

**A SYSTEMATIC TRANSFORMATION APPROACH FROM REQUIREMENTS TO
DESIGN ENGINEERING FOR WEB APPLICATIONS**

By

SANGEETA SRIVASTAVA

COLLEGE OF ENGINEERING (DEPARTMENT OF COMPUTER SCIENCE)

Submitted



**IN PARTIAL FULFILLMENT OF THE REQUIREMENT OF THE DEGREE OF
DOCTORATE OF SCIENCE**

TO

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

DEHRADUN

June, 2016

Under the Guidance of

Dr. Vandana Gupta

Associate Professor

**Department of Computer Science,
Kalindi College, Delhi University**

Acknowledgement

“Sometimes our light goes out but is blown into flame by another human being. Each of us owes deepest thanks to those who have rekindled this light.”

Albert Schweitzer (1875-1965)

This Higher Doctoral thesis is the culmination of a long interest in design engineering and hard work of numerous people over the years. There are lots of people that I need to pay my gratitude and thanks to for helping me in my research work. I owe a debt of profound gratitude to my supervisor, Dr. Vandana Gupta, for her continuous guidance, constructive criticism and supervision which were an instrument of inspiration for me to complete this work. I am extremely grateful to her. She generously allowed me to work under her at the cost of her precious time without which this work would not have been possible.

I record my heartfelt gratitude towards, Wing Commander PK Gupta, Associate Director & Head - Centre for Continuing Education (CCE), University of Petroleum & Energy Studies for his valuable feedback and great support throughout my research.

I further record my heartfelt gratitude and special thanks to Dr. Anjali Midha Sharan, Program Director, University of Petroleum & Energy Studies and Ms. Rakhi Ruhel, Program Manager-PhD, University of Petroleum & Energy Studies for their advice and great support throughout my research work.

My special thanks go to my colleagues who have been one of the greatest resources to rely on. I would like to thank their participation in countless, interesting and clarifying discussions during this research. I shall truly be failing in my gratitude if I do not remember the invaluable support extended in one form or the other by the non-teaching staff of Department of Computer Science. Apart from this, the institution provided extensive computer facilities that made the progress of the work very smooth and convenient.

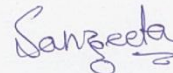
I am very grateful to my husband, Mr. Rajiv Saxena, who inspired me at the most difficult hours as a consequence of which this thesis became possible. He always supported me throughout my studies and was always confident of me. My greatest debt is towards my children Ritwik, Ritika and Rohan who cooperated with me in this endeavor and spared me all the time that actually belonged to them. I also take this opportunity to thank my family members. They all have expressed words of wisdom and great inspiration. Their care and love keeps me going during hard times.

Last but not least, I thank Almighty for giving me an opportunity to complete my research work.

Sangeeta Srivastava

DECLARATION

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text".



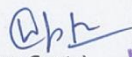
(Dr. Sangeeta Srivastava)

18.06.2016

THESIS COMPLETION CERTIFICATE

This is to certify that the thesis on "**A Systematic Transformation Approach from Requirements to Design Engineering for Web Applications**" by **Dr. Sangeeta Srivastava** in Partial completion of the requirements for the award of the Degree of Doctorate of Sciences (Computer Science) is an original work carried out by her under my supervision and guidance.

It is certified that the work has not been submitted anywhere else for the award of any other diploma or degree of this or any other University.

Mentor 
(**Dr. Vandana Gupta**)
Associate Professor,
Department of Computer Science,
Kalindi College, Delhi University

KALINDI COLLEGE
East Patel Nagar
New Delhi-110008

Place: **New Delhi**
Date: **20/6/16**

Table of Contents

Acknowledgement.....	i
List of Figures.....	xi
List of Tables.....	xiv
List of Abbreviations.....	xv
Executive Summary.....	xvii
Chapter 1.....	1
1.1 Web Applications.....	1
1.2 Web Application Development.....	2
1.3 Need for a Good Software Design Model.....	4
1.4 UML (Unified Modeling Language).....	10
1.5 Research Contributions.....	14
1.6 Organization of Thesis.....	22
Chapter 2.....	25
2.1 Background.....	25
2.2 Web Based Engineering.....	26
2.3 Model based Software Development.....	27
2.3.1. Model based "Web engineering methods".....	27
2.3.2. The Schwabe & Rossi's Hypermedia Design Model.....	29
2.3.3. The De Troyer & Leune's "Web Site Design Method".....	30
2.3.4. The Ceri et al's Web Modeling Language.....	33
2.3.5. The Gomez et al's Object Oriented Hypermedia.....	33
2.3.6. The Koch et al's UML-based Web Engineering.....	34
2.3.7. SHDM.....	36

2.3.8.	NDT (Navigational Development Techniques).....	37
2.3.9.	Analysis of the MDWE Methods	37
2.4	Model Based Web Engineering Approaches.....	39
2.4.1.	The Web Software Architecture (WebSA) Approach.....	40
2.4.2.	The Midas' Hypertext Modeling Method.....	41
2.4.3.	The Pastor et al's Object-Oriented Web Solutions Approach.....	42
2.4.4.	UML-Based Web Engineering -UWE Approach.....	43
2.4.5.	Ceri et al's Web Modeling Language and WebRatio	44
2.4.6.	Visser & Groenewegen et al's Domain Specific Language for web applications ..	46
2.4.7.	DSL for generating Web application (MarTE).....	47
2.4.8.	HERA	48
2.4.9.	Garrigos' Adaptive Object Oriented Hypermedia (A-OOH) approach.....	50
2.5	Analysis of Existing approaches wrt Web Engineering.....	52
2.6	Other Approaches	56
Chapter 3.....		59
3.1	Introduction	59
3.2	The WebGRL Metamodel.....	61
3.2.1.	GORE in Web Application Development.....	61
3.2.2.	URN.....	63
3.2.3.	Web Specific User Requirements Notation	64
3.2.4.	WebGRL Notation.....	65
3.2.5.	WebUCM Notation	68
3.3	The Object Oriented Hypermedia (OO-H) Method.....	71
3.4	Design Process.....	74

3.4.1.	Pattern Catalog.....	75
3.4.2.	Navigational Access Diagram	77
3.4.3.	Abstract Presentation Diagram	79
3.5	The Overview of AOO-H Method.....	82
3.5.1.	Requirements	85
3.5.2.	Domain Analysis and Design	85
3.5.3.	Navigation Design	86
3.5.3.1.	Navigation Access Diagram.....	86
3.5.3.2.	The NAD Metamodel	91
3.6	Presentation Design	94
3.6.1.	Design Presentation Diagram	94
3.6.2.	DPD Metamodel	98
Chapter 4.....		103
4.1	INTRODUCTION	103
4.2	The EAOO-H model.....	106
4.2.1.	Discussion on the Extension of the A-OOH approach to EAOO-H approach	108
4.3	The EAOO-H "Domain model (DM)"	109
4.3.1.	The EAOO-H Domain Model Concepts	109
4.3.2.	Deriving the EAOO-H Domain Model	115
4.3.3.	The Transformation Rules to derive the EAOO-H Domain Model from Content WebGRL Diagram.....	115
4.3.4.	The Transformation Method for transformation from Content WebGRL diagram to EAOO-H Domain Model	117
4.3.5.	The Transformation Algorithm of Content WebGRL diagram to EAOO-H Domain Model	118

4.3.6.	# EAOO-H Domain Model Transformation Algorithm.....	119
4.4	Example of Transformation from WebGRL Content diagram to EAOO-H Domain Model for an Online Book Store	120
4.4.1.	Case Study Of Online Bookstore -Requirements Specification	121
4.4.2.	The Content WebGRL Diagram.....	122
4.4.3.	The Method for deriving the EAOO-H Domain Model.....	123
Chapter 5.....		126
5.1	Introduction	126
5.2	The Enhanced A-OOH (EAOO-H) Navigation Model.....	128
5.2.1.	The Navigation Metamodel	130
5.2.2.	Navigation Access Diagram (NAD).....	132
5.2.2.1.	The Navigation Node.....	133
5.2.2.2.	Navigational Links (NL)	137
5.3	Deriving the EAOO-H Navigation Model	139
5.3.1.	The Transformation Rules to derive the EAOO-H Navigation Model from Navigation WebGRL Diagram.	140
5.3.2.	The Transformational Method for transformation from the Navigation WebGRL Diagram to the Navigation EAOO-H Model	141
5.3.3.	The Transformation Algorithm of Navigation WebGRL diagram to EAOO-H Navigation Model	143
5.3.4.	# EAOO-H Navigation Model Transformation Algorithm.....	144
5.4	Example of Transformation from WebGRL Content diagram to EAOO-H Navigation Model for an Online Book Store.....	146
5.4.1.	Case Study Of Online Bookstore -Requirements Specification	146
5.4.2.	The Navigation WebGRL Diagram for Online Bookstore	148

5.4.3.	The Method for deriving the EAOO-H Navigation Model m	148
Chapter 6.....		151
6.1	Introduction	151
6.2	The EAOO-H Presentation Model.....	154
6.3	The Presentation Design Model.....	156
6.4	The EDPD Metamodel.....	156
6.4.1.	The Enhanced Design Presentation Diagram	160
6.4.2.	Main Elements in the Level Zero of the EDPD.....	161
6.4.3.	Main Elements in the Level One of the EDPD.....	164
6.5	Deriving the EAOO-H Presentation Model.....	167
6.5.1.	The Transformation Rules for deriving the EAOO-H Presentation Model from the Presentation WebGRL Model.....	168
6.5.2.	The Transformation Method for deriving the EAOO-H Presentation Model from the Presentation WebGRL Model.....	169
6.5.3.	The Transformation Algorithm of deriving the Presentation EAOO-H model from Presentation WebGRL Diagram.....	172
6.5.4.	# EAOO-H Presentation Model Transformation Algorithm	173
6.6	Example of Online Bookstore	174
Chapter 7.....		179
7.1	Introduction	179
7.2	UML Profile for the EAOO-H Design Models	181
7.2.1.	Stereotypes	182
7.2.2.	Tagged Values	183
7.2.3.	Constraints	184
7.2.4.	Package.....	185

7.2.5.	Association.....	185
7.2.6.	Class	185
7.2.7.	Property	186
7.2.8.	Dependencies	186
7.3	UML Profile for the EA00-H Domain Model	187
7.4	UML Profile for the EA00-H Navigational model	191
7.5	UML Profile for the EA00-H Presentation model	198
Chapter 8.....		204
8.1	Introduction	204
8.2	Web Application Design Tool.....	205
8.2.1.	WAD Tool Technical details	205
8.2.2.	Features of WAD Tool	206
8.2.3.	Design of the WAD tool	209
8.2.4.	Working of WAD Tool	211
8.3	Early Requirements Analysis of Web Based Education System	215
8.4	Conclusion.....	218
Chapter 9.....		220
9.1.	Main Contributions of the thesis.....	220
9.2	Future Directions	226
Bibliography		228
CURRICULAM VITAE & LIST OF PUBLICATIONS		239

List of Figures

Figure 1.1 Block Diagram of EA00-H Design Models Framework.....	22
Figure 2.1 General scheme of MDWE Methods	28
Figure 3.1 Enhancement of URN Metamodel	65
Figure 3.2 WebGRL metamodel.....	67
Figure 3.3 WebUCM metamodel	69
Figure 3.4 The OO-H Overview	73
Figure 3.5 OOH Method - The design process	75
Figure 3.6 Sample NAD Diagram for Participation in Paperlist	79
Figure 3.7 Sample APD for Paper list	81
Figure 3.8 Navigation Class.....	87
Figure 3.9 Navigational Target	87
Figure 3.10 C-Collection.....	88
Figure 3.11 Transversal Link	89
Figure 3.12 Service Link	89
Figure 3.13 Index Design Pattern	90
Figure 3.14 NAD Metamodel	91
Figure 3.15 Navigational Link metaclass.....	91
Figure 3.16 Filter defined in a Navigational Link	92
Figure 3.17 Automatic and Manual Links	93
Figure 3.18 Navigational Node metaclass.....	94
Figure 3.19 Presentation Page.....	95
Figure 3.20 Page Chunk	96
Figure 3.21 Window, FrameSet and Frame elements	96
Figure 3.22 BoxLayout.....	97
Figure 3.23 BorderLayout	97
Figure 3.24 GridBagLayout	98

Figure 3.25 Presentation Node Subtypes	99
Figure 3.26 Presentation Link types	100
Figure 3.27 Layout and IncludeCell metaclasses	100
Figure 4.1 Showing Content Goals, tasks and Resources	108
Figure 4.2 Domain Class	110
Figure 4.3 Example of the Author Domain Class	111
Figure 4.4 Example of relationships between domain classes	112
Figure 4.5 Constraint on Attribute	114
Figure 4.6 Constraint on association between classes	115
Figure 4.7 The BaseWebGRL diagram for Online Bookstore	122
Figure 4.8 Content WebGRL diagram	123
Figure 4.9 The Domain Model for Online Bookstore	124
Figure 5.1 The Navigation Metamodel	131
Figure 5.2 Navigation Class	133
Figure 5.3 Index Class and shorthand for index	134
Figure 5.4 Guided Tour class and shorthand for guided tour	135
Figure 5.5 Query Class and Shorthand for Query	135
Figure 5.6 Showall Class and shorthand for showall	136
Figure 5.7 Menu class and shorthand for Menu	137
Figure 5.8 Transversal Link	138
Figure 5.9 Service Link	139
Figure 5.10 The BaseWebGRL diagram for Online Bookstore	147
Figure 5.11 Navigation WebGRL diagram	148
Figure 5.12 The Navigation Model	149
Figure. 6.1 DPD Main Elements	157
Figure 6.2 Presentation Node Subtypes	158
Figure 6.3 Presentation Link types	160
Figure 6.4 Presentation Page	161
Figure 6.5 Main Elements in the Level 0 of the EDPD	162

Figure 6.6Page Chunk	163
Figure 6.7Window, FrameSet and Frame elements	163
Figure 6.8Presentation Class.....	164
Figure 6.9Presentation Class and the Interface Components.....	165
Figure 6.10BoxLayout.....	166
Figure 6.11BorderLayout.....	166
Figure 6.12GridBagLayout	167
Figure 6.13The Online BookStore Presentation WebGRL diagram	174
Figure 6.14BookStore Home Presentation Page.....	175
Figure 6.15 Level 1 of the DPD for The BookStore Home Presentation Page.	176
Figure 6.16Expanded Level 1 of the DPD for The BookStore Home Presentation Page.....	177
Figure 7.1UML Profile for the EAOO-H Design Models	182
Figure 7.2UML Profile for Domain Model.....	189
Figure 7.3The Navigation Metamodel	195
Figure 7.4 UML profile for NAD	196
Figure 7.5UML Profile for EDPD	200
Figure 8.1 WAD Tool.....	212
Figure 8.2Input Information for the Content Model of WAD Tool	213
Figure 8.3Input Information for the Navigation Model of WAD Tool	214
Figure 8.4Base WebGRL diagram for Web based Education System.....	216
Figure 8.5 Content WebGRL diagram for Web based Education System.....	217
Figure 8.6Output Content Design Model diagram for web based education system.....	217

List of Tables

Table 4.1 Transformation of Content web GRL diagram to Domain model	116
Table 5.1 The Transformation of Navigation WebGRL diagram to NM	142
Table 6.1 Transformation from Presentation WebGRL Diagram to Presentation Model or the DPD.	170
Table 7.1 UML METACLASS for WebGRL Content Model and Domain Model	190
Table 7.2 UML Metaclass for Navigation WebGRL and Navigation Model.....	197
Table 7.3 UML PROFILE FOR Presentation WebGRL and Presentation Model	202

List of Abbreviations

A-OOH	Adaptive Object Oriented Hypermedia
DM	Domain Model
DPD	Design Presentation Diagram
EAOO-H	Enhanced Adaptive Object Oriented Hypermedia
FR	Functional Requirements
FREQ	Forward Reasoning and Evaluation- Qualitative
FREN	Forward Reasoning and Evaluation- Numeric
GORE	Goal Oriented Requirements Engineering
GRL	Goal oriented Requirements Language
HTML	Hyper Text Markup Language
ISD	Information Systems Development
NAD	Navigation Access Diagram
NDT	Navigational Development Technique
NFR	Non-functional Requirements
NM	Navigation Model
OOH	Object Oriented Hypermedia
OOHDM	Object Oriented Hypermedia Design Model
PM	Presentation Model
RE	Requirements Engineering
SDLC	Software Development Life Cycle
UCM	Use Case Maps
UML	Unified Modeling Language
URN	User Requirements Notation
UWE	UML Based Web Engineering
WebGRE	Web specific Goal driven Requirements Engineering
WebGRL	Web specific Goal oriented Requirements Language
WebML	Web Modeling Language

WebUCM Web specific Use Case Maps
WebURN Web specific User Requirements Notation

Executive Summary

In the recent past web engineering has emerged as a very important discipline that focuses on the future of web application development. The web applications range from being content oriented systems to business process and workflow based systems. As more and more web applications are being developed and the fact that they are being used in a multitude of fields which span our everyday lives their complexity is increasing. Web applications are different from traditional information systems in many ways. The varied usability and user requirements of web applications make it imperative to engineer them systematically. Therefore, a systematic method of web application development is required to convert these complex requirements of web applications into simple front ends for a naive user. Uptill now ad-hoc methods of web application development were being followed resulting in a product that requires repeated iterations to meet the stakeholders' goals. As such we need to incorporate the structured method of web application development as in the case of conventional software developments where the five distinct stages of software development namely the requirements, design, construction, implementation and testing are followed.

