help users to orientate themselves into the application. For this reason, modelling of *Guidance* methods should be based on user characteristics that inform about their preferences, goals, interests and, in general, long-term characteristics, in order to provide high-quality suggestions or guides to future navigation. On the other side, modelling of *Orientation* methods should be based on characteristics that describe more immediate goals of users, their navigational actions during a browsing session, visited pages, submitted search actions and, in general, short-term characteristics, in order to provide a more effective orientation during a specific browsing session.

From the Adaptive Navigation Methods classified by the Adaptive Hypermedia community in [8], this proposal does not include a modelling strategy to the ***Personalized Views*** method, as seen in Fig. 5.20. This method has been mostly used as a complement of Information Retrieval systems [78], with the aim to support the organization of retrieved information nodes into the page or interaction unit. However, most implementations that make use of this method are *adaptable*, i.e., their initially proposed organization of contents can be adapted, but only through explicit intervention of the own user. Some examples of this type of adaptable Web applications are the well-known *My Yahoo* (http://my.yahoo.com) and *iGoogle* (http://www.google.com/ig). With few exceptions, applications that adopt a *Personalized Views* method do not adapt the organization of retrieved contents in an automatic way, i.e., they are not *adaptive*, which are the focus of the present proposal.

A possible modelling strategy for this method should consider the adaptive access to the different sections of a Web page. By modelling those sections as different navigational views of the domain schema of the application, at least an adaptive *Visibility* property should be incorporated to the *Abstract Information Unit (AIU)* structure. However, this property would make possible only to show and hide a given section. The adaptation of its position in the interaction unit obeys to interface preferences and usability criteria that are out of the scope of the present work.

In summary, these modelling strategies are conceived to help developers in the conceptual specification of Adaptive Web Applications, by providing them with guidelines in the use of the introduced Adaptive Primitives. As another complement of these primitives, in the next chapter we introduce a Requirements Model that supports the specification of Adaptivity Requirements.

Chapter 6

Requirements Specification for Adaptive Web Applications

6.1 Introduction

The introduced set of Adaptive Primitives provides OOWS method with conceptual elements to model Adaptive Web Applications. However, the definition of a solid proposal for the development of these systems is not complete if the modelling decisions adopted in the use of these primitives are not supported by the specification of the corresponding requirements. For most of the current Model-Driven proposals of Adaptive Web Applications, adaptivity is only a concern from the conceptual modelling phase on, and modelling decisions are not subordinated to well-specified requirements. In order to provide a more complete support to the development of this kind of systems, OOWS method needs to incorporate a requirements model that deals with the specification of adaptivity requirements.

OO-Method faces the specification of functional requirements of the application from different perspectives [71, 73, 72]. OOWS describes the navigational requirements of Web applications through the definition of a hierarchy of tasks and a detailed description of each elementary task in terms of user-system interactions. Besides these kinds of requirements, the development of an Adaptive Web Application needs a clear specification of the adaptive features that the system will provide, the distinct groups of their potential users and the specific characteristics on which the adaptivity will be based.

This chapter introduces a Requirements Model for Adaptive Web Applications, extending the proposals of OO-Method and OOWS through conceptual structures that support the specification of the following two groups of requirements:

1. ***User-related Requirements***: this group describes the information about potential users of the application that is relevant to the application's goals and, in particular, to its adaptation purposes. It comprises the definition and classification of groups of similar users, along with their distinctive characteristics.
2. ***Adaptivity Requirements***: this group describes the adaptive functionality that the system is required to provide to its users, by adapting the access to contents and links to their particular characteristics.

In the next section, we present a description of the current approach of OOWS to Requirements Specification. From this description, we analyze which are its main deficiencies in relation to the support of the specification of Adaptivity Requirements. Next, the proposal for specifying User-related and Adaptivity Requirements is introduced. Finally, we describe how the Requirements Specification process is affected by the new proposed specifications.

6.2 OOWS Requirements Specification Phase

OO-Method proposes different approaches to the specification of requirements, adopting different perspectives: it describes functional requirements by means of a hierarchical structure, which is detailed through Use Case diagrams [71]; it adopts an organization modelling approach by using the *i** framework [73]; it adopts a business process modelling through the *BPMN* notation [79]; and it proposes an strategy of natural language processing from the definition of Use Case diagrams [72].

All these strategies share the specification of functional requirements, by means of the definition of a hierarchical structure that classifies the relevant functions of the system. In this work, we adopt this approach as the main proposal of requirements specification of OO-Method. Furthermore, it is used as the functional connection with the OOWS proposal of navigational requirements specification [24].

6.2.1 Specification of Functional Requirements

According to the taxonomy of requirements of Web Applications introduced in [80], the OO-Method approach for the specification of functional requirements provides high-level tools to describe the following two groups of requirements:

- **Data Requirements**, also known as structural requirements, which establish how information is stored and administrated by the application.
- **Transactional Requirements**, also known as behavioural requirements, which express what the application has to compute internally, without considering interface and interaction aspects.

The specification of these requirements is made by applying the following three techniques:

- **Mission Statement**: in a general sense, it describes the purpose of the system in one or two sentences and the system's major responsibilities.
- **Function Refinement Tree**: it describes the main functional groups of the system, which contribute to the fulfilment of the specified mission.

- **Use Cases Definition**: it specifies the behavior of the elementary functions at a very high-level, showing the communication between the environment (actors) and the system.

The following subsections present a more detailed description of these techniques.

6.2.1.1 Mission Statement

The mission of the system is the most general service (the main goal) that the system provides. The *Mission Statement* (or *Statement of Purpose*, according to the *Yourdon System Method* (YSM) [81]) is a high-level description of the nature and purpose of the system. It consists of three elements:

- *Definition.* This is a short description of the kind of system to be designed.
- *Purpose.* This is an overall indication of the purpose of the system.
- *Responsibilities.* Each responsibility is a general functionality of the system.

A possible Mission Statement for the specification of an on-line Bookstore is the following:

"(Definition) *The system to be developed is an E-commerce Web application.* (Purpose) *The main purpose of the system is to provide support for the on-line sale of Books.* (Responsibilities) *To achieve this, the user must be able to consult information about the products, add them to the shopping cart and send purchase orders. The user must also be able to consult the state of her shopping cart at any time. The system administrator must be able to manage the information of products and clients."*

The main goal of defining the Mission Statement is that customers, users, designers, programmers, testers, and all other stakeholders keep the system mission in mind during the entire software development process, to avoid disagreements about what is the actual purpose of the system.

6.2.1.2 Function Refinement Tree

The functions that the system will provide to its environment are organized in a refinement hierarchy, called ***Function Refinement Tree (FRT)***. The root of this hierarchy is the *Mission Statement*, whereas the leaves are the *elementary functions*. The intermediate nodes are groups of elementary functions and usually represent a kind of activity or a business area that the system supports. The distinction between intermediate and leaf nodes can be made by following the definition of Wieringa [82]: *"a function is regarded as elementary if it is triggered by an event sent by a user of the system (actor) or by the occurrence of a temporal event"*.

FRT is used to represent a hierarchical decomposition of business functions of a system independently from the actual system structure. Moreover, it gives the entry point for building the Use Case Model, avoiding to start from scratch and the potential problem of mixing the abstraction level of Use Cases. Figure 6.1 shows an example of an FRT for an online bookstore.

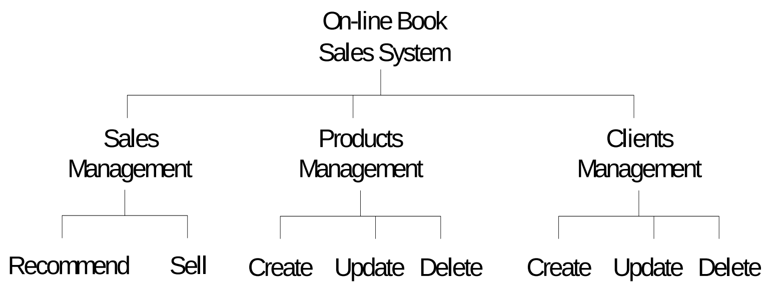


Figure 6.1: Example of a Function Refinement Tree

6.2.1.3 Use Cases Definition

Each elementary function from the Function Refinement Tree is detailed in a Use Case Diagram. Introduced in [83], the ***Use Case*** concept represents an interaction between the system and an external entity during the execution of a function. The main benefits of Use Cases can be summarized in the following points:

- They only describe the externally visible behavior of a system, thus avoiding solution bias and premature design.
- They can be used for both elicitation and description of requirements.
- They inherit the comfort and ease of natural language specifications, but avoid many of the problems of a purely narrative specification.

A Use Case Diagram shows the distinct Use Cases that describe how the system should interact with the actors (external entities) to achieve the corresponding function. Figure 6.2 shows the specification of a Use Case Diagram corresponding to the `Sell` elementary function from the FRT shown in Fig. 6.1.

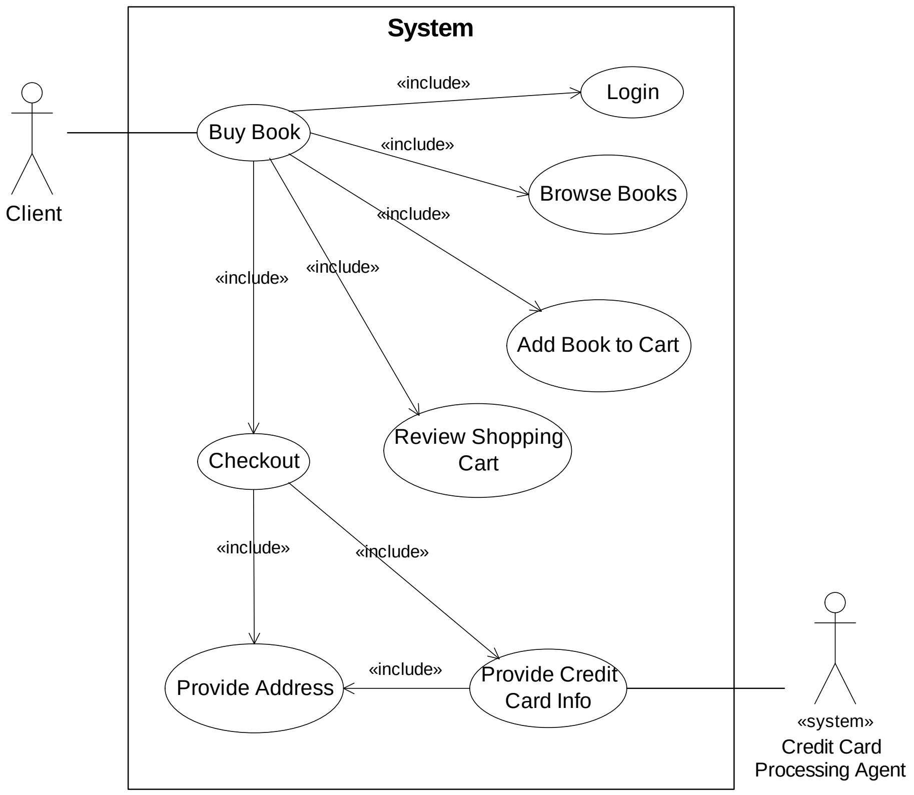


Figure 6.2: Example of a Use Case Diagram

In the example of Fig. 6.2, the provision of the `Sell` function involves the execution of the `Buy Book` Use Case by the `Client` actor. The execution of this process in turn involves performing other use cases (what is graphically represented through ≪include≫ relationships), which represent the different processes in which the actor must interact with the system to obtain the corresponding functionality.

The use case definition depicts a very preliminary definition of different categories of the users that are allowed to access the system functionality, from the set of defined Use Case's actors. As seen in Fig. 6.2, from both actors of the diagram, the `Client` actor clearly refers to a group of users. This is not the case of `Credit Card Processing Agent` actor, which rather refers to an external software component.

6.2.2 Specification of Navigational Requirements

The proposal described above, as other proposals of requirements engineering based on Use Cases (such as the proposals by Constantine et al. [84], Jaaksi [85] or Rosenberg et al. [86]) and on Scenarios (Leite et al. [87]), allows specifying structural and behavioural requirements. However, a web application requires considering other specific aspects that are not properly addressed by these methods [88], specially in the specification of navigational aspects.

OOWS incorporates a proposal [24] to specify navigational requirements. It makes use of the *task* metaphor [89], which has been widely used for capturing functional requirements, with the goal of capturing the navigational structure of the system, by introducing information about the interaction between the user and the system.

The specification of these requirements is made through the following mechanisms:

- *Tasks Diagram*, which classifies the navigational tasks that the system must provide;
- *Elementary Tasks Description*, which describes each elementary task from the previous diagram;

- *Achievement of Elementary Task*, which specifies the user-system interactions needed to achieve a given elementary task, through an UML-compliant *Activity Diagram* [90].
- *Data Description*, which specifies the data items that are stored and reachable by users.
- *Users Specification*, which describes the different groups of users that can interact with the system.

In the following subsections, each of these mechanisms is described.

6.2.2.1 Tasks Diagram

In this step, the tasks that the Web application must provide to the users are identified and specified by means of a ***Tasks Diagram***. Just like the OO-Method's Function Refinement Tree (FRT), this diagram is a hierarchical tree, whose base is the ***Statement of Purpose*** or ***Mission Statement*** of the system. This is considered the most general task, from which a progressive refinement is performed until the level of elementary tasks is reached. An ***elementary task*** is a task in which atomic actions are obtained when it is decomposed.

Complementing this diagram, temporal constraints between two sibling tasks are expressed through CTT (ConcurTaskTree) temporal constraints[91]. The main constraints that this proposal considers are:

- T1 [ ]$\gg$ T2, *Enabling with information passing*: when T1 task terminates it provides a value for task T2 besides activating it;
- T1 |$\gg$ T2, *Suspend-Resume*: T1 can be interrupted by T2. Once T2 terminates, T1 can be resumed;
- T1 [ ] T2, *Choice*: one of the tasks can be chosen and only the chosen task can be performed; and
- T1*, *Iteration*: the task is iterative.

Figure 6.3 shows the *Tasks Diagram* of an online bookstore. For visibility purposes, elementary tasks have been circled with a thicker line. The statement of purpose of the application corresponds to `Purchase Products` expression. It is decomposed into two excluding tasks (related by a *"Choice"* dependency), which describe the purchase and information administration tasks, respectively. `Purchase` is decomposed into `Make Purchase` and `Identify User`, which are related by a *"Suspend-Resume"* dependency, i.e., a user can give his/her identification while the purchase is being made, and then resume it. `Make Purchase` is decomposed into `Collect Items` and `Checkout`, related by an '*'Enabling with Information Passing"* dependency, i.e., books must be firstly collected into the shopping cart before the checkout.

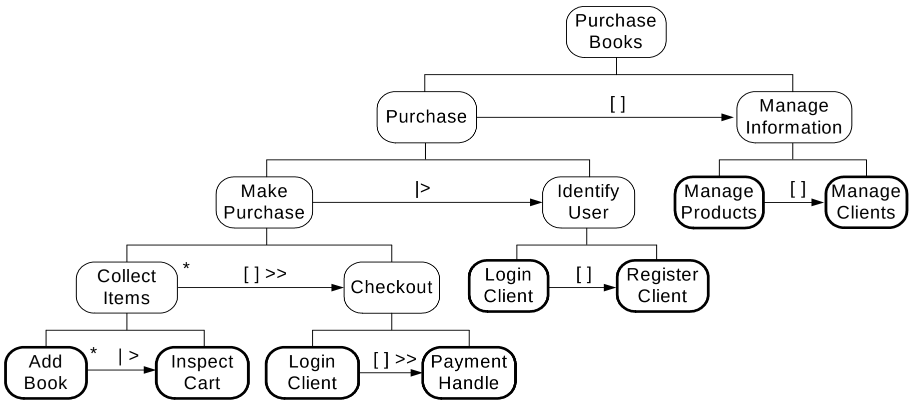


Figure 6.3: Tasks Diagram

6.2.2.2 Elementary Tasks Description

Relevant aspects of each elementary task are specified through a tabular description, which contains the following characteristics:

- *Goals*: objectives that must be reached from the task's achievement.
- *Users*: the groups of users that are allowed to achieve the task.
- *Frequency*: information of the relative frequency of the task's occurrence.

- *Additional Constraints*: non-functional conditions that must be fulfilled in order to complete the task.
- *Achievement Description*: the sequence of user-system interactions that are needed to achieve the task.

Task	:	Add Book
Description	:	The user adds a Book to her Shopping Cart
Users	:	Anonymous, Clients
Frequency	:	100 times/hour
Additional Constraints	:	-------------

Figure 6.4: Documentation of elementary tasks

Figure 6.4 shows a partial description of the `Add Book` elementary task. Due to its higher complexity, the description of the task's achievement deserves special attention, and it is described in the following section.

6.2.2.3 Achievement of Elementary Tasks

The achievement of an elementary task is described in terms of the interaction that users require of web applications. Along with describing the actions performed by the user and by the system, this proposal introduces the concept of ***Interaction Point (IP)***, representing the exact moment when the user interacts with the system. This concept allows specifying the sequence of activities that can be executed and also the navigational steps that must be followed to do it.

An elementary task is described as a process where the system carries out several actions, sometimes delaying them in order to interact with the user by means of an IP.

1. ***Output Interaction***: the system provides the user with access to data or operations, both related to an ***entity*** (an object of the real world that belongs to the system domain [24]). In the first case, the user

can select a particular element from the corresponding entity (called ***information instance***) and obtain more information related to her selection; in the second case, the user can activate the operation, and the system carries out an action as a result.

2. ***Input Interaction***: the system requests the user to introduce information of an entity, and uses it as an input to perform a specific action (for instance, carrying out an on-line purchase from the provided information about a client). This kind of interaction only allows users to input the requested information.

The *actions* that the system performs in the achievement of a given task can be classified into the following types:

1. *Functionality Execution*, which are actions that change the system state.
2. *Information Search*, which are actions that only query the system state.

To describe the tasks that systems provide in terms of the interactions with users, these concepts are included in UML activity diagrams [90]. Each elementary task is represented by one activity, by considering the following characteristics (see Fig. 6.5):

- Each node (action) represents an Interaction Point (IP) (solid line) or a System Action (dashed line). IPs are stereotyped with the *Output* or the *Input* keyword to indicate the interaction type. System actions are stereotyped with the *Function* or the *Search* keywords to indicate their types.
- In the case of Output IPs, the number of information instances that the IP includes (cardinality) is depicted with a small circle in the top right side of the primitive.
- In the case of Input IPs, in order to indicate that this kind of IP exclusively depends on a system action (the user introduces information which allows the system to correctly perform an action) and does

not take part in the general process of the task, nodes that represent both elements (input IP and system action) are encapsulated in dashed squares.

- Each arc represents:

  1. a *user action* if the arc source is an IP; or
  2. a *node sequence* if the arc source is a system action.

  If an arc represents a user action (the arc source is an IP), it can either be an activation of an operation (if the arc target is a system action) or an information selection (if the arc target is another IP).

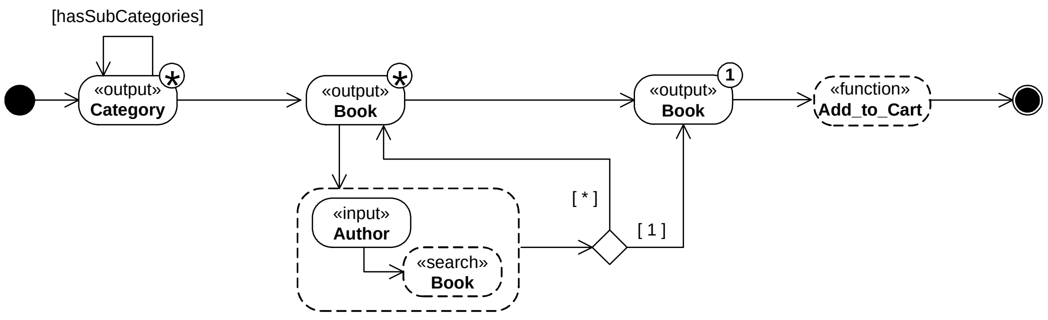


Figure 6.5: Specification of an elementary task achievement

Figure 6.5 shows the Activity Diagram that describes the achievement of the `Add Book` elementary task. This task starts with an Output IP, where the system provides the user with a list (cardinality *) of book categories. From this list, the user can select a `Category`. If the selected `Category` has sub-`categories`, the system again provides the user with a list of (sub-)`categories`; if not, the system informs about the `Books` of the selected `Category` by means of an Output IP. The user may perform two actions from this IP: (a) select a `Book`, and the system provides the user with a description of the selected `Book`; or (b) Activate a search operation, and the system performs a system action which searches for the `Books` of an `Author`. To do this, the user must introduce the name of an `author` in the Input IP. If the search returns only one `Book`, the system provides the user with its detailed description; otherwise, the system provides the user with a set of `Books`. Finally, when the user has obtained a `Book` description, she can activate the `Add_to_Cart` operation, and the system performs an action which adds the selected `Book` to the shopping cart.

6.2.2.4 Data Description

The information that must be stored in the system is defined by means of a template technique based on CRC Cards [92]. One data template is defined for each entity from the Activity Diagram, comprising an *identifier*, the *entity name*, and a specific *data section*.

The *Data Section* describes each specific feature associated to the entity, through its name, a description, and the IPs (if any) in which the feature is considered as part of the user-system interaction. To identify an IP, the following notation is adopted: *Output (Entity, Cardinality)*, for Output IPs, and *Input (Entity, System Action)*, for Inputs IPs.

Identifier	O1	
Specific Data	Book	
Name	***Description***	***IPs***
ISBN	ISBN code	Output(Book, 1)
title	Title of the book	Output(Book, *); Output(Book, 1)
date	Publication date	Output(Book, *); Output(Book, 1)
summary	Brief summary of the book	Output(Book, 1)
cover image	Image of the book's cover	Output(Book, *); Output(Book, 1)
pages	Number of pages of the book	Output(Book, 1)
author's name	Name of the book's author	Output(Book, *); Output(Book, 1)
price	Purchase price of the book	Output(Book, *); Output(Book, 1)
keyword	List of keywords associated to the book	

Figure 6.6: Data template for *Book* entity

According to the template in Fig. 6.6, the information about a `Book` that the system manages is: its `ISBN` code number, a `summary` of the book and its number of `pages`, which are only shown in Output(Book,1); its `title`, `date` of publication, a `cover`'s `image`, its `author` and `price`, which are shown in the Output(Book,1) and Output(Book,*) IPs; and the list of associated `keywords`, not shown in any IP.

6.2.2.5 Users Specification

Not included in the Requirements Specification phase, but as a preliminary step of the Navigational Modelling phase, OOWS defines the groups of users that can interact with the system. Types of users are organized in a hierarchical way by means of inheritance relationships. In this context, child types of users can inherit the navigational semantics associated to their parent, which permits to reuse navigational descriptions. This model categorizes the types of users into three groups:

1. *Anonymous users* (depicted with the '?' symbol). This type of users represents users that are not logged into the system.
2. *Registered users* (depicted with a padlock symbol). They refer to users that are identified in the system as valid users.
3. *Generic users* (depicted with a cross symbol).These types of users are used in the diagram as a mechanism to represent abstract users (users that can not be instantiated).

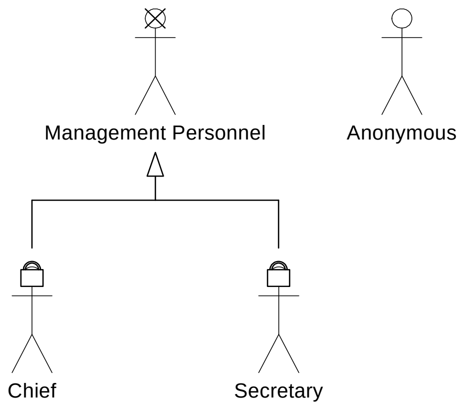


Figure 6.7: OOWS User Diagram

An example of a graphical representation of this model is presented in Fig. 6.7. In the example, the `Management Personnel` user has been defined as a generic user. This means that this kind of user will be associated to a navigational model which will be enriched by the models defined for their inherited types of users. This inheritance mechanism allows reusing navigational models that have been defined by different groups of users.

6.2.3 Adaptivity requirements in OOWS

The current approach of OOWS method to Requirements Specification provides developers with a set of high-level diagrams that allow specifying mainly the transactional or functional requirements of the application. From a global perspective, the *Function Refinement Tree* shows the set of tasks that the system is committed to provide and the hierarchical relationships among them. Meanwhile, the definition of *use cases* for each elementary task provides a detailed specification of the interactions with the system environment that are needed to fulfil the task. These two diagrams guide the definition of the business logic of the application and, with the complement of other techniques of information requirements, of the data repository.

In order to support the specification of requirements related to the navigational capabilities of hypermedia applications, the OOWS Web Application Method, as an extension of OO-Method, proposes the incorporation of two mechanisms: the first one, called *Task Diagram*, may be seen as an extension of the hierarchical Function Refinement Tree, by incorporating temporal dependencies between pairs of tasks. The second diagram is an extension of the UML Activity Diagram [90], oriented to describe the specific navigational requirements of each elementary task from the Task Diagram.

In the specification of requirements of Adaptive Web Applications, Requirements Engineers face the need to describe different alternatives to access the contents and links of the application, and to associate each of them to different users, according to their particular characteristics.

However, the OO-Method and OOWS approaches do not support neither the description of the relevant user groups, nor the relevant characteristics of those users. Only mechanisms to obtain a very preliminary definition of user hierarchies are provided, with the sole purpose of supporting different access levels to the whole application. In this way, the requirements specification task is made regardless of user characteristics. Definition of adaptivity requirements is not possible in this scenario.

For this reason, the OOWS proposal of Requirements Specification must be enhanced in order to support adaptivity requirements. This enhancement must comprise mechanisms to describe the required information of the relevant users of the application, the different variants of presentation of contents and links that the adaptive application will provide; and the conditions that users must fulfil to gain access to those variants.

The following sections introduce the proposed process of Requirements Specification for Adaptive Web Applications, as an extension of OOWS Requirements Model. This process is composed of three main phases:

1. ***Specification of User-related Requirements***. In this phase, the requirements of information about the intended users of the application are specified. This is made by defining the relevant user groups of the application and the user characteristics that must be captured and maintained in order to define the adaptive functionalities of the application.

2. ***Identification and Description of Adaptive Tasks***. This phase adopts the *Task Diagram* from OOWS to describe the set of tasks provided by the system. From the defined elementary tasks, Adaptive Tasks are identified, i.e., those tasks whose fulfilment requires the implementation of adaptive functionalitites. The specification of the adaptive interactions that must be offered in each adaptive task is made by adopting the *Use Cases* approach from the OO-Method proposal.

3. ***Specification of Adaptive Task Achievement***. In this phase, the interactions of each Adaptive Task are described through UML Activity Diagrams. The specification of the adaptive access to information and functionality is made through OCL [56] constraints, which are defined in terms of the user characteristics described in the first phase. The data that are involved in the described interactions are described as proposed by OOWS, through CRC Cards, but also including user-dependent constraints that allow defining variants of the inclusion of data items in different interaction units.

6.3 Specification of User-related Requirements

User-related requirements refer to the required information of the intended users of the application. These requirements correspond to characteristics of the potential users that the system needs to manage in order to achieve an adaptive interaction. These characteristics may belong to one of the following groups:

- *Personal characteristics*, which describe users in terms of individual and objective facts that are independent of the application domain. For instance: name, address, age, city of residence, etc.
- *Domain-dependent characteristics*, which describe users in terms of their relationship with the knowledge domain of the application. For instance: knowledge level about a domain concept, preferences for a given type of products, etc.
- *Behaviour characteristics*, which describe users in terms of their actions when interacting with the applications. For instance: visited pages, submitted searches, executed operations.

Once identified, these characteristics are organized by means of two mechanisms, which describe the relevant users from two perspectives:

- In terms of their shared characteristics, through the hierarchical *User Stereotypes Diagram*;
- In terms of their distinctive characteristics, through the tabular *User Specifications.*

The *User Stereotypes Diagram* allows defining groups of similar users, according to their characteristics in common. Users from the same group, however, may present some differences. Characteristics that are important to consider for each user group and whose values may be different for different users, are described through the *User Specifications.* Both perspectives are described in the following subsections.

6.3.1 User Stereotypes Diagram

From the set of possible users of a Web application, many of them likely share some characteristics: age, expertise in a certain discipline, preferences for a given product, frequent navigational paths when interacting with a given Web application, etc. Developers of Adaptive Web Applications can take advantage of this similarity, by grouping shared features in ***user groups or stereotypes***.

A *stereotype* is a cluster of characteristics (as described by by Rich in [23]). This concept is frequently used to characterize people, when a person gets an idea of how a given type of person is by assigning it a set of characteristics. To deal with the complexity of characterizing the world, people use stereotypes as a way of simplification and categorization. Stereotypes are a means of representing partial descriptions of frequently occurring situations. When it comes to describing people, their significative characteristics are characterized, while omitting the nonsignificant ones. Rich concludes that *"the use of stereotypes, when combined with the ability to record explicit statements by the user about himself and to make direct inferences about a user from his behavior, may provide a powerful mechanism for creating computer systems that can react differently to different users"* [23].

The use of stereotypes in existing adaptive hypermedia systems and the intuitiveness of the obtained descriptions strongly supports the decision to incorporate this concept in the specification of user-related requirements.

The present work proposes to specify the relevant user groups of an Adaptive Web Application by means of a hierarchical ***User Stereotypes Diagram***. This is made by performing the following steps:

1. ***Identification of relevant stereotypes***. In this step, user groups (*stereotypes*) that are relevant to the application are identified. Stereotypes may range from very general ones, such as man or woman, which might be appropriate in a wide variety of domains, to very specific ones, such as reader of premium client, which are appropriate only in specific systems and domains.

2. ***Definition of hierarchical tree of stereotypes***. From the identified stereotypes, the most general ones may be classified into several more specific ones. In this step, this classification is expressed by defining generalization relationships between a given stereotype and the set of its sub-stereotypes, which inherit the characteristics of the "super-stereotype".

3. ***Identification of stereotype's triggers***. The trigger of a stereotype is an event whose occurrence signals the appropriateness of the stereotype. A trigger may be a performed operation or the fulfilment of a condition. When a given user activates a trigger, that user is classified into the corresponding stereotype. In this step, triggers for each stereotype are defined.

From the first two steps, we obtain a *Hierarchical Tree*, whose nodes correspond to the detected stereotypes and whose branches represent the partial "*generalization of*" ordering relation. By complementing this graph with the needed stereotype's triggers, the *User Stereotypes Diagram* is finally obtained. As a general rule, the present proposal compels to define the *USER* stereotype as the most general stereotype for any Adaptive Web Application. It represents the set of all individuals that are potential users of the system, and has no trigger associated. *USER* is therefore the *root* of any Diagram of User Stereotype.

By adopting the terminology of the *Tree* Data Structure, we may characterize the relationships between stereotypes in the following way. Let us consider A, B as stereotypes from the same User Stereotypes Diagram D. We define the following concepts:

- A is the *parent* of B if A is one step higher in the hierarchy and closer to the root stereotype of D.
- A is a *child* of B if B is the parent of A.
- A and B are *siblings* if they have the same parent stereotype in D.
- A is an *ancestor* of B if A is the parent of B or A is the ancestor of the parent of B.

Note that the characteristics that two users from the same stereotype have in common are, at least, the trigger of that stereotype plus the set of triggers of the ancestors of that stereotype.

Figure 6.8 shows a User Stereotypes Diagram for the case study corresponding to an online bookstore.

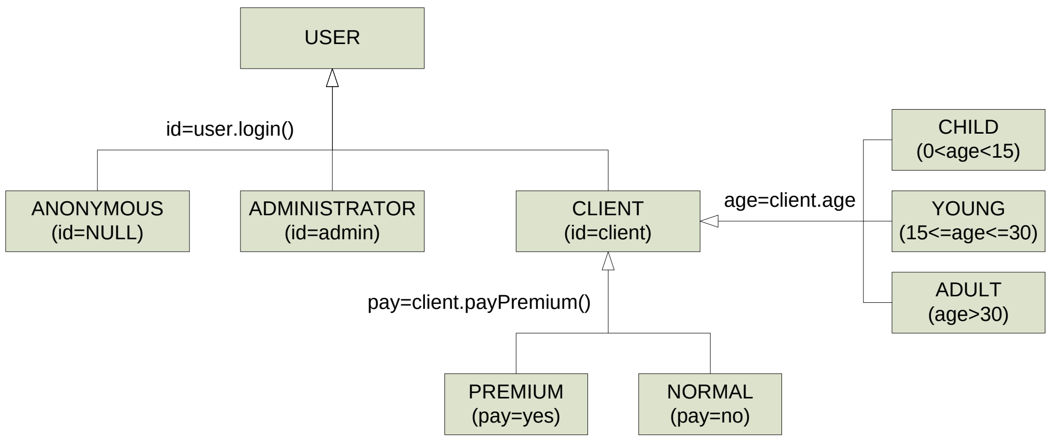


Figure 6.8: User Stereotypes Diagram for the case study

In the diagram of Fig. 6.8, the most general stereotype is `USER`, which characterizes anyone accessing the Web application. This stereotype is a generalization of three more specific stereotypes: `ANONYMOUS`, which comprises those users that previously identified as registered users of the application and/or that have not been logged into the system yet; `CLIENT`, which describes the users registered as clients; and `ADMINISTRATOR`, which represents those users that perform the tasks of system administration. To decide in which of those stereotypes a user is classified, `login()` operation is defined as a trigger. Returned values identify a given user into one of the three stereotypes. Let us note that a user previously registered as `CLIENT` or `ADMINISTRATOR` may be considered into the `ANONYMOUS` stereotype if she has not previously executed the corresponding trigger.

The proposed User Stereotypes Diagram allows classifying users from a given stereotype according to different criteria. For instance, from the `CLIENT` stereotype, two criteria of classification are adopted:

- By considering the user's `age` as a trigger, she can be classified into `CHILD`, `YOUNG` or `ADULT` stereotypes.
- By considering the execution of a given payment operation (`payPremium()`) as a trigger, the user can be classified into `NORMAL` stereotype, which are the default client category; or into `PREMIUM` stereotype, which are clients with access to special functionalities of the application.

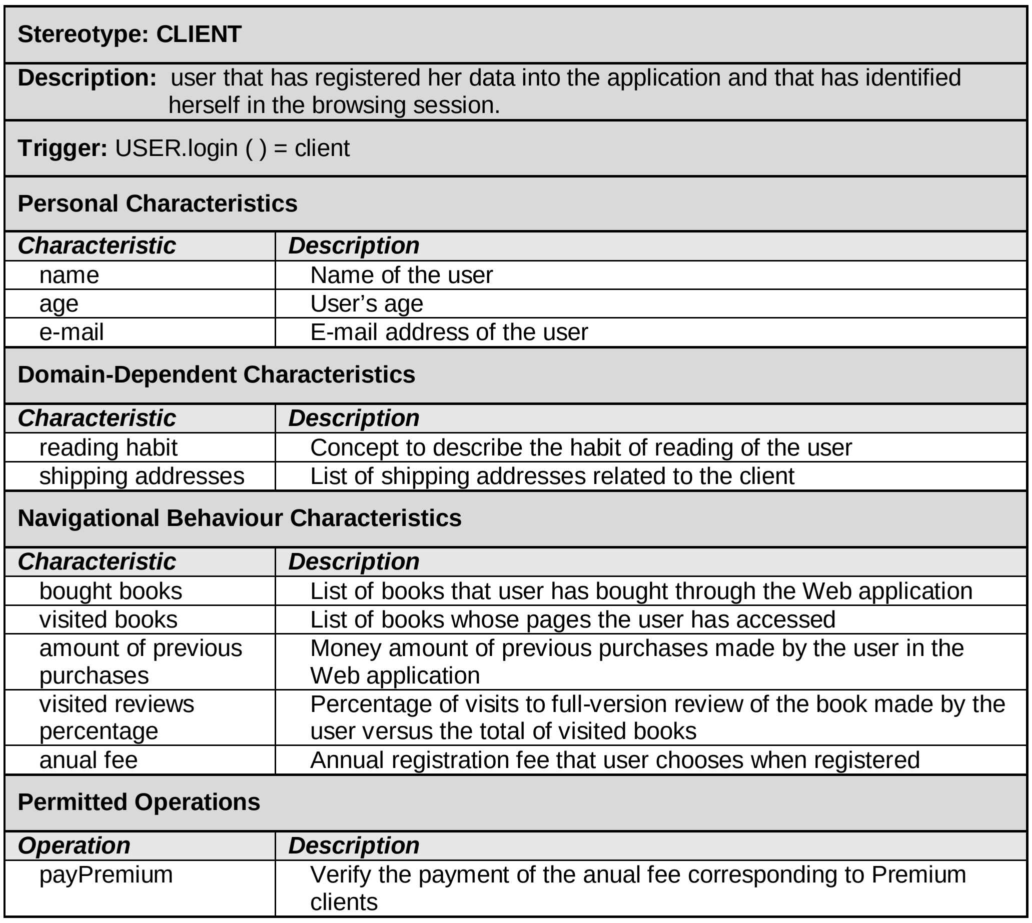
The possibility to define more than one criterion of inheritance from a unique stereotype permits a richer description of the intended users from distinct, complementary perspectives.

6.3.2 Users Specification

Once similar users have been grouped into stereotypes, the required user characteristics that allow distinguishing one user from another must also be specified. In the ***Users Specification*** phase, the developer defines the set of characteristics that are needed to be valuated in order to describe and distinguish individual users from each stereotype. Furthermore, it includes the operations that users from the stereotype are allowed to execute.

Characteristics and operations are specified and ordered in tabular descriptions. Figure 6.9 illustrates the specification of a given stereotype (`CLIENT`, in the example). It includes three fields that identify the described stereotype: *Name* (the name of the stereotype), *Description* (a brief description of the users that are classified into the stereotype) and *Trigger* (the condition that a user must be evaluated to be classified into the stereotype). As the figure shows, the Stereotypes Specification describe the characteristics of the stereotype by considering four categories:

1. **Personal or Domain-Independent User Characteristics**. This group refers to those features that are intrinsic to a given user as an individual. Particular values of these characteristics are independent of specific domains. Typical features from this group are: *name*, *date of birth*, *city of residence*, etc.



Stereotype: CLIENT	
Description: user that has registered her data into the application and that has identified herself in the browsing session.	
Trigger: USER.login ( ) = client	
Personal Characteristics	
Characteristic	***Description***
name	Name of the user
age	User's age
e-mail	E-mail address of the user
Domain-Dependent Characteristics	
Characteristic	***Description***
reading habit	Concept to describe the habit of reading of the user
shipping addresses	List of shipping addresses related to the client
Navigational Behaviour Characteristics	
Characteristic	***Description***
bought books	List of books that user has bought through the Web application
visited books	List of books whose pages the user has accessed
amount of previous purchases	Money amount of previous purchases made by the user in the Web application
visited reviews percentage	Percentage of visits to full-version review of the book made by the user versus the total of visited books
anual fee	Annual registration fee that user chooses when registered
Permitted Operations	
Operation	***Description***
payPremium	Verify the payment of the anual fee corresponding to Premium clients

Figure 6.9: `CLIENT` stereotype specification

These characteristics allow identifying a user and achieve an initial, permanent adaptation, for instance, to obtain recommendations about places near to her *city of residence*.

2. **Domain-Dependent Characteristics**. This group comprises those user features that are related to the knowledge domain of the specific application that is being developed. For instance: *shipping address* (e-commerce domain); *knowledge level* (e-learning).

   These characteristics may be explicitly provided by the user or be inferred from her navigational behaviour. For instance, in an online bookstore, values for a "*Favourite Authors*" characteristic may be submitted by the user or inferred from the books that she has previously bought.

   Characteristics from this group allow achieving a more precise adaptation, focused on the relation (knowledge, interest, preference, etc.) that a user has with the specific application domain.

3. **Navigational Behaviour Characteristics**. This group comprises the actions that user performs when interacting with the application and that have significance to its adaptation purposes. For instance, *reviewed contents* (e-learning), *purchased items* (e-commerce).

   These characteristics are defined as a consequence of a *page access* (e.g., *reviewed contents*) or an *operation execution* (e.g., *purchased items*, performed by a given user. Therefore, their specification give clues about navigational paths and operations that must be considered in the application modelling.

   Characteristics from this group are the basis to achieve run-time adaptation, i.e., adaptation based on the current user's navigational actions. Their values allow keeping the user information constantly updated and, as a consequence, achieving a more effective adaptation.

4. **Permitted Operations**. This group comprises those operations that can be executed by users from the stereotype. For instance, operation to rate a product, submit the answers of an evaluation, etc.

The User specification of a given stereotype must include the characteristics and/or operations on which triggers of its child stereotypes are based.

Once information and classification of the intended users of the application have been specified, it is possible to describe the adaptive functionalities that system needs to provide. In the following phase, the set of adaptive tasks of the application are specified.

6.4 Identification and Description of Adaptive Tasks

This phase extends the current OOWS proposal to describe the tasks the system provides to users, incorporating the specification of ***Adaptive Tasks***. We use this term to describe those tasks that incorporate some adaptive functionality, in which the access to data items and links are different to different users. Some tasks make use of information about users, like traditional *"login"* task, but the provided interaction possibilities are the same for any user. For this reason, that kind of tasks is not considered into this category.

As a first step, the adaptive tasks included in the *OOWS Task Diagram* are identified and documented, indicating the kind of users that are allowed to access each of them. Afterwards, in case of complex adaptive tasks, alternative interactions are described by means of a *Use Case Diagram.*

6.4.1 Identification of Adaptive Tasks

Following the OOWS strategy to specify the set of tasks that a Web application must provide, we adopt the hierarchical description of the *Task Diagram*, which also describes temporal dependencies between pairs of tasks. From the elementary tasks of the diagram (leaves of the tree), adaptive tasks are identified and graphically depicted with the stick-man symbol at the left of the primitive.

Figure 6.10 shows the Tasks Diagram of an adaptive online bookstore. It is equivalent to the diagram corresponding to the non-adaptive version of the application, previously shown in Fig. 6.3. However, the adaptive version includes the graphical clues that indicate whether an elementary task

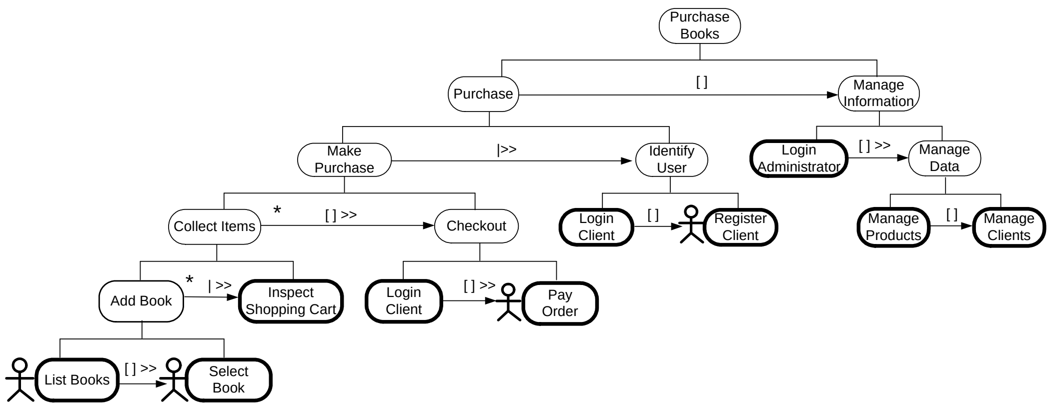


Figure 6.10: Tasks Diagram with Adaptive Elementary Tasks

includes some adaptive features. Due to its high complexity, the `Add Book` task, which is elementary in the first diagram, has been refined into two adaptive elementary tasks. Some of the elementary tasks that make use of user information, as `‘‘Login Client’’` task, have not been considered as adaptive tasks, assuming that their fulfilment is identical to any user, i.e., they do not provide adaptive interactions. In any case, the adaptive condition of elementary task may be changed if adaptive features arise when more refined requirements of elementary tasks are specified.

6.4.2 Description of Adaptive Tasks

Complexity of some adaptive tasks may not be properly described through the Task Diagram. For instance, `List Books` task shows a list of books from which users may select one book to visualize deeper information about it and eventually add it to the shopping cart. However, books may be listed according to multiple, non-adaptive or adaptive criteria: new released books (non-adaptive), recently viewed or recommended books (adaptive). If necessary, the specification of adaptive tasks may be refined through a Use Case Diagram.

Figure 6.11 shows the detailed specification of `List Books` adaptive task through a Use Case Diagram. By means of `extends` relationships, different criteria to list books are represented by means of respective use cases. One

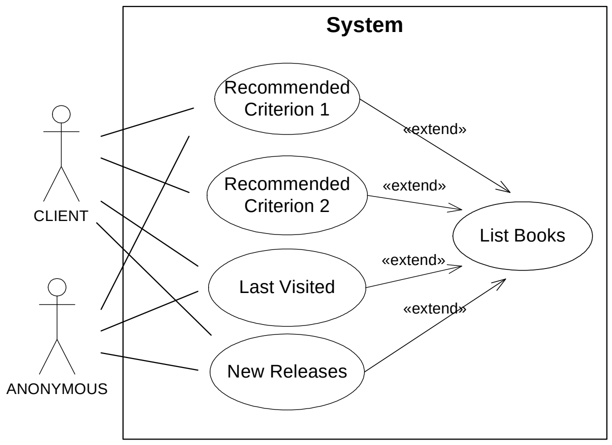


Figure 6.11: Use Cases Diagram of the `List Books` Adaptive Task

of them is available to users from `CLIENT` stereotype only, while the others are available to `CLIENT` and `ANONYMOUS` users.

The tabular documentation of elementary tasks used in OOWS proposal is maintained in this proposal, but with some modifications: a new *Adaptivity* field is added, which indicates the adaptivity quality of each elementary task; in *Users* field (see Subsection 6.2.2), the specification of the users allowed to access the task is made in terms of the defined *User Stereotypes.* Optionally, a constraint defined in terms of user characteristics may be added, to provide a more precise specification; finally, a Use Cases field is incorporated, which corresponds to the list of use cases that composes the adaptive task. This field may be left empty if the complexity of the adaptive task does not require a detailed description through a Use Case Diagram.

Figure 6.12 presents the description of the `List Books` adaptive task. *Adaptivity* field indicates that this task has some adaptive features; *Users* field indicates the stereotype to which users must belong to access the task, along with fulfilling the included constraint; finally, an *Additional Constraint* is included, which specifies a business restriction to the task.

In the case of complex adaptive tasks specified through a Use Case Diagram, each of the Use Cases that extends the definition of the task is documented through a similar table, as shown in the example of Fig. 6.13. In this case, one of the criterion adopted to list books is detailed: listed books

Task	List Books
Adaptivity	Adaptive
Users	ANONYMOUS, CLIENT
Description	The system provides users with a indexed list of books to select, according to different criteria of selection and order
Frequency	300 times/hour
Additional Constraints	--
Use Cases	– Recommended Criterion 1 – Recommended Criterion 2 – New Releases – Last Visited

Figure 6.12: Description of an Adaptive Elementary Task

are adaptively recommended to the user, according to their similarity with books she has previously visited and excluding those she has already bought. This criterion is only valid to users from the `CLIENT` stereotype that have visited the information of at least one book. As an additional constraint, recommended books must be in stock.

Use Case	Recommended Criterion 1
Task	List Books
Adaptivity	Adaptive
Users	– CLIENT users, with at least one visited book; – ANONYMOUS users
Description	– For CLIENT users, the system provides the user with recommendations of books, ordered according the number of keywords in common with books previously visited by user, and excluding those books that the user has already bought. – For ANONYMOUS users, the system provides the user with a list of the books in stock, alphabetically ordered by title.
Additional Constraints	Recommended books must be in stock

Figure 6.13: Description of Use Cases from Adaptive Elementary Task

6.5 Specification of Adaptive Task Achievement

The goals of the two previous phases of this requirements specification proposal are to describe the required user information to specify adaptive functionality and to determine which are the tasks through which that functionality is provided, respectively. The goal of this third phase is to specify the required adaptive functionality itself.

The enhancements introduced to the OOWS proposal in the previous chapter are focused on extending the conceptual descriptions of the navigation, in order to describe an adaptive access to contents and links. For this reason, the proposal for specifying adaptivity requirements mainly affects the OOWS proposal for the specification of navigational requirements, through the use of extended ***UML Activity Diagrams***.

To support the modelling decisions that can be made from the use of the introduced navigational structures, the OOWS proposal must be extended in order to fulfil the following conditions:

1. *Requirements specification based on user characteristics.* In OOWS Activity Diagrams, access to information is expressed through Interaction Points (IP), while access to operations is described through System Actions. The required adaptive access to both aspects must be expressed through constraints defined in terms of user characteristics. The definition of User Stereotypes and User Specifications in the phase of User-related Requirements is the basis of that specification. In particular, the consideration of characteristics related to the navigational behaviour of the user (visited pages, executed operations) is specially important, because these features contain constantly updated information about the current user.

2. *Support to the adaptive access to entity instances.* The specification of requirements that involve the adaptive access to one or several instances from an entity must be also provided. With the aim of privileging the access to the most relevant contents, some methods may require to postpone the access to certain entity instances until some conditions

are fulfilled by the user. These requirements support the modelling of methods based on the *visibility of navigational relationship* property (see Section 4.5.5) and the *mixed relationship* primitive (see Section 4.5.4): *Additional Explanations*, *Prerequisite Explanations* and *Local Orientation.*

3. *Support to the adaptive access to attributes and operations.* Some adaptive methods involves the adaptive access to specific data and operations related to an entity instance, specially when complementary explanations about a concept must be provided according to the current user's characteristics. The specification of this kind of requirements supports the modelling of adaptive methods based on the *adaptive visibility of navigational attributes and operations* primitive (see Section 4.5.7): *Additional Explanations*, *Prerequisite Explanations.*

4. *Support to the adaptive filtering of multiple instances.* Resulting specifications must be able to express requirements of adaptive filtering of multiple information instances of an entity, allowing the access only to the most relevant or pertinent to a particular user. This expressiveness is needed to support the modelling of adaptive methods based on the *adaptive population filter* primitive (see Section 4.5.2): *Global Guidance*, *Local Guidance*, *Global Orientation*, *Local Orientation* and *Comparative Explanations.*

5. *Support to the adaptive ordering of multiple instances.* Analogously, it must be possible to express requirements that involve the adaptive ordering of multiple information instances of an entity, in such a way to facilitate the access to the most relevant or pertinent information to a particular user. This is needed to support the modelling of adaptive methods based on the *adaptive ordering pattern* primitive (see Section 4.5.3): *Global Guidance*, *Local Guidance*, *Global Orientation* and *Local Orientation.*

To satisfy these conditions, the structures considered by the OOWS requirements model [24] in its Activity Diagram (and presented in Section 6.2.2) have been enhanced, through the definition of access constraints in terms of user characteristics. For each of the described conditions, we propose the following solutions, respectively:

1. Constraints that define the adaptive access to data and functionality are expressed in terms of the user stereotypes and characteristics specified in the User-related Requirements phase. To refer a particular user, we adopt the *#user#* keyword; a particular user's characteristic is represented through the expression *#user#.<characteristic name>*; and the conditional expression to verify whether a user belongs to a given stereotype is written as #user# ∈ *<STEREOTYPE NAME>*. These kinds of expressions are included in boolean conditions that users must fulfil to gain access to a certain adaptive feature. For example, the following expression describes a certain subset of users, according to the characteristics of the *CLIENT* stereotype specification of Fig. 6.9.

   #user# ∈ CLIENT and #user#.prevPurchaseAmount >= 100

   This expression describes those users that belong to the `CLIENT` stereotype and whose amount of previous purchases (`prevPurchaseAmount` characteristic) is greater than or equal to `100` (euros, dollars, etc.).

2. To support the adaptive access to entity instances, the ***Access Condition*** constraint is introduced. This constraint represents the condition that users must fulfil in order to access a given Interaction Point.

   This primitive supports the specification of requirements whose fulfilment needs the adaptation of the access to one or more instances from a given entity.

   In Activity Diagrams, *Access Conditions* are expressed as conditions on the *flows* that directly lead to the Interaction Point whose access is adaptively restricted.

   To illustrate this kind of constraints, Fig. 6.14 incorporates the expression of the previous item as an Access Condition to the `Special Offer` Interaction Point. This condition specifies the adaptivity requirement of giving access to the information of special offers only to users that fulfills the mentioned condition. Graphically, this Access Condition is located on the flow that leads to the constrained IP.

3. The access to the operations required to fulfil a given task is expressed through a *System Action* structure. To specify the adaptive access to a given operation, an ***Access Condition*** is defined on the flow that

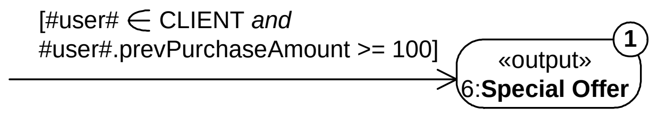


Figure 6.14: Access Condition for Interaction Points

directly leads to the corresponding System Action, in the same way that it is defined to represent the adaptive access to Interaction Points. Access Condition expresses the constraint that users must fulfil in order to access a given operation.

Figure 6.15 shows an Access Condition that only permits to access the `Get_Discount` System Action to those users that belong to the `PREMIUM` stereotype.

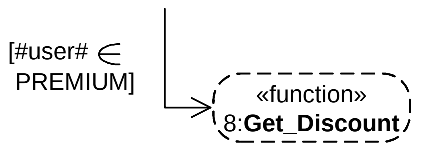


Figure 6.15: Access Condition for System Actions

4. To support the adaptive filtering of multiple instances, a ***Selection Precondition*** constraint is introduced. It allows representing different criteria adopted to select multiple instances from an entity. According to these criteria, different set of instances can be accessed by different groups of users through the same Interaction Point.