A number of requirements approaches for web application development exist, however their focus is mostly on the functional requirements the non-functional requirements are completely overlooked in these approaches leading to design conflicts. Keeping this in mind we need a web oriented requirements approach that captures the requirements comprehensively. We use the WebGRL approach based on Goal Oriented Requirements Engineering of (Chawla et al 2014) as it captures both the functional and non-functional requirements of the stakeholders comprehensively using a goal oriented approach specifically tuned for web application development. This approach is an enhancement of the existing GRL approach such that it has the ability to capture the different facets of web application requirements. Web applications cover the four different features of web based applications, namely the Conceptual Design,

the Navigation Design, the Presentation Design and Implementation. The WebGRL approach of Chawla et al helps in capturing all these four features of a web application in the requirements phase itself. Once the requirements have been captured in totality we now proceed to the next stage of SDLC i.e. the Design stage.

The design helps in translating “requirements specifications” into a special working system. Software design is a procedure and model. On account of web applications improvement designing frequently requires considering the stylish, useful, financial and sociopolitical dimensions of both the design object and design process. Beginning stage is elaborating model of application area, which decides the look of web application. This is done during concept designing stage. A crucial recognizing aspect of Web application is the concept of navigation in which application user navigates in a space. The Presentation Design phase describes the layout of the various objects.

The software design process follows the requirements specification process and acts as a link between the construction and the requirements phase of a SDLC. This incorporates the algorithm design, low-level modules and the high-level, architectural design. As the requirements for the web application development are available with us we now present a strategy for the transformation of the WebGRL approach into the design models for the design stage which are UML compliant. UML is a graphic notation for expressing object oriented design. This presents the ease of conversion of a platform independent UML compliant models into Platform Specific Models based on object oriented approaches.

The problems identified in web application development are:

- Lack of use of a methodical approach that follows a software development process model in the case of web engineering.
- Lack of early requirements analysis for web application development.
- Lack of choice of appropriate design model that captures the web requirements both functional and nonfunctional comprehensively to produce a high quality software design.

- Need for a transformation approach that captures both the "functional and non-functional requirements" comprehensively for "requirements engineering" phase and transform it losslessly and seamlessly into the design phase.
- Use of design models that follow a model driven architecture and provide a structured way for conversion of the design models into a platform independent model which can use a model transformation language to convert it into a platform specific model thus providing the code reusability.

The problem faced by web development engineers is that existing web engineering approaches lack a methodical web application development approach that follows a software development process model with detailed requirements elicitation and designing which leads to a low quality and high cost of the product under development due to multiple iterations and unsatisfactory performance of the product.

The issues faced in web application development can be solved if proper procedure of both requirements and design engineering is followed where both functional and non-functional requirement are taken care of in the requirements stage and a watertight transformation of these requirements is done into the design stage where the design models provide a structured means of further conversion of the design models into a platform independent models.

The Information Systems development stages also need to be tuned to meet the requirements of web application development. Just like the WebGRL approach has been defined for capturing requirements of web applications, similarly, we need to define a design model that is suited for web application development. For this we either start from scratch and define a new design model or use an existing design model by modifying it to suit the needs of web application design development. We choose the A-OOH design model of (Garrigos 2009) as it has the capability to capture different perspectives of the web application development and enhance it to the EA00-H Design approach to ensure that all the facets of the requirements of web application are easily convertible into the design models for web applications. Next, we need to define the transformation strategy for transforming the requirements from the requirements phase to the design phase automatically that results in a high quality low cost

better standard of designs of web applications. This thesis accomplishes the design engineering of web applications using a step by step approach as explained below:-

- Enhancement of the AOO-H model into the EAOO-H Design Model to capture the GORE based WebGRL specific diagrams of requirements engineering
- Transformation of content WebGRL diagram to EAOO-H domain model.
- Transformation of navigation WebGRL diagram to EAOO-H Navigation model
- Transformation of presentation WebGRL diagram to EAOO-H presentation model
- Further a UML Profile for the three EAOO-H Design models has defined to enable the transformation of the design phase into the coding phase easily.
- Web Application Design tool has been developed to support the transformation process from web requirements engineering into web design engineering.

In our research work we present a systematic approach for web application development as in the case of traditional Information systems. Different phases of Information Systems development of proper requirements engineering and software design phases suitable for web application development are followed. Similarly, a suitable design model for web application development is chosen that has the capability to capture different perspectives of the web application development. This design model has been enhanced to define a new EAOO-H Design model which is more comprehensive and suitable for web application design development for the WebGRL approach. A transformation strategy is defined for transforming the different types of requirements of web applications namely the content, navigation and presentation requirements for web applications from the requirements phase to their respective design phases. This transformation strategy ensures a seamless conversion of the requirements into the design phase without any loss of information and traceability resulting in a high quality, low cost and better standard of designs of web applications.

This thesis has presented several contributions that help in losslessly converting the requirements captured in totality for web applications into a holistic design model suited for web applications. The enhancement of the existing A-OOH design model to the EAOO-H design model has been done to capture the web specific requirements completely. Further,

transformation strategy for flawless conversion of the web specific requirements to model the WebGRL content, navigation and presentation model from the requirements to the design phase has been covered in the research. As a result the web design engineer gets complete guidance and support for modeling the requirements into the design phase and the output thus produced is a platform independent model that can be easily converted into a platform specific model in the coding phase. The enhanced web design model and the requirements to design phase transformation method has attempted to address most of the problems faced in web application development as discussed in the beginning of this thesis. The proposed solution supports the design engineering of web application right from the requirements engineering phase to the coding phase. To enable real-world usage of the transformation process, a usable and extensible tool has been developed that supports the conversion of the requirements engineering models into their respective design models.

Chapter 1

Introduction

Web applications have permeated extensively into our everyday lives and has become intrinsic to information systems development. There is a lot of emphasis on the development of information systems that are web oriented. As a result the web application development has become the focus area of research for the past few years due to their increased importance and complexity. However, there still exist a lot of problems and the process has not evolved as in the case of traditional information systems development. This thesis illuminates the problems faced in web application development like the absence of systematic process of development of web applications and identifies the key area of web application development which needs to be improved and automates the entire process enabling it to capture a web application comprehensively by focusing on the requirements analysis and design phase and contributes by developing a web specific transformation of WebGRL models to EAOO-H Design models approach that is more thorough, systematic, complete and consistent. This results in producing a better quality web applications that follows a systematic automated approach at a much lower cost with reduced iterations, thereby improving the web application being developed.

1.1 Web Applications

A "web application is a program which is used by web browser and browser-supported programming language and depends on web browser to render application (as per Wikipedia, Accessed Jan-2014)". Web applications is a vital area of our lives in today's times.

We reach out to internet for everything starting from seeking information, business, online shopping or social networking. The web applications have transformed being just information source to interactive and extremely significant applications. Even the enterprise and business applications that specifically need robustness, performance and scalability are being built as web applications nowadays because of their reachability and convenience. Web applications are gaining popularity because they are easy to renew and maintain without disturbing or mounting the software on computers of customers or clients. They have cross platform compatibility that increases the reachability and flexibility to use these applications. With the increased reliance on web applications, it is imperative to study the existing web application development methods that are used to create these systems.

Web applications are unlike the conventional information systems in many ways. They differ in the information that they have to offer (content), the method of access of information, transition from one information to another (navigation), handling various transactions (business process) and adaptability to the user's profile or geographic location. Web applications continue to be content driven, focused on visual creativity and incorporate multimedia for an appealing look and feel. The audience that web application caters to is heterogeneous and unbound; the end users are at different locations, different age groups and have different goals and preferences. The varied usability and user requirements of web based applications make it imperative to engineer them systematically.

The qualitative expectations and constraints specific to web applications are different from information systems; hence they need to be explored further in the area of development of web applications. Therefore, web application development is an interesting area of research with unique requirements and development methods.

1.2 Web Application Development

The web applications range from being content oriented systems to business process and workflow based systems. The increasing demand of web based applications and the need of

diverse kinds of web applications have been gaining attention. The development of web applications has undergone an exponential growth in the last decade. It is observed that, web application development is done by a wide variety of developers like amateurs with no sound knowledge of software development, graphic designers, writers, freelancers and IT professionals. This is because a wide variety of web publishing tools and applications are available that enable creation of web application without prior knowledge of programming (Creswell, 1994). The traditional software engineering practices like prototyping are often used for developing web applications. In practice, mainly impromptu methods are still being used in designing web applications rather than engineering them. The focus of all these methods is mainly on website design aspects and quick delivery of the application but the quality and the stakeholders' requirements are overlooked. As large number of existing web based applications have been built in an extemporized way, it has led to issues of maintainability, quality and reliability. Similar problems were faced in traditional information systems development and this resulted in the emergence of software engineering so that the information systems could be developed in a structured and a systematic way.

Similarly, there has been some progress towards re-framing the creation of both design and installation of web applications as a controlled and structured endeavor. Web engineering discipline has emerged that focuses on web application development. If we can ensure a systematic process like Software Development Life Cycle (SDLC) of web application development as in the case of traditional information systems this would lead us to a better design, better quality of the software product and lower budget costs and time.

"Requirement Engineering (RE)" is the first most important activity of the SDLC for any information systems development, which includes understanding problem area, solution determination, and specification of an answer that is testable, justifiable, viable, and that fulfills project quantity guidelines (ISO: 9001, 1989). This stage is utilized to decipher the ambiguous, partial and flawed needs and wishes of the potential users of software into complete, exact and formal specifications. Therefore a detailed requirements analysis which takes care of both "functional and non-functional requirements" has been put forth by the

research community as it results in lesser number of iterations and leads to a better quality product.

Likewise, we need to carry out the other phases of the SDLC approach systematically to develop a high quality web application. Skipping any of the phases and using an ad-hoc approach might cost less time and money. However, the quality of the output and the satisfaction level of the user would be low. Therefore we need a methodical approach that captures the requirements in totality as stated above and further translates it flawlessly into the design phase and later on into the coding phase. As we have already emphasized above on the need for an exhaustive approach for capturing the requirements phase now we insist on using a design phase that holistically captures the requirements of the stake holders without losses. In order to do so we need a design approach that is tuned to capture these requirements for web applications and transforms them completely into the design phase without loss of information captured in the earlier phase. Therefore, firstly we need a good design model that losslessly captures the requirements specified in requirement engineering phase and the requirement model into specified design model without any loss of information captured in requirement phase. Thirdly output of the design phase is structured in order to ensure easy transformation from the design phase into the coding phase thereby providing a complete solution to web application development that is both systematic and comprehensive.

1.3 Need for a Good Software Design Model

The design of a system is essentially a blueprint or a plan for a solution for the system (Jalote 2005). The software design process follows the requirements specification process and acts as a link between the construction and the requirements phase of a SDLC. In the software design phase the emphasis is on capturing the requirements in totality without any losses and producing the software design output which is easily adaptable into a programming language by the programmer. This incorporates the algorithm design, low-level modules and the high-level, architectural design.

Therefore software design is a very important phase of the SDLC and acts as a bridge between the requirements and the construction phase while providing traceability and validation to the requirements of the SRS.

Software design is defined by "(Freeman et al 2004)as the procedure by which an operator makes a specification of a software artifact, planned to achieve objectives, utilizing an arrangement of primitive components and subject to constraints". As per "(Ralph and Wand 2009)Software design may refer to all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as a stylized software engineering process."

"One of the essential parts of software design is theSoftware Requirements Analysis (SRA). It is a part of the software development process which is used in software engineering". The design helps in translating "requirements specifications" into a special working system. So a "semi-automated" or user centered software, uses client familiarity of design to determine the specifications and in case of completely automated software the design maybe a flow chart or text describing for determining these specifications. "So there exist semi-standard methods like Unified Modeling Language and Fundamental modeling concepts where some documentation of the plan is a product of the software design which would depend on the technology used for the design.So Software design output might be platform-independent or platform-specific depending on the technology used".

"There are different kinds of software designs, the IEEE Std. 610.12-1990 Standard Glossary of Software Engineering Terminology defines the following as follows :

a)Architectural Design:It is a collection of hardware and software components and their interfaces. It identifies the structure for developing the information system.

b)Detailed Design:The Process of refining and expanding the preliminary design of a system or component to the extent that the design is sufficiently complete to begin implementation.

c) Functional Design: the process of defining the working relationships among the components of a system.

d) Preliminary Design: the process of analyzing design alternatives and defining the architecture, components, interfaces, and timing/sizing estimates for a system or components. So Software design is not just the higher level architectural view, but also low-level component and implementation issues".

Software design is a procedure and model. The design procedure is a series of steps that designer portrays in the software to be fabricated. Creative expertise, past experience, a feeling of what makes good programming, and complete dedication to quality are vital achievement factors for a component design. The design model is architect's plans for a building and gives direction to building every subtle element. Essentially, the design model thus made gives an assortment of various perspectives for the software development. The designer is empowered by the fundamental design principles to explore the design process (Davis 95).

A set of the variables that administer a good design are that design ought to unmistakably be verifiable, complete (actualizes every one of the specifications), and traceable (all design components can be traced to some requirements). Be that as it may, the two most imperative properties that worry designers are effectiveness and simplicity. Proficiency of any system is dependent on the best possible utilization of scarce assets by the system. The requirement for efficiency emerges because of cost contemplation's. On the off chance that a few assets are rare and costly, it is attractive that those assets be utilized effectively. Simplicity is maybe the most imperative quality criteria for information systems.

We have seen that upkeep of software is typically very costly. We have ascertained one of the objective as its maintainability. The design of a system is a standout amongst the most critical elements influencing the maintainability of a system. Amid support, the initial step a maintainer needs to attempt is to comprehend the system to be maintained. When a maintainer has an exhaustive comprehension of the diverse modules of the system, how they are

interconnected, and how changing one will influence the others in case the change be embraced. A simple and understandable design will go far in making the occupation of the maintainer simpler. These criteria are not independent, and escalating one might unfavorably affect another. In this manner, design choices most of the time include tradeoffs. It is the designers business to perceive the tradeoffs and accomplish the best sense of balance. Essential goal of the design procedure is to deliver designs that are easy to understand. Making a simple and efficient design of an expansive system can be a very complex job that requires great engineering judgment. As designing is essentially an artistic action, it cannot be decreased to a progression of steps that can be basically taken after, however guidelines can be made available.