   This primitive is used to specify requirements whose fulfilment involves the presentation of a list of multiple instances of an entity in a single page, in which the shown instances vary according to the characteristics of the user that is accessing the page.

In Activity Diagrams, *Selection Preconditions* are defined on flows that directly connect to the *Output IP* from which the filtered list of instances is accessed. The precondition is formed by the set of adopted selection criteria, each of them associated to different set of users. For the graphical notation of *Selection Preconditions*, we use the *Selection* operation from *UML Activity Diagrams* [90].

Figure 6.16 shows a *Selection Precondition*, which filters the set of instances of *Book* IP that are presented to different users. The described condition indicates that a given book (identified as *self*) is only shown if does not belong to the set of books that the current user already owns. Graphically, the precondition is located on the flow that directly leads to the constrained IP.

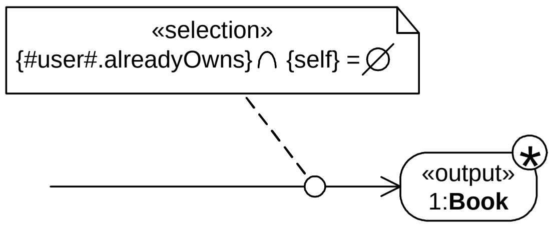


Figure 6.16: Selection Precondition for Interaction Points

5. To support the adaptive ordering of multiple instances, the ***Sorting Precondition*** constraint is introduced. It allows representing the different criteria that are adopted to order multiple instances from an entity. According to these criteria, the same set of instances can be accessed by different groups of users in the same Interaction Unit, but ordered according to criteria that are suited to each user's needs.

   This primitive is used to specify requirements that involves the presentation of a list of multiple instances of an entity in a single page, in which the presentation order of the selected instances vary from one user to another.

   In Activity Diagrams, *Sorting Preconditions* are defined on flows that directly connect to the *Output IP* from which the list of instances is accessed. The precondition is formed by the set of adopted ordering criteria, which are attributed to different set of users through *if-then* clauses. For the graphical notation of *Sorting Preconditions*, we use the *Transformation* operation from *UML Activity Diagrams* [90].

Figure 6.17 shows a *Sorting Precondition*, which defines an ordering criterion for the presentation of the instances of the `Book` IP. The described condition counts the keywords that a given book (identified as `self`) has in common with the set of books that a given user has already visited. Graphically, the precondition is located on the flow that directly leads to the constrained IP.

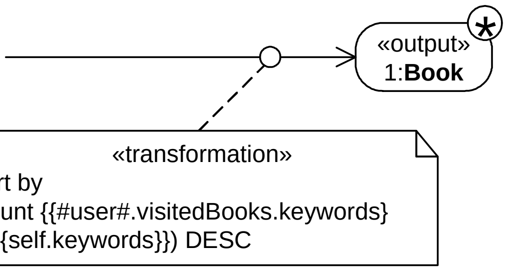


Figure 6.17: Sorting Precondition for Interaction Points

To exemplify the specification of the achievement of adaptive tasks, let us consider the requirements specifications for two adaptive tasks from the Task Diagram shown in Fig. 6.10: `List Books` task, considering the *Recommended Criterion 1* use case documented in Fig. 6.13, and the `Select Book` task.

The Activity Diagram for *Recommended Criterion 1* use case is shown in Fig. 6.18 and it can be read as follows[1]:

1. If the user belongs to `CLIENT` stereotype, a list of `Book` instances is provided in the (1:Book,*) IP. An `Access Condition` (#user# ∈ `CLIENT`) describes this constraint. The list must not include those books that the user has already bought. This is expressed through a *Selection Precondition* (≪selection≫ expression). Finally, the list must be ordered according to the number of keywords that displayed books have

[1]We propose the following textual notation for IPs: *(number:Entity,Cardinality)*

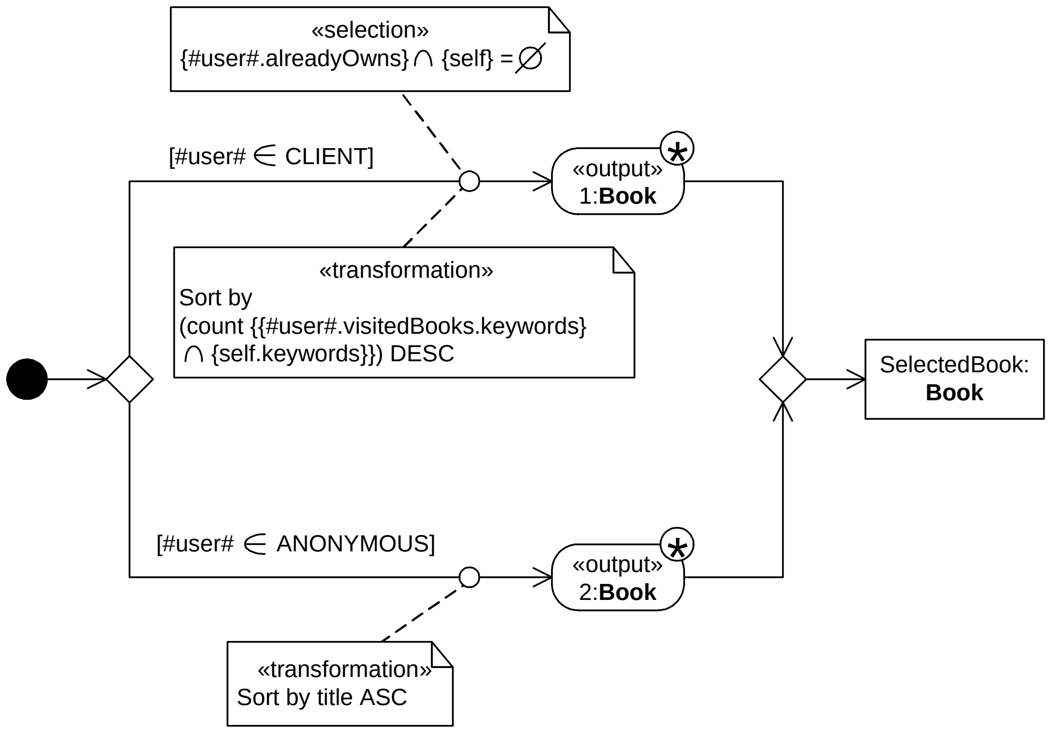


Figure 6.18: Activity Diagram of a Use Case of the `List Books` task

in common with those books whose Web pages the client has visited. A *Sorting Precondition* expresses this requirement (≪transformation≫ expression).

2. If the user belongs to `ANONYMOUS` stereotype, the list of books is provided in the (2:Book,*) IP. This is also described through an *Access Condition* (#user# ∈ `ANONYMOUS`). Due to that IP has no *Selection Precondition*, it thus includes the full list of books in stock, ordered alphabetically by their titles (according to a *Sorting Precondition*).

The task concludes when the user select a `Book` instance from the corresponding list. As shown in the Task Diagram of Fig. 6.10, there is a *"enabling with information passing"* temporal dependency between the `List Books` and `Add Book` tasks. The passed information is precisely the selected book, as shown in the framed expression at the right of the diagram.

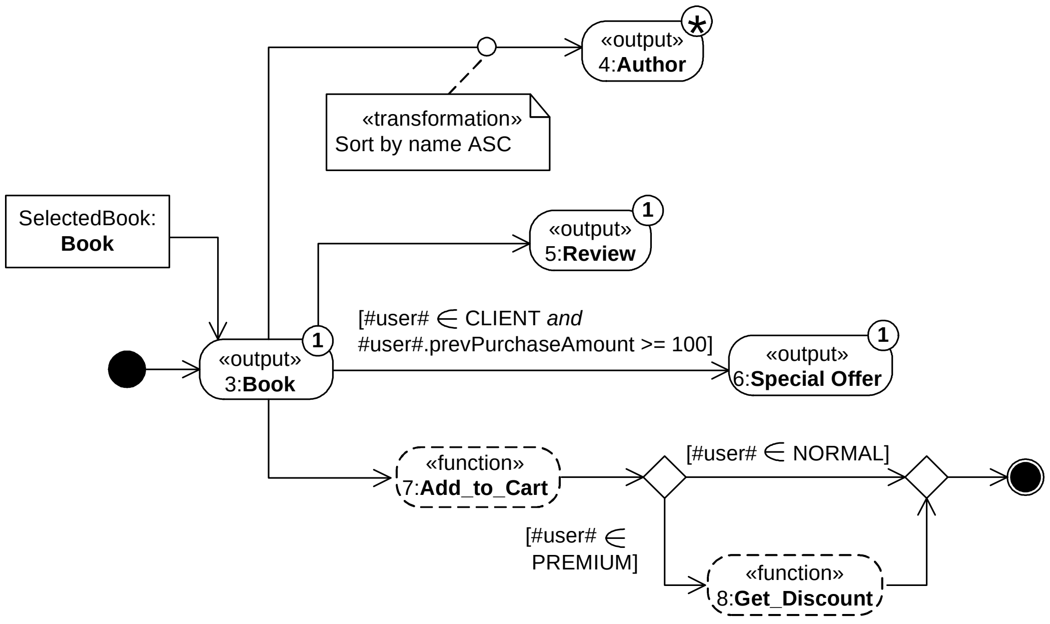


Figure 6.19: Activity Diagram of *Add Book* task

Figure 6.19 shows the Activity Diagram of the `Add Book` task. It starts receiving the selected book from the `List Books` task. Detailed information of the selected instance may be accessed through (3:Book,1) IP. From this IP, the user has the following choices:

- She may access a set of instances of the `Author` entity, corresponding to the list of author of the selected book, through the (4:Author,*) IP. This set of instances is alphabetically ordered, through a *Sorting Precondition*.
- She may access an instance of `Review` entity, corresponding to a specialized review of the selected book, through the (5:Review,1) IP.
- She may access an instance of `Special Offer` entity, corresponding to a special offer related to the selected book, through the (6:Special Offer,1) IP. The access to this IP is constrained by means of an *Access Condition*, based on the amount of previous purchases made by the user (#user#.prevPurchaseAmount $\geq$ 100).
- She may activate the (7:Add_to_Cart) function, and the system will add the selected book to the shopping cart. If the amount of previous purchases made by the user is at least $`100`, a special discount is applied. This adaptive action is expressed through the (8:Get_Discount) function, which is only available to users from the `PREMIUM` stereotype. This is expressed through an *Access Condition* (#user# $\in$ `PREMIUM`).

6.5.1 Extended Data Description

In the OOWS requirements proposal, the information that must be stored by the system is specified through the use of a template technique, which is based on techniques such as *NDT* [93] and *CRC Cards* [92], and on the template-based approach presented in [94].

A ***Data Template*** is defined for each entity included in the activity diagrams. It comprises the following parts: an identifier, the entity name, a description of the entity, and a specific data section. This section contains a list of features associated to the entity. Each feature is specified through its name, its description and the list of IPs in which it is included.

Identifier:	O1	
Entity:	Book	
Description:	Book on sale in the bookstore	
Name	*Description*	*IPs*
ISBN	ISBN code	(3:Book,1)
title	Title of the book	(3:Book,1) (1:Book,*) (2:Book,*)
date	Publication date	(3:Book,1) (1:Book,*) (2:Book,*)
summary	Brief summary of book	(3:Book,1)
cover image	Image of book's cover	(3:Book,1) (1:Book,*) (2:Book,*)
pages	Number of pages	(3:Book,1)
price	Purchase price	(3:Book,1) (1:Book,*) (2:Book,*)
discount	Discount on book to PREMIUM users	IF #user# ∈ PREMIUM THEN (3:Book,1) ENDIF
keyword	List of keywords associated to book	

Figure 6.20: Data description of *Book* entity

The specification of the adaptive access to an individual entity feature is made by incorporating an ***Access Condition*** in the list of IPs through which the feature may be accessed. This constraint is based on user characteristics previously specified. From the list of possible IPs, the particular feature may be accessed through one IP or another depending on the fulfilment of this condition by the current user. In some cases, the access to the feature may be forbidden. Figure 6.20 shows the Data Description corresponding to the `Book` entity, from which it is possible to distinguish some features that are included in all of the IPs corresponding to `Book`; others, like `ISBN` and `summary`, can only be accessed in some of them; and also one feature (`keywords`) is not included in any IP (it may be used by internal system actions only). In the case of the `discount` feature, it can be reachable only if the user fulfills a given *Access Condition*, which ask users for a certain amount of previous purchases to access the represented `discount` associated to a `Book`.

Figure 6.21 shows the data description of the `Review` entity. In this case, the `title` and `author` features are included in two IPs, corresponding to the description of a particular book and its specialized review, respectively. However, the `text` feature (the full version of the review) may be accessed in

Identifier:	O2	
Entity:	Review	
Description:	Review of a book by a specialist	
Name	*Description*	*IPs*
title	Title of the review	(5:Review,1) (3:Book,1)
author	Name of the review's author	(5:Review,1) (3:Book,1)
text	Text of the review	IF userConstraint_1 THEN (5:Review,1) ELSE (3:Book,1) ENDIF

* userConstraint_1 = ((#user#.visitedReviewsPercentage < 80)
OR (#user#.visitedBooks->count() < 20))

Figure 6.21: Data description of the `Review` entity

different IPs, depending on the fulfilment of another *Access Condition*. For a first-time user, the full review is shown apart from the information of the corresponding book (i.e., the`text` feature is shown in (5:Review,1)); however, if the user has visited information of at least 20 books and accessed the full review in at least a 80% of them, the full review is shown together with the rest of the book's features (i.e., `text` is shown in (3:Book,1)).

6.6 Conclusions

Modelling decisions about adaptivity respond to very concrete interaction requirements. The proposal presented in this chapter supports the modelling decisions that can be made through the use of the introduced Adaptive Primitives, by means of a Requirements Model for Adaptive Web Applications.

The introduction of a proposal to specify the requirements of Adaptive Web Applications is a distinctive characteristic of the present work in comparison with most of the Model-Driven approaches to Web Adaptivity. Most of the existing requirements specification proposals do not consider adaptivity requirements or, at least, not with the proper level of detail. A methodological strategy to specify these requirements permits to support the complex modelling decisions that can be made through the use of the introduced Adaptive Primitives. The distinction between User-related and Adaptivity Requirements permits to separate the concerns related to the adaptive functionality that must be provided from those related to the user information that needs to be obtained and updated, in order to implement that functionality. This distinction is often blurred in later stages of the development cycle, hindering the traceability of the system requirements. In summary, the complexity of the conceptual modelling of Adaptive Web Applications needs for higher level mechanisms, which permit to specify the requirements to which it is subordinated.

This adaptivity requirements proposal extends the OOWS approach to specify functional and navigational requirements of Web applications. Although the proposed extensions do not introduce drastic modifications to the already existing models, the spectrum of requirements whose specification is supported increases remarkably. This is a proof that the complexity of adaptivity issues decreases as the abstraction level from which they are considered raises.

Only one Adaptive Primitive is not supported by this proposal: the *Ranking of Navigational Relationships*. This structure assigns a relevance ranking to the set of navigational relationships of an AIU. The specification of this characteristic is subordinated to the complete specification of the inner structure of the corresponding AIU. For this reason, it must be integrated with the specification of the structural requirements, which is out of the immediate scope of this thesis. A possible alternative is the adoption of a textual specification of this kind of requirements, complementing the activity diagram corresponding to the task whose fulfilment this information supports.

As stated in [71], the purpose of a requirements model for Information Systems is to favor the understanding of the system to be built and to provide techniques to capture its desired properties. The notation that is used to express the requirements specification must be: (a) simple enough to permit people without formal training to understand and review them; and (b) precise enough so that the resulting specifications can be the source to derive the building blocks of a conceptual schema.

In this chapter, we have considered these two characteristics: the User Stereotype Diagrams, the enhanced Activity Diagrams, and their corresponding User and Data Specifications descriptions, are intended to capture complex requirements of Adaptive Web Applications, by means of intuitive and easy-to-understand schemas. Furthermore, they must serve as a basis to derive the conceptual specifications of users and adaptive features in the corresponding schemas of the OOWS Conceptual Modelling phase. In the next chapter, we introduce a set of traceability rules that supports this modelling process, by means of correspondences between adaptivity requirements specifications and conceptual schemas.

Chapter 7

From Adaptivity Requirements to Navigational Schemas

7.1 Introduction

This chapter presents a set of modelling rules that permit to obtain conceptual specifications of an Adaptive Web Application from its corresponding requirements specifications.

In Chapter 4 of this dissertation, a User Modelling proposal and a set of Adaptive Primitives were introduced, in order to support the high-level description of the adaptive features of a Web application. Chapter 6 presented a set of conceptual mechanisms to specify the User-related and Adaptivity Requirements. The present chapter analyzes the correspondences between different occurrences of requirements specifications and the conceptual modelling of adaptive features.

Firstly, we have analyzed the correspondences between the User-based requirements specifications, which are made through the User Stereotypes Diagram and the User Specifications, and the User Schema of the application. As a result, a set of guidelines to model the specified requirements has been proposed.

Afterwards, we have studied the correspondences between Adaptivity Requirements Diagrams and the OOWS Navigational Model, enhanced with the introduced Adaptive Primitives. In this case, a set of modelling rules has been defined[1].These rules associate different occurrences of requirements diagrams to a corresponding navigational description in OOWS.

In the following sections, we describe how the requirements of Adaptive Web Applications specified through each of the introduced and extended diagrams may be described in an OOWS Navigational Schema. In the case of Activity Diagrams and Task Descriptions, the associated set of modelling rules that support this process is described in detail.

7.2 Fixing a Model Transformation strategy

In Chapter 6, we have introduced a set of diagrams to specify requirements of Adaptive Web applications based on the OOWS approach. Resulting specifications provide developers with knowledge that gives support to the navigational modelling of the described requirements. With this purpose, some of the introduced diagrams are determinant in the definition of the navigational structure, while other diagrams only permit to propose some guidelines or suggestions to a proper navigational modelling of the expressed requirements. The following subsections describe how each primitive contained in the OOWS requirements diagrams is transformed in its corresponding navigational modelling counterpart.

7.2.1 Transformations associated to the User Stereotypes Diagram

Relevant groups of users with similar characteristics are specified in the *User Stereotype Diagram* (see Section 6.3.1). By means of this hierarchical struc-

[1]We mean by "modeling rules" those rules that specify the mappings between conceptual primitives at the requirements level (source) and their corresponding representation at the conceptual model level (destination). In terms of Model Driven Architecture concepts, there are Model to Model rules, that we refer to as "modelling rules"

ture, any eventual user of the application is classified into one stereotype, permitting to obtain a general idea of her characteristics and to provide a coarse-grained adaptation.

The definition of this diagram directly influences the modelling of the application domain. According to their relevance and particular criteria of modelling, a stereotype may be modelled as a *class* of the Class Diagram, or only be considered to define adaptive constraints in the Navigational Schema. In the first case, an inheritance relationship between two stereotypes may be also modelled as a *generalization dependency* between the corresponding classes.

7.2.2 Transformations associated to the User Specifications

User Specifications (see Section 6.3.2) complement the taxonomy of relevant user groups defined in the User Stereotype Diagram. A *User Specifications* table is defined for each stereotype, describing the specific characteristics that allow distinguishing two users from the same stereotype, along with the operations that they are allowed to access.

If a given stereotype has been modelled as a *class* in the Class Diagram of the application, the attributes and operations of this class are defined from the characteristics and operations included in the corresponding tabular specification. Eventually, such class may also include attributes and operations from ancestor stereotypes.

A User Specification table comprises different categories of descriptors. Each of these categories provides the following associated components (conceptual primitives) to the Class Diagram of the application:

- Characteristics from *"Personal or Domain-Independent"* category are likely to be modelled as attributes of its stereotype's class, in the *User Description View* of the Class Diagram. This is because these user characteristics are intrinsically associated to the stereotype they belong to, and have no relationship with specific domains.

- *"Domain-Dependent"* characteristics may be modelled through classes and attributes, and included into *User Description View* of the Class Diagram. However, if some of these characteristics may be described in terms of user actions, corresponding primitives may be included in *User Behaviour View.*

- Characteristics from *"Navigational Behaviour"* category are mainly modelled through *classes* and *attributes* that represent the information about user actions that needs to be captured and/or stored, in order to fulfill adaptivity goals. Furthermore, some of them may be modelled through associations between domain classes and stereotype-based classes. In this way, some navigational actions may be modelled in terms of the domain concept affected by the action and the person that executes it. This kind of associations may also connect a domain concept with an instance of the *Browsing Session Class* (see Section 4.3.2), to describe more volatile actions or the interactions of an anonymous user. All the obtained primitives from this category of user characteristics are included in the *User Behaviour View* of the Class Diagram.

- The set of *"Permitted Operations"* may be modelled directly as *operations* and included in the class that describes the corresponding stereotype, or rather being decomposed into several operations distributed among different classes, according to responsibility assignment decisions. Along with the signature of the modelled operation, another structures may complete the description of these operations. For instance, an association may be created or an attribute value may be updated as a consequence of the execution of such operations. These aspects must be also considered in the Class Diagram.

- Finally, the characteristics and operations that compose the expression of each stereotype's *trigger* must be modelled in the Class Diagram, as attributes and operations of domain or stereotype-based classes. In navigational schemas, conditional expressions to verify whether a user belongs to a given stereotype are defined in terms of these primitives.

7.2.3 Transformations associated to the Task Diagram

As an intermediate step of the Requirements Specification Phase, the definition of the hierarchical *Task Diagram* (Section 6.2.2) mainly supports a refined specification of requirements. By organizing and classifying the set of possible tasks that the system must provide, this diagram allows determining which are the elementary tasks that must be detailed in the respective Activity Diagrams. Furthermore, *temporal constraints* between two tasks lead to define synchronization mechanisms in their diagrams.

For this reason, navigational modelling phase is only indirectly affected by the specifications made through the Task Diagram. Only temporal constraints between pairs of tasks may be considered by developers in the distribution of AIU's among different navigational contexts. These constraints may influence the coexistence of two AIU's into the same context. For instance, if two tasks (or their corresponding ancestor tasks) are related by a *Choice* temporal constraint, i.e., only one of them can be performed, then the inclusion of AIU's that support their respective fulfilment into the same context may be avoided. Analogously, two AIU's that support tasks with *Enabling with Information Passing* or *Suspend-Resume* constraints between them should be preferably placed in different contexts.

However, the final composition of navigational contexts must also consider other requirements, like usability aspects. The influence of temporal constraints in the inner structure of navigational nodes is an interest topic that deserves further attention.

7.2.4 Transformations associated to the Use Case Diagrams

Use Case Diagrams allows specifying the interaction requirements of complex elementary tasks. A given task may be decomposed into different use cases, which may specify different alternatives of execution (through ≪*extends*≫ relationships) or describe subtasks that concur to its fulfilment (through ≪*includes*≫ relationships) and that have not been included in the Task Diagram. This is specially useful in the specification of Adaptive Web Applications, where a given task may be fulfilled in different ways according to the characteristics of a given user.

Each use case that composes a given task is detailed through an Activity Diagram, just like simpler elementary tasks. The navigational modelling of these diagrams generates a set of AIU's that may be included in contexts that support the fulfilment of the task, representing navigational steps of the task or alternative navigational solutions.

7.2.5 Transformations associated to the Activity Diagrams

The requirements diagram that mostly influences the navigational modelling of Adaptive Web Applications is the *Activity Diagram* (introduced in Section 6.5). From the specifications of the user-system interactions that are needed to perform a given task, the inner structure of navigational nodes may be almost completely obtained. In the case of Adaptive Tasks, these diagrams include user-dependent constraints, which are modelled through the adaptive primitives introduced in this work. The influence of these diagrams in the confection of OOWS Navigational Schemas is described through modelling rules introduced in the following section.

7.2.6 Transformations associated to the Data Descriptions

Complementing the descriptions made through Activity Diagrams, *Data Descriptions* (Section 6.5.1) specify the features or characteristics of the information entities that are involved in the specified user-system interactions. From the inclusion of user-centered constraints, the modelling of several adaptive features may be derived. The corresponding modelling rules that this work proposes are described in the next section.

7.3 Modelling Rules: From Adaptivity Requirements to Adaptive Navigational Schemas

As stated above, the specifications of requirements that are made through Activity Diagrams and Data Descriptions are the main source of information to define the inner navigational structure of an Adaptive Web Application. These requirements rule the obtention of the navigational schema of the application. Due to the modelling of adaptive features is made through the OOWS Navigational Schema, which has been extended with the introduced Adaptive Primitives, the adaptivity requirements of the application are mainly described through these mechanisms.

The mapping from the specification of these requirements to their corresponding OOWS navigational descriptions is described by means of a set of *modelling rules*. These rules associate modelling strategies to different occurrences of these diagrams. Each rule is specified by means of a textual description, which comprises the following two fields: (a) *Requirement Specification* (*RS*), which describes an occurrence of the extended OOWS Activity Diagram and/or Data Descriptions; and *OOWS Modelling Solution* (*O-MS*), which describes the proposed way to model the specified requirement in terms of the current and recently introduced structures of the OOWS Navigational Model.

The textual definition of each rule is supported by a graphical definition: RS field is described by means of a segment of an Activity Diagram or Data Description, while its corresponding *O-MS* field is described through the OOWS navigational primitives.

We illustrate the application of the introduced modelling rules by means of a set of requirements from the previously introduced case study. As an input, we consider the adaptivity requirements that have been illustrated in the *Activity Diagrams* of Figs. 6.18 and 6.19, and through *Data Descriptions* for each of the included entities. These specifications correspond to the *"List Books"* (one of its use cases) and *"Add Book"* tasks from the *Task Description* of Fig. 6.3. Descriptions of the intended users of the application and their characteristics, which are included in modelled adaptivity constraints, are obtained from the *User Stereotypes Diagram* of Fig. 6.8 and *User Specifications* like the one shown in Fig. 6.9.

Goals of proposed modelling rules

The definition of these rules has two main goals: (a) to guide and facilitate the complex task of modelling the navigational structure of Adaptive Web Applications, providing developers with modelling patterns that may be easily detected and modelled; and (b) to support the traceability of adaptivity requirements, describing the effects of the specification of this kind of requirements in the analysis and design of the navigational aspects of the adaptive application. The introduced modelling rules are also intended to serve as a basis for the definition of an automatic model-to-model transformation of adaptivity requirements in conceptual schemas, which would support even more the mentioned goals.

Previous work

These rules complement the set of rules defined by Valderas et al. in [24] and [95], which allows deriving the navigational structure of non-adaptive Web Applications from their navigational requirements specifications. However, these works are based on a notion of navigational context composed of only one navigational view of the Class Diagram. In this way, specified requirements permit to specify not only the inner navigational structure of each context, but also an important part of the corresponding navigational map.