The software development procedure is seen as a transformation function that changes the given inputs into the required outputs, and the focal issue of designing software systems is thought to be appropriately planning this transformation task. Because of this perspective of programming, the structured design methodology is basically function oriented and depends intensely on function abstraction and decomposition.

On account of web applications improvement designing frequently requires considering the stylish, useful, financial and sociopolitical dimensions of both the design object and design process. It might include extensive research, thought, modeling, intuitive modification, and re-design. Conventional software development procedures, or information systems utilizing databases, do not help in facilitating the hypertext metaphor.

For instance, they do not use links and incorporate hypertext into the interface. Moreover, as the size, intricacy and number of applications grows, a systematic methodology is required that helps managing the multifaceted nature, and permits progression and reuse of beforehand accumulated design knowledge. Delivering applications in which the client might profit by the hypertext worldview for exploring through Web-sites, while performing complex transactions over its data base, is a troublesome task (Gellersen 1997) . In (Bieber 1997), the authors bring up the fact that while numerous organizations will adopt the World Wide Web, there is a danger of losing well understood favorable circumstances of hypertext in the web .

In case of new items supporting the development of web applications the proposed approaches design them as they are to be used in non-web environment. Some of the proposed approaches are as follows:-

Parc-Place's Visual Wave (VWave 96) and Applied Reasoning's Classic Blend (CBlend 96) use a shared database but neglect the navigational feature totally. A sound navigational structure is one of the keys to achievement in hypermedia applications. When the client reaches a target he can achieve a sought target point, and take the maximum advantage from the application. At the point when navigation is consolidated with mixed media information, new issues come up (Hardman93).

In multimedia presentation, web application interface is mind boggling, not just do we have to indicate which interface articles ought to be executed, additionally the path in which those interface items will cooperate with the rest of the application. For instance, we have to recognize when an interface activity causes navigation to another page, when it is only a local interface impact. Experience accumulated in building interactive applications demonstrates that utilizing object-oriented strategy is a key approach, particularly on account of large, evolving applications".

Numerous organizations are presently changing interactive applications to the WWW (or to intranets utilizing the WWW conventions) , which involves including hypertext usefulness on top of the already present application behavior but sometimes troublesome since various design concerns must be obliged. Current design approaches tend to disregard one of those facets. The object oriented approaches don't give primitives to determining navigation, while hypermedia design approaches underline basic perspectives, overlooking computational behavior.

There is a developing accord about the exercises the software product should perform are analysis or modeling, design, implementation and testing. Software lifecycle models used for determining the sequencing of processes and product included in the advancement of an application.

Web applications is not inherently unlike from the one utilized when building traditional applications. It should be fulfilled in a unifying framework. In ultimate application, navigation and functional behavior must be flawlessly incorporated. Then again, amid the design process we ought to have the ability to change design choices related with the application's navigational structure and the domain model. In general most of the application environments don't give full backing to object oriented concepts. Therefore, our design model ought to effortlessly transform into existing platforms.

Hypermedia applications has four exercises processes the Conceptual Design, the Navigation Design, the Presentation Design and Implementation. These exercises are performed in a blend of various styles of development. In every step a model is constructed or enhanced. Beginning stage is elaborating model of application area, which decides the look of web application. This is done during concept designing stage, the concept model signify two sorts of objects:

- a) those that will be in the end seen as nodes in the navigational model (Jacobson 92)
- b) those that will give computation backing to the application, encapsulating behaviors. For example, algorithms access to databases and so forth.

Subsequent model might serve a base for some applications and would exclude any navigation related data. A crucial recognizing aspect of Web application is the concept of navigation in which application user navigates in a space. Navigation has its own behavior, execution function over and above browsing or navigation, e.g., upgrades or calculations. Another stride of organizing the navigation is given by assembling objects of navigational space in important sets called "Navigational Contexts". There exist a few conceivable criteria for characterizing such as accumulations of nodes based on class attributes and relationships .

In the case of Navigation design we characterize how navigation will continue determining conversions in navigation, i.e. the grouping of available "navigational objects" at a given time. Navigational object is not straightforwardly seen by user; rather, they are contacted by means of interface objects. The Presentation Design phase describes the layout of the various objects.

Once the logic structure of the interface is defined, the Presentation Model allows specifying the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages.

At long last, the Implementation stage is in charge of mapping the conceptual object, the navigation object and the interface object into a runtime environment being focused on. This might include characterizing HTML pages, script in some language and query to a relational database and so forth.

Therefore we can conclude from the above given details of a web application that though the concepts of web development are very different from that of a traditional software system. However, we need a software development methodology similar to that of a traditional information systems that follows a generic process model where all the different phases of software development are given due weightage.

There is a lot of emphasis on using a systematic, comprehensive and a good quality requirements approach for any information systems development. This is so, because requirement engineering is one of the most important phases of an information systems development. Web applications being very different in nature we need a requirement engineering approach that captures the different features of a web application development like the content, navigation, presentation and abstraction comprehensively. Therefore, we have need a web based requirements approach to understand the requirements of a web application development that has been tuned to gather and analyze the requirements of a web application to produce a high quality SRS.

1.4 UML (Unified Modeling Language)

A design methodology is a methodical way to producing a design by using a set of techniques and guidelines. Most design practices concentrate on the system design, and do not diminish the design action to a grouping of steps that can be aimlessly trailed by the designer. Designers are helped by the presence of modeling languages. These are utilized to represent

data, information or systems in an organization that is characterized by consistent set of instructions. A modeling language can be graphic or text. One of the graphic modeling language for software design is "Unified Model Language (UML)". UML is a broad-spectrum modeling language to depict the softwares in both systematic and behavioral manner. It has a graphic notation and takes into account expansion with an UML Profile.

Modeling is a design of the software product before coding (OMG's UML.org).It is an essential part of an information system development. A model in an information systems development is analogous to blueprints and other plans in the construction of a building. Utilizing a model, guarantees that a system design underpins prerequisites for a) security, b) scalability, c) robustness, d) extendibility and other attributes, before execution in code which makes change troublesome and costly to make and modeling is one of the best way to envision your design and verify it against requirement before coding.

Design methodologies have two viewpoints for them a) "language" and b)"notation" to express the design, especially while being created, and philosophy for adding to the design. Design is an imaginative and repetitive task, a great notation ought to help the designer amid the design action. This implies the notation ought to permit the designer to concisely catch the key features of design and refine it later and permit easy correspondence to empower conceptualizing. In cases of a good notation regularly the strategy for design turns into set of instructions, and the notation turns into main-tool for making design. UML is a graphic notation for expressing object oriented design.It's known as a model language and not notation as it permits expression of different viewpoints of the system, and not just design that must be actualized. For design, a description of classes that exists in the system may surface. Nonetheless, while modeling, amid the design process, the designerlikewise tries to comprehend how diverse classes are related and how they communicate to give the chosen function. This viewpoint of modeling assembles design that will probably fulfill every one of the requirement of the system. Because of ability of UML to make distinctive models, it has turned into a guide to understand the system, design the system, and in addition a notation for expressing the design.These qualities of the UML and the fact that it is a standard which is

used very frequently for Object Oriented technology we have tried to make our output UML Compliant.

We use the WebGRL approach based on "Goal Oriented Requirements Engineering" to capture the requirements of the stake holders using goals. This approach is enhanced to suit the requirements of the web applications by the WebGRL approach. Therefore, using the WebGRL approach as a base to capture the requirements for a web application we present a strategy for the transformation of these WebGRL diagrams into the design models which are UML compliant. Therefore, the last step of the output of our transformation approach delivers an UML profile which offers some assistance in specifications, envisions, and documents model of software system, which includes the structure and design such that it meets these requirement, so that transformation into the next phase of SDLC approach i.e. Coding is also taken care of by using structured design models that support platform independent models which can be easily converted into platform specific models. Therefore the use of this systematic approach of web application development where the different phases are dealt with in an organized manner as in the case of SDLC would result in lesser iterations and a high quality and low cost output.

Web applications are very crucial applications that reach out to a vast and varied audience. Engineering the web applications methodically has become even more important with the increased dependency on them. Thus, as in information systems, we need to focus on a methodical approach as in the case of traditional Information systems by using a software development process model like SDLC where the different phases of software development are dealt in detail separately. The problems identified in web application development are:

1. Use of ad-hoc approaches for web application development that give fast results, however the quality of the product is questionable.
2. Lack of use of a methodical approach that follows a software development process model in the case of web engineering.
3. Lack of early requirements analysis for web application development.

4. Lack of current approaches that consider both function and non-function requirement in web engineering that are tuned to capture the requirements phase completely especially the non-functional requirements which have a huge impact on the quality of the web application developed.
5. Lack of choice of appropriate design model that captures the web requirements both functional and nonfunctional comprehensively to produce a high quality software design.
6. Need for a transformation approach that captures both the "functional and non-functional requirements" comprehensively for "requirements engineering" phase and transform it losslessly and seamlessly into the design phase.
7. Use of design models that follow a model driven architecture and provide a structured way for conversion of the design models into a platform independent model which can use a model transformation language to convert it into a platform specific model thus providing the code reusability.

Summarizing the problems, it can be stated that

The existing web engineering approaches lack a methodical web application development approach that follows a software development process model with detailed requirements elicitation and designing that leads to a low quality and high cost of the product under development due to multiple iterations and unsatisfactory performance of the product, therefore we need a web application design approach that is methodical and captures the requirements in totality and further translates these requirements flawlessly into design models that support a structured and systematic approach of a platform independent model to ensure a complete and high quality software.

The issues faced in web application development can be solved if proper procedure of requirements engineering is followed where both *function and non-function requirement are taken care and watertight transformation of these requirements is done into the design phase where the design models provide a structured means of further conversion of the design models into a platform independent models.* There are many requirements engineering approaches that are prevalent we use the "goal oriented requirements engineering" based WebGRL methodology for requirements engineering phase as a base to transform these

WebGRL diagrams into the design models which are UML compliant thereby ensuring easy portability into software code.

1.5 Research Contributions

For solving the issues that are faced in web Application Development a proper requirements engineering methodology should be used that captures the requirements of the web application being developed exhaustively. Secondly, we need a design approach that collaborates with the requirements phase to capture them losslessly with traceability and validity. The design stage is one of the most vital stages in the software engineering life cycle. A good design ensures that the requirements have been captured completely and the different facets of the web application are represented with the help of design models to provide an insight into the possible solution for the problem domain. Further, it also ensures that a structured means of conversion into the coding phase exists. Therefore, the software design is a very important phase of the SDLC which acts as a smooth bridge between the requirements phase and the coding phase. A good design would lead to a simple, low cost and efficient architecture and the detailed design of the problem. There is increased need of a good design approach that gives a holistic view of the web application development as their importance and complexity is increasing. In this research thesis we focus on the following:-

1. Ad-hoc approach of web application development to be replaced with a systematic process of development as in the case of traditional 'Information systems development'.
2. Different phases of Information Systems development consisting of proper requirements engineering and software design phases suitable for web application development to be followed.
3. The Information Systems development phases to be tuned to meet the requirements of web application development. Therefore, we need to choose a requirements

approach which has already captured the requirements process comprehensively. As the WebGRL is an enhanced version of grl that captures both function and non-function requirement in a comprehensive manner which is tuned specifically for web application development. It will be used as an input to the design phase.

4. Similarly, we need to define a design model that is suited for web application development therefore we choose AOOH as it has the capability to capture different perspectives of the web application development and enhance it to the EAOO-H Design model.
5. Next, we need to define the transformation strategy for transforming the requirements from the requirements phase to the design phase automatically that results in a high quality low cost better standard of designs of web applications.
6. An Automatic transformations method to be followed from one phase of the web application development to the other phase as it reduces time of development and eases the work of the design engineer resulting in a structured, systematic, consistent and comprehensive web application development approach that has lower cost, lesser time and has better quality reliability and traceability.

Web based design approaches have been gaining a lot of recognition and research interest for their ability to capture web based application development comprehensively and holistically in a systematic manner. Web engineering concepts and approaches and their analysis has been discussed in the next chapter in detail. It is important to mention here one of the very recent and popular design methodology, i.e. "A-OOH". It is an "Adaptive Object Oriented Hypermedia" design approach that follows the OO-H methodology for web development. A-OOH design techniques can be applied to overcome the web application development problems. A structured and systematic approach that incorporates detailed requirements engineering and later transforms these requirements into a good comprehensive design would ensure lesser iterations, easy evolution, a complete and high quality web application

development that focuses both on the requirements and the design phases as in the case of traditional software development.

This thesis proposes that the solution of these problems can be found if *an all-embracing, exhaustive design approach that incorporates a good comprehensive requirements approach and seamlessly translates it into a good design for web application development* is followed.

This would result in a very high quality and a reliable web application development output as the focus is extensively on the two early phases of requirements and design engineering. Also the design model output being a platform independent model the output can be easily translated into coding. The goal oriented requirements engineering along with the Object Oriented based design model techniques when applied for web application development would create better quality product with lesser number of iterations thus, saving cost and time of the project. This is due to the fact that a comprehensive requirements and design engineering techniques provide an in depth insight and the understanding of the system is better for both the developer and the stakeholder. For web applications, which are very critical, complex and indispensable systems in today's times an efficient design engineering technique that captures an all-embracing requirements engineering is required. This thesis accomplishes the design engineering of web applications using a step by step approach as explained below.

1. Enhancement of the AOO-H model into the EAOO-H Design Model to capture the GORE based WebGRL specific diagrams of requirements engineering

First and the foremost, the unique concerns of web applications need to be understood. The web applications are distinct from "traditional information systems" and their concerns are also different. In the case of "web application development" the facets of content, presentation and navigation etc. play a major role in defining the requirements. Therefore we need a suitable "web application development" approach that captures the requirements in entirety based on the different facets stated above. In the case of an "information systems

development" the emphasis is on capturing both the "functional and non-functional requirements" through goals and softgoals. The goals represent the explicit requirements and the softgoals represent some of the unstated or implicit requirements that are often overlooked while development of software. These hidden requirements effect even the functional requirements and their corresponding qualities or constraints. Though a number of web engineering approaches like OOHD, UWE etc. exist based on the factors of the capability of the WebGRL approach of (Srivastava & Chawla 2010) to capture both the functional as well as nonfunctional requirements with respect to web applications comprehensively and its ability to resolve conflicts and ambiguities we have chosen the WebGRL approach as a base for providing input to our design approach. In order to use the WebGRL diagrams for stating the requirements captured in the requirement phase we need a good design approach that is dedicated to web application development and also has the capability to capture and transform the web application requirements seamlessly into the design phase. Further, the output of the design phase should be such that it provides the ease of transformation into the coding phase. Keeping this in mind we need a design approach for web applications that acts as a smooth bridge between the requirements and coding phase and helps the requirements to be translated easily with the help of the design engineering into the coding phase. as stated above a number of web based Design models exist, however we have chosen the A-OOH model of Garrigos to capture the WebGRL input as it is web based and consists of domain, navigation and presentation, and adaptation models as part of the AOOH design models. In order to capture the WebGRL requirements in totality without any loss of information of the requirements phase the A-OOH model of Garrigos has been extended and enhanced to the EA00-H model by us in this thesis. The EA00-H model not only captures the WebGRL requirements losslessly it allows seamless transformation from the requirements phase to the design phase. The EA00-H or the Enhanced AOOH model developed by us has been described in detail in Chapter 4, 5, and 6 of this thesis.

2. Transformation of content WebGRL diagram to EAOO-H domain model.

In the second step we proceed to transform the requirements captured by the WebGRL specific diagrams into their corresponding specific design models. The WebGRL diagram is refined into its constituent Content, Navigation, Presentation, Adaptation and Business process specific WebGRL diagrams that represent the different facets of a web application. In the design phase we take these specific WebGRL diagrams one by one and transform them into a corresponding design model. Our EAOO-H design model has been designed in a manner that it captures the different facets of the WebGRL specific diagrams into the corresponding Domain, navigation and Presentation models. In the first step of the transformation process of the requirements model into the design model we start with the Content WebGRL diagram as an input to the design model and define a transformation algorithm that converts it seamlessly into the corresponding EAOO-H Domain Model. The transformation strategy and the Enhanced AOOH Domain model have been described in the Chapter 4 of this thesis.

3. Transformation of navigation WebGRL diagram to EAOO-HNavigation Model

In the third step we continue to transform the requirements captured by the WebGRL specific diagrams into their corresponding specific design models. In this step of the transformation process of the requirements model into the design model we transform the navigation WebGRL diagram as an input to the design model and define a transformation algorithm that converts it seamlessly into the corresponding "EAOO-H navigation Model. The Navigation model" is one of the most important part of the design model. It captures the navigation between the different web pages as well as the navigation between the components of a web page and other websites. Therefore we need to process the navigation requirements of the navigation WebGRL diagram intelligently such that the navigation between the pages is well directed, user friendly and does not turn into a chaos when the user transits from one webpage to

another. Keeping these factors in mind the transformation algorithm and the Enhanced AOOH Navigation model have been described in the Chapter 5 of this thesis.

4. Transformation of presentation WebGRL diagram to EAOO-H presentation model

In the fourth step we go on to transform the requirements captured by the Presentation WebGRL diagrams into its corresponding Presentation design model. In this step of the transformation process of the requirements model into the design model we define the transformation algorithm for transforming the presentation WebGRL diagram as an input to the Presentation design model that converts it flawlessly into the corresponding EAOO-H Presentation Model. The Presentation model represents a very important aspect of the web application development. . The Presentation model is a two tier structure. It describes how the presentation of the content is to be distributed amongst the various web pages at one level and the layout of the content components within a web page in another level. As a result we need to process the presentation requirements of the presentation WebGRL diagram smartly such that the presentation of the web site is eye catching and pleasing to the user while maintaining a certain amount of consistency between the different webpages such that they appear as a collection of the same website. Therefore, the transformation strategy for the Enhanced AOOH Presentation model has been dealt with accordingly and the same has been described in the Chapter 6 of this thesis.

5. UML Profile for the WebGRL and EAOO-H Design Models

A good design should act as a smooth link between the requirements and the coding phase. Therefore in this step we move from the transformation of the requirements into the design phase towards making the design easily transformable into the coding phase. This quality of a design output goes a long way in the web application development as it ensures a good transformation strategy in the movement from the requirements phase into the coding phase. In order to do so we further define a UML Profile for the three EAOO-H Design models defined by us in chapters 4, 5 and 6. Such that the output of the design phase is UML Compliant that is a platform independent model. UML being a standard provides easy

mechanism of transformation from a "Platform Independent Model into a Platform Specific Model" using some constraint language like OCL etc. The UML Profile for the three design models has been described in detail in the seventh chapter of the thesis.

6. Transformation Tool for WebGRL to EAOO-H Design Model

A transformation tool for transforming the specific WebGRL models of content, navigation and presentation diagrams into the corresponding domain, navigation and presentation EAOO-H design models using our transformation algorithms has been developed. This tool automates the entire transformation process of transforming the requirements into the design phase and reduces the work of the design engineer considerably.

These steps produce a design model output UML Compliant or a Platform Independent model that can be easily transformed into a platform specific model with traceability to web specific requirements. The Transformation strategy used ensures a smooth and seamless transformation of requirements into the design phase and further into the coding phase. This covering a major portion of the web application development which is automated resulting in greater efficiency lesser time, cost, iterations and reduction of time and work in the case of the design engineer.

This thesis enriches the A-OOH Design model by enhancing it in order to capture the specific WebGRL requirements completely. The Transformation strategy is defined for each of the constituent specific WebGRL diagrams namely the content, navigation and presentation models transformation into the corresponding EAOO-H Design model. Further a UML Profile for the three EAOO-H Design models has defined to enable the transformation of the design phase into the coding phase easily. Web Application Design tool has been developed to support the transformation process from web requirements engineering into web design engineering.

The contributions of the thesis are:

1. The enhancement of the AOOH model to EAOO-H Design model in order to capture the constituent specific WebGRL diagrams requirements completely in the web application design phase.
2. Defining a transformation algorithm for transforming the Content WebGRL requirements into the EAOO-H Domain Design Model seamlessly without any losses.
3. Defining a transformation algorithm for transforming the Navigation WebGRL requirements into the EAOO-H Navigation Design Model seamlessly without any losses. This transformation algorithm deals with the navigational aspect of the web application being developed and has to ensure integral responsibility of capturing the WebGRL navigational requirements.
4. Defining a transformation algorithm for transforming the Presentation WebGRL requirements into the EAOO-H Presentation Design Model seamlessly without any losses. This transformation algorithm deals with the presentation side of the web application being developed and has to ensure an appealing layout of the web application development while also ensuring that WebGRL presentational requirements have been captured in totality.
5. Defining a UML Profile for the three design models namely the Domain, Navigation and Presentation model to produce a UML Compliant Platform Independent Model as an output of the design phase
6. Enable seamless transition to the coding phase.

The diagram of the transformation proposed in this thesis is shown in figure 1.1. This figure sums up the contributions of the thesis by showing the WebGRE input to the design phase and the output produced by it. The tentative goals and qualitative expectations are transformed to selected solution in the form of EAOO-H Design models that is UML compliant, thereby enabling ease of transformation to the coding phase and acting as a strong bridge between the requirements and the coding phase. A seamless transformation from the requirements to the coding phase thereby ensures traceability to the requirements phase

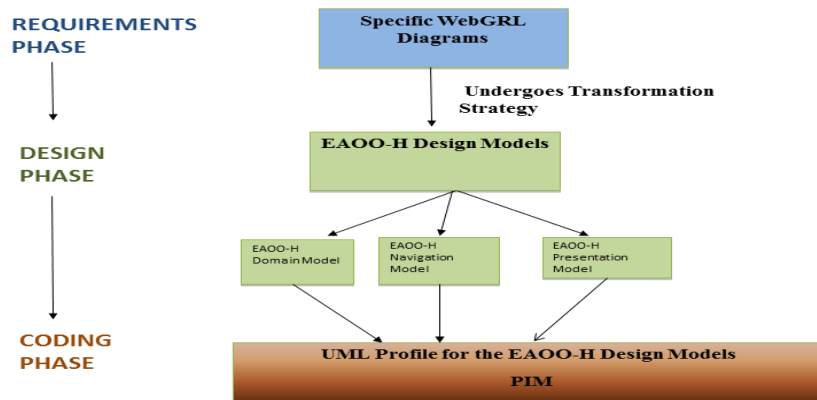


Figure 1.1 Block Diagram of EAOO-H Design Models Framework

This thesis enhances an existing design model and defines a comprehensive and systematic transformation approach for transforming the web specific goal driven requirements engineering into a web based design model. A new fully automated transformation strategy from requirements to the design phase has been proposed in this thesis, with a UML Compliant output of the design models which is a Platform Independent Model as such it provides ease of transformation to the coding phase of the SDLC resulting in a systematic methodology for transformation between the two different phases of SDLC that is more complete, consistent and traceable for the web application leading to a better quality product.

1.6 Organization of Thesis

The details of the chapter layout are as follows:

Chapter II

Web Engineering based on scientific and engineering principles, aims at establishing a structured way to effectively create, implement and maintain high quality Web-based systems forms the basis of this chapter. In this chapter, the literature review has been presented that formed the basis and motivation of this thesis. Various web Application development techniques for Model Driven Web Engineering like the OOHDM, SHDM, WEBSA, HM3 have been discussed and compared in detail in this chapter and suitability of A-OOH for web application development is justified.

Chapter III

The Requirements and Design Approaches which form the base for our input and transformation process have been described in detail in this chapter. WebGRL Model used as a requirements engineering approach for providing the input to the design phase has been discussed and justified. Further the OOH and the A-OOH Design Models which have been described. These design models have been enhanced to form the EA00-H Design Model in the later chapters which has been used for the transformation strategy for transforming the requirements into design for a web applications

Chapter IV

This Chapter presents the Enhancement of A-OOH Domain Design Model for capturing the requirements represented by the content WebGRL diagram. Further, the transformation of these content based requirements into the Domain Design Model using the transformation algorithm defined by us has been presented. The transformation strategy has also been illustrated with the help of an example.

Chapter V

The web specific navigation requirements for a web application are captured by the navigational WebGRL diagram. In this chapter Enhancement of the A-OOH Navigational Model to capture the navigational requirements is done. The navigational WebGRL diagram is used as an input for the navigational transformation process, further the transformational algorithm transforms the navigational WebGRL diagram into the EA00-H Navigational Design Model.

Chapter VI

In this chapter, the algorithm for transformation of the Presentation WebGRL Diagram into the EA00-H Presentation Design model is presented. In order to capture the Presentational requirements for a web application in total after the transformation into the design model the

EAOO-H Presentation Design Model is enhanced. The EAOO-H Design Model and the Presentation Transformation Algorithm has been presented in Chapter 6.

Chapter VII

In Chapter 7 the UML Profile for the three EAOO-H Constituent design models i.e. the Domain, Navigation and Presentation Models is presented. The advantage of using a UML Profile is the fact that it is a UML Compliant output which is a standard and the same can be easily implemented.

Chapter VIII

The implementation of the requirements to design transformation approach has been done by the transformation tool. The tool has been built upon Microsoft Visio platform and visual basic language has been used to implement the algorithms proposed in chapter 4, 5 and 6. The features, design and working of the tool has been explained with the help of online bookstore application example.

Chapter IX

Finally, the conclusion is presented that highlights the main contributions of this thesis. The comparative analysis of the problems faced in web applications and the accomplishments in achieving the solutions are discussed. The future directions of work in the area of web specific design engineering are also offered.

The *list of publications* produced by this research endeavor is presented in the end along with the *references* used throughout thesis.

Chapter 2

Literature Review

Web application development is based on the specification of structure, behavior, navigation, and presentation aspects. Recently attempts have been made to use a model -centric process for web application. These attempts encourage software developers to focus on problem domain specific modeling and analysis and structured software reuse to develop a high quality and standardized design model based on a systematic and structured web engineering process. In this chapter, review of existing methodologies for creation of web applications is studied. We discuss the various Model Driven Web Engineering approaches which propose a structured model to present the three models, the data model that depicts the content of the web application, a navigation model depicting navigation used for connecting pages and the presentation model with respect to their contribution to the model transformations in the Web engineering process.

2.1 Background

The Web application development technology have turned into a noteworthy usage and release platform for an expansive assortment of applications. These web applications range from straightforward web sites to supply-chain management systems, economic web sites, e-governance, onlineeducation and so on among others. Such web applications, besides offering their inborn services, additionally show the more intricate behavior of dispersed applications. This has prompted the advancement and extension of the area of Web Engineering.

2.2 Web Based Engineering

Web engineering is concerned with the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment, and maintenance of high quality web-based systems and applications (Murugesan, Deshpande, Hansen, & Ginige, 2001).

This includes theoretical principles and systematic, disciplined, qualitative and quantifiable approaches towards the cost-effective development and evolution of high-quality, ubiquitously usable web-based systems and applications. It fundamentally concerns the technology which enables the construction of web applications. Web engineering, while rooted in computer science and engineering, draws from a diverse range of other disciplines, such as information science, information systems, management and business, among others. The web engineering process and tools brought improvement in creation of web applications and followed a more structured approach. The web specific requirements and phases were identified and dealt with. Some of the popular web engineering methodologies are discussed briefly in the following paragraphs.

When developing a Web application, it is necessary to specify structure, behavior, navigation, and presentation aspects. Traditional engineering methods failed to specify navigational and presentation issues, which is why some people have proposed specific approaches to tailor these two aspects of Web applications. Such approaches are called Web Engineering Methods. In recent years, there have been several attempts to promote web application development as a model-centric process. These attempts encourage software developers to focus on problem domain specific modeling and analysis and structured software reuse to develop a high quality and standardized design model based on a systematic and structured web engineering process. In this chapter we present some of the most relevant model-driven Web engineering methods that have been proposed, and discuss and analyze the advantages and disadvantages of such methods keeping in mind the requirements engineering phase, current tendencies and best practices in Web application development.

2.3 Model based Software Development

The application of models to software development has been a tradition for a long time, and it has become more popular since the development of the Unified Modeling Language (UML) (OMG UML 2009). Model Driven Software Development (MDS) presents an approach in which models do not only constitute documentation, but are considered to be similar to code artifacts because their implementation is automated. Since those models are highly coupled to the domain of applications, MDS aims at finding domain specific abstractions that can be specified through formal modeling, providing models that can be understood by domain experts. MDS is a software paradigm with roots in Software Product Lines (SPL) (SEI 2010) "engineering, which is the discipline of designing and building families of applications for a specific purpose or market segment" (Stahl & Voelter 2006). In order to apply the domain-specific model concept, there are certain requirements that need to be taken into account: Domain Specific Languages (DSL) that allow expressing the models, transformation languages to express the model-to-code transformations, and code generators to obtain executable code on several platforms. MDS's idea is to give models a central role. "Web Engineering based on scientific and engineering principles, aims at establishing systematic approaches to successfully develop, deploy and maintain high quality Web-based systems. It also incorporates well-known software engineering principles and practices" (Murugan et al 2001) from diverse areas such as human-computer interaction (HCI), system analysis and design, requirements engineering, hypermedia engineering and data.

2.3.1. Model based "Web engineering methods"

As "model-based initiatives such as MDS grew in popularity within the software development community, several Web Engineering approaches began to change their notations and processes to be MDS compliant. As stated in" (Koch et al 2008), this change implied a redesign in Web modeling languages; a reorganization of the set of models to be built in a modular and platform independent way; planning the development processes in terms of model transformations; and adopting standards"

"such as UML, Meta-Object Facility (MOF") (OMG MOF 2009), XML Metadata Interchange (XMI), or Query/ View/Transformation (QVT). MDS D compliance turned into a discipline within Web Engineering called Model-Driven Web Engineering (MDWE). MDWE adopts some of the techniques proposed in MDS D in order to generate Web applications": (1) the construction of metamodels and models in the Web applications domain; (2) the definition and implementation of model-to-model transformations and model to text transformations with the purpose of obtaining some parts of the entire implementation; and (3) the adaptation or development of CASE tools to support the creation and transformation of models and the generation of code. In this way, MDWE aims at bridging the gap between the high level design models and the low-level Web implementation code" (Gellerson & Gaedke 1999).

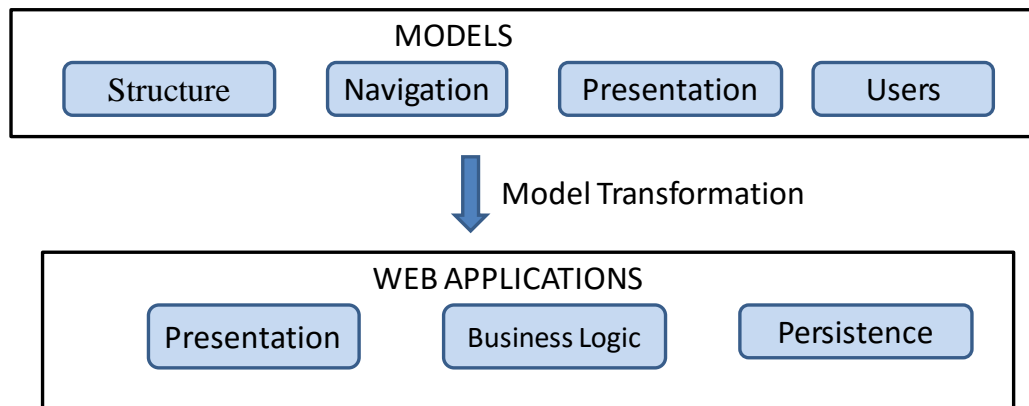


Figure 2.1 General scheme of MDWE Methods

Figure 1 portrays a general plan pursued by a large portion of the MDWE techniques, These MDWE techniques propose a systematic model that is used to present the data, navigation and presentation using the "data model", a "navigation model" for depicting navigation used for connecting pages and the "presentation model" characterizing the layout of the website components. A user model is utilized as a part of some methodologies. As most MDWE techniques depend on conventional Web Engineering

strategies, we first present the conventional techniques, and later continue to reevaluate Model Driven Web Engineering approaches. Later on, we present the arguments for each of the Model Driven Software Development based strategy displayed below.