In our proposal, we have adopted the extended definition of Navigational Contexts (introduced by Valderas et al. in [96]), as containers of several views of the structural schema. Each of these views (Abstract Interaction Units, AIU's, in OOWS) supports the fulfilment of a given task. From the requirements specification of an only task, we are not able to model the whole navigational node that supports its fulfilment. For this reason, the modelling rules we are proposing are focused on defining the inner structure of these navigational views. When modelling rules affect the modelling of the global navigational structure of the application, it will be explicitly stated.

Scope of the modelling rules

Modelling decisions proposed in this chapter are focused on those aspects of the application that are directly related to the provision of adaptive functionality. The modelling of the whole navigational structure of the final Adaptive Web Application needs to consider a broader set of requirements. Primitives that may be obtained from the application of the proposed rules are: *navigational classes* and *navigational relationships*, along with Adaptive Primitives: *adaptive population filters*, *adaptive ordering patterns*, *visibility of attributes and operations*, *adaptive behaviour of relationships* and *visibility of navigational relationships.*

In the following sections, the introduced modelling rules are described. Rules 1 to 6 support the modelling of the inner navigational structure of the Web application. They do not specifically tackle the modelling of adaptive features, but help to define the conceptual structures in which these characteristics are defined.

Modelling rules 7 to 12 directly support the conceptual descriptions of the adaptive features of a Web application. Most of these adaptive features are based on the definition of expressions that constrain the access to *Output IP*'s, *System Actions* and *Entity Features.* Although these constraints may be defined in terms of any domain concept, the description of adaptive features demands that this definition is made in terms of user characteristics, specified in the previous phase of User-related Requirements Specification.

As a general precondition, the application of these rules assumes that the structural diagram has been already defined. Each entity considered in the Activity Diagram has a correspondent Class in the Class Diagram, from which navigational views may be defined.

7.3.1 Modelling Rule 1: Single-Object AIU's

RS:	An *Output IP* that informs about a single instance of an *ent1* entity (cardinality 1)
O-MS:	A *Single-Object AIU*, whose manager class is a navigational view of the class that represents *ent1* entity.

Description:

This requirement describes the need to provide users with access to detailed information about a single entity instance. In OOWS Navigational Schemas, this is expressed by means of a *Single-Object AIU*, which allows accessing the information of one instance from its manager class. As the most important concept in this view of the application, this manager class must be a navigational view of the structural class that corresponds to the represented entity. Figure 7.1 shows the graphical definition of this rule, in terms of the respective diagrams.

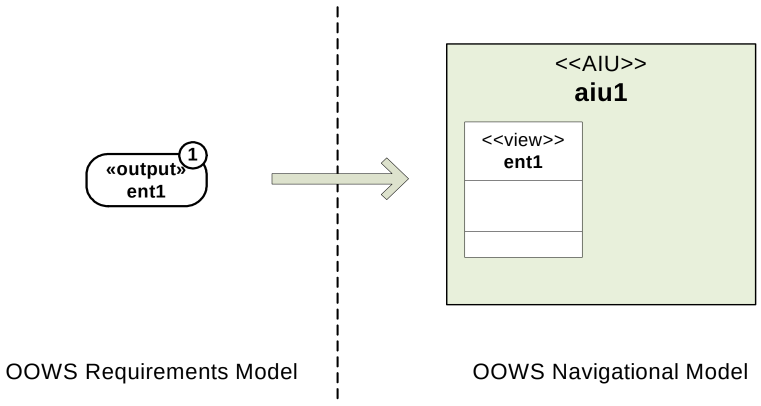


Figure 7.1: Graphical definition of Modelling Rule 1

Example:

Figure 7.2 shows the application of this rule to the two considered activity diagrams of the studied case. Each included *Output IP* with cardinality 1 is modelled through respective Single-Object AIU's, whose manager classes correspond to the respective entities. Navigational contexts that contain these AIU's are not defined from these requirements, and they are included for illustration and future reference purposes only.

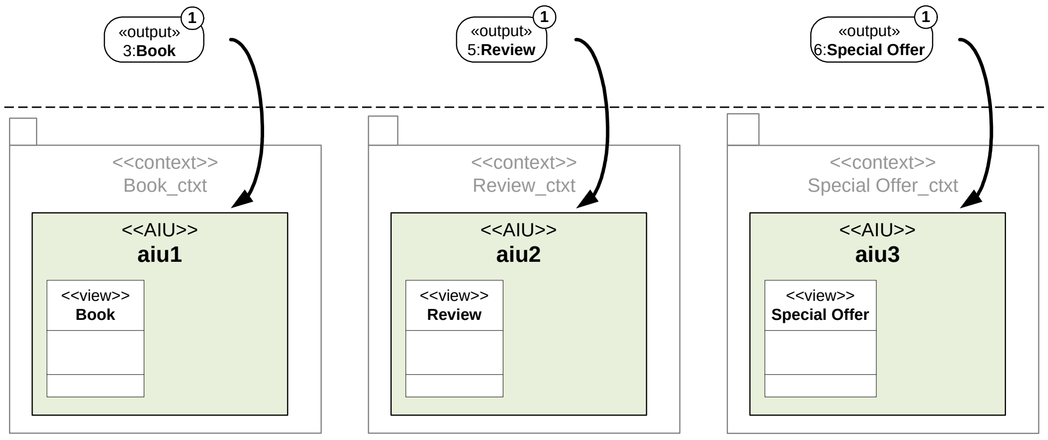


Figure 7.2: Definition of AIU's through Modelling Rule 1

7.3.2 Modelling Rule 2: List AIU's

RS: An *Output IP* (let us name it *outIP1*) that informs about multiple instances of an *ent1* entity (cardinality *).

O-MS: A *List AIU*, whose manager class is a navigational view of the class that represents the *ent1* entity. If the outgoing flow of *outIP1* leads to another *Output IP* that informs about one instance from *ent1*, then a reflexive context relationship is added to the manager class.

Description:

The requirement expresses the need to provide a list of multiple instances of a given entity. In OOWS, this is modelled through a *List AIU*. The special case indicates the additional requirement of providing navigation from that list, i.e., the requirement of an index. This is made by including the a *reflexive context relationship*, which allows navigating to another context that contains the detailed information of the selected instance.

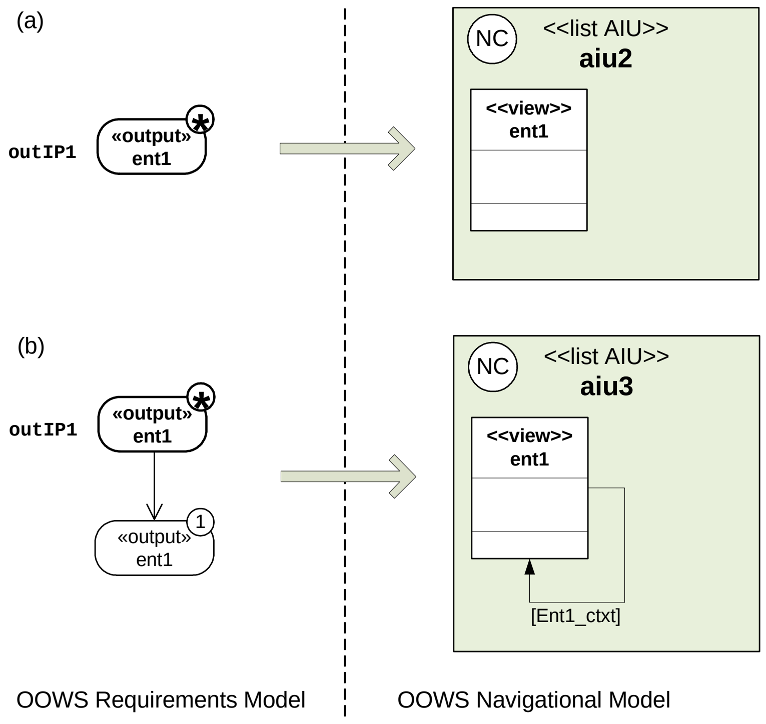


Figure 7.3: Graphical definition of Modelling Rule 2

Figure 7.3 shows the graphical correspondence of this modelling rule. The upper right section shows the modelling of the general case, while the lower right section shows the modelling of the derived index, including the reflexive context relationship that leads to *"Ent1_ctxt"*, which symbolizes a context that contains an AIU with detailed information about *ent1* instances.

Example:

Three *List AIU*'s are obtained from the application of this rule in the studied case, as Fig. 7.4 shows. In the case of `(1:Book,*)` and `(2:Book,*)`, both are linked to an Output IP with cardinality 1, which also informs about the `Book` entity (`(3:Book,1)`). For this reason, *reflexive context relationships* are associated to manager classes of the corresponding AIU's, leading to the context that contains the AIU generated from `(3:Book,1)`. In this way, two indexes to detailed information of `Book` instances have been modelled (case (b) from Fig. 7.3). In the other side, `(4:Author,*)` IP does not lead to any other IP, so the corresponding *List AIU* represents a non-indexed list of instances (case (a) from Fig. 7.3).

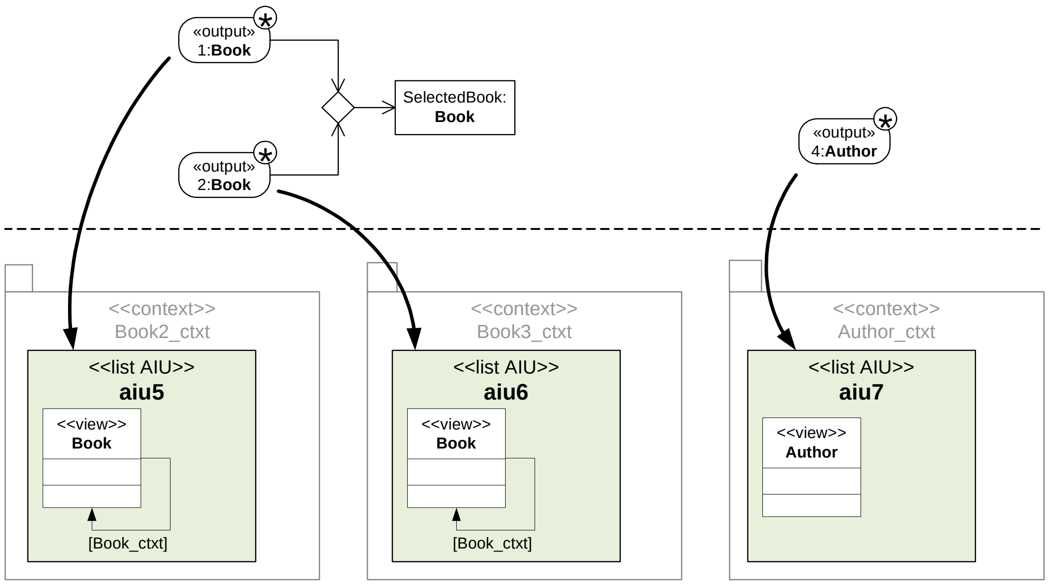


Figure 7.4: Definition of AIU's through Modelling Rule 2

7.3.3 Modelling Rule 3: Context Relationships

RS:	A *direct flow* from *outIP1* to *outIP2 Output IP's*, which inform of instances from *ent1* and *ent2* entities, respectively. Both Output IP's have been modelled through *aiu1* and *aiu2* AIU's, respectively.
O-MS:	A *context relationship* in *aiu1*, from its manager class to a new *complementary class* of *ent2*. This relationship leads to a context that contains *aiu2*. As a consequence, a *navigational link* is created from the context that contains *aiu1* to the context that contains *aiu2*.

Description:

The direct flow from *outIP1* to *outIP2* indicates the need to retrieve information from *ent2* that is associated to instances from *ent1*. However, the previous existence of respective AIU's leads to provide a navigational path between them, by defining the mentioned context relationship and complementary class. This modelling solution affects the global navigational description, by the definition of the corresponding *navigational link* that describes this new navigational alternative.

Figure 7.5 shows the graphical definition of this rule. In the upper right section, *ent2* complementary class is included, along with the corresponding context relationship, whose *context attribute* corresponds to the context that contains *aiu2* (modelled from *outputIP2*). In the lower right section, the derived *navigational link* between both contexts is depicted.

Example:

In the example, this rule is applied to all the *Output IP*'s with flows that enter from `(3:Book,1)` IP. As Fig. 7.6 shows, they are modelled as *complemen-*

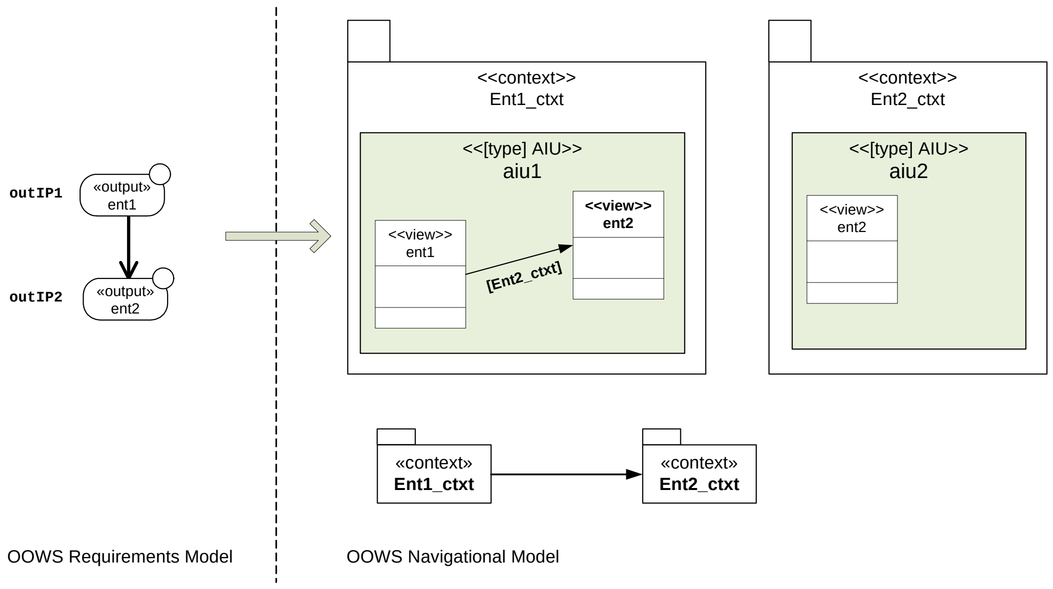


Figure 7.5: Graphical definition of Modelling Rule 3

tary classes and their flows are transformed into *context relationships*, which lead to the contexts containing the corresponding AIU's previously generated through Modelling Rules 1 and 2. As a result, the `Author`, `Review` and `Special Offer` classes complement the main information provided through the `Book` manager class, also providing the possibility to navigate for detailed information provided in other AIU's.

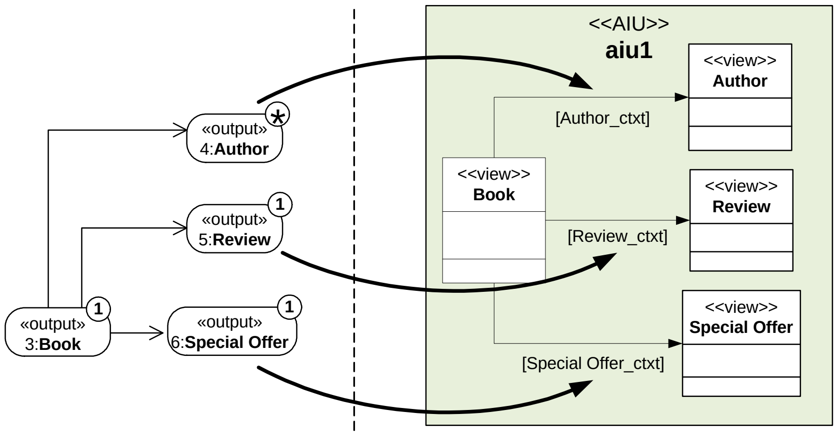


Figure 7.6: Definition of Context Relationships through Modelling Rule 3

7.3.4 Modelling Rule 4: Navigational Operations

RS:	A *System Action*
O-MS:	A *navigational operation*, which is included in the navigational class that corresponds to the closer *Output IP* that leads to it.

Description:

The inclusion of a *System Action* describes the access to execute a given service or operation. In the OOWS Navigational Model, this is represented through a *navigational operation*. The access to this operation is packed together with the data provided in the last previous interaction with the user, i.e., the (manager or complementary) navigational class derived from the closer *Output IP* that leads to the *System Action*. The graphical definition of this rule is shown in Fig. 7.7.

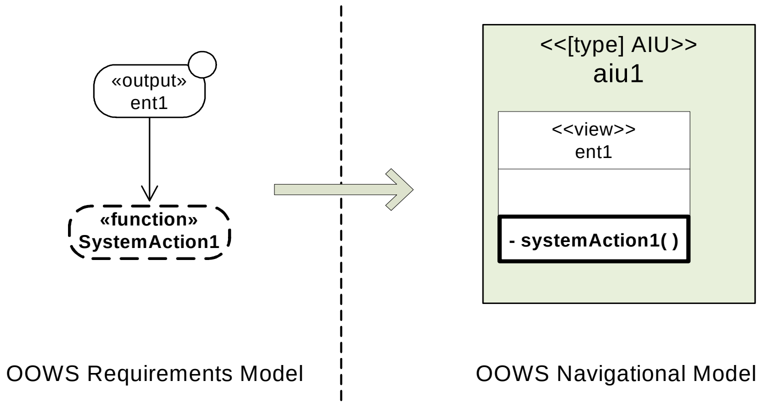


Figure 7.7: Graphical definition of Modelling Rule 4

Example:

The considered activity diagrams of the example only includes two *System Actions*, both reachable from `(3:Book,1)` IP. For this reason, these actions are included as *navigational operations* of the `Book` manager class, in the AIU generated from that Output IP, as Fig. 7.8 shows.

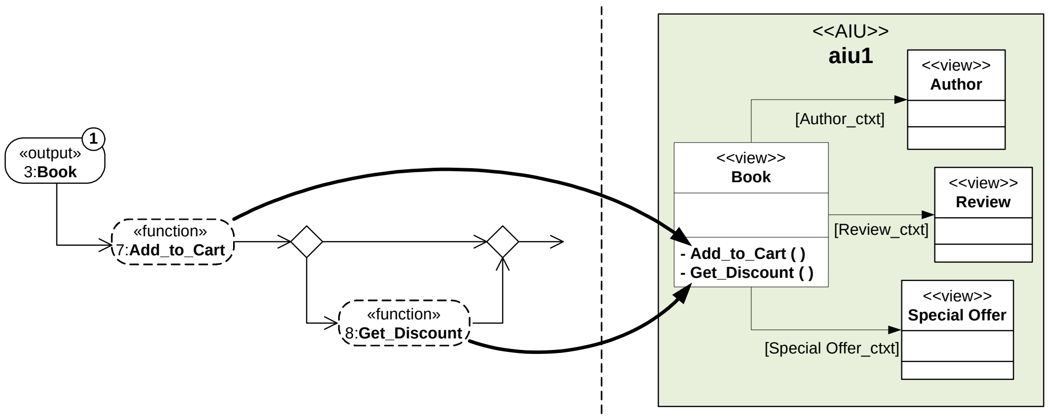


Figure 7.8: Definition of Navigational Operations through Modelling Rule 4

7.3.5 Modelling Rule 5: Navigational Attributes

RS: An *entity feature*, from *ent1* entity.

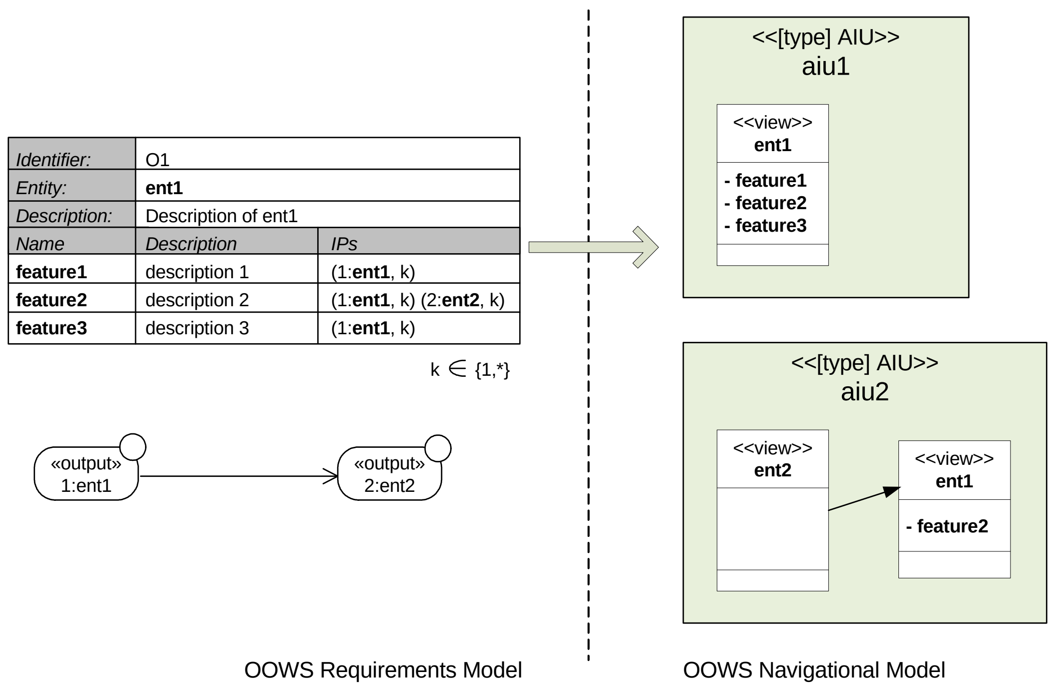
O-MS: A *navigational attribute*, included in a navigational class according to the following options:

1. If the feature may be accessed from an *Output IP* corresponding to *ent1*, then the associated navigational attribute is included into the *manager class* of the AIU obtained from that *Output IP*.
2. If the feature may be accessed from an *Output IP* of a different entity, but directly connected to an *Output IP* of *ent1*, then the associated navigational attribute is included into the *complementary class* corresponding to that entity, in the AIU obtained from that *Output IP*. If that complementary class does not already exist, it must be created.

Description:

This rule allows detailing the data that the system is requested to provide in the different Output IP's. Each entity feature is a navigational attribute. If a feature is requested in a Output IP from its entity, then the associated attribute is part of the main information of the corresponding AIU, i.e., it is an attribute of the manager class. On the contrary, if it is requested in an Output IP from other entity, then it complements the main information of that other entity, and the attribute is part of a complementary class of the corresponding AIU.

Figure 7.9 shows the graphical definition of this rule. In the *Data Description* of *ent1*, the three described features may be accessed from an Output IP of *ent1*, then they are incorporated in the manager class of the corresponding AIU (upper right section of Fig. 7.9). Furthermore, *feature2* may be accessed



Identifier:	O1	
Entity:	**ent1**	
Description:	Description of ent1	
Name	*Description*	*IPs*
feature1	description 1	(1:**ent1**, k)
feature2	description 2	(1:**ent1**, k) (2:**ent2**, k)
feature3	description 3	(1:**ent1**, k)

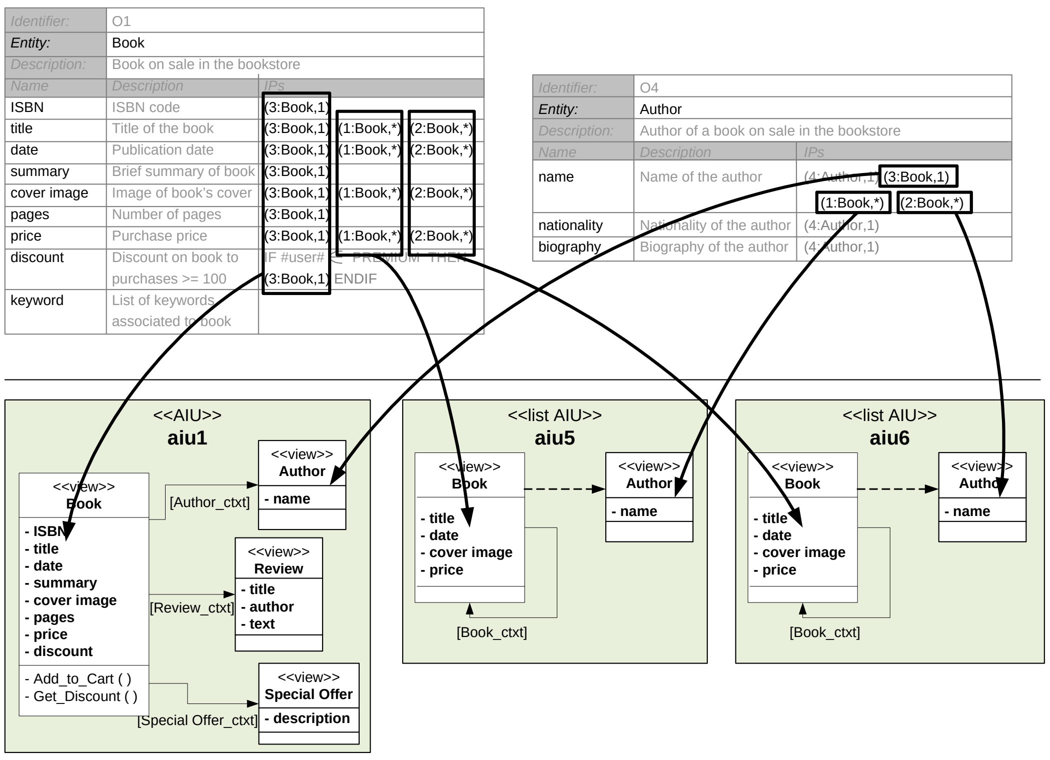
k ∈ {1,*}

Figure 7.9: Graphical definition of Modelling Rule 5

from an Output IP that informs of instances from other entity (*ent2*). For this reason, a complementary class of *ent1* is created in the corresponding AIU, including the *feature2* navigational attribute (lower right section of the figure).

Example:

By applying this rule, the attributes of the defined navigational classes are modelled. Figure 7.10 shows the transformation of features included in *Data Descriptions* of the `Book` and `Author` entities. `Book` features are included in Output IP's that inform about `Book` entities. This corresponds to the Modelling Solution 1 of this rule. For this reason, these features are modelled as navigational attributes of the manager classes from the respectively generated AIU's. In the case of the `Author` entity, transformations consider only a feature included in IP's about other entity (`Book`). As the Modelling Solution 2 of this rule indicates, corresponding navigational attributes are included in the `Author` complementary classes. Navigational attributes of the `Review` and `Special Offer` complementary classes are also obtained according to Modelling Solution 2.