The most illustrative methodologies of conventional Web engineering are Web engineering methods. The Web engineering methods are as per the following:- (a) The "Object-Oriented Hypermedia Design Method" developed in 1995 (b) The "Web Site Design Method" presented in 1995 (c) "Web Modeling Language" of 2000 (d) "Object-Oriented Hypermedia" (OO-H) presented in 2000 (e) "UML-Based Web Engineering" put forth in 2002 (f) "Semantic Hypermedia Design Method" (SHDM) presented in 2004 (g) "Navigational Design Techniques" (NDT) put forth in 2008.

2.3.2. The Schwabe & Rossi's Hypermedia Design Model

The Schwabe & Rossi's Object-Oriented Hypermedia Design Model (OOHDM) (Schwabe & Rossi 1995)(Schwabe & Rossi 1998),(Schwabe 1995), (Schwabe et al 2002) is based on Object oriented paradigm. As every object-oriented modeling proposal, it promotes the development of new applications reusing existing components. OOHDM is a four steps process. The first step is a domain analysis in which an application conceptual model is built. The second step consists of a navigational design describing the navigation structure of a hypermedia application in terms of navigational contexts. These contexts are inferred from navigation classes such as nodes, links, and guided tours. Nodes represent logic views of conceptual classes, and links are derived from conceptual relationships. The abstract interface design is the third step and proposes the construction of perceptible objects, such as an image or a map, in terms of interface classes. These perceptible objects are mapped to navigational objects in order to give the last ones a perceptible appearance. In the fourth step, interface objects are mapped to implementation objects involving architectural specifications.

2.3.3. The De Troyer & Leune's "Web Site Design Method"

"Web Site Design Method (WSDM) was started in 1998/1995 by De Troyer and Leune (De Troyer and Leune, 1998)" and is known as one of the primary webpage design techniques. It encourages the development of Web sites and Web applications systematically. WSDM is a user focused methodology, it takes as a beginning stage the distinctive users with their diverse requirements and goals.

The group of users of the site is arranged into a levels as per these requirements, and the navigation structure of the site is based upon this order. WSDM likewise gives the designer a reasonable technique on the most proficient method to develop these design models. It comprises of four stages namely the conceptual design, user modeling, implementation design, and the actual execution.

WSDM differentiates between the conceptual design i.e. the task analysis, modeling data and functionality, and the conceptual navigation schema and the installation design i.e. gathering of data onto pages, layout and presentation, which further makes a differentiates between the execution plan for gathering into pages, layout and presentation and the real execution e.g. java servlets, php code, html, XML etc. and so forth and the relating design models. A more detailed explanation of WSDM can be found in (De Troyer and Casteleyn, 2001) and (De Troyer and Casteleyn, 2004) .

The first Stage of WSDM categorizes the web users based on the business or the procedure for which the website is going to be constructed. A user modeling stage is performed, comprising of a arranging a group of users together based on their behavior. Amid categorization the various user classes are identified, depending on the requirements of the possible guest users, which are grouped into different user levels of importance. While gathering people behavior, it depicts the pertinent qualities for the diverse group of user classes. Informational requirements are then used to characterize user classes in to subsets. Further, data requirements and behavior of the target group are used to dissect the user classes. On the off chance that users have

diverse attributes within a user class, this is isolated into points of view which are used to symbolize the various dissimilar usability requirements.

The second stage is used to properly display the data requirements presented in the user class portrayals by building conceptual models for the diverse user classes. There are two substages in the conceptual design namely the navigation design and the task and data modeling. In the task and data modeling substage, the user requirements, which were casually indicated in the past stage, are described. For every requirement, a task model is determined. WSDM utilizes the concurrent task tree strategy of (Paterno et al, 1997) for its task modeling in light of the fact that it permits not just the progressively partition of task into its subtasks as well as to determine temporal relationships between various subtasks. Next, for each rudimentary task, an object chunk is generated. This object chunk is a small conceptual schema to formally depict the information or usefulness required by a specific rudimentary task. At first, WSDM utilized a broadened variant of ORM (Halpin, 1995) to model object chunks, yet as of now OWL is utilized.

In the substage called navigational design, an abstract conceptual navigational structure of the Website is constructed. A navigational model comprises of nodes, which gather data and/or functionality into a cluster that sensibly belong together. Links between these nodes indicate navigation paths. WSDM gives a structured technique to acquiring the navigation structure of a Website: to begin with, the user classes at different levels are mapped onto a navigational model using a 1:1 mapping, making a navigation track for every group of user class. Next, every navigation track is further refined utilizing the data as a part of the task models. Nodes are made for rudimentary tasks; navigation links are utilized to mirror the relationships between tasks and the relevant object chunks are associated with the nodes . A navigation track is depicted in the form of links and components that guide the path clients of a specific point of view can explore through the accessible data. Three sorts of components are explained: A data segment with the data that relate to a particular point of view; a

navigation element that comprises of a cluster of links; and an outer segment which points to a segment in another website.

In the third stage the user interface of the webpage is composed with the goal of making a reliable, satisfying, and productive output for the conceptual design made in the past stages. WSDM proposes some policies and guiding principles keeping in mind the end goal to accomplish an implementation model, similar to those portrayed in (December et al 1995). The objective of the implementation design is to provide, for the conceptual design, the essential execution details. There are three substages in the implementation as stated below. The subject of the principal subphase, the Site Structure Design, is to choose how the nodes, characterized in the navigation model, will be assembled into pages. Note that in this stage we characterize abstract pages; every abstract page results in multiple instances of concrete pages during execution. If required, diverse Site Structures can be indicated, reflecting for instance distinctive devices, contexts or platforms. The objective of the following sub stage, the Presentation Design, is to depict the arrangement (i.e., position and style) for all pages. WSDM gives three unique arrangements of modeling concepts to depict the design of a page. Every set gives an alternate level of abstraction. Amid the Template and Style design, the designer determines layout that will be utilized for various sorts of pages. Templates characterize the general design of a page. To determine the Website style (text styles, hues, illustrations, and so on), WSDM right now utilizes Style Sheets. In the Page Design sub stage the designer portrays where the data on a page (characterized in the object chunks) ought to be situated and how it ought to be laid out. It is likewise chosen how and where the links characterized in the navigational design ought to be exhibited. A template characterized in the past substage is taken as a beginning point for a page .

The last stage is the final execution of the Web application as indicated by the design made in the past stages. Keeping in mind the end goal to store the data connected with the webpage, creators proposed utilizing strategies for database supported sites

portrayed in (Greenspun et al 1997). In this stage, the final execution, all the data gathered so far is taken as information for the web application development.

WSDM describes two implementation architectures: an XML-based transformation pipeline, capable of generating static Web sites, and a Java-based server-side implementation also capable of generating dynamic Web sites. WSDM also has support for several other design concerns: localization, semantic annotations, accessibility and adaptation.

2.3.4. The Ceri et al's Web Modeling Language

The Web Modeling Language (WebML) (Ceri et al 2000) is a notation to specify complex Web sites at conceptual level. It proposes four models: (1) a structural model defining the data of the application in terms of relevant entities and relationships, which is compatible with UML and Entity/Relationship (E/R) notations. (2) A hypertext model that describes hypertexts that can be published in the site. Each hypertext defines a site view and consists of a composition sub-model and a navigation sub-model. The first model specifies the pages that make up the hypertext and the content units that structure a page, and the second one describes how pages and content units are linked to form the hypertext. (3) A presentation model that states the distribution and graphical appearance of the pages, independently of the output device. And (4) a personalization model in which users and groups of users are modeled as predefined entities called User and Group.

2.3.5. The Gomez et al's Object Oriented Hypermedia

The Object Oriented Hypermedia (OO-H) (Gomez et al 2000), (Cachero & Gomez 2002) Method is a generic model, integrated into OO-Method (Pastor et al 1997, 2001), for the semantic structure of Web interfaces. It defines the abstract interaction model of the user interface, the information which each type of user can access, and the navigation paths from one information view to another. Since OO-Method captures the statics of the system, OO-H addresses particularities associated with the

design of web interfaces by adding several constructs for navigation and interface. OO-H provides an interface execution model in order to determine the way of implementing the conceptual model in a given development environment. In OO-H, the navigation model is represented by means of a Navigation Access Diagram (NAD). The NAD is built starting from the filtering and enriching of the information provided by the class diagram that is captured in the conceptual modeling phase of the OO-Method. As each type of user has a different system view and can activate different services, each one needs a corresponding NAD. The main components of the NAD are navigation classes, navigation targets, navigation links, and collections. Navigation classes, which are based on the classes identified during the conceptual modeling phase, contain the attributes relevant to the considered user and view, and the services capable of being invoked by the actual user of the NAD. A navigation target is a set of navigation classes which provide the user with a coherent view of the system. OO-H bases its navigation targets on user requirements, instead of on the physical presentation of the information. Navigation links are defined by a name, a source navigation class, a target navigation class, associated navigation patterns, and associated navigation filters. Navigation patterns (Bernstein 1998) are a mechanism for a web user interface to share its knowledge about the way of showing the information objects to the user. Navigation filters restrict the order, the quantity or the quality of the target objects. Collections are structures, with a set of filters and a set of navigation patterns associated, which abstract some concepts regarding both external and internal navigations, and are useful limiting the interaction options between the user and the application. Regarding the execution model for a target development environment, OO-H focuses on defining how to implement the interface information associated to web environments, since OO-Method has already defined an execution strategy. We will discuss the OO-H model in detail in the next chapter.

2.3.6. The Koch et al's UML-based Web Engineering

"UML-based Web Engineering (UWE)" (Koch & Wirsing 2001), (Koch & Kraus 2002) is a development process for Web applications which focuses on systematic"

design, personalization, and semi-automatic generation. Based on UML and the UML extension mechanism, it defines navigation and presentation models which are supplemented by other UML diagrams and UML modeling elements within an iterative and incremental approach based on the Unified Software Development Process (Jacobson, Booch, Rumbaugh 1999). The main modeling activities in UWE are the requirements analysis, conceptual, navigation and presentation design, supplemented with task and deployment modeling and visualization of Web scenarios. The task models and state charts of Web scenarios are included to model the dynamic aspects of the application. In UWE, requirements of a Web application can be specified by using a use case model. The static view of the system, also known as conceptual model, is represented using a UML class diagram which is built based on the use cases and the detailed description of these use cases with activity diagrams (in a textual form). The navigation model is represented as UML stereotyped class diagrams and consists of two components: the navigation space model and the navigation structure model. The former specifies which object can be visited by navigation, while the latter defines how these objects are reached. The modeling of the navigation is built following a set of guidelines defined in (Henicker & Koch 2000). The presentation model is represented using a particular form of a class diagram that uses the UML composition notation for classes and also stereotyped classes. This model describes where and how navigation objects will be presented to the user. For the presentation model, UWE uses a set of stereotypes that consists of the stereotypes text, button, image, audio, anchor, collection, and anchored collection. In UWE, state chart diagrams are used in order to visualize navigation scenarios that allow detailing of the navigation structure model by specifying the event that triggers the transitions, defining guard conditions, and including the actions to be performed. UWE also proposes the use of sequence diagrams to show presentation flows such as interaction between windows and frames. UWE uses activity diagrams for task modeling; here, a task represents one or more actions that a user may perform to achieve a goal

(Harmelen 2001). It also extends the concept of task by including actions performed by the system.

2.3.7. SHDM

Just like the The Object-Oriented Hypermedia Design Method (OOHDM) (Schwabe and Rossi, 1995; Schwabe and Rossi, 1998) and its successor, the Semantic Hypermedia Design Method (SHDM) (Schwabe and Moura, 2003; Schwabe et al. 2004) allows the concise specification and implementation of hypermedia (Web) applications. This is achieved based on various models describing conceptual, navigation and interface aspects of these applications, and the mapping of these models into running applications, in various environments.

SHDM is a model-driven approach to design Web applications using five different design phases (the same as OOHDM) explained above. The difference with OOHDM is that SHDM uses semantic Web technology to represent its models. In the Requirements Gathering phase, the requirements are obtained forming scenarios that are abstracted into use cases. The analysis of these use cases determines the user profiles and the tasks that the application will support. For each use case a user interaction diagram is defined, representing the interaction between the user and the application. Although the information for personalization is not explicitly extracted yet, this analysis could be extended to do so.

In the Conceptual Design phase a conceptual model is defined with the domain objects, relationships and required functionality. The SHDM conceptual model uses semantic Web technology as RDF and OWL. In the Navigational Design phase the navigation structure of the application is described in terms of navigational contexts, which are induced from navigation classes such as nodes, links, indexes and guided tours. There are two models: the navigational class model, which describes which information can be accessed and the navigational context model, which describes how this information can be reached. In SHDM this views are defined using the RDF

Query Language (RQL). Different navigational models may be built for the same conceptual schema to express different views on the same domain.

The Abstract Interface Design makes perceptible to the user the navigational structure of the application through the application interface, which is done by defining an abstract interface model. In SHDM, the abstract interface model is specified using the Abstract Widget Ontology. The specification of the concrete interface design (look and feel and layout) is left to the graphic designer, since it is totally dependent on the particular hardware and software runtime environment.

Finally in the implementation phase, the interface objects are mapped to the implementation objects and may involve elaborated architectures, generating the actual Web site implementation code.

2.3.8. NDT (Navigational Development Techniques)

(Escalona and Aragón 2008) is mainly based on the early phases of web engineering. It starts with the overall objectives of the system and capture, definition and validation of requirements is followed. The requirements are categorized into information storage requirements, functional requirements, interaction requirements or non-functional requirements. Then analysis models are created using the metamodel of UWE that are depicted with class diagrams, use cases etc. Though requirements analysis has been taken up in this technique but the usage of textual templates for use cases is very tedious. Also, it doesn't cover all the requirements like personalization in depth. The non-functional requirements are discussed and specified in isolation and their interrelationships with other requirements and effect is not covered in detail.

2.3.9. Analysis of the MDWE Methods

Most of the methodologies except NDT don't carry out or implementation requirements analysis phase in detail (Aguilar, Garrigos, Mazon, Trujillo, 2010). Even when requirements analysis is carried out like in NDT, the non-functional requirements are mostly considered in seclusion and their influence on other

requirements is not analyzed. The rationale for choice of appropriate design alternative is not there.

In the Requirements Gathering phase, the requirements are obtained forming scenarios that are abstracted into use cases. The analysis of these use cases determines the user profiles and the tasks that the application will support. For each use case a user interaction diagram is defined, representing the interaction between the user and the application. Although the information for personalization is not explicitly extracted yet, this analysis could be extended to do so.

In OOHDM, a hypermedia application is built in a five-step process supporting an incremental or prototype process model. Each step focuses on a particular design concern, and an object-oriented model is built. Classification, aggregation and generalization/specialization are used throughout the process to enhance abstraction power and reuse opportunities. SHDM is a model-driven approach to design Web applications using five different design phases, same as OOHDM. The difference with OOHDM is that SHDM uses semantic Web technology to represent its models.