Identifier:	O1			
Entity:	Book			
Description:	Book on sale in the bookstore			
Name	*Description*	*IPs*		
ISBN	ISBN code	(3:Book,1)		
title	Title of the book	(3:Book,1)	(1:Book,*)	(2:Book,*)
date	Publication date	(3:Book,1)	(1:Book,*)	(2:Book,*)
summary	Brief summary of book	(3:Book,1)		
cover image	Image of book's cover	(3:Book,1)	(1:Book,*)	(2:Book,*)
pages	Number of pages	(3:Book,1)		
price	Purchase price	(3:Book,1)	(1:Book,*)	(2:Book,*)
discount	Discount on book to purchases >= 100	IF #user# ∈ PREMIUM THEN (3:Book,1) ENDIF		
keyword	List of keywords associated to book			

Identifier:	O4	
Entity:	Author	
Description:	Author of a book on sale in the bookstore	
Name	*Description*	*IPs*
name	Name of the author	(4:Author,1) (3:Book,1) (1:Book,*) (2:Book,*)
nationality	Nationality of the author	(4:Author,1)
biography	Biography of the author	(4:Author,1)

Figure 7.10: Definition of Navigational Attributes through Modelling Rule 5

7.3.6 Modelling Rule 6: Context-Dependency Relationships

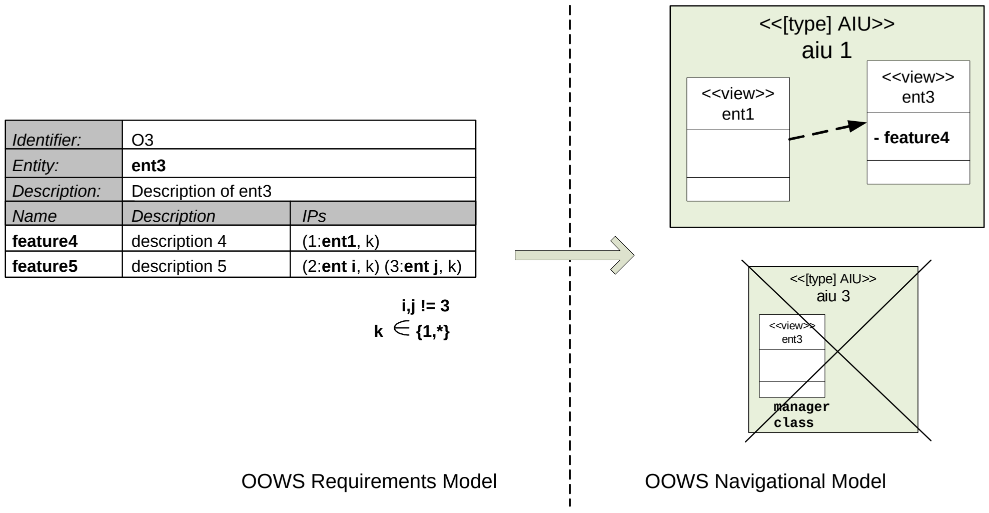
RS: An *ent3 entity* whose features are available only through *Output IP*'s that inform about other entities, different than *ent3*

O-MS: The definition of one *context-dependency relationship* leading to each complementary classes defined from *ent3*. Described *ent3* features are including as *navigational attributes* in such complementary classes. Any previously defined AIU whose manager class corresponds to *ent3 must be discarded.*

Description:

This rule describes how to model data about an entity that is required only to complement information of other entities, and no detailed information about its instances is required. By applying previous modelling rules 1 and 2, AIU's from *Output IP*'s that specifically inform about *ent3* may have been created. However, this complementary role discards the usefulness of such AIU's, suggesting their deletion. As a consequence, navigational relationships that lead to complementary classes defined from *ent3* only retrieve information, with no navigational options. For this reason, those relationships are classified into the context-dependency group and described features are only visible through the corresponding complementary classes.

Figure 7.11 describes this rule. In the *Data Description* of *ent 3*, the two included features may not be accessed from any *Output IP* of *ent3*. In the upper right section of the figure, the inclusion of one *ent3* feature is shown, while in the lower part the deletion of a generic AIU, previously created from *ent3*, is represented.



Identifier:	O3	
Entity:	**ent3**	
Description:	Description of ent3	
Name	*Description*	*IPs*
feature4	description 4	(1:**ent1**, k)
feature5	description 5	(2:**ent i**, k) (3:**ent j**, k)

Figure 7.11: Graphical definition of Modelling Rule 6

Example:

Data Description of the `Special Offer` entity shows a unique feature, only included in `(3:Book,1)`. By applying this rule (see Fig. 7.12), this feature is modelled as a *navigational attribute* of the complementary class defined from `(6:Special Offer,1)` through Modelling Rule 2 (see Fig. 7.4). The *Single-Object AIU* defined from the same IP (through Modelling Rule 1, see Fig. 7.2) becomes unnecessary, because its unique navigational class has no attributes nor operations, and thus it is discarded. The context relationship firstly defined to connect the `Special Offer` class to the corresponding manager class is replaced by a *context-dependency relationship*, through which no navigational capability is provided.

Identifier:	O3	
Entity:	Special Offer	
Description:	Extra benefit associated to the purchase of a book	
Name	*Description*	*IPs*
description	Description of the Offer	(3:Book,1)

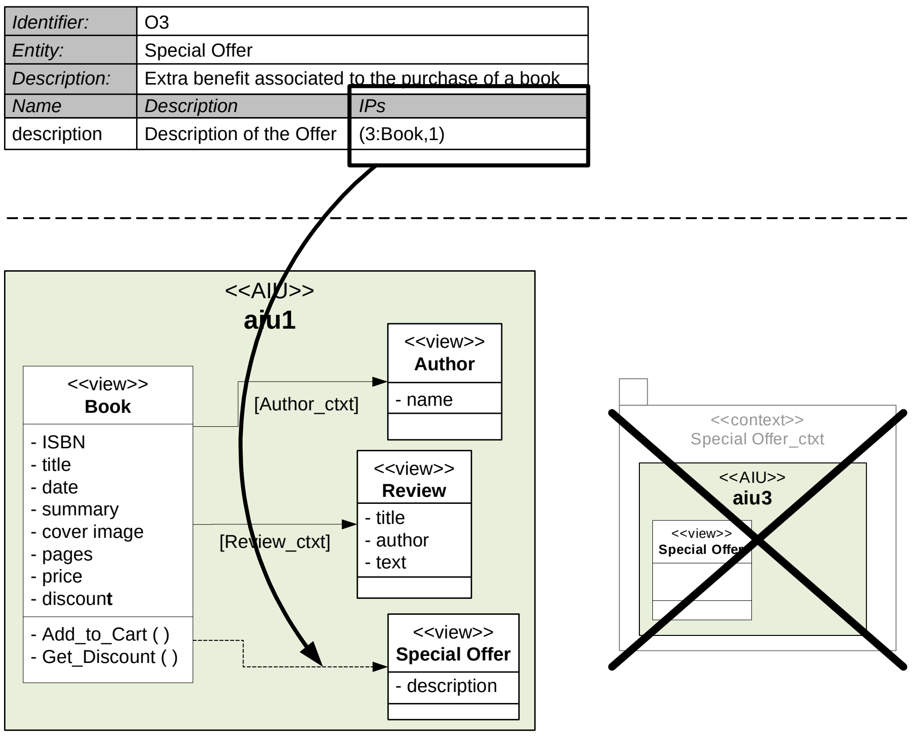


Figure 7.12: Definition of Context-Dependency Relationships through Modelling Rule 6

7.3.7 Modelling Rule 7: Adaptive Population Filters

RS: A *Selection Precondition* defined on the flow that leads to an *outIP* Output IP.

O-MS: An *Adaptive Population Filter* defined in the navigational class modelled from *outIP*. If two navigational classes have been derived from *outIP*, the filter is defined only in the generated *complementary class*.

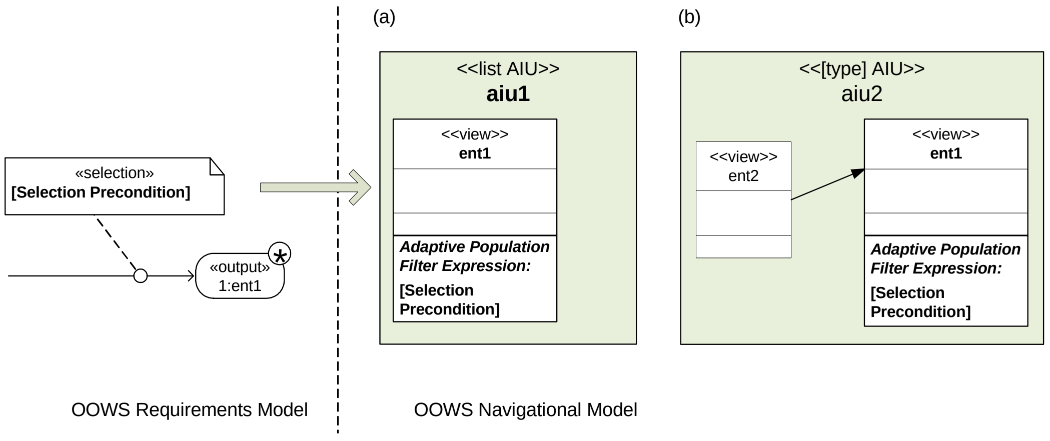


Figure 7.13: Graphical definition of Modelling Rule 7

Description:

Selection Precondition expresses the requirement of selecting some instances from the list provided in a given Output IP. The selection criteria are included as the filtering expression of the *Population Filter* structure of the OOWS Navigational Model. The adaptive character of the filter is determined by the inclusion of user characteristics in those criteria. When a given *Output IP* has generated a manager class and a complementary class, the filter should be applied when users firstly access data about the corresponding entity. For this reason, the definition of *Population Filters* in complementary classes is preferred. Figure 7.13 shows the ways to model that structure, depending on the type of navigational class (manager or complementary) generated from the affected Output IP.

Example:

The only occurrence of this requirement is the *Selection Precondition* defined for `(1:Book,*)` IP. Expressed in a semi-formal language, it prevents the access to information of `books` that the current user already owns. This precondition is modelled as an *Adaptive Population Filter* of the `Book` manager class of `aiu1`, describing this condition in OCL language, at the bottom of the primitive (description has been split for readability purposes).

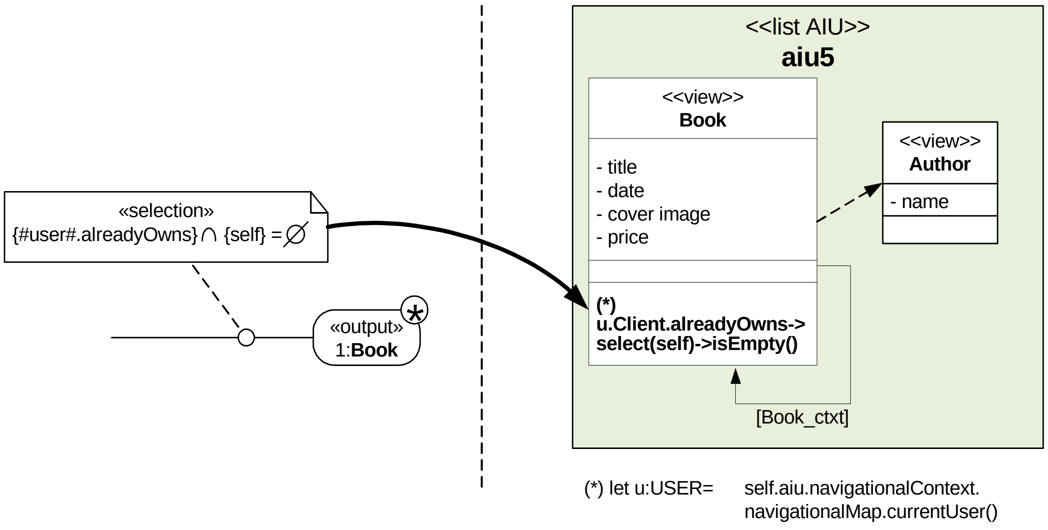


Figure 7.14: Definition of Adaptive Population Filters through Modelling Rule 7

7.3.8 Modelling Rule 8: Adaptive Ordering Patterns

RS:	A *Sorting Precondition* defined on the flow that leads to an Output IP.
O-MS:	An *Adaptive Ordering Pattern* defined in the navigational class modelled from *outIP*. If two navigational classes have been derived from *outIP*, the ordering pattern is defined only in the generated *complementary class*.

Description:

Sorting Precondition has been incorporated to OOWS Activity Diagram to express the requirement of providing the list of instances that may be accessed from an Output IP with a given order of presentation. The ordering criteria defined in the requirement are included as the ordering expression of the *Ordering Pattern* structure of the OOWS Navigational Model. The adaptive character of both structures is given by the inclusion of user characteristics in those criteria. As in the previous rule, it is suggested to associate the *Sorting Precondition* to the complementary class, in case of two navigational classes generated from the affected Output IP. Figure 7.15 shows the ways to model that structure, depending on the type of navigational class generated.

Example:

The three Output IP's with multiple cardinality of the considered activity diagrams are defined by this requirement. Figure 7.16 shows the application of this rule to two of them. In the left side of the figure, the specified *Sorting Precondition* describes an adaptive criterion to order the multiple instances that may be accessed in `(1:Book,*)` IP: the similarity of the presented books with those already visited by the current user, in terms of number of keywords in common. This criterion is modelled by defining an *Adaptive Ordering Pattern* in the navigational class derived from that IP

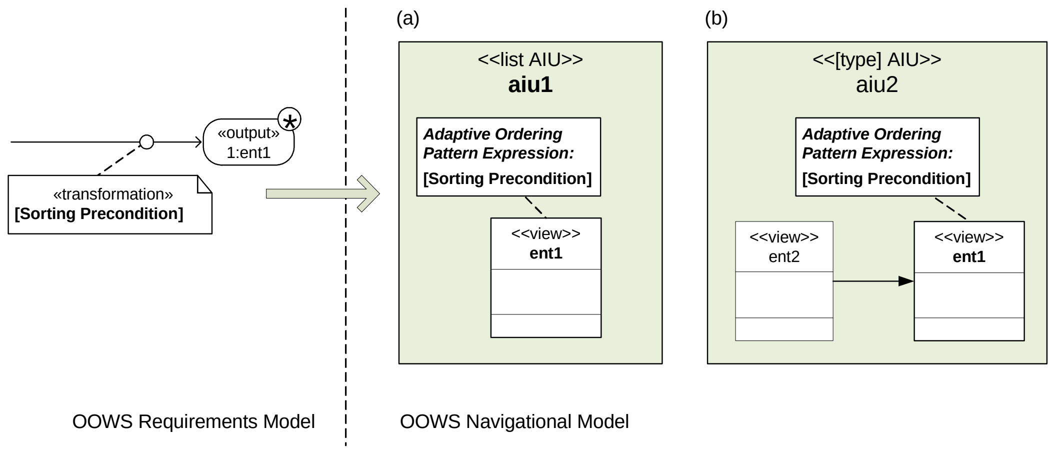


Figure 7.15: Graphical definition of Modelling Rule 8

(`Book` manager class of `aiu5` AIU) and expressed in OCL, in terms of the structures of the User Schema. Meanwhile, the right side of the figure shows a non-adaptive *Sorting Precondition* that affects the `Author` instances that are shown in `(4:Author,*)` IP, ordering according to `name` feature alphabetically. As seen in Fig. 7.4 and Fig. 7.6, two navigational classes have been modelled from this IP. The *Ordering Pattern* is defined in the derived complementary class only, and expressed in terms of the corresponding navigational attribute.

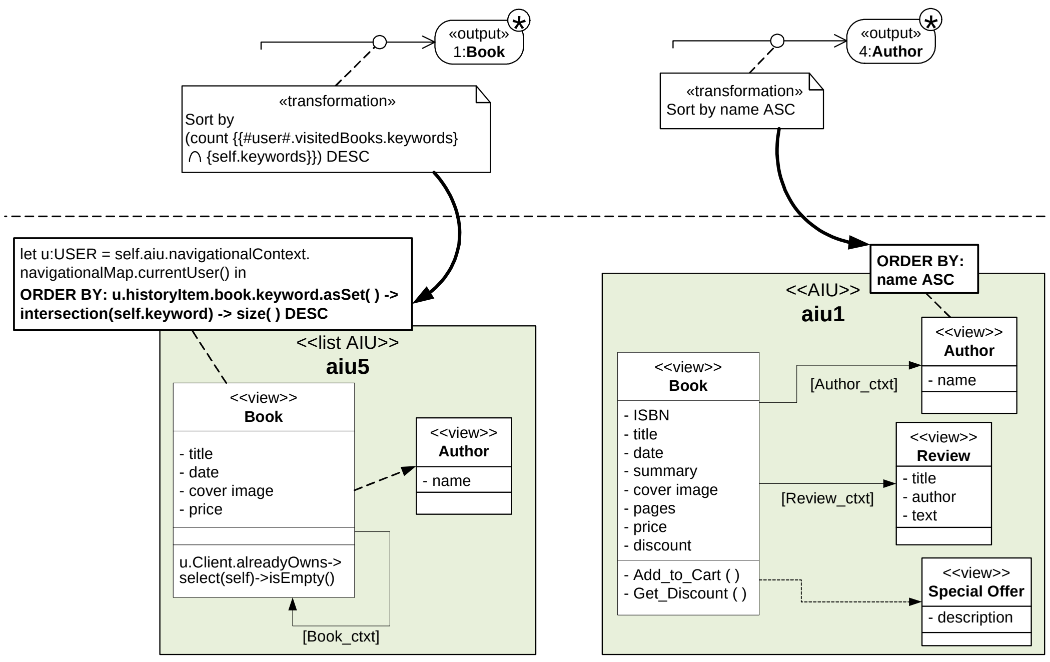


Figure 7.16: Definition of Adaptive Ordering Patterns through Modelling Rule 8

7.3.9 Modelling Rule 9: Adaptive Visibility of Navigational Relationships

RS: An *Access Condition* defined on the flow that leads to a given Output IP, from which a complementary class has been defined.

O-MS: An *Adaptive Visibility Condition* of the navigational relationship that allows accessing the complementary class whose modelling has been derived from that Output IP.

Description:

This requirement expresses the need to constraint the access to the information provided in a given Output IP. In OOWS, this is expressed by defining an Adaptive Visibility Condition on the navigational relationship that leads to the complementary class that contains the information. If it is not already created, we propose to define this relationship as a context relationship, by default, providing the possibility to navigate to another context. The use of other modelling rules may cause a modification of this type.

Figure 7.17 shows the graphical definition of this rule. The specified *Access Condition* is incorporated as the visibility expression of the context relationship that leads to *ent2* complementary class, generated from the constrained Output IP.

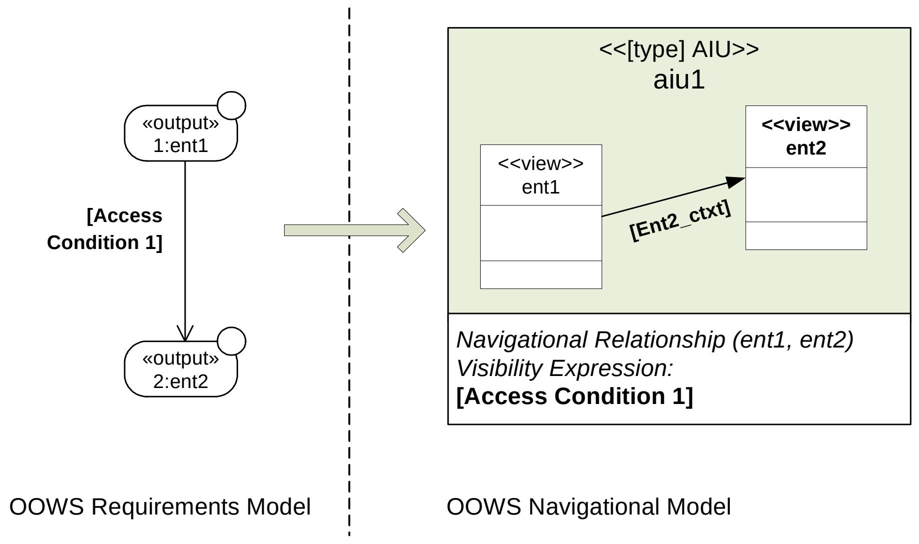
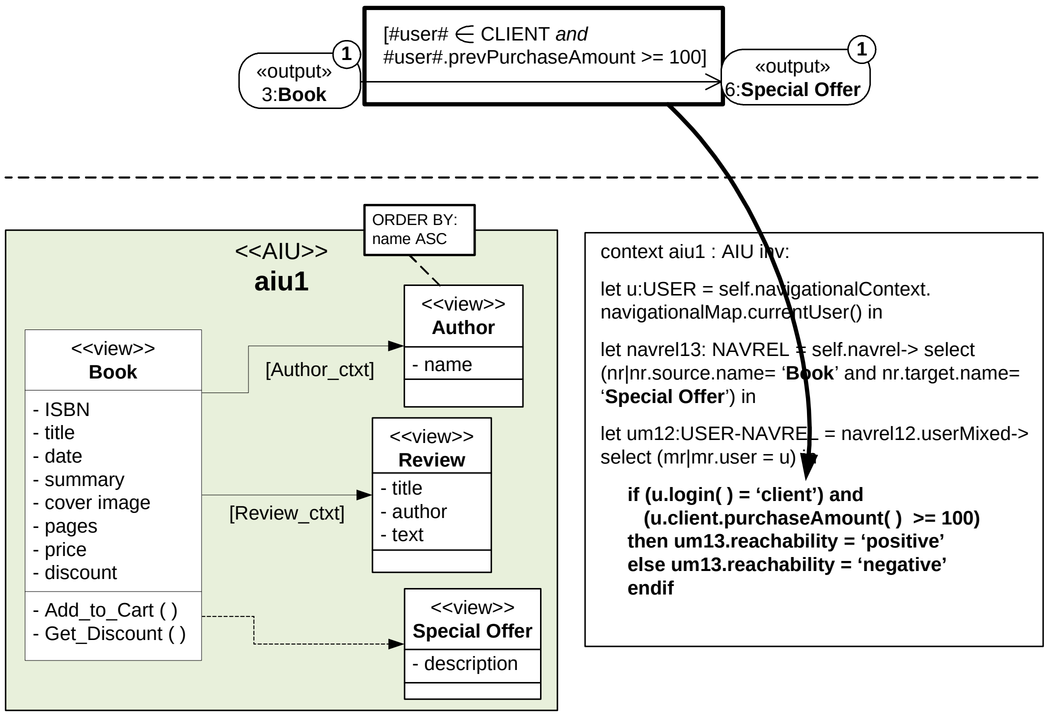


Figure 7.17: Graphical definition of Modelling Rule 9

Example:

In the example, the only occurrence of this requirement is an *Access Condition* that constrains the access to `(6:Special Offer,1)` IP. This condition expresses the requirement to adapt the access to information of an `Special Offer` associated to a given `book`: only `CLIENT` users with an amount of previous purchases of at least 100 (euros). By applying Modelling Rule 9, this condition is incorporated as the *visibility* condition of the context-dependency relationship that leads to the `Special Offer` complementary class. Figure 7.18 shows the complete definition of the Visibility property of this relationship, in OCL.



```
context aiu1 : AIU inv:
let u:USER = self.navigationalContext.
navigationalMap.currentUser() in
let navrel13: NAVREL = self.navrel-> select
(nr|nr.source.name= 'Book' and nr.target.name=
'Special Offer') in
let um12:USER-NAVREL = navrel12.userMixed->
select (mr|mr.user = u) in
    if (u.login( ) = 'client') and
      (u.client.purchaseAmount( )  >= 100)
    then um13.reachability = 'positive'
    else um13.reachability = 'negative'
    endif
```

Figure 7.18: Definition of Adaptive Visibility of Navigational Relationships, through Modelling Rule 9

7.3.10 Modelling Rule 10: Multiple Adaptive Filtering and Ordering

RS: Two Output IP's that fulfil the following conditions:

1. Both Output IP's inform about multiple instances (cardinality *) from the same entity;
2. Flows that lead to each of them come from the same target, being previously divided by a decision structure. This implies the definition of an *Access Condition* on the respective flows; and
3. If any, flows born from both Output IP's lead to the same target, and are joined by a merge structure;

O-MS: AIU's previously modelled from Output IP's are *merged*. *Selection Preconditions* that may be defined on their respective entering flows are incorporated as an *Adaptive Population Filter* of the modelled navigational class; analogously, respective *Sorting Preconditions* are incorporated as an *Adaptive Ordering Pattern* of the navigational class. In both cases, preconditions from each flow are associated to the respective *Access Condition*, by means of an *if-then-else* clause.

Description:

This requirement describes the need to provide users with a list of instances of the same entity, but adopting different sorting and ordering criteria in their presentation, according to the characteristics of a given user. Due to both Output IP's provides access to the same information, their modelling is unified through the same navigational class in the corresponding AIU. To decide which of both criteria must be adopted, the characteristics of the current user are evaluated through the respective *Access Conditions*.

The first condition of this rule leads to apply the *Derivation Rule 2*. This means that the navigational class from both Output IP's may correspond to (a) the manager class of a List AIU, or (b) a complementary class. Figure 7.19 shows the graphical definition of this rule for the case (a). In that case, the access conditions are only used to decide the selection and ordering criteria to adopt.

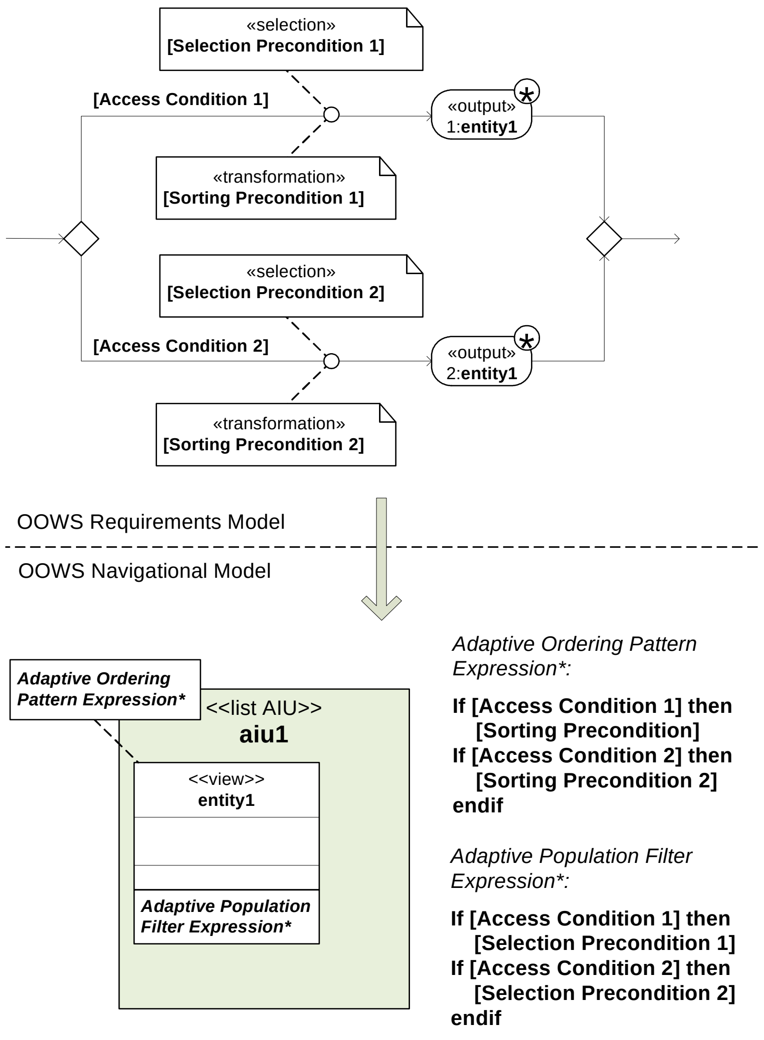

«selection»
[Selection Precondition 1]
[Access Condition 1]
«output»
1:entity1
«transformation»
[Sorting Precondition 1]
«selection»
[Selection Precondition 2]
[Access Condition 2]
«output»
2:entity1
«transformation»
[Sorting Precondition 2]
OOWS Requirements Model
OOWS Navigational Model
Adaptive Ordering Pattern Expression*
<<list AIU>>
aiu1
<<view>>
entity1
Adaptive Population Filter Expression*
Adaptive Ordering Pattern Expression*:
If [Access Condition 1] then [Sorting Precondition]
If [Access Condition 2] then [Sorting Precondition 2]
endif
Adaptive Population Filter Expression*:
If [Access Condition 1] then [Selection Precondition 1]
If [Access Condition 2] then [Selection Precondition 2]
endif

Figure 7.19: Graphical definition of Modelling Rule 10, case (a)

The case (b) of this rule, joined to the second condition of the current rule, leads to apply the *Derivation Rule 8*. As a consequence, the *Adaptive Visibility Condition* of the navigational relationship that leads to the obtained complementary class is composed of the union of the respective *Access Conditions* of both Output IPs. In this way, access to the complementary class are guaranteed to the fulfilment of any of both conditions. Figure 7.20 shows the graphical definition for this case.