To understand and improve the web development process it is imperative to take up the requirement analysis phase thoroughly. The Object-Oriented Hypermedia Design Method (OOHDM) (Schwabe and Rossi, 1995; Schwabe and Rossi, 1998) and its successor, the Semantic Hypermedia Design Method (SHDM) (Schwabe and Moura, 2003; Schwabe et al. 2004) allows the concise specification and implementation of hypermedia (Web) applications. This is achieved based on various models describing information (conceptual), navigation and interface aspects of these applications, and the mapping of these models into running applications, in various environments.

WSDM is audience centric. The audience of the site is categorized into a hierarchy. The requirements, and the navigation structure of the site is based upon this hierarchy. Even though there are distinct conceptual design (task analysis, modeling data and functionality, and the conceptual navigation schema) phase and the implementation

design phase (grouping information onto pages, layout and presentation) and the actual implementation (e.g. java servlets, php code, html, XML, ...)etc. corresponding design models. The emphasis on the audience requirements tilts the WSDM towards the audience as compared to the other stakeholders.

The WebML is a notation to specify complex Web sites at conceptual level. Therefore the concentration on defining these complex structures in the different phases.

The OO-H Method is a generic model, integrated into OO-Method, for the semantic structure of Web interfaces. It defines the abstract interaction model of the user interface, the information which each type of user can access, and the navigation paths from one information view to another and captures the statics of the system. This is one of the most suitable and well defined WSDM which ensures easy transformation to an Object Oriented approach in the design and implementation phases.

The UWE is a development process for Web applications which focuses on systematic design, personalization, and semi-automatic generation. Based on UML and the UML extension mechanism, it defines navigation and presentation models which are supplemented by other UML diagrams and UML modeling elements within an iterative and incremental approach based on the Unified Software Development Process. It captures both dynamics and statics of the system. It also has a smooth transition from the requirements to the design and further implementation phase.

2.4 Model Based Web Engineering Approaches

There exist multiple Model based Web engineering (MBWE) approaches. The vast majority of the methodologies portrayed are all well renowned and generally referenced by the Web application development research group. They speak of the subjects of enthusiasm for Web building during the recent couple of years. Some of the different MBWE methodologies described below are as follows:-

(1)The "WebSA" approach of Beigbeder & Castro

- (2) The "MIDAS HM3" approach of Cacero et al
- (3) The "OOWS" approach of Pastor and Valverde et al
- (4) The "UWE" approach of Koch et al
- (5) The "WebML" and "WebRatio" approach of Ceri et al
- (6) The "WebDSL" approach of Visser 2008 & Groenewegen et al
- (7) "MarTE" of Cadavid
- (8) The HERA approach of Houben et al
- (9) The A-OOH approach of Garrigos et al.

2.4.1. The Web Software Architecture (WebSA) Approach

The "approaches are described in terms of the models they proposed that are common to Web Engineering and its diagramming notation; the consideration of architectural models; and model transformations The Web Software Architecture (WebSA") (Beigbeder & Castro 2004), (Beigbeder 2007) is a model-driven approach that defines an instance of the MDA development process for the Web application domain. It groups the Web application model into three viewpoints: requirements, functional, and architectural viewpoints. Regarding common models to Web engineering, WebSA uses models proposed in two approaches: UWE and OO-H. These models correspond to the requirements and functional viewpoints and consist of a structural model and a navigational model. The structural model is built using a UML class diagram, while the navigational model is built using a UML class diagram and a UML profile. The architectural viewpoint, the main contribution of the approach, includes a logical architecture view and a physical architecture view. It is made up of three models: (1) the subsystem model, which determines the layers of the application, (2) the Web component configuration model, which represents each subsystem in terms of abstract components and abstract connectors, and (3) the web component integration model that allows the designer to determine the low level platform-independent component that make up the final application. The MDA-based development process establishes four phases of the development life cycle: analysis; platform independent design ,

where a platform independent (PIM) model is built; platform specific design, where a platform specific model (PSM) is built; and implementation. In the analysis phase, the Web application specification is divided into functional models and conceptual architecture models. The first ones reflect the functional analysis, and the second ones define the system architecture based on the concept of conceptual architecture (Nowack 2000). In order to get to the platform independent design phase, a PIM-to-PIM transformation is performed providing a set of artifacts in which the conceptual elements of the analysis phase are mapped to concrete elements where the information about functionality and architecture is integrated. These models of the second phase are then transformed into Platform Specific Models (PSM) by means of several PIM-to-PSM transformations that generate the specification of the Web application for a given platform. In the final phase, a PSMto-Code transformation, implemented by means of templates, is performed.

2.4.2. The Midas' Hypertext Modeling Method

The Hypertext Modeling Method of MIDAS (HM3)(Cachero et al 2006) is a methodological framework for agile development of Web information systems based on MDA. It proposes to model the system by specifying Computation Independent Models (CIMs), PIMs, PSMs, and the mapping rules between these models. It proposes to model the system according to three basic aspects: hypertext, content, and behavior. However, it does not propose any strategy for modeling architectural issues. All the models in MIDAS are made using UML as notation, as well as the use of UML profiles. HM3 defines a new UML profile to support the Hypertext modeling, and it uses this profile to specify the metamodels for the user services model, the extended user services model, the extended slices model, and the extended navigation model it proposes. Besides, the approach defines the transformation rules in a declarative style and then maps them to graph rules with the intention of automating these rules with existing facilities to automate graph transformations.

2.4.3. The Pastor et al's Object-Oriented Web Solutions Approach

The Object-Oriented Web Solutions (OOWS) (Pastor et al 2006),(Valverde et al 2007) is based in OO-Method, which is a method that combines formal specifications with conventional object modeling techniques to specify information systems. OOWS integrates navigation and presentation designs into the object-oriented conceptual modeling provided by OO-Method. OOWS allows specifying functional, navigational, and presentational aspects of Web application requirements by using graphics schemes and high abstraction level primitives. Using conceptual schemes as input, a methodology is defined in order to bring the problem space to the solution space by defining matches between conceptual modeling abstractions and final software components. The structural, dynamic, and functional models come from OO-Method; OOWS complements them with a navigational model and a presentation model. The structural model is defined by a class diagram. The dynamic model, a state charts diagrams, describes the different valid sequences of an object life-cycle for each system class, and it also represents interaction between objects by means of sequence diagrams. The functional model captures the semantics of state changes in order to define the effects of a service using a formal specification. Before the creation of the navigation and presentation models, OOWS defines a user diagram to describe the types of users that can interact with the system and the visibility they can have on the attributes and operations of the classes. Once users are identified, a system structured view is created for each class of the structure diagram in terms of attributes, operations, and relationships visibility, which forms the navigation diagram. The presentation model consists of several patterns associated to the primitives of navigational context (navigation classes, links): information paging, order criteria, and information organization. The last one is made of four patterns: record, table, master-detail, and tree. In order to develop the application, OOWS takes as a basis the OO-Method structural and behavioral models and generates the persistence and application layers by using the OlivaNova model transformation engine (Care Technologies

2010). The presentation layer is generated by an OOWS transformation process, and all the artifacts are generated to be deployed on .NET platforms.

2.4.4. UML-Based Web Engineering -UWE Approach

The methodical approach of UWE (UML based Web Engineering) (Koch, 2008) is used for developing web applications using Unified Process model. UWE models are based on the same basic models that are used by UML like "use case diagrams", "activity diagrams", "class diagrams" but with web specific enhancements. The requirements analysis phase is modeled using use case diagrams and activity diagrams. The requirements model is then further refined to content model, presentation model, process model and navigation model. The adaptation requirement is modeled using aspect oriented methodology.

The disjointed handling of Web application concerns as compared to the traditional information system is a fundamental component of this methodology. Accordingly, distinctive models are worked for every perspective i.e the navigation, content or domain, presentation and business logic. The information connected with the Web applications is represented with the help of a conceptual model. The classes of the objects that will be utilized as a part of the Web application are spoken to by instances of <<conceptual class>> which is a subclass of the UML Class. The connections between conceptual classes are represented by associations between these classes. Thereafter, based on the conceptual model the navigation model is designed representing the navigation paths within the web application. A navigation model can be improved by the aftereffects of the business process which manages the business process rationale of a Web application and happens in the process model. The presentation model is utilized to portray the arrangement of the Web pages with respect to the navigation nodes. Adaptivity in UWE gives a reference model to versatile hypermedia frameworks.

As expressed before, MDA methodology of UWE proposes a few models to exhibit to navigation, structure, and presentation. These models find their basis in UML diagrams –such as "sequence diagrams", "state charts" and classes in the utilization of "UML profiles". By and by, it doesn't consider any design model. In this model-driven rendition of UWE (Kraus et al 2007), the content and presentation models are interpreted into Platform independent model utilizing model translation guidelines executed as a part of the "Atlas Transformation Language (ATL)". With a specific end goal to make models implementable, UWE proposes a virtual machine based on top of the controller of the Spring Framework in order to incorporate the navigation within the web application to the business processes. An exhaustive explanation of portrayal of CIMs, PIMs, CIM-to-PIM translations, and PIM-to-PIM translations represented with the help of as ATL translation rules is incorporated into the UWE augmentation given in (Kraus 2007). UWE likewise presents Magic UWE (Busch and Koch, 2009) as a tool support that aides in development and execution of UWE models. UWE be that as it may, doesn't examine the requirements for decision of choices, settling conflicts or determining design decisions, the requirements are represented with the help of Use case graphs and activity diagrams. The nonfunctional requirements are likewise viewed as less important and their part is not sufficiently given significance in this methodology.

2.4.5. Ceri et al's Web Modeling Language and WebRatio

The Web Modeling Language of (Ceri et al, 2002; Ceri et al, 2000) is an illustrative language that is used to depict the content structure and the association as well as the presentation aspect of the information using one or more hypertexts. There are two models which are used for the designing of the web application using WebML namely the "data model" and the "hypertext model". The data model is used to determine the data schema depicting the associations between the website data. The WebML Data Model receives the UML class diagrams or the Entity-Relationship (E-R) primitives. The hypertext model permits portraying how information, assessed beforehand is distributed in the hypertext model.

The hypertext is characterized with the help of pages, links and units, sorted out into construction modules called site "views" and area. A site view is intended to address a particular group of requirements using hypertext. The website view is made up of "areas" which further subdivided into subareas and "pages". Storage units of data conveyed to the user are pages. They are composed of information or data units, which are the basic bits of data extricated from the information catalog by method for inquiries, and presented inside the pages.

One or more entity instances are used to represent the data units. In WebML both the content and the navigation structure are represented in the hypertext model utilizing a predefined set of modeling elements. Views of a site are created by the data units and pages interconnected by links. Associated with a link can be related variables that exchange the info data to the target units. There are a few predefined operation units, for case for actuating content managing operations or exterior Web services like construction, erasure, and upgrade of "entities" and "relations" (Navathe 2004).

WebML additionally gives units to the meaning of session parameters. The entire library of units is open and permits addition of new units in XML. It also contains various information entry units for various types of client inputs. XML Files are used to store all intra unit logical data of link variables. WebML primitives are given a XML-based notation alongwith a visual representation for representing extra properties, not easily expressible graphically.

The XML helps the organization of the same design into numerous rendering arrangements, for example, HTML, "WML" (Hjelm et al, 1998), "VoiceXML" (W3C, 2004) "SALT" (SALTforum.org, 2005) or for mutli-model interactions. WebML of (Ceri et al 2000) is used to denote complex web applications in a conceptual manner. It gives description in the form of views- conceptual, navigational, presentational etc. XML (eXtensible Markup Language) is used to model different point of views. A detailed requirement analysis is not there the authors propose that the same can be

done by using UML diagrams. The non-functional requirements are not captured or analyzed in this methodology.

WebML is supported by a CASE tool, named WebRatio (Ceri et al, 2002) which allows the automatic generation of the application code. WebRatio is a design layer for hypertext and data design that uses XML for intrinsic representation of the models. It likewise permits making "XSL" templates from "XHTML" generalized templates, for defining the presentation layer in light of XHTML. These templates are connected with WebML pages keeping in mind the end goal to characterize a presentation style of the application. A code generator, is used to associate the design layer with a J2EE-based runtime layer to translate XML into executable code. The dependence between the XML units in the data layer are generated by the XSL translations into dynamic XML page templates and files. The language has been stretched out in the new version of WebML, to incorporate new constructs that permit denoting application where Web services can be invoked, the navigation can be driven by procedure models, and page content and navigation can be adjusted to construct context aware Web applications (Brambilla 2008), (Ceri et al 2007).

2.4.6. Visser & Groenewegen et al's Domain Specific Language for web applications

The next work is a DSL for the implementation of dynamic web application called WebDSL (Visser et al 2008), (Groenewegen et al 2008). It consists of sub-languages for the specification of data models and for the definition of pages for viewing and editing objects in the data model. WebDSL uses entity definitions syntax in order to describe the data model of a Web application. It also proposes textual constructs for page definitions specifying a presentation of a Web page and its associated entities. The navigation between pages is defined by means of navigational elements that specify linked pages. WebDSL also proposes higher abstraction level constructs for access control and workflow. The access control is governed by rules that

determine access to the application components. It also allows representing users in order to generate authentication components. The workflow abstraction, based on WebWorkFlow (Hemel et al 2009), defines activities between different actors which result in task pages, task lists, status pages, and navigation between them. The model transformations in WebDSL are implemented using Stratego/XL which is a rewriting system that integrates model-to-model, model-to-code, and code-to-code transformations. The WebDSL generator consists of a set of rules that rewrite extensions of the WebDSL core language to more primitive language constructs by means of a technique of compilation by normalization (Kats et al 2008).

2.4.7. DSL for generating Web application (MarTE)

MarTE (Cadavid et al 2009) uses a DSL to generate web application from UML domain models (Selic 2007). It describes the language's semantics, abstract syntax, and concrete syntax and frames it within a MDSM based transformation tool to generate web applications. The semantics of the DSL is referred as the meaning of web application elements that allow providing a well fit human-computer interaction (Molina 2003) to generated applications. They describe concepts like web forms, web list, master-detail, lookup, defined selection, and primary key, claiming that with these artifacts it is possible to generate fully functional web applications that perform data manipulation operations (Create, Retrieve, Update, Delete, Exists and List) and that are ready for deployment. The abstract syntax is supported by a metamodel called Web Application MetaModel (WAMM). It describes all the global components needed to generate a complete web application in an object oriented programming language. Authors aim at WAMM as a generic web application platform, which any web application could be defined in. WAMM is divided in two parts: a structure part that contains the structure of the domain objects and the relationships between them; and an application part containing the relations between the domain objects and the web user interface. The concrete syntax consists of a UML profile called WebApp Profile. It offers a mechanism to mark the UML domain model in order to provide a good human-computer interaction to the generated application. Such profile is made of

several stereotypes and tagged values that are used to mark classes, attributes, and relationships in the domain model. These stereotypes are: Form, List, Master-Detail, Lookup, Defined Selection, and Primary Key. Each stereotype contains several tagged values which define specific characteristics of the user interface elements derived from the stereotypes through a transformation process. As an example, the Form stereotype, which applies to classes, is used when there is the need of manipulating in a web form the information of a single record based on the marked class. The transformation process for generating web applications is based on ATL and JET, and uses some technologies in order to integrate the DSL into the Eclipse Platform (Eclipse Foundation 2009).

2.4.8. **HERA**

The Hera methodology (Houben et al, 2004) is a model-driven methodology for designing and developing Web Information Systems (WIS). From the gathering of requirements to the maintenance of the operational application, most information system design methodologies distinguish several phases in the design process. The development of a WIS is different in several aspects, and these aspects are the central focus of the Hera project. Like other WIS methodologies, Hera includes a phase in which the hypermedia (Web) navigation is specified. Hera considers navigation in connection to the dataintensive nature of the modern WIS in its presentation generation phase. Before, Hera's integration and data retrieval phase considers how to select and obtain the data from the storage part of the application. This includes transforming the data from these sources into the format (syntax and semantics) used in the application. Also, the handling of the interaction from users is specified: querying, navigation, or application-specific user interaction.

Hera's aim is to facilitate the automatic execution of the design: it should be possible to program the WIS in such a way that it can automatically execute the process specified by the design. Hera uses several models to capture the different design aspects. Because these models are considered Web metadata descriptions that specify

different aspects of a WIS, they chose to use the Web metadata language, i.e. RDFS) (Resource Description Framework), to represent all models and their instances. The choice is also justified by the RDF(S) extensibility and flexibility properties that enabled allow us to extend the language with model specific primitives to achieve the desired power of expression. As RDF(S) doesn't impose a strict data typing mechanism it proved to be very useful in dealing with semistructured (Web) data.

The Semantic Layer indicates the information content within the WIS with the help of Conceptual Model (CM) which characterizes the contents particular to the application domain. It likewise characterizes the integration procedure that accumulates the information from various sources by using the Integration Model (IM). – The Application Layer determines the abstract hypermedia view on the information is in the form of an Application Model (AM) speaking to the navigation structure given in the hypermedia presentation. Essential component in this model is a slice, an important presentation unit that gatherings characteristics from potentially distinctive CM concepts.

Recursively Slices can be characterized as follows: – "The Presentation Layer" that determines the presentation points of interest from the definitions in the "Application Layer" that are required for delivering a presentation for an actual platform, similar to HTML, WML, or SMIL. The "Presentation Model" (PM) fundamental element is the "region", an abstraction of a particular area of the user's browsing device for presenting the data contained in a certain slice. Arrangement of data is associated with regions like a table layout or a vertical/horizontal layout and a certain style (e.g., font size, font colour, link color etc.). Regions can be recursively characterized. It lays accentuation independently on the abstract interface design and the navigational design. It experiences the accompanying stages: design of conceptual classes, design of navigational structure, design of the abstract interface and execution. Its main contribution is represented by separating concerns and presenting a new navigational class diagrams to represent the navigation model.

By representing one model in terms of another, the associations between the distinctive models is presented with clarity putting it in a favorable position for model-driven transformations. Populating the diverse models with information and changing them as indicated by the associations between the models results in a programmed execution of the design while implementing it at the instance level, and eventually to the creation of the hypermedia presentation.

The models in the phase of integration and data retrieval obtain the data from the sources. In reaction to a user query a conceptual model instance is produced with the data for which the application is going to generate a presentation. The models in the phase of presentation generation generate a hypermedia presentation for the retrieved data. The result of the user query, represented by the conceptual model instance, is transformed into a presentation in the specific format of the user's browser (e.g. HTML, WML, or SMIL).

2.4.9. Garrigos' Adaptive Object Oriented Hypermedia (A-OOH) approach

The Unified Process identifies five core workflows that occur during the software development process (Requirements, Analysis, Design, Implementation and Test). The workflows are described separately in the process for clarity but they do in fact run concurrently, interacting and using each other's artifacts. The Adaptive Hypermedia Approach is based on the OO-H approach stated above. It captures the "navigation view" and the "presentation view" of the OO-H approach and further extends the OO-H approach by including another view i.e. the adaptivity or the personalization view.

The AOO-H case considers the following workflows:

1. Requirements: In this stage the requirements for each type of user are gathered, including the personalization (adaptation) requirements. The development process in A-OOH starts with a workflow specifying the functional requirements similar to most we can find in methodologies for the design of other types of software applications.

This activity of requirement gathering for every user type manages the rest of the activities of the development process. The functional requirements considered can be requirements related to the content, to the structure or to the presentation. The adaptation requirements can also relate to the content, structure or presentation of the Website.

2. Analysis and Design: In this stage all the activities related to the analysis and design of the software product are included:

a. Domain Analysis: From the user requirements and the designer knowledge of the domain, the relevant concepts for the application are gathered. The domain model (DM) is defined. It specifies the structure of the Web application domain data. It is expressed in OO-H as an UML compliant class diagram. It encapsulates the structure and functionality required of the relevant concepts of the application and reflects the static part of the system .

b. Domain Design: The domain analysis model has to be refined in consecutive iterations with new helper classes, attribute types, parameters in the methods... etc.

c. Navigation Design: The domain information is the main input for the design navigation activity, where the navigational paths are defined to fulfill the different functional requirements and the organization of that information in abstract pages. In A-OOH instead of using the notation of the NAD presented in OO-H, an extension of UML by means of a UML profile is used. This profile is defined by a set of stereotypes and tagged values to represent the NAD concepts in UML notation. The main advantage of using a standard language (UML) is that the learning curve for the designer is smaller. The NAD is composed of Navigational Nodes, which represent restricted views of the domain concepts, and their relationships indicating the navigation paths the user can follow in the final Website (Navigational Links). Each Node has associated a (owner) Root Concept from the DM attached to it by the notation: "Node:DM.RootConcept".

d. Presentation Design: Once the logic structure of the interface is defined, OO-H allows to specify the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages. During the presentation design, the concepts related with the abstract structure of the site and the specific details of presentations are gathered. The Presentation Model is defined in this activity. It is captured by one or more Design Presentation Diagrams (i.e. DPDs). There should be one DPD for each NAD defined in the system. This diagram enriches the Navigation Action Diagram described in previous section. The DPD uses UML notation. In this section the DPD is presented describing its main elements, then the MOF metamodel in which the DPD is based is described. Finally the UML profile defined to extend the UML concepts in order to represent the DPD elements is presented.

e. Adaptation Design: In parallel to the other sub-phases an adaptation design phase is performed, which allows to specify the adaptation (or personalization) strategies to be performed.

3. Implementation: Implementation is the following workflow considered in A-OOH where the final application is generated.

4. Test: The goal of this workflow is verifying that the implementation work as intended.

As a result of performing the activities of each of the design process phases, in A-OOH we get a set of models, reflecting views of the interface to be generated. A model can evolve along the different phases over time.

2.5 Analysis of Existing approaches wrt Web Engineering

This is an analysis of existing approaches with regard to requirements engineering. These approaches concur on their definitions of a structure model, a navigation model, and a

presentation model. However, besides these, this section contains mention of some special cases in which extra models are defined, or in which the common models are defined perfectly.

One such case is of the MarTE. In this, the tagged values in the UML profile used to mark the structural model represent the navigation and presentation characteristics of a Web application. The user finds this feature of MarTE simple and easy to use, but is restricted to the navigation and presentation characteristics defined by the authors .

This is an analysis of existing methodologies as to necessities building web applications. These methodologies agree on their meanings of a "structure model", a "navigation model", and a "presentation model". Be that as it may, other than these, this section contains notice of some extraordinary cases in which additional models are characterized, or in which the basic models are characterized flawlessly.

One such case is of the "MarTE" which uses the UML Profile to design the three models which are easy to use yet is limited to the navigation and presentation qualities characterized by the creators. The utilization of general models in different methodologies can help the understanding of Web application designs, and devise standard ideas for Web application development. Likewise, this could prompt making of standards for Web application models serialization so as to permit trading models between various tools, as in XMI.

When it comes to defining modeling notation, "Unified Modeling Language (UML)", a widely accepted standard for model software, is used by all approaches, except WebML and WebDSL. This is because a developer, used to UML, can learn a UML-based Web modeling language faster than an unknown language.

However, WebML, which has a proprietary notation (DSL) to define each model, brings with it the graphic constructs or abstractions similar to actual objects, to represent the elements of the Web application domain in a better way. Developers may take more time to learn this, but it is balanced out by its intuitiveness. Furthermore, building tools to support modeling Web

application with these notations becomes easier by using technologies such as Eclipse Modeling Framework (EMF) or Graphic Modeling Framework (GMF) .

Learning and using WebDSL is more difficult as it is not very common and lacks the intuitiveness of the graphic constructs because of its textual form.

With regards to characterizing modeling representation, Unified Modeling Language (UML), a broadly acknowledged standard for developing software, is utilized by all methodologies, due to the fact that for a web engineer it is easier to use a language in which he is proficient in place of a new one.

Nonetheless, WebML, which has graphical notation based proprietary language the DSL for defining the models, is able to represent the components of the web application in a better manner. Engineers might take time to realize this opportunity, yet it is offset by its instinct. Learning and utilizing WebDSL is more troublesome as it is not regular and does not have the instinct of the realistic develops as a result of its textual structure.

The other part of this investigation manages how the component utilized by creators does not isolate the particular of the objective design from the strategy changes. In general the structural design and the execution platform and in the next stage the method translations are defined. Such a methodology results in interdependency between the two stages of model translation and structural design.

In the WebSA methodology there exists a way for translation and platform modularization to support different structural designs. They define two transformations: one in which the structural and the functional designs are merged and the second in which the integration model is converted to different "Platform Specific Models" (PSM).

If a structural model is defined it would lead to the integration of that methodology with many different web methodologies. In general it has been seen that the existing web engineering methodologies support presentation pattern catalogs for Web application development, or

XML technology for the implementation of the navigation and presentation aspects. However they lack the characteristics of Web 2.0 for presentation.

If Web engineering methods have these types of technologies, it could enhance the usability and attractiveness of the applications generated for users.

The main strategy for the ones portrayed in this chapter, which introduces works with respect to the utilization of current Web interface methodology, is WebML. It proposes a methodology (Brambilla et al 2008) to apply the RUX-Method (Linaje et al 2007) presentation model. The drawback of utilizing the RUX-Method is that it further enhances the complexity of the translations. Nevertheless, it does make this conversion possible, therefore the benefits are more than the disadvantages.

Summarizing the problems, it can be stated that

The existing web engineering approaches lack detailed requirements elicitation of functional and non-functional requirements and analysis in the early phase, that lead to an incomplete and inconsistent Software Requirements Specification (SRS), thus need is to provide a structured and systematic approach of requirements engineering in web Application Development to ensure a complete and high quality software requirements specification.

Web Engineering, an emerging new discipline, advocates a process and a systematic approach to development of high quality Web-based systems. In this context Web Design Methodologies appeared, giving solutions both for designers and for users. However, new challenges appeared, like the need of continuous evolution, or the different needs and goals of the users. Adapting the structure and the information content and services for concrete users (or for different user groups) tackles the aforementioned (navigation, comprehension and usability) problems. It has therefore become a vital feature of modern Web applications .

The issues faced in web application development can be solved if proper procedure of requirements engineering is followed as in the case of traditional software development.

There are many requirements engineering approaches that are prevalent based on viewpoints, scenarios, aspects, goals etc. The goal oriented requirements engineering (GORE) approaches have been gaining a lot of popularity because of their thorough analytical capabilities. Our vision is that if GORE in combination with the GRL Metamodel and URN is used for Web Engineering Software Development it would provide a good alternative approach with respect to the approaches stated above that focuses on both the functional and non-functional requirements and results in a high quality output.

2.6 Other Approaches

There are other many Web design methods supporting (in some way) adaptation. We briefly summarize how some of them (Araneus, W2000 and WUML) support adaptation.

Araneus (Atzeni et al, 1998; Mecca et al, 1999) is a methodology tuned to the design and development of Web applications. It includes a development process centered in the authoring of the application and a proprietary notation. In this approach personalization is treated by means of user roles: they consider that there should be a personal page for every user, the users are organized into user groups. Each group will be defined as an actor in the use case diagram and will have associated a different navigational model.

W2000 (Baresi et al, 2001) represents a framework for the design of internet applications by means of combining UML elements with HDM elements (Garzotto et al, 1991; Garzotto and Paolini, 1993). In this approach it is considered that different users have a different view of the project in terms of data and operations. W2000 organizes the design activity in a set of independent tasks. Each activity produces a model that gives a description of some facet of the Web based project. In the activity of visibility design it is specified to whom should be visible certain operations, information structures and navigational paths. Therefore this approach supports personalization only in base of user roles. Different application views are assigned to different user roles, in which is specified what is available for each of them.

WUML (Kappel, 2001), in the context of ubiquitous computing, has personalization based on context. The authors propose an object oriented framework. WUML (Kappel, 2001), in the setting of universal processing, has personalization in light of context as the authors have proposed an Object Oriented System. The framework is made up of four models namely the context, profile, rule and event model. The context and profile model can be extended by the designer, to give a more descriptive narration about the application environment that would lead to trigger on change in the real customization. Context represents current and previous data of the application environment for automatic supervision. Profiles represent the established data which is explicitly given by a designer or a user or is transparently acquired by the system itself. Customization rules, in terms of the event/condition/action (ECA) mechanism, are used for specifying the actual customization. These customization rules can be attached to any Web application modeling using UML as the basic formalism.

They also, give definite data about the earth of an application and trigger the real customization when nature changes. Setting speaks to present and recorded data about the earth of the application which is consequently observed. Profiles cover more steady data which is unequivocally given by an architect or a client (e.g. client inclinations) or is straightforwardly gained by the framework itself (e.g. use measurements). Customization rules, regarding the occasion/condition/activity (ECA) component, are utilized for indicating the real customization. These principles are indicated inside of an UML explanation, utilizing the generalization <<CustomizationRule>>. The detail of such a guideline contains a one of a kind name, a reference to one or more prerequisites, and an ECA triplet. The explanation is appended to those models components being liable to customization. Accordingly the customization guidelines can be connected to any Web application demonstrating utilizing UML as the essential formalism.

For the requirements phase we use the WebGORE framework of (Chawla 2011) and based on the Literature review presented above in this chapter we use the OO-H and the EAOO-H methodology of Webengineering as a base and enhance the AOO-H approach to EAOO-H

approach. Both the WebGRL metamodel OO-H and the AOO-H approaches are explained in detail in the next chapter.

Chapter 3

TheWEBGRL Model and the A-OOH Design Approach

We discussed the existing methodologies for web development in the previous chapter, now we present our reasons for selecting the WebGRL approach for capturing the requirements for the requirements stage and the A-OOH Design model as a base for our conversion strategy to the design stage. In this chapter, a review of existing methodologies the WebGRL and the A-OOH methodologies is done for providing an insight into these existing web requirements and web design approaches respectively. Further, we have presented our reasons for the choice of these approaches for the translation from the requirements stage into the design stage for the creation of "web applications". WebGRL is a "Goal Oriented Requirements Engineering" Approach that captures the functional and nonfunctional requirements comprehensively and also evaluates and resolves conflicts. As such it acts as a good source of input for the transformation from the requirements stage into the design stage. We also discuss the A-OOH and the OO-H design approach for web application development and focus on the AOO-H design approach, used as a base for our requirements to design transformation strategy in detail.

3.1 Introduction

As we probably are aware Web Engineering concentrates on "design notations and visual languages" that can be utilized for the development of organized, usable and viable Web applications. Designing an information based Website adds up to determining its qualities in

terms of different orthogonal reflections. The fundamental orthogonal models for Web application design includes information structure, content organization, navigation, and presentation model. In the previous chapter we talked about the different web engineering techniques. Web Engineering a new discipline proposed a procedure and a structured method of developing Web-based systems. Therefore the Web Design Methodologies provide an answer for both designers and users. Be that as it may, the issue confronted in all these WMDE methodologies of consistent advancement, changing prerequisites according to the clients and the different partners and adjusting the structure, content and services for real worldusers.

Our emphasis is on the need for incorporating a larger role of Requirements Engineering in the Web application Developments. The MDWE approaches discussed in the previous chapter incorporate the requirements gathering phase, however they tend to capture the functional requirements only which results in a product which is not very close to the real world expectations of the different user groups. The absence of detailed requirements elicitation of "functional, non-functional requirements" and analysis in early phase, leads to an Utterance web application development, higher costs and low quality web applications that do not satisfy the non-functional requirements of the stakeholders. Therefore, a structured and systematic approach that incorporates detailed requirements engineering in web Application Development would ensure lesser Utterances, easy evolution, a complete and high quality web application development that focuses both on the requirements and the design phases as in the case of traditional software development.