Example:

In the example, `(1:Book,*)` and `(2:Book,*)` IP fulfil the conditions of the Requirements Specification fields of this rule: (a) both inform of multiple instances of the `Book` entity; (b) their access is ruled by a *decision* structure, with corresponding *Access Conditions* to both entering flows; and (c) their outgoing flows are joined through a *merge* structure, leading to the same target. Both IP's have generated respective List AIU's, with manager classes corresponding to `Book` (`aiu5` and `aiu6` in Fig. 7.21). For this reason, Modelling Rule 10 for the case (a) is applied. Both AIU's are merged, and one of them (`aiu6`) is discarded. Respective Access Conditions are modelled as conditional constraints that allow deciding which of both *Ordering Patterns* must be applied. In this case, for `CLIENT` users, the list of `books` are ordered according to an adaptive criterion, while `ANONYMOUS` users are provided with a list of `books` alphabetically ordered by their `titles`. Note that these conditions are considered through checking the *triggers* of the corresponding stereotypes. From the previously defined AIU's, only one had an *Adaptive Population Filter*, so it is preserved in the merged manager class, but adding the conditional clause corresponding to its *Access Condition*.

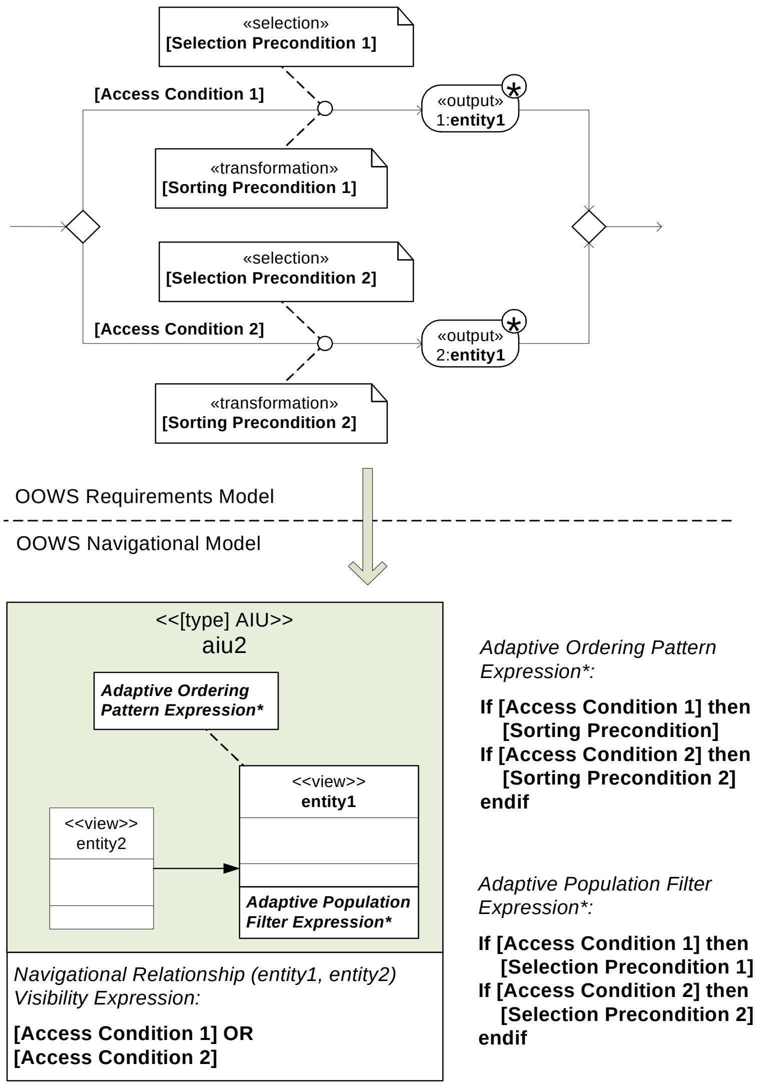

«selection»
[Selection Precondition 1]
[Access Condition 1]
«output»
1:entity1
«transformation»
[Sorting Precondition 1]
«selection»
[Selection Precondition 2]
[Access Condition 2]
«output»
2:entity1
«transformation»
[Sorting Precondition 2]
OOWS Requirements Model
OOWS Navigational Model
<<[type] AIU>>
aiu2
Adaptive Ordering Pattern Expression*
<<view>>
entity1
<<view>>
entity2
Adaptive Population Filter Expression*
Navigational Relationship (entity1, entity2) Visibility Expression:
[Access Condition 1] OR
[Access Condition 2]
Adaptive Ordering Pattern Expression*:
If [Access Condition 1] then
[Sorting Precondition]
If [Access Condition 2] then
[Sorting Precondition 2]
endif
Adaptive Population Filter Expression*:
If [Access Condition 1] then
[Selection Precondition 1]
If [Access Condition 2] then
[Selection Precondition 2]
endif

Figure 7.20: Graphical definition of Modelling Rule 10, case (b)

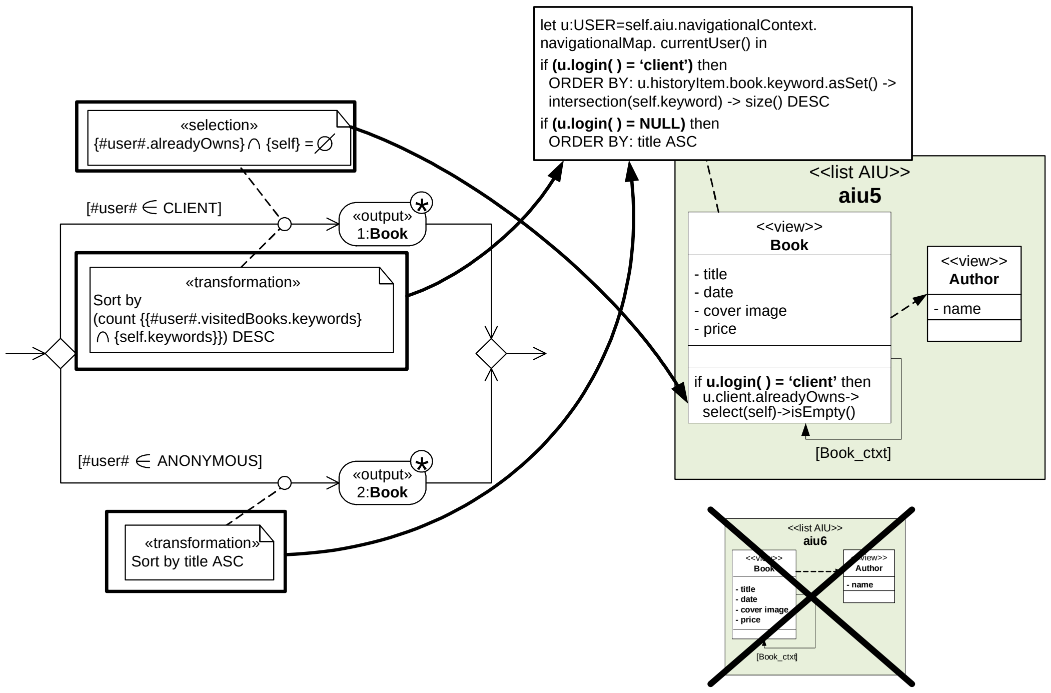


Figure 7.21: Multiple Adaptive Ordering and Filtering criteria through Modelling Rule 10

7.3.11 Modelling Rule 11: Adaptive Visibility of Navigational Operations

RS:	An *Access Condition* defined on the flow that leads to a given System Action.
O-MS:	An *Adaptive Visibility* condition of the navigational operation that is derived from that *System Action*.

Description:

The conditional access to a System Action is modelled as the condition visibility of the corresponding navigational operation, by means of the Adaptive Visibility property, as is shown in the graphical definition of Fig. 7.22.

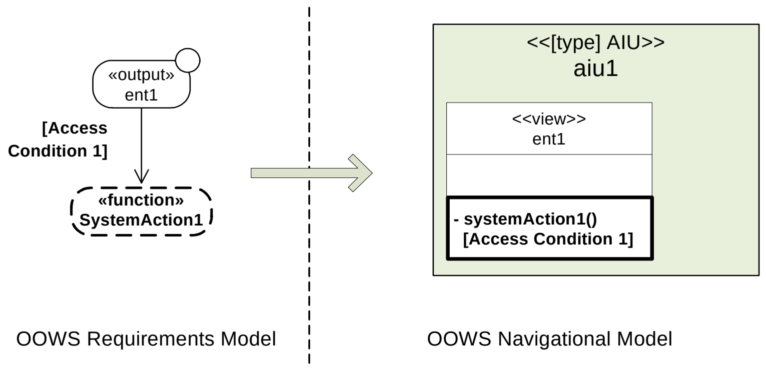
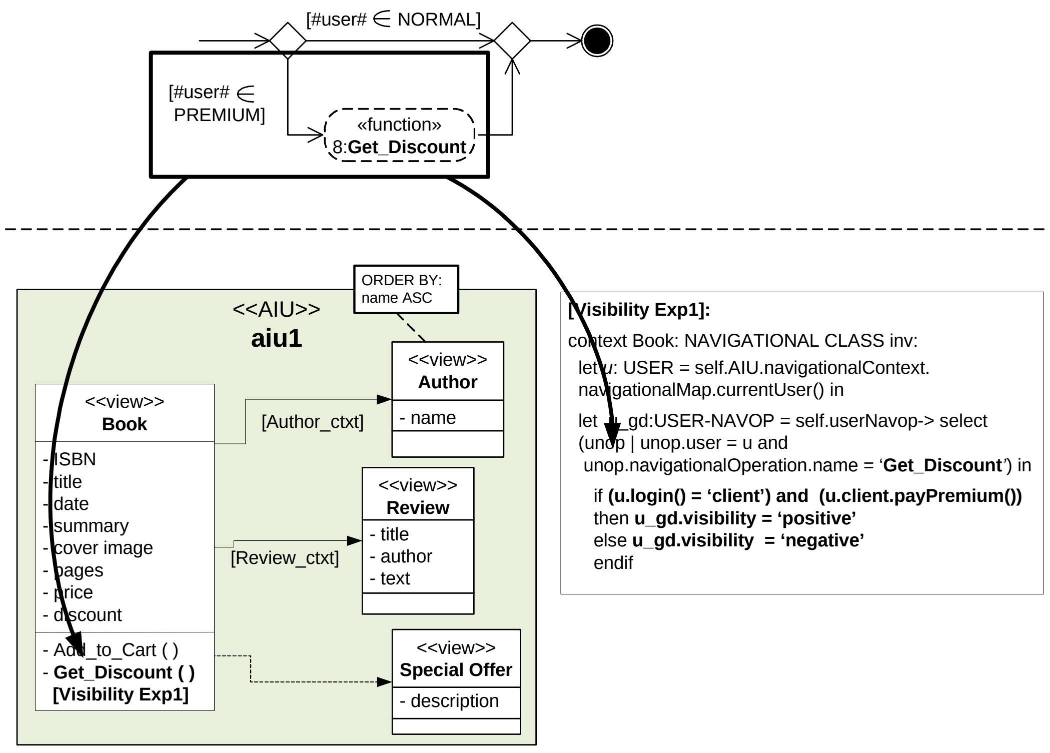


Figure 7.22: Graphical definition of Modelling Rule 11

Example:

The access to `(8:Get_Discount)` *System Action* is only allowed to `PREMIUM` users. This adaptivity requirement is specified through an *Access Condition*, and modelled as the *Visibility* expression of the corresponding navigational operation. Figure 7.23 shows this transformation. At the right of the AIU, the *Adaptive Visibility Expression* of the operation is shown. Bold text shows the adaptivity condition itself, in which the Access Condition is modelled through the triggers that a user needs to fulfil to be classified as a `PREMIUM` user. In such case, the *visibility* of the operation is positive.



```
[Visibility Exp1]:
context Book: NAVIGATIONAL CLASS inv:
  let u: USER = self.AIU.navigationalContext.
  navigationalMap.currentUser() in
  let u_gd:USER-NAVOP = self.userNavop-> select
  (unop | unop.user = u and
   unop.navigationalOperation.name = 'Get_Discount') in
    if (u.login() = 'client') and  (u.client.payPremium())
    then u_gd.visibility = 'positive'
    else u_gd.visibility  = 'negative'
    endif
```

Figure 7.23: Adaptive Visibility of Navigational Operations through Modelling Rule 11

7.3.12 Modelling Rule 12: Adaptive Visibility of Navigational Attributes - Mixed Relationships

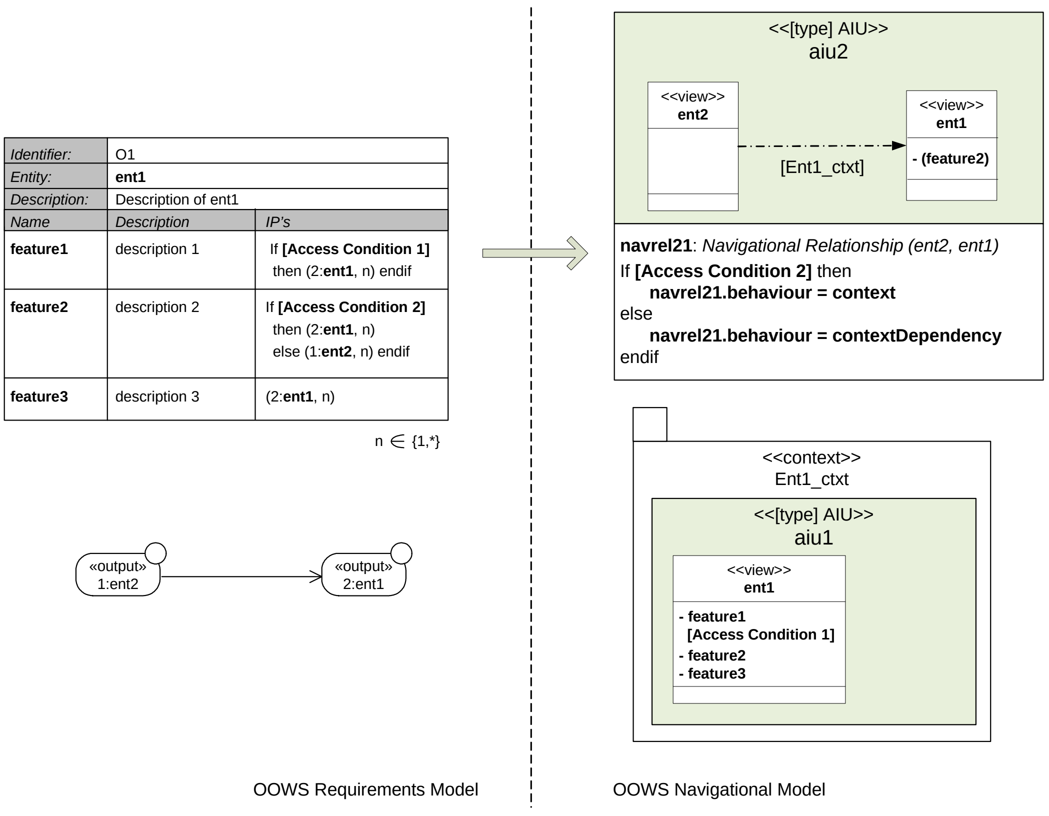
RS: An *Access Condition* defined on the Data Description of a given entity feature, constraining the Output IPs through which that feature may be accessed.

O-MS: This characteristic may be modelled according to one of the following two possible scenarios:

1. The Access Condition defines one possible Output IP for a given entity feature, by means of an *if-then* clause. In this case, the *Access Condition* is incorporated as an *Adaptive Visibility Condition* of the corresponding navigational attribute.

2. The Access Condition defines two possible Output IP's for a given entity feature, by means of an *if-then-else* clause: one of these Output IP's has derived the modelling of the manager class of a given AIU and the other has derived the modelling of the complementary class of another AIU. In this case, two navigational attributes corresponding to that entity feature are modelled, each included into the respective modelled navigational classes. Furthermore, a Mixed Relationship is defined in the second AIU, with the following characteristics:

    - it leads to the complementary class derived from one of the considered Output IP's.
    - its *conditional attribute* is the navigational attribute corresponding to the conditioned entity feature and incorporated in the complementary class.
    - its *context attribute* corresponds to the navigational context that contains the AIU whose manager class has been derived from the other Output IP.



Identifier:	O1	
Entity:	**ent1**	
Description:	Description of ent1	
Name	*Description*	*IP's*
feature1	description 1	If **[Access Condition 1]** then (2:**ent1**, n) endif
feature2	description 2	If **[Access Condition 2]** then (2:**ent1**, n) else (1:**ent2**, n) endif
feature3	description 3	(2:**ent1**, n)

n ∈ {1,*}

navrel21: *Navigational Relationship (ent2, ent1)*
If **[Access Condition 2]** then
    navrel21.behaviour = context
else
    navrel21.behaviour = contextDependency
endif

Figure 7.24: Graphical definition of Modelling Rule 12

Description:

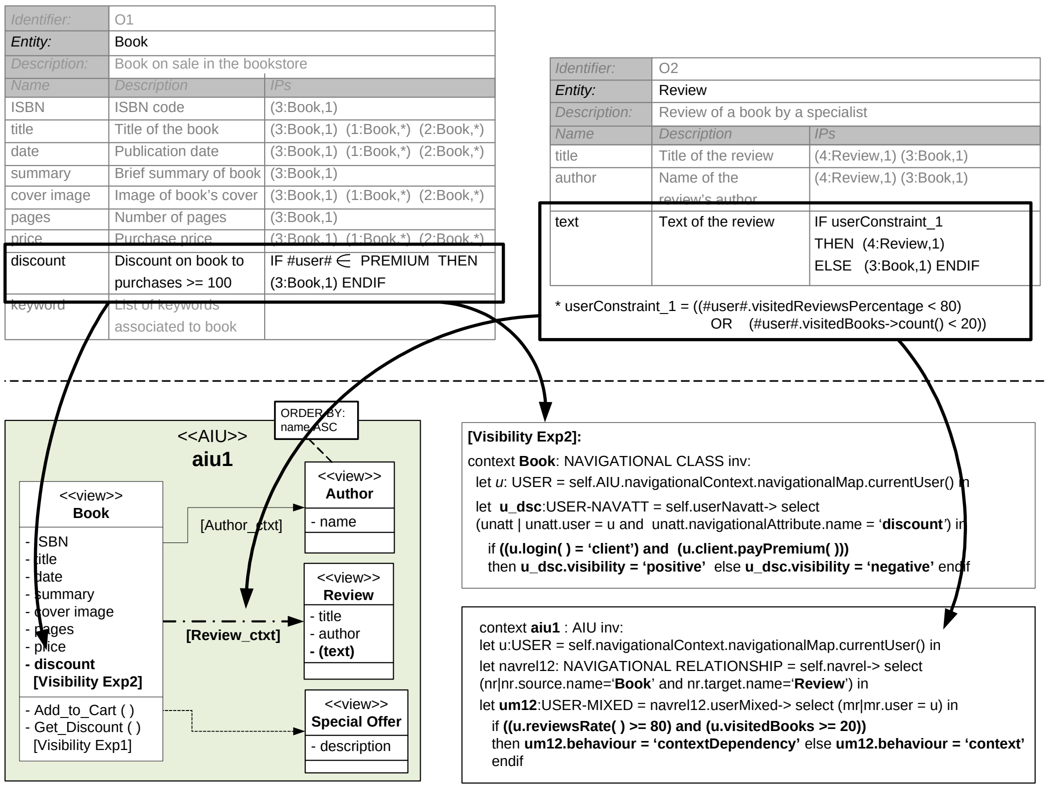
This requirement constraints the availability of a given feature in an Output IP, according to the fulfilment of an Access Condition. In the first scenario of this rule, i.e.., if there is only one Output IP through which the feature may be accessed, the *Access Condition* is incorporated as an *Adaptive Visibility constraint* in the definition of the corresponding attribute. The modelling solution for this scenario is shown in the upper right section of Fig. 7.24.

The second scenario is more complex. As described in *Modelling Rule 6*, the derived navigational attribute may be incorporated in a manager class or in a complementary class, depending on whether the listed Output IP informs about its corresponding entity or about another one, respectively. If the defined Access Condition forces to decide between these two alternatives, then a *mixed relationship* must be defined. This allows hiding the presentation of the corresponding attribute, being only visible in the corresponding manager class through a previous navigational action, or showing the attribute in the defined complementary class, avoiding the possibility of navigation. This modelling solution is shown in the lower part of the graphical definition of this rule, in Fig. 7.24.

Example:

The two described scenarios of this rule are illustrated in Fig. 7.25. The first scenario is present in *Data Description* of the `Book` entity, where the access to `discount` feature in a sole Output IP is constrained through an *Access Condition* (the `discount` associated to a given `book` is only provided to `PREMIUM` users). This condition is modelled in the Adaptive Visibility expression of the corresponding navigational attribute (`VisibilityExp2`, in the figure).

The second scenario is exemplified through the right-side transformation, from *Data Description* of the `Review` entity. Its `text` feature may be adaptively accessed from two different Output IP's: one of them has generated an AIU whose manager class is a view of `Review` (`aiu2`, in Fig. 7.2); the other has been modelled through an AIU whose manager class corresponds to `Book` (`aiu1`, in Fig. 7.2). In the first case, the corresponding `text` navigational attribute is part of the manager class of the AIU; in the second case, is included in a complementary class. Applying Modelling Rule 12, a *mixed relationship* is defined, leading to this complementary class. If a user fulfills the specified *Access Condition*, modelled as an OCL Expression, then the relationship adopts the *context-dependency behaviour* and `text` attribute is shown in the complementary class, avoiding the navigational step to access it; if the condition is not fulfilled, the relationship adopts the context behaviour and a navigational step is needed to access the value of `text` , from another AIU. In the example, the included *Access Condition* permits to visualize the `text` of the `Review` in the same page of the detailed information of the related `Book`, if the user has accessed at least 20 pages of different `books` and has navigated to access the related `review` in at least the 80% of these visits.



Identifier:	O1	
Entity:	Book	
Description:	Book on sale in the bookstore	
Name	*Description*	*IPs*
ISBN	ISBN code	(3:Book,1)
title	Title of the book	(3:Book,1) (1:Book,*) (2:Book,*)
date	Publication date	(3:Book,1) (1:Book,*) (2:Book,*)
summary	Brief summary of book	(3:Book,1)
cover image	Image of book's cover	(3:Book,1) (1:Book,*) (2:Book,*)
pages	Number of pages	(3:Book,1)
price	Purchase price	(3:Book,1) (1:Book,*) (2:Book,*)
discount	Discount on book to purchases >= 100	IF #user# ∈ PREMIUM THEN (3:Book,1) ENDIF
keyword	List of keywords associated to book	

Identifier:	O2	
Entity:	Review	
Description:	Review of a book by a specialist	
Name	*Description*	*IPs*
title	Title of the review	(4:Review,1) (3:Book,1)
author	Name of the review's author	(4:Review,1) (3:Book,1)
text	Text of the review	IF userConstraint_1 THEN (4:Review,1) ELSE (3:Book,1) ENDIF

```
* userConstraint_1 = ((#user#.visitedReviewsPercentage < 80)
                       OR    (#user#.visitedBooks->count() < 20))
```

```
[Visibility Exp2]:
context Book: NAVIGATIONAL CLASS inv:
  let u: USER = self.AIU.navigationalContext.navigationalMap.currentUser() in
  let  u_dsc:USER-NAVATT = self.userNavatt-> select
  (unatt | unatt.user = u and  unatt.navigationalAttribute.name = 'discount') in
     if ((u.login( ) = 'client') and  (u.client.payPremium( )))
     then u_dsc.visibility = 'positive'  else u_dsc.visibility = 'negative' endif
```

```
context aiu1 : AIU inv:
let u:USER = self.navigationalContext.navigationalMap.currentUser() in
let navrel12: NAVIGATIONAL RELATIONSHIP = self.navrel-> select
(nr|nr.source.name='Book' and nr.target.name='Review') in
let um12:USER-MIXED = navrel12.userMixed-> select (mr|mr.user = u) in
   if ((u.reviewsRate( ) >= 80) and (u.visitedBooks >= 20))
   then um12.behaviour = 'contextDependency' else um12.behaviour = 'context'
   endif
```

Figure 7.25: Adaptive Visibility of Navigational Attributes and Mixed Relationships through Modelling Rule 12

7.4 Conclusions

The modelling strategy presented in this chapter, based on the correspondences between adaptivity requirements and Adaptive Primitives, aims to fill the gap between the corresponding phases of the development of Adaptive Web Applications, which is present in most of the Model-Driven approaches to Web Adaptivity. This gap makes difficult to trace these requirements along the rest of the development process.

In particular, requirements specified through ***Activity Diagrams*** and ***Data Descriptions*** have an great influence in the navigational modelling. For this reason, these specifications receive special attention in this analysis. An *Activity Diagram* expresses the different navigational alternatives that the system provides to the user in the fulfilment of a given task, defining the access possibilities to contents, system actions and navigational paths. Furthermore, in this diagram the user-centered constraints that rule this kind of access are specified. In other words, it is the main container of adaptivity requirements. As a complement of this diagram, *Data Descriptions* specify the information required to fulfil a task and the specific interactions through which this information is available to the user. These specifications also include user-centered constraints that allow describing several Adaptive Primitives.

This chapter completes the Problem Specification stage of OOWS method to describe the adaptive features of a Web application. The correspondences between Requirements Specification and Conceptual Modelling phases permit to keep a permanent trace of the defined requirements along the whole development process of Adaptive Web Applications.

Chapter 8

Conclusions

The present thesis has introduced a Model-Driven proposal to the development of Adaptive Web Applications. This is made by extending an existing method of Web development (the OOWS Method), through the definition of conceptual tools that support the specification of a set of adaptive methods and techniques, which have been successfully implemented in traditional hypermedia systems.

In the following sections, we present: a brief review of the main parts of this thesis; a summary of its main contributions; a brief discussion of issues related to the implementation of the presented proposal; and a list of future research works that can complement this thesis.

8.1 Summary

A new type of conceptual primitives has been incorporated to the OOWS Navigational Model. These primitives, called ***Adaptive Primitives***, support the modelling of different *Adaptive Techniques*. In this way, an OOWS navigational schema describes the navigational aspects of a Web application considering different ways to adapt the presentation of contents and hyperlinks to the characteristics of different users. In some cases, the introduced primitives enhance the definition of already existing navigational primitives;

in others, it was necessary to completely define a new one. However, in both scenarios the compliance with the existing modelling proposal was carefully preserved, with the goal of allowing that a non-adaptive OOWS specification can evolve into a diagram of an Adaptive Web Application with no need of serious changes.

In OOWS, like in most of the Model-Driven approaches to Web development, navigational schemas are defined as views of the structural description of the problem domain. The definition of Adaptive (navigational) Primitives in terms of characteristics of users and their context led us to incorporate a ***User Model*** into the OOWS Class Diagram. Complementing the modelling of domain concepts, users are described through classes and associations, considering their personal features, their relationship with the particular application domain and the different navigational actions that they perform during a browsing session. This description can be continuously updated according to the feedback obtained from those navigational actions.

Adaptive Methods define different ways to provide adaptive functionality at a high abstraction level, in terms of the adaptation of the presentation of contents and links. Several of these methods are supported by the introduced Adaptive Primitives, and a set of *modelling strategies* to support their description are proposed. These strategies are intended to become a valuable help to developers in the adoption of this proposal to model Adaptive Web Applications.

New conceptual primitives augment the spectrum of modelling decisions that developers may take. To support these new decisions, a ***Requirements Model*** has been proposed. Taking as a basis a set of diagrams included in OO-Method and OOWS proposals for requirements specifications of non-adaptive applications, we extracted and organized those diagrams, proposed some enhancements and incorporated new specific diagrams. As a result, the proposal supports the specification of User-related Requirements, i.e., the user information that needs to be stored and updated to fulfil the adaptivity goals. By means of a *User Stereotypes Diagram*, relevant user groups are defined, to provide similar users with similar interaction alternatives. To specifically describe a given user, the required characteristics are specified through the *User Specification* diagram. The specification of *Adaptivity Requirements* is supported by the enhancements introduced to the OOWS proposal for navigational requirements, through the incorporation of user-centered constraints into the descriptions of the interactions needed to fulfil a given navigational task. In this case, the scope of navigational requirements that may be expressed through the enhanced diagram is highly extended, leading to the obtention of more complex navigational schemas and richer implementations. This strengthens the adopted decision to face the Adaptive Web Applications development from a high abstraction level.

Aimed at supporting the navigational modelling tasks, this work proposes a set of ***traceability rules***, which provides guidelines to model different occurrences of an *Adaptivity Requirements specification*, obtaining a schema that describes those requirements in terms of navigational structures.

8.2 Main Contributions

To summarize the main contributions of this thesis, we present the following list of solutions that this work provides to a set of problems detected in the model-driven development of Adaptive Web Applications.

1. **Problem**: Adaptivity in Web Applications has major influence on their navigation aspect. Because of this, a highly expressive Navigational Model is needed, which provide conceptual tools to adapt the contents and hyperlinks to be displayed in a navigational view, according to the current user's characteristics.

   Contribution: Enrichment of the conceptual model of the OOWS Model-Driven Web Development Method [97], by defining the navigational structures and properties (Adaptive Primitives) that allow specifying Adaptive Methods at a high-abstraction level. We adopt an object-oriented modelling approach by the need to define operations that capture user information over domain concepts, providing a higher expressiveness in terms of the functional dimension of the Web application.

2. **Problem**: A Model-Driven approach to Adaptivity requires to describe at a high abstraction level the characteristics of the potential users of

an Adaptive Web Application, their association with the application domain concepts and the needed functionality to keep track of the user-application interactions.

Contribution: An object-oriented User Model, which provide the conceptual tools to model: (a) the information of the user that is relevant to the application; (b) the relevance, interest, preferences, etc. that domain objects have for a given user; (c) the navigational behaviour of the user, i.e., the interactions that involve navigation between Web pages, along with the operations that keep the model updated, according to the navigational actions that user performs during a browsing session.

3. **Problem**: User-related and Adaptivity Requirements of Adaptive Web Applications are usually specified through textual descriptions, with low expressiveness and precision.

   Contribution: A *Requirements Model*, which provides conceptual tools to the specification of:

   - *User Requirements*, i.e., those requirements related to the potential users of the application. Supporting a coarse-grained adaptivity, groups of similar users are identified and hierarchically ordered, by adopting a stereotype-based approach [23]; to support a fine-grained adaptivity, specific user characteristics are defined, comprising personal, application-related and navigational behaviour features.
   - *Adaptivity Requirements*, i.e., those requirements that describe the adaptive functionalities that the application should provide to the user. By adopting a task-based approach, tasks for each user group are identified, classified and detailed, specifying the adaptive features of the involved user-application interactions.

4. **Problem**: There exists a gap between specifications of User and Adaptivity Requirements and the conceptual specification of the Navigation, which makes difficult to trace these requirements along the development process.

   Contribution: A set of Model to Model rules to systematically obtain the conceptual specifications of the adaptive features of a Web page, from their adaptivity requirements. The defined rules allow mapping specifications from the proposed Requirements Model to an OOWS Navigational schema.

8.3 Discussion about Implementation Issues

The enrichment of the interaction experience provided by Adaptive Web Applications motivates to face the complexity that their development involves. The provision of conceptual support to Adaptive Techniques is intended to bring the rationale of Adaptive Hypermedia field to the Model-Driven development of Web applications, under the belief that this convergence may help to promote the currently limited use of engineering methods to develop this kind of systems.

In this work, the definition of Adaptive Primitives was made by preserving the nature of the OOWS Navigational Model. The enhancements to already defined primitives and the introduction of user-centered constraints to provide them with adaptive behaviour, are defined as views of the structures included in the Class Diagram of the application. In this way, adaptivity becomes a key factor in the modelling of the navigation of a Web application.

However, as Paul De Bra, who is one of the lead researchers of Adaptive Hypermedia, states in [12]: *"A number of experiments have been conducted to demonstrate the benefits of adaptive hypermedia, mostly of adaptation of link structures. (...) we argue that the benefits of using an AHS are a result of careful authoring, more than of the adaptive techniques themselves"*. Precisely, in order to give support to the authoring process, the set of modelling strategies to describe Adaptive Methods was defined. In the application of the present proposal in some case studies, the set of defined Adaptive Primitives was introduced to a group of developers. However, when these modelling strategies were presented, they obtained the corresponding navigational schemas in a very easy way.

Considering the compliance of the introduced primitives with the OOWS method, the implementation of Adaptive Web Applications through the present proposal does not demand the consideration of many additional implementation issues, in comparison with the implementation of OOWS-based, non-adaptive Web applications. This is an important advantage of this proposal.

Currently, the present Adaptivity proposal has been implemented through different case studies: an online bookstore like the illustrated along this dissertation; a recommender system of music artists, based on permanently updated preferences and evaluations of proposed items; a News portal, which recommends the reading of news by capturing the interest areas of users through their previously reviewed news; and a travel agency site, which provides users with different trip advices, according to their profile, travelling experience and the evaluation of proposed travels and package tours. These applications has been implemented in different platforms and languages, like *PHP* [98] and *JSP* [99]. However, all of these implementations share the same implementation concerns.

In the consideration of user characteristics, this implementation approach abstracts the way in which information about users is captured. The storing of user information is made in the same way than the rest of the domain concepts, through the corresponding structures of the used Database Management System. The information of navigational behaviour that keeps the user model updated, is captured by means of different mechanisms of the respective implementation languages. In the case of adaptive features that are provided to non-authenticated (anonymous) users, the use of *HTTP cookies* [100] is a good alternative to manage the user information. Navigational actions of these users are encoded in a piece of text (cookie) by the system and sent back to the browser. In this way, every time the user accesses a page, the server receives the cookie and can use the encoded information to provide the desired adaptation.

In the implementation of the introduced Adaptive Primitives, we must distinguish those primitives that support Adaptive Presentation Techniques and those that support Adaptive Navigation Techniques:

- The implementation of primitives that support the specification of Adaptive Presentation Techniques must face the difficulty of altering the presentation template of the adaptive Web page.

  Pages that different users can access are defined at design-time through the *Adaptive Reachability of Navigational Contexts* primitive. In this case, the system assigns the navigational map to users once they have been authenticated, through mechanisms like HTTP authentication [101].

In the case of *Adaptive Visibility of Navigational Attributes and Operations Adaptive Visibility of Navigational Relationships* and *Adaptive Mixed Relationships* primitives, their implementation includes user-centered conditions that rule the displaying of the intended data items and operation anchors. Scripting languages of Web development, like PHP [98], JSP [99] or VBScript [102], support the definition of conditions for the displaying of these elements.

Finally, the *Adaptive Ranking of Navigational Relationships* primitive has been implemented through the use of different presentation templates for the corresponding Web page. Each of these templates considers the displaying of the ranked clusters of information in different order. Once authenticated, a user is assigned with their corresponding template for that page.

Along with the mentioned hypertext scripting languages, the definition of different templates of presentation can be made through *Cascade Style Sheets (CSS)* templates [103] and adaptively be assigned to different groups of users.

- Primitives that support the specification of Adaptive Navigation Techniques are mainly based on the selection and ordering of multiple instances of a given class, with the goal of providing users with orientation and guidance about their current position into the hyperspace and their best alternatives for future navigation. The implementation of these primitives involves the incorporation of selection and ordering expressions as part of the queries that must be made to the database when the corresponding dynamic page is generated. For instance, in a *SELECT* sentence of SQL [104], an adaptive selection expression can be incorporated as part of the *WHERE* statement, which imposes a condition to the selection of data; and an adaptive ordering expression may be translated to the *ORDER BY* statement, which defines a presentation order of the selected instances.

One of the main problems that developers of these implementations faced in the adoption of our proposal was the implementation of adaptive conditions. These correspond to the OCL constraints of the Adaptive Primitives. The translation of OCL expressions into sentences of query languages like SQL [104] must be supported by automatic mechanisms that facilitate this task.

Another concern that developers dealt with was the usability of the provided adaptive features. Although this aspect is out of the immediate scope of this proposal, the presented modelling approach should support the definition of usability guidelines that help to guide users in their adaptive interaction experience. For instance, textual or graphical clues may be provided to inform users when a given adaptive functionality is provided, and which are the conditions that they must fulfil in order to gain access to the different implementation alternatives. Furthermore, and with the goal of not imposing an adaptive solution to users, they can be provided with mechanisms to deactivate the tracking of their navigational actions and the adaptivity features themselves whenever they want.

Privacy is another issue that must be carefully considered in the development of Adaptive Web Applications. Users have little assurance that personal information that is provided at a Web application might not be misused, and usually are reluctant to submit personal information. Implemented solutions must keep their users permanently informed about the use of their personal information in the provision of adaptive functionality. This is one of the main factor that can decide the success of this kind of systems.

8.4 Publications

The work related to this thesis has been published in two international conferences, two international workshops and one national workshop.

8.4.1 International Conferences

- **Gonzalo Rojas**, Pedro Valderas, Vicente Pelechano (2006). *Describing Adaptive Navigation Requirements Of Web Applications.* IV International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems - **AH2006**. Dublin, Ireland, June 20-23, 2006, Proceedings. Lecture Notes in Computer Science 4018, Springer.
- **Gonzalo Rojas**, Vicente Pelechano (2005). *A Methodological Approach for Incorporating Adaptive Navigation Techniques into Web Applications.* VI International Conference on Web Information Systems Engineering - **WISE 2005**. New York, NY, USA, November 20-22, 2005, Proceedings. Lecture Notes in Computer Science 3806, Springer.

8.4.2 International Workshops

- **Gonzalo Rojas**, Vicente Pelechano, Joan Fons (2005). *A Model-Driven Approach to include Adaptive Navigational Techniques in Web Applications.* V International Workshop on Web Oriented Software Technologies - **IWWOST'05**. The 17th Conference on Advanced Information Systems Engineering (CAiSE'05)), Porto, Portugal.
- **Gonzalo Rojas**, Vicente Pelechano, Joan Fons (2004). *Navigational Properties and User Attributes for Modelling Adaptive Web Applications.* Engineering the Adaptive Web - **EAW'04** Workshop Proceedings. Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2004), Eindhoven, The Netherlands.

8.4.3 National Workshops

- **Gonzalo Rojas**, Vicente Pelechano, V.(2006). *Systematic Derivation of Navigational Specifications from Adaptivity Requirements.* IV Taller en Sistemas Hipermedia Colaborativos y Adaptativos. Sitges, España, 03 de octubre de 2006.

8.5 Future research

The proposal presented in this thesis may be complemented by different research works. Some of them are:

- The set of proposed adaptivity primitives may be enhanced in order to support the modelling of Adaptive Methods that are not considered in this work. *Personalized views* method, which describes the adaptation of the work space by selecting the relevant clusters of information that a Web page presents to a user, is already implemented in some Web applications, but mainly in an adaptable way, i.e., the adaptation is made explicitly by the user (e.g., *iGoogle*, *http://www.google.com/ig*). The provision of user-centered constraints that rule the way to display such segments may support the adaptation of this kind of applications in an adaptive way, with no explicit intervention of the user required.

- The goal of the definition of traceability rules from Adaptivity Requirements to OOWS Navigational Schemas is to facilitate the task of the modeller, by providing guidelines to obtain the navigational description of the adaptive features of the application. The application of these rules is intended to be complemented with a modelling strategy to obtain the Structural, Functional and Dynamic schemas of the application, along with its global navigational structure. To support the traceability of adaptivity requirements to navigational schemas, the consideration of the defined rules in an automatic Model-to-Model transformation proposal should be a challenging contribution.

- As a major goal that has oriented the development of this work, we propose to integrate this proposal in the *OO-Method Automatic Code Generation Method* [20], in order to automatically generate Adaptive Web Applications from conceptual schemas. This requires to define the formal specification of the introduced adaptivity primitives and the corresponding transformation rules to different implementation platforms.

- In order to rapidly validate the navigational modelling of OOWS-based Adaptive Web Applications, we propose the development of a CASE tool that allows obtaining prototypes of the application, from partial or complete navigational schemas. In this way, these schemas can be rapidly validated, with development time and effort saving. Furthermore, this would permit to capture the feedback of users from their interaction with the generated application, in order to improve adaptivity constraints or recommendation algorithms.

Appendix A

The OOWS Navigational Model for Non-Adaptive Web Applications

The *OOWS Navigational Model* [19] provides developers with conceptual primitives that allow specifying the navigational aspects of an Hypermedia Application. The OOWS Method [97] proposes the achievement of this specification in two main phases: (a) *Authoring-in-the-Large*, where the global structure of nodes and hyperlinks is specified; and (b) *Authoring-in-the-Small*, where the inner structure of each node is detailed.

In the following sections, we present a review of the OOWS Navigational Model, based on these two phases. This review has the goal of describing the conceptual framework on which the present proposal of adaptivity modelling is based.

A.1 Authoring-in-the-Large: Global Structure of the Navigation

In the ***Authoring-in-the-Large*** phase, the modeller describes the navigational paths that different kinds of users are allowed to explore, through the definition of a set of Navigational Maps. The primitives of the OOWS Navigational Model that are used in this step are:

A.1.1 Navigational Map

A ***Navigational Map*** represents a global view of the Web application defined for a given group of users. It is expressed as a directed graph, in which the nodes, called *Navigational Contexts*, contain the information and functionality reachable by the corresponding users. The arcs of the graph, called *Navigational Links*, define all the possible navigational paths among the set of Contexts the user are allowed to browse through. A *stickman* symbol, which represents the group of users that have access to the navigational map, is the starting point of the graph.

A.1.2 Navigational Contexts

A ***Navigational Context*** represents an interaction point between users and the application. Each context contains at least one navigational view of the *Class Diagram* of the application, where users have access to attributes and operations with some semantic closeness among them. Graphically, a Navigational Context is represented as a UML package, stereotyped with the ≪context≫ keyword. There are two types of navigational contexts:

1. ***Exploration Contexts***. An *Exploration Context* is reachable from any other navigational context. Its inclusion in a Navigational Map implicitly defines a navigational link that points it, called *exploration link* (see next subsection). From the set of Exploration Context of a map, the one whose content is displayed when the user logs in the

system is called ***Home Context***. Graphically, an Exploration Context is depicted as a UML package marked with an "E" label.

2. ***Sequence Contexts***. A Sequence Context is a navigational context that only can be accessed from other navigational context(s), by following one or more predefined navigational paths. These paths are determined by linking different contexts through *sequence links* (see next subsection). Graphically, a Sequence Context is depicted as a UML package marked with an "S" label.

A.1.3 Navigational Links

In a Navigational Map, the navigational contexts are connected through the so-called ***Navigational Links***. These links define the paths that the corresponding users are allowed to cross during their browsing process. There are two types of navigational links:

1. ***Sequence Links***. They define a semantic navigation between contexts, by carrying the information of an object selected in the source navigational context to the target context. Graphically, Sequence Links are represented with solid arrows.

2. ***Exploration Links***. They represent an intentional change of task made by the user. In contrast to the navigation defined by sequence links, this navigation does not involve carrying information between contexts. These links are implicitly defined by the set of exploration contexts of a navigational map. They are graphically represented as dashed arrows from the *stickman* symbol. The exploration link that leads to the *Home Context* of the map is labelled with an "H".

Figure A.1 shows an example of a Navigational Map, corresponding to an online bookstore and defined for a *Client* user type. It is composed of five exploration contexts (those with "E" label) and four sequence contexts (with "S" label). The two types of navigational links are included: dashed arrows from the stickman symbol representing the exploration links of the map, and solid arrows that denote the sequence links. The Exploration Context Main

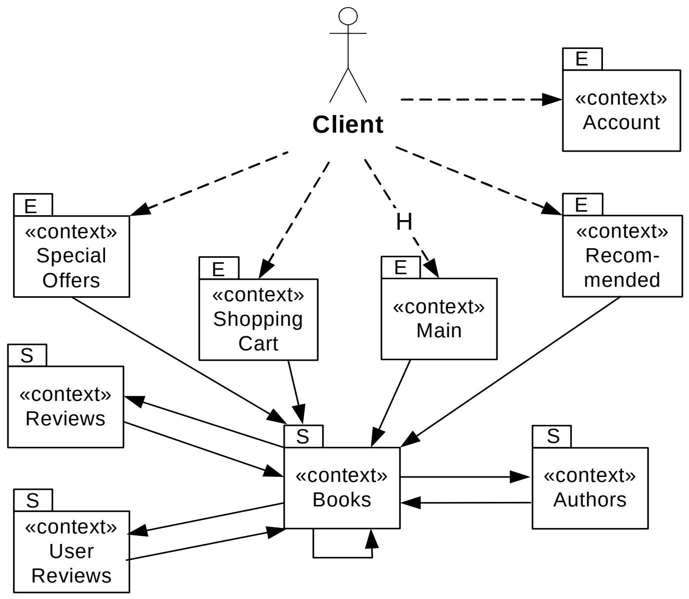


Figure A.1: Example of OOWS navigational map and the two types of navigational contexts

is the Home Context of the Map, which is indicated through the "H" letter in the exploration link that points it.

A.2 Authoring-in-the-Small: Inner structure of Navigational Contexts

Navigational maps defined in the *Authoring-in-the-Large* phase provide a global view of the navigational paths that users are allowed to follow. In the ***Authoring-in-the-Small*** phase, a more detailed specification is provided, by defining the inner structure of each navigational context of a navigational map, i.e., the way that navigational paths can be accessed from the different pages of the hypermedia network. Different navigational views of the structural schema are contained in multiple interaction units, and complemented with conceptual structures specially oriented to support the browsing process.

The conceptual primitives of the OOWS Navigational Model that are used in the *Authoring-in-the-Small* phase are the following:

A.2.1 Abstract Information Units (AIU)

Each navigational context is the abstraction of a Web page of the final application. To model the multiple sections of a Web page, OOWS defines a Navigational Context as an aggregation of several ***Abstract Information Units (AIU)*** [96]. Each AIU is a navigational view of the structural schema (Class Diagram), which provides a set of cohesive data and operations, representing the retrieval and displaying of chunks of related information.

In a contextual navigation action, information of a selected object is carried from the source context to a target one. This information must be captured by an AIU in order to show specific information about the selected object. OOWS distinguishes AIUs of two types, according to whether they support contextual navigational or not:

1. ***Contextual AIU*** (graphically labelled with a circled "C"), which instantiates its manager class to the object received as contextual information; and
2. ***Non-Contextual AIU*** (labelled with a circled "NC"), which does not depends on the contextual information received by its containing context.

A.2.2 List AIU and Single-Object AIU

From the different content units that a Web page can aggregate, some of them may show information about one single instance, while others may present a list of data and operations related to several instances.

To model the presentation of information related to multiple objects, currently OOWS proposes the definition of ***indexes*** [19], which are structures that provide an indexed access to the population of objects from a navigational class. Indexes create a list of summarized information, allowing the user to choose one instance from the list. However, the use of this structure makes difficult to express the adaptivity characteristics of such index list.

In this work, we propose to define two types of Abstract Information Units (AIUs), according to the number of objects that are retrieved in each one when its corresponding context is accessed. These two types are the following:

1. **List AIU**. This kind of AIU provides access to data and operations of multiple objects. In the specification of this structure, two modelling scenarios must be distinguished:

   - When the AIU is ***contextual***, the contextual information arrives to the context through a relationship with an (X:N) cardinality in the corresponding relationship-end, and indexed access about only the referred objects is provided.
   - If the AIU is ***non-contextual***, no contextual information arrives and information about all the objects from the manager class can be accessed, being only constrained by possible population filters defined on the AIU or presentation aspects.

   A list of multiple objects may also provide an indexed access to further information of a selected object. To model an ***index***, the manager class of the *List AIU* must have a reflexive (context) navigational relationship, which provides access to the detailed information of the selected instance, by navigating to a context that contains a Single-Object AIU.

2. **Single-Object AIU**. This AIU provides access to information about only one single object from the manager class. These AIUs are always contextual, and the contextual information arrives through a relationship with (X:1) cardinality in the corresponding end.

Graphically, *List AIU*s are represented by a UML package stereotyped with the ≪list AIU≫ keyword, while the Single-object AIU is simply labelled with the ≪AIU≫ keyword.

A.2.3 Navigational Classes

AIUs are composed of ***Navigational Classes***, which represent views over classes from the Class Diagram. These views are represented graphically as UML classes that are stereotyped with the "view" keyword and that contain the set of attributes and operations that will be available to the user.

In each AIU, two kinds of navigational classes are included:

1. **Manager Class**: Each AIU must include one and only one manager class, which provides the main information about the most important concept of that application view.
2. **Complementary Classes**: They are optional navigational classes that provide information complementary to the provided by the manager class.

Attributes and operations included in both types of navigational classes are named as **Navigational Attributes** and **Navigational Operations**, respectively.

Navigational operations may provide navigation capabilities, along with their intended functionality. A **Service Link** defines a navigation that will automatically be performed after the execution of a navigational operation. Beside the signature of the operation, the name of the context to which the navigation leads is included between brackets.

A.2.4 Navigational Relationships

All navigational classes are related by unidirectional binary relationships, called ***Navigational Relationships***. These relationships are defined over existing association or inheritance relationships defined in the Class Diagram. To eliminate any possible ambiguity in the case of multiple relationships between two classes, navigational relationships must include the role name of the relationship (depicted graphically as /role-attribute/) as well as the attribute used as anchor to move between navigational contexts. Moreover, depending on the navigational capability of the navigational relationship these can be of two types:

1. ***Context Dependency Relationship*** (graphically represented using dashed arrows), which represents a basic information retrieval by crossing a structural relationship between classes. When a context dependency relationship is defined, all the object instances related to the origin class object are retrieved; and
2. ***Context Relationship*** (graphically represented using solid arrows), which represents a recovery of the information provided by the target complementary class, plus a contextual navigation to a target context. This navigation creates a *Sequence Link* in the corresponding Navigational Map.

Due to the navigation capabilities that provide, *Context relationships* have some special properties:

- A ***context attribute*** that indicates the target context of the navigation (depicted as [target context]).
- A ***link attribute*** that specifies the attribute used as the "anchor" to activate the navigation to the target context. The link attribute is usually an attribute of the target navigational class. For usability reasons, it is sometimes interesting to define the anchor as a "label" (static text).

Figure A.2 shows the *Authoring-in-the-Small* specification of the *Book* navigational context included in the navigational map of Fig. A.1. This context represents a Web page that shows detailed information of a book on-sale in an Web-based bookstore, also including a list of books recently added to the bookstore stock. Each content unit is modelled through a respective AIU:

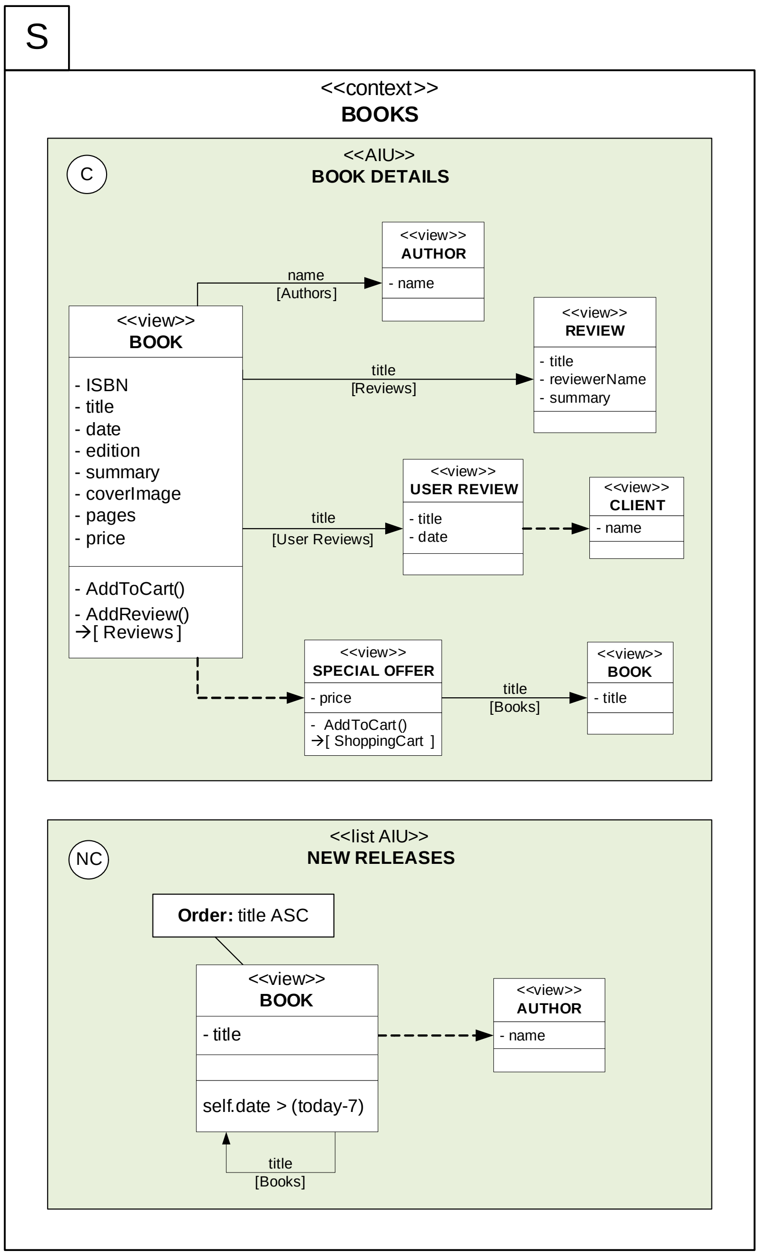


Figure A.2: Example of Navigational Context, composed of a Single-Object AIU and a List AIU

1. *Book Details*, whose navigational classes show detailed information of a particular book. This AIU is *contextual*, i.e., the contextual information received by *Book* (sequential) context determines which entity instance will be shown. Furthermore, this is a *Single-Object AIU*, being intended to display information about one entity instance only.

2. *New Releases*, whose navigational classes shows few data and operations of a list of *Book* instances. This AIU is *non-contextual*, i.e., the displayed *Book* instances are independent of the contextual information received by the context. It is also a *List AIU*, where the summarized information of each book contains a link to another context, i.e., it describes an *index*. As the reflexive relationship of Fig. A.2 indicates, the user will be led to the same *Book* context, where *Book Details* will show information about the selected book.

Each AIUs of the example has a navigational view of *Book* class as its *manager class*. However, in *Book Details* AIU, this navigational class includes more *navigational attributes* than the corresponding to *New Releases* AIU, along with two *navigational operations*. *AddReview()* operation from the manager class of *Book Details* has a value for *Service Link*: once this operation is executed, the user is led to *Review* navigational context.

Book Details also contains much more complementary information than the other AIU, through the inclusion of six *complementary classes*, in comparison with only one from *New Releases*. Both AIUs have *navigational relationships* defined between their respective *Book* and *Author* navigational classes. In *Book Details*, this relationship is a *context* one, with *name* as a *link attribute* and *Author* as a *context attribute*. This means that values of *name* attribute from *Author* class are used as anchor of the navigation to other context (*Author* context), where detailed information about the author of the currently shown book is provided. On the contrary, *Book* and *Author* navigational classes from *New Releases* AIU are connected through a *context dependency relationship*, which allows retrieving the name of the associated author, but with no navigational capabilities.

Another OOWS structures that enrich the specification of pages that show multiple instances of the same entity, i.e., those pages that are modelled through navigational contexts containing *List AIU*s, are the *Population Filter* and the *Ordering Pattern.*

A.2.5 Population Filters

When a list of multiple objects is shown in a Web page, the number of instances included can be inconveniently high, causing information overloading and problems of presentation. To provide a mechanism to select a subset of instances to be included in such a list, OOWS introduces the ***Population Filter*** structure.

A *Population Filter* is a constraint defined over a navigational class. The system only shows information of those objects that fulfil that constraint, which is expressed in OCL [56] language. It gains special importance when it is defined on the manager class of an AIU, applying its condition to the main objects of the navigational view. In this case and according to the type of AIU on which the filter is defined, we distinguish two special modelling scenarios:

- If the AIU is *contextual*, the *population filter* provides an additional condition to the instances to be shown, along with the contextual selection, i.e., the condition defined through the contextual information that the corresponding context receives.
- If the AIU is *non-contextual*, the *population filter* is the only way to select the instances to be shown. If no filter is defined, the AIU would allow showing the instances corresponding to all the objects of the manager class.

In both types of AIUs, the definition of population filters on complementary classes has the same effect of imposing an additional constraint to the retrieved objects, along with the already defined through the corresponding navigational relationships.

Graphically, this constraint is depicted at the bottom section of the navigational class primitive. In *New Releases* AIU from Fig. A.2, a *Population Filter* defined on the manager class constrains the retrieved *Book* instances to those whose value of *releasedate* attribute belongs to the last week.

Even when population filters are more useful when multiple objects are retrieved, it can be also defined when only one object is involved. In this sense, a special case occurs when a population filter is defined on the manager class of a Single-object AIU, when the filter may forbid the access to the only object that is going to be shown. However, the population filter is more frequently defined on the manager class of a List AIU.

A.2.6 Ordering Pattern

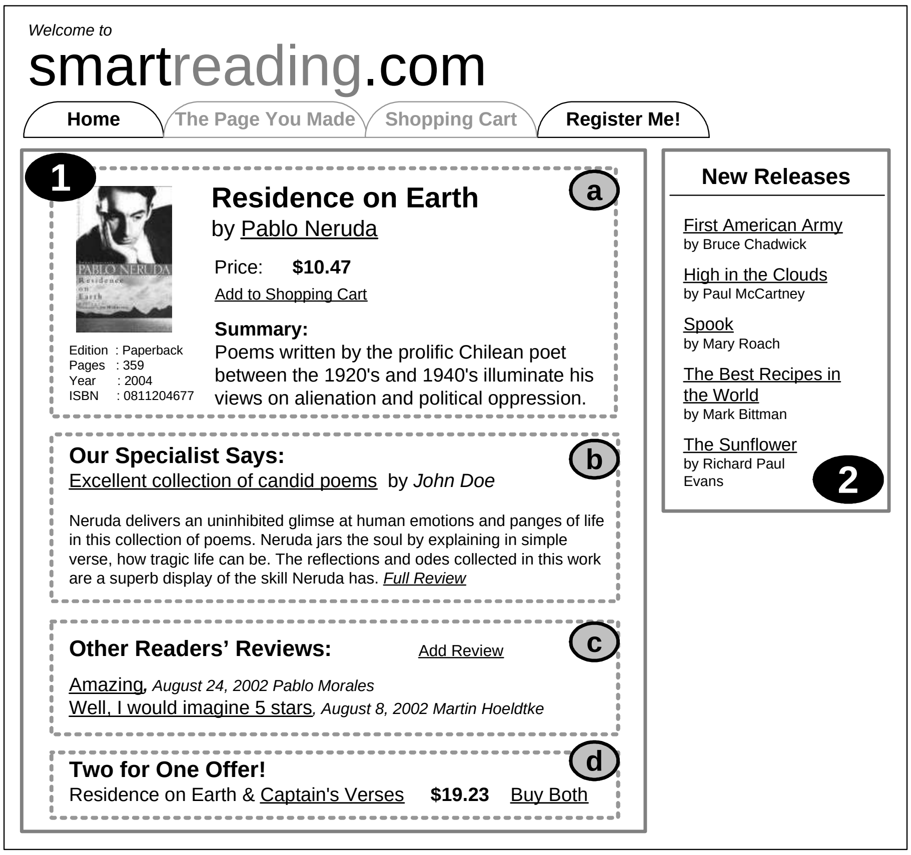
When a list of multiple instances is displayed, a certain order on its presentation may help users to find the most relevant items quickly. OOWS includes the ***Ordering Pattern*** structure to sort the information of multiple instances from a navigational class. An ordering pattern comprises multiple ***ordering statements***, each of which containing the specification of an order criterion and a direction of order, which can be ascending or descending. These criteria are specified through an OCL constraint, expressed in terms of structures from the Class Diagram. When the information of a navigational class is required, the associated ordering pattern (if any) is applied, invoking each of its ordering statements in a certain order of priority, defined in the corresponding OCL expression at modelling time.

According to the type of navigational class on which the Ordering Pattern is defined, we distinguish two special cases:

- If it is defined on a manager class, the Ordering Pattern allows ordering the information of instances from the main class of the AIU. The implementation of the modelled interaction unit corresponds to an ordered index. This case is only possible for List AIUs; the manager class of a Single-object AIU only represent the retrieval of a single instance, when no ordering criterion is needed.
- If it is defined on a complementary class, the Ordering Pattern allows

ordering information of objects that complement the main information of the interaction unit. This case is possible for Single-object and List AIUs, but restricted to the retrieval of multiple objects.

Graphically, the *Ordering Pattern*'s expression is included in a rectangular label attached to the upper part of the corresponding navigational class. In Fig. A.2, instances shown through *New Releases* AIU are ordered by values of *title* attribute from its manager class, in ascending order.



Welcome to

smartreading.com

Residence on Earth
by Pablo Neruda

Price: **$10.47**
Add to Shopping Cart

Edition : Paperback
Pages : 359
Year : 2004
ISBN : 0811204677

Summary:
Poems written by the prolific Chilean poet between the 1920's and 1940's illuminate his views on alienation and political oppression.

Our Specialist Says:
Excellent collection of candid poems by *John Doe*

Neruda delivers an uninhibited glimse at human emotions and panges of life in this collection of poems. Neruda jars the soul by explaining in simple verse, how tragic life can be. The reflections and odes collected in this work are a superb display of the skill Neruda has. *Full Review*

Other Readers' Reviews: Add Review

Amazing, *August 24, 2002 Pablo Morales*
Well, I would imagine 5 stars, *August 8, 2002 Martin Hoeldtke*

Two for One Offer!
Residence on Earth & Captain's Verses **$19.23** Buy Both

New Releases

First American Army
by Bruce Chadwick

High in the Clouds
by Paul McCartney

Spook
by Mary Roach

The Best Recipes in the World
by Mark Bittman

The Sunflower
by Richard Paul Evans

Figure A.3: Implementation of *Books* context

Figure A.3 shows an implementation of the *Books* Navigational Context. Frames *1* and *2* show the data modelled in both AIUs, respectively. The structure of *Book Details* AIU can be distinguished through the dashed frames. Frame *a* presents the main data of the book, corresponding to attributes and operations of the *Book* manager class; the rest of frames show the information retrieved from the complementary classes.

Bibliography

[1] De Bra, P., Eklund, J., Brusilovsky, P.: Adaptive Hypertext and Hypermedia. Web Site of Adaptive Hypermedia community, http://wwwis.win.tue.nl/ah/ Retrieved on Sep 13, 2007.

[2] Mendes, E., Mosley, N.: Web Engineering. Springer, Berlin, Germany (2005)

[3] Jacobs, I.: About the World Wide Web Consortium (W3C). W3C Web Site, http://www.w3.org/Consortium/Overview.html Retrieved on Sep 25, 2007.

[4] Netcraft Ltd.: September 2007 Web Server Survey. Netcraft Web Site Published on Sep 3, 2007.

[5] Nielsen, J.: 100 Million Websites. Jakob Nielsen's Alertbox - Web Site, http://www.useit.com/alertbox/web-growth.html Published on Nov 6, 2006.

[6] Nielsen, J.: Designing Web Usability: The Practice of Simplicity. New Riders Publishing, Thousand Oaks, CA, USA (1999)

[7] W3C Quality Assurance IG: About the Quality Assurance Interest Group at W3C. Official Web Site, URL: http://www.w3.org/QA/IG Retrieved on Oct 1, 2007.

[8] Brusilovsky, P.: Adaptive Hypermedia. User Modeling and User-Adapted Interaction 11 (1-2) (2001) 87–110

[9] De Bra, P., Brusilovsky, P., Houben, G.J.: Adaptive hypermedia: from systems to framework. ACM Comput. Surv. **31**(4) (1999) 12

[10] Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction **6**(2-3) (1996) 87–129

[11] Brusilovsky, P.: Adaptive Navigation Support: From Adaptive Hypermedia to the Adaptive Web and Beyond. PsychNology Journal **2**(1) (2004) 7–23

[12] De Bra, P.: Pros and Cons of Adaptive Hypermedia in Web-Based Education. CyberPsychology & Behavior **3**(1) (2000) 71–77

[13] Internet World Stats: Internet Usage Statistics - The Big Picture: World Internet Users and Population Stats. Official Web Site, URL: http://www.internetworldstats.com/stats.htm Retrieved on Oct 1, 2007.

[14] Ginige, A., Murugesan, S.: Guest Editors' Introduction: Web EngineeringAn Introduction. IEEE MultiMedia **08**(1) (2001) 14–18

[15] Brusilovsky, P., Maybury, M.T.: From adaptive hypermedia to the adaptive web. Commun. ACM **45**(5) (2002) 30–33

[16] Amazon.com, Inc.: Amazon.com: Recommendations. Official Web Site, URL: http://www.amazon.com/gp/help/customer/ display.html?nodeId=13316081 Retrieved on Oct 3, 2007.

[17] Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing **7**(1) (2003) 76–80

[18] Rossi, G., Pastor, O., Schwabe, D., Olsina, L.: Web Engineering: Modelling and Implementing Web Applications. Springer (2007)

[19] Fons, J., Pelechano, V., Albert, M., Pastor, O.: Development of Web Applications from Web Enhanced Conceptual Schemas. In: Proceedings of ER 2003. LNCS 2813, Springer (2003) 232–245

[20] Pastor, O., Gómez, J., Insfrán, E., Pelechano, V.: The OO-Method Approach for Information Systems Modeling: from Object-Oriented Conceptual Modeling to Automated Programming. Information Systems 26 (7) (2001) 507–534

[21] Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)

[22] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-oriented Modeling and Design. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1991)

[23] Rich, E.: User Modeling via Stereotypes. International Journal of Cognitive Science **3** (1979) 329–354

[24] Valderas, P., Fons, J., Pelechano, V.: Transforming Web Requirements into Navigational Models: A MDA Based Approach. In: Proceedings of ER 2005. LNCS 3716, Springer (2005) 320–336

[25] Ginige, A., Lowe, D.B., Robertson, J.: Hypermedia Authoring. IEEE MultiMedia **02**(4) (1995) 24–35

[26] Toffler, A.: Future Shock. Random House, USA (1970)

[27] Edwards, D.M., Hardman, L.: Lost in hyperspace: cognitive mapping and navigation in a hypertext environment. In: Hypertext: theory into practice. Intellect Books, Exeter, UK (1999) 90–105

[28] Lieberman, H., Selker, T.: Out of context: Computer systems that adapt to, and learn from, context. IBM Systems Journal **39**(3&4) (2000) 617–632

[29] User Modeling Inc.: User Modeling Inc. (UM Inc.) Home Page. Official Web Site, http://www.um.org/ Retrieved on Aug 21, 2007.

[30] Koch, N., Wirsing, M.: The Munich Reference Model for Adaptive Hypermedia Applications. In: Proceedings of AH 2002. LNCS 2347, Springer (2002) 213–222

[31] Schilit, B., Theimer, M.: Disseminating active map information to mobile hosts. IEEE Network **8**(5) (1994) 22–32

[32] Brown, P.: The stick-e document: a framework for creating context-aware applications. Electronic Publishing **8**(2-3) (1995) 259–272

[33] Dey, A.K.: Context-Aware Computing: The CyberDesk Project. In: AAAI '98 Spring Symposium Stanford University. (1998)

[34] Ward, A., Jones, A., Hopper, A.: A New Location Technique for the Active Office. IEEE Personnel Communications **4**(5) (October 1997) 42–47

[35] Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing **5**(1) (2001) 4–7

[36] ACM's Special Interest Group on Computer-Human Interaction: ACM/SIGCHI. Official Web Site, http://sigchi.org/ Retrieved on Sep 17, 2007.

[37] Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. In: Proceedings of Workshop on The What, Who, Where, When, and How of Context-Awareness, The Hague, The Netherlands (april 2000)

[38] Jameson, A.: Adaptive Interfaces and Agents. In: The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. 2 edn. Erlbaum (2006) 305–330

[39] Koch, N.: Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process. PhD thesis, Fakultät für Mathematik und Informatik, Ludwig-Maximilians-Universitä München (December 2000)

[40] Berlanga, A.J., García, F.J.: Using IMS LD for Characterizing Techniques and Rules in Adaptive Educational Hypermedia Systems. In: UNFOLD/Prolearn joint workshop Current State on IMS Learning Design. (September 2005) 61–79

[41] Schwinger, W., Koch, N.: Modeling Web Applications. In: Web Engineering: The Discipline of Systematic Development of Web Applications. Wiley (July 2006) 39–64

[42] Schwinger, W.: A Survey on Web Modelling Approaches for Ubiquitous Web Applications. International Journal of Web Information Systems (1) (2008)

[43] Rojas, G., Valderas, P., Pelechano, V.: Describing Adaptive Navigation Requirements of Web Applications. In Wade, V.P., Ashman, H., Smyth, B., eds.: Adaptive Hypermedia and Adaptive Web-Based Systems, 4th International Conference, AH 2006, Dublin, Ireland, June 21-23, 2006, Proceedings. Volume 4018 of Lecture Notes in Computer Science., Springer (2006) 318–322

[44] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2002)

[45] Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. IEEE Internet Computing **6**(4) (2002) 20–30

[46] Chen, P.P.S.: The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems **1**(1) (1976) 9–36

[47] Ceri, S., Fraternali, P., Paraboschi, S.: Data-Driven, One-To-One Web Site Generation for Data-Intensive Applications. In: Proceedings of VLDB'99, Morgan Kaufmann (1999) 615–626

[48] Ceri, S., Daniel, F., Matera, M., Facca, F.M.: Model-driven Development of Context-Aware Web Applications. In: ACM Transactions on Internet Technology (ACM TOIT), volume 7, number 2. (2007)

[49] Ceri, S., Daniel, F., Demaldé, V., Facca, F.M.: An Approach to User-Behavior-Aware Web Applications. In: Proceedings of ICWE 2005. LNCS 3579, Springer (2005) 417–428

[50] Dayal, U.: Active Database Management Systems. In: Third International Conference on Data and Knowledge Bases: Improving Usability and Responsiveness, Jerusalem, Israel, Morgan Kaufmann (1988) 150–169

[51] Schwabe, D., Rossi, G.: An object oriented approach to web-based applications design. Theory and Practice of Object Systems **4**(4) (1998) 207–225

[52] Rossi, G., Schwabe, D., Guimarães, R.: Designing Personalized Web Applications. In: Proceedings of WWW 2001. (2001) 275–284

[53] Güell, N., Schwabe, D., Vilain, P.: Modeling interactions and navigation in web applications. In: ER '00: Proceedings of the Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling, London, UK, Springer-Verlag (2000) 115–127

[54] Schwabe, D., Guimarães, R., Rossi, G.: Cohesive design of personalized web applications. IEEE Internet Computing **6**(2) (2002) 34–43

[55] Koch, N.: Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process. PhD thesis, LMU Munich (2000)

[56] Object Management Group: UML 2.0 Object Constraint Language (OCL). www.uml.org (2003)

[57] Halasz, F., Schwartz, M.: The Dexter hypertext reference model. Communications of the ACM **37**(2) (1994) 30–39

[58] Koch, N., Wirsing, M.: Software Engineering for Adaptive Hypermedia Applications? In: Third Workshop on Adaptive Hypertext and Hypermedia, Sonthofen, Germany (July 2001)

[59] Troyer, O.D., Leune, C.J.: Wsdm: A user centered design method for web sites. Computer Networks **30**(1-7) (1998) 85–94

[60] De Troyer, O., Casteleyn, S., Plessers, P.: WSDM: Web Semantics Design Method. In: Web Engineering: Modelling and Implementing Web Applications. Volume 12. Springer (2007)

[61] Casteleyn, S.: Designer Specified Self Re-organizing Websites. PhD thesis, Departement Computerwetenschappen - Vrije Universiteit Brussel, Brussels, Belgium (2005)

[62] Gómez, J., Cachero, C., Pastor, O.: Conceptual modeling of device-independent web applications. IEEE MultiMedia **8**(2) (2001) 26–39

[63] Garrigós, I., Casteleyn, S., Gómez, J.: A structured approach to personalize websites using the oo-h personalization framework. In Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M., eds.: APWeb. Volume 3399 of Lecture Notes in Computer Science., Springer (2005) 695–706

[64] Garrigós, I., Gómez, J., Barna, P., Houben, G.J.: A Reusable Personalization Model in Web Application Design. In: WISM 2005, ICWE 2005 Workshop Proc. (2005)

[65] Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. J. Web Eng. **2**(1-2) (2003) 3–26

[66] Frasincar, F., Houben, G.J., Vdovjak, R.: Specification Framework for Engineering Adaptive Web Applications. In: Proceedings of WWW 2002, Web Engineering Track. (2002)

[67] Frasincar, F., Houben, G.J.: Hypermedia presentation adaptation on the semantic web. In De Bra, P., Brusilovsky, P., Conejo, R., eds.: AH. Volume 2347 of Lecture Notes in Computer Science., Springer (2002) 133–142

[68] Brickley, D., Guha, R.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C, http://www.w3.org/TR/rdf-schema/. (February 2004)

[69] Frasincar, F., Barna, P., Houben, G.J., Fiala, Z.: Adaptation and reuse in designing web information systems. In: ITCC (1), IEEE Computer Society (2004) 387–291

[70] Mens, T., Gorp, P.V.: A Taxonomy of Model Transformation. Electr. Notes Theor. Comput. Sci. **152** (2006) 125–142

[71] Insfrán, E.: A Requirements Engineering Approach for Object-Oriented Conceptual Modeling. PhD thesis, Department of Information Systems and Computation, Technical University of Valencia, Spain (2003)

[72] Díaz, I., Moreno, L., Fuentes, I., Pastor, O.: Integrating Natural Language Techniques in OO-Method. In Gelbukh, A.F., ed.: CICLing. Volume 3406 of Lecture Notes in Computer Science., Springer (2005) 560–571

[73] Estrada, H., Rebollar, A.M., Pastor, O., Mylopoulos, J.: An Empirical Evaluation of the i* Framework in a Model-Based Software Generation

Environment. In Dubois, E., Pohl, K., eds.: CAiSE. Volume 4001 of Lecture Notes in Computer Science., Springer (2006) 513–527

[74] Wieringa, R.J.: Requirements engineering: frameworks for understanding. John Wiley & Sons, Inc., New York, NY, USA (1996)

[75] Kobsa, A., Koenemann, J., Pohl, W.: Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships. The Knowledge Engineering Review **16**(2) (2001) 111–155

[76] Atkinson, C., Kühne, T.: Meta-Level Independent Modeling. In: International Workshop Model Engineering (in Conjunction with ECOOP'2000), Cannes, France (2000)

[77] Gonzalez-Perez, C., Henderson-Sellers, B.: Templates and Resources in Software Development Methodologies. Journal of Object Technology **4**(4) (2005) 173–190

[78] Lalmas, M.: ACM Special Interest Group on Information Retrieval, ACM SIGIR. Official Web Site Retrieved on Jul 14, 2007.

[79] Object Management Group: Business Process Modeling Notation (BPMN) Specification: OMG Final Adopted Specification. Object Management Group - Business Process Management Initiative, Web Site (January 2006)

[80] Escalona, M.J., Koch, N.: Requirements Engineering for Web Applications - A Comparative Study. Journal of Web Engineering **2**(3) (2004) 193–212

[81] Corporate Yourdon Inc.: Yourdon systems method: model-driven systems development. Yourdon Press, Upper Saddle River, NJ, USA (1993)

[82] Wieringa, R.: Postmodern Software Design with NYAM: Not Yet Another Method. In: RTSE '97: Proceedings of the International Workshop on Requirements Targeting Software and Systems Engineering, London, UK, Springer-Verlag (1998) 69–94

[83] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G.: Object Oriented Software Engineering: A Use Case Driven Approach. ACM Press. Addison-Wesley Professional, Reading, Massachusetts (1993)

[84] Constantine, L.L., Lockwood, L.A.: Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. ACM Press. Addison-Wesley Professional (1999)

[85] Jaaksi, A.: Our cases with use cases. Journal of Object-Oriented Programming **10**(9) (1998) 58–65

[86] Rosenberg, D., Scott, K.: Use Case Driven Object Modeling with UML: A Practical Approach. Addison-Wesley Object Technology Series. Addison-Wesley Professional (1999)

[87] Leite, J., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G., Oliveros, A.: Enhancing a requirements baseline with scenarios. In: RE '97: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97), Washington, DC, USA, IEEE Computer Society (1997) 44

[88] Jeenicke, M., Bleek, W.G., Klischewski, R.: Revealing Web User Requirements through e-Prototyping. In: SEKE. (2003) 9–16

[89] Lauesen, S.: Task Descriptions as Functional Requirements. IEEE Software **20**(2) (2003) 58–65

[90] Object Management Group: Activities. In: Unified Modeling Language: Superstructure Specification, v2.0. Object Management Group, Inc. (August 2005) 285–406

[91] Paternò, F., Mancini, C., Meniconi, S.: ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In: Proceedings of INTERACT '97. (1997) 362–369

[92] Wirfs-Brock, R., Wilkerson, B., Wiener, L.: Designing Object-Oriented Software. Prentice-Hall (1990)

[93] Escalona, M.J.: Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software. PhD thesis, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Spain (2004) in Spanish.

[94] Toro, A.D., Jiménez, B.B., Cortés, A.R., Bonilla, M.T.: A Requirements Elicitation Approach Based in Templates and Patterns. In: WER. (1999) 17–29

[95] Valderas, P., Fons, J., Pelechano, V.: Developing E-Commerce Applications from Task-Based Descriptions. In: Proceedings of EC-Web 2005. LNCS 3590, Springer (2005) 65–75

[96] Valderas, P., Fons, J., Pelechano, V.: Modelling Content Aggregation for Developing e-Commerce Web Sites. In: Proceedings of EC-Web 2004. LNCS 3182, Springer (2004) 259–267

[97] Pastor, O., Pelechano, V., Fons, J., Abrahão, S.: Conceptual Modelling of Web Applications: the OOWS approach. In: Web Engineering - Theory and Practice of Metrics and Measurement for Web Development. Springer (2005) 277–301

[98] The PHP Group: PHP Hypertext Processor. Official Web Site, http://www.php.net/ Retrieved on Dec 14, 2007.

[99] Sun Microsystems, Inc.: JavaServer Pages Technology. Official Web Site, http://java.sun.com/products/jsp/ Retrieved on Dec 14, 2007.

[100] Kristol, D.M.: HTTP Cookies: Standards, privacy, and politics. ACM Transactions on Internet Technology (TOIT) **1**(2) (2001) 151–198

[101] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Stewart, L.: HTTP Authentication: Basic and Digest Access Authentication (1999)

[102] Read, D., Kingsley-Hughes, K., Wilton, P., Updegrave, S., Nelson, E., Maharry, D.: VBScript Programmer's Reference. Wrox Press Ltd., Birmingham, UK, UK (1999)

[103] Wium Lie, H., Bos, B.: Cascading style sheets: designing for the Web. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1997)

[104] Date, C.J.: A guide to the SQL standard: a user's guide to the standard relational language SQL. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1987)