There are many requirements engineering approaches that are prevalent based on viewpoints, scenarios, aspects, goals etc. The "goal oriented requirements engineering (GORE)" approaches have been gaining a lot of popularity because of their thorough analytical capabilities. GORE in combination with the GRL Metamodel and URN is used for Web Engineering Software Development it would provide a good alternative approach with respect to the approaches stated in the previous chapter that concentrates on both the "functional and non-functional requirements" and results in a high quality output. The GRL Metamodel and

URN have been enhanced to suit the web application development into Web GRL and Web URN by (Chawla& Srivastava et al 2011). Using the WebGRL Metamodel for the Requirements Engineering phase we produce the specific WebGRL Diagrams that capture the goals well of the Web Application development.

Further the specific WebGRL diagrams are seamlessly transformed into the EAOO-H design models based on the Object oriented paradigm into the content, navigation, presentation models. This Web engineering approach being UML Compliant further leads to the conversion of the different models of the EAOO-H approach into a UML Profile which can be easily implemented using any "Object Oriented Programming Language".

3.2 The WebGRL Metamodel

In this section we briefly describe the WebGRL Metamodel which we have chosen as our base for the requirements engineering phase of Web Engineering. The WebGRL Metamodel incorporates the various phases of Requirements Engineering like Requirements Gathering, Requirements Elicitation and Requirements Analysis in detail. The WebGRL metamodel finds its basis in Goal Oriented Requirements Engineering. The goal oriented approaches are very successful for requirements engineering due to their comprehensive requirements analysis. Goals are a better way of acquiring, translating, clarifying the requirements from the stakeholders rather than directly jumping to list of requirements. The stakeholders are more glued to their needs and desires in real world set up. They can imagine easily and interact in terms of goals. Goals allow traceability of rationale, both forward and backward, resolve conflicting view points and help in exploring the design alternatives depending upon the relevance, resource, effectiveness. Goals support reuse, because there are traceable links from requirements to the design phase. There is lot of potential of reuse.

3.2.1. GORE in Web Application Development

There has been lot of work on "goal oriented requirements engineering" in the information system development area. The goal oriented approach is effective because it captures the intentions of the stakeholders thoroughly and various issues like feature

interaction, conflicting requirements, design decisions are resolved in the Requirement phase only. The work is mostly for generic systems and the specific needs of web applications are not catered entirely by these approaches. There are a number of approaches that have been developed for goal oriented analysis and applied to web Systems as well, one of them is GRL and URN.

It is found that "GORE approaches are better at reasoning about the design rationale and functionality of a system. It is found that GORE approaches are very comprehensive and detailed. Goal-driven requirements engineering takes the view that requirements should initially focus on the why and how questions rather than on the question of what needs to be implemented. Traditional analysis and design methods focused on the functionality of the system to be built and its interactions with users. Instead of asking what the system needs to do, goal-driven methods ask why a certain functionality is needed and how it can be implemented. Thus, goal-driven methods give a rationale for system functionality by answering why certain functionality is needed while also tracking different implementation alternatives and the criteria for the selection among these alternatives. Another important requirements engineering technique is scenario based requirements analysis, which is important to understand the working of the system. It is also established that goal and scenario coupling is beneficial for requirements analysis and these approaches work in coherence of each other. There has been a massive amount of work on linking goals and scenarios together (Lamsweerde, 2001), (Lamsweerde, 2004), (Potts 1995), (Rolland, Grosz and Kla, 1999). The obvious reason is that scenarios and goals have complementary characteristics; the former are concrete, narrative, procedural, and leave intended properties implicit; the latter are abstract, declarative, and make intended properties explicit. Scenarios and goals thus complement each other nicely for requirements elicitation and validation".

The requirements analysis phase being the most important phase of a software development, it needs the most attention and time. "User Requirements Notation is the

only goal and scenario based standard", hence it is preferred for using in web requirements engineering. The notation needs to be enhanced for web specific requirements. We need a web specific goal driven requirements engineering approach which categorizes web specific functional and non-functional concerns and develops goal driven notation for web specific requirements. These objectives if met would lead to a better software requirements specification and enable smooth transition to design phase.

3.2.2. URN

URN refers to User Requirements Notation; it has been standardized in ITU standards Z-series in 2012 (ITU-T, 2012). It is a combination of two notations Goal Requirements Language (GRL) and Use Case Maps (UCM). User Requirement notation aims to capture goals and decision rationale that finally shape a system and model dynamic systems where behavior may change at run time (Amyot, 2003). It describes scenarios as first class entities without reference to system sub-components. It can capture the user requirements even when very little design detail is available. It also facilitates seamless transition from requirement specification to a high level design. It provides insights at the requirement level to enable the designers to reason about the feature interaction and performance tradeoffs early in the design process. The two components of URN- GRL and UCM focus on goals and scenarios both. GRL is Goal Requirement Language that focuses on goal analysis, that help in defining the goals including the non-functional requirements, evaluating them, resolving conflicts etc. UCM stands for Use Case Maps that are the visual notation for scenarios. UCM notation employs scenario paths to illustrate causal relationships among responsibilities. The combination of GRL and UCM helps to improve the definition of new goals and satisfy them. UCM provides a behavioral framework for evaluating and making architectural decisions at high level of design. It also supports dynamic refinement at behavior and structure levels.

User Requirements Notation is a goal and scenario based standard for early requirements analysis more suitable for the traditional Information systems development. It has been enhanced for requirements that are unique and distinctive for web applications. The new notation is called web specific User Requirements Notation and helps in requirements analysis of web application. The web specific notation for modelling requirements and web artefacts help the web designer for a smoother transition to the design phase. Different web engineering approaches use different notations for modelling web specific requirements. **Web specific User Requirements Notation (WebURN)** has been proposed which is a web specific goal driven notation that can be used for modelling web specific requirements for goal and scenario based analysis.

3.2.3. **Web Specific User Requirements Notation**

The existing standard 'User Requirements Notation' (URN) has been enhanced to suit the web specific needs. The two parts of URN- 'Goal Requirements Language' (GRL) & 'Use Case Maps' (UCM) have been enhanced suitably for modelling goals and responsibilities to a notation termed as Web specific User Requirements Notation (WebURN). The extended notation for analysing goals is named as WebGRL and the Use case maps have been enhanced for incorporating navigation in the scenarios and termed as WebUCM. The enhanced notation clearly depicts web specific "functional&non-functionalrequirements". The enhancement of User Requirements Notation Metamodel has been done for web specific GRL and UCM models to WebGRL and WebUCM metamodel respectively. A metamodel is a more generic representation of the requirements model. It is a system of meta-concepts, interrelationships between these and constraints. A GRL and UCM metamodel for example have all the elements of a GRL and UCM diagrams respectively. The GRL metamodel has all the intentional elements (goal, task, resource), intentional relations (decomposition, contribution, dependency, means end), the inter-relationships, the cardinality of relations. The enhancement of URN metamodel to WebGRL and WebUCM metamodel and the enrichments are shown in figure 3.1.

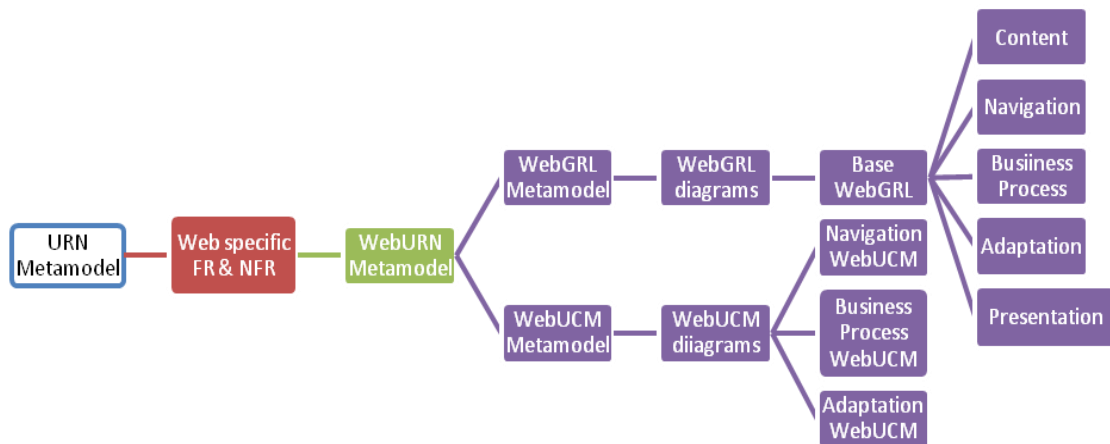


Figure 3.1. Enhancement of URN Metamodel

3.2.4. WebGRL Notation

WebGRL notation is the enhancement of GRL metamodel from the URN standard. The web specific functional and non-functional requirements have been incorporated in the GRL metamodel to create WebGRL notation that addresses the web specific modelling issues precisely. WebGRL notation consists of Intentional Elements and intentional Links like in GRL. The intentional elements are Goal, Softgoal, Task, and Resource. The intentional links are of mainly four types: decomposition links, contribution links, means end links and dependency links. The goals and softgoals have been enhanced from GRL notation to suit the web specific needs. The metamodel for WebGRL notation is shown in figure 3.2. The metamodel consists of the basic elements of a GRL notation: the GRL graph, actor, intentional elements, element links etc. In WebGRL notation, the types of goals and softgoals have been extended to incorporate web specific functional and non-functional requirements respectively. They are represented using enumeration of goal and softgoal types. The enhancements in GRL metamodel for WebGRL notation are listed below:

1. The goals have been given additional attributes: *ftype*, *priority* and *satisfaction*. The attribute '*ftype*' stands for functionality type. The functionality can

be either of the following: - generic, content, navigation, business process, adaptation or presentation. The functionality type is kept as an enumeration which means any one of the above mentioned types can be selected for a goal type. The *priority* of the goal is also added as an attribute that can carry any of 6 values either numeric or qualitative. The *satisfaction* attribute has been added that is used for evaluation of WebGRL diagram. The satisfaction value can either be initialized or computed with the help of algorithms.

2. The softgoals have also been given additional attributes: *sftype*, *priority* and *satisfaction*. The *sftype* refers to the type of softgoal concern and is directly related to non-functional requirements of the category of softgoal concern. The requirements classification proposed in the previous chapter has been used for categorizing the softgoals. The *sftype* attribute can take up any of the following enumerated values: product, project, environment, legal, actors, organizational and functionality softgoal type. The functionality softgoals are further divided into content, navigation, business process, adaptation and presentation softgoal types. These softgoal types would refer to a non-functional requirement about which the softgoal is being depicted. The softgoals also carry priority and satisfaction attributes like goals.

3. The satisfaction values have been enhanced from GRL metamodel to accommodate 7 point scale rather than 5 point scale to give a more detailed estimate of satisfaction of goal or softgoal in the qualitative analysis. The values are FS > PS > WS > UN > WD > PD > FD. The quantitative values of satisfaction range from -100 to +100.

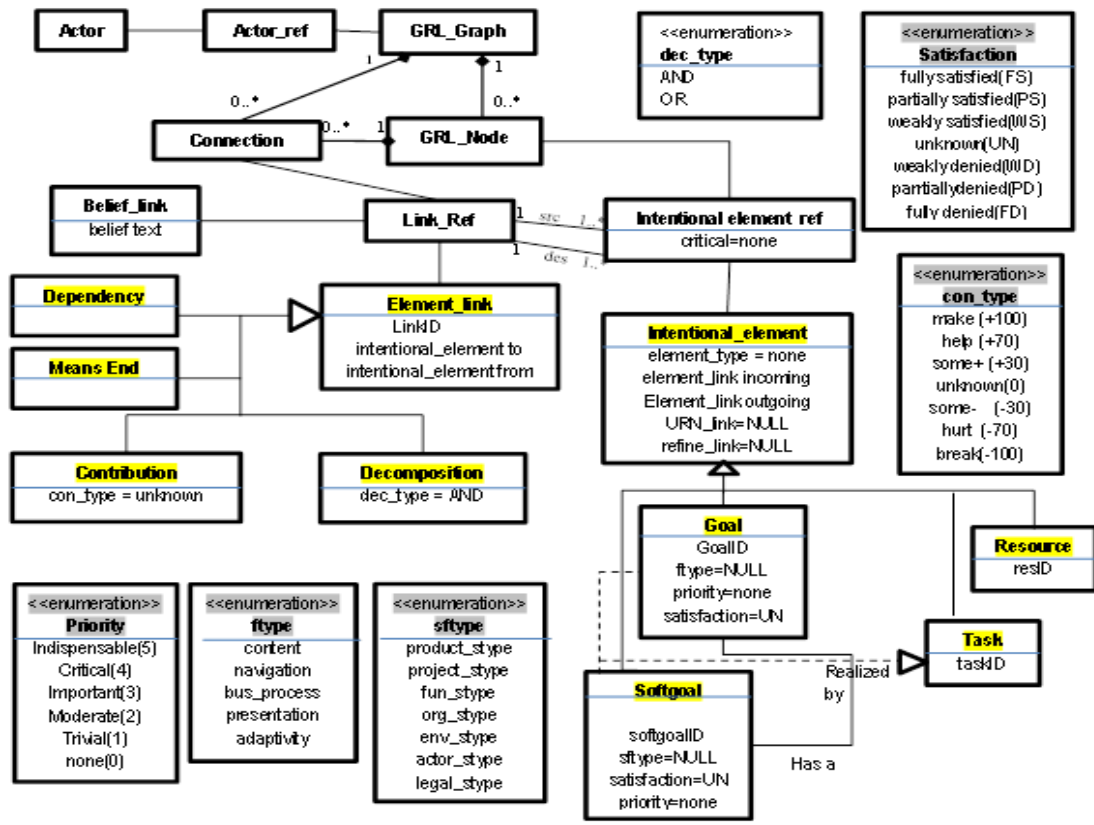


Figure 3.2. WebGRL metamodel

The WebGRL notation has been derived from the WebGRL metamodel wherein the goal and softgoals have been enhanced from being generic to web specific requirements as shown in the metamodel. The "web specific functional and non-functional requirements" classification has been used as the basis for the notation. The tasks & resources are represented in a similar way as in GRL i.e. a Hexagon and Resource as a rectangle respectively. WebGRL notation consists of links that connect two or more intentional elements. They are decomposition links, contribution links, dependency links and means end links. The "WebGRL notation is presented in Table 3.1 as in(Chawla et al 2015)".

Table 3.1. Enhancement of GRL- WebGRL notation

Enhanced WebGRL Notation					
Goal	Notation	Softgoals	Notation	Other concerns	Notation
				Legal	
Content		Content		Product	
Navigation		Navigation		Project Specific	
Business process		Business process		Organizational	
Adaptation		Adaptation		Environmental	
Presentation		Presentation		Actors	

3.2.5. WebUCM Notation

WebUCM is an enhancement of Use Case Maps that form a part of the URN standard. UCM is used to represent scenarios. Use Case Maps consist of a path with start and end point; responsibilities and stubs to represent activities; and other path elements like forks and joins. WebUCM aims to depict the walkthrough of the web application. A web application has distinctive design elements like forms, hyperlinks, images, multimedia etc. The responsibilities in UCM have been enhanced and refined according to web specific needs in the WebUCM diagrams. The metamodel of WebUCM is shown in figure 3.3. The prime difference between UCM metamodel and WebUCM is the types of responsibilities that represent the activity point examples being hyperlink, image, menu etc. The stubs have been categorized into generic and business process stub. Business process stub can be used when navigation UCM has to

refer to a detailed business process or a transaction to be carried out that can be explained with a separate WebUCM diagram.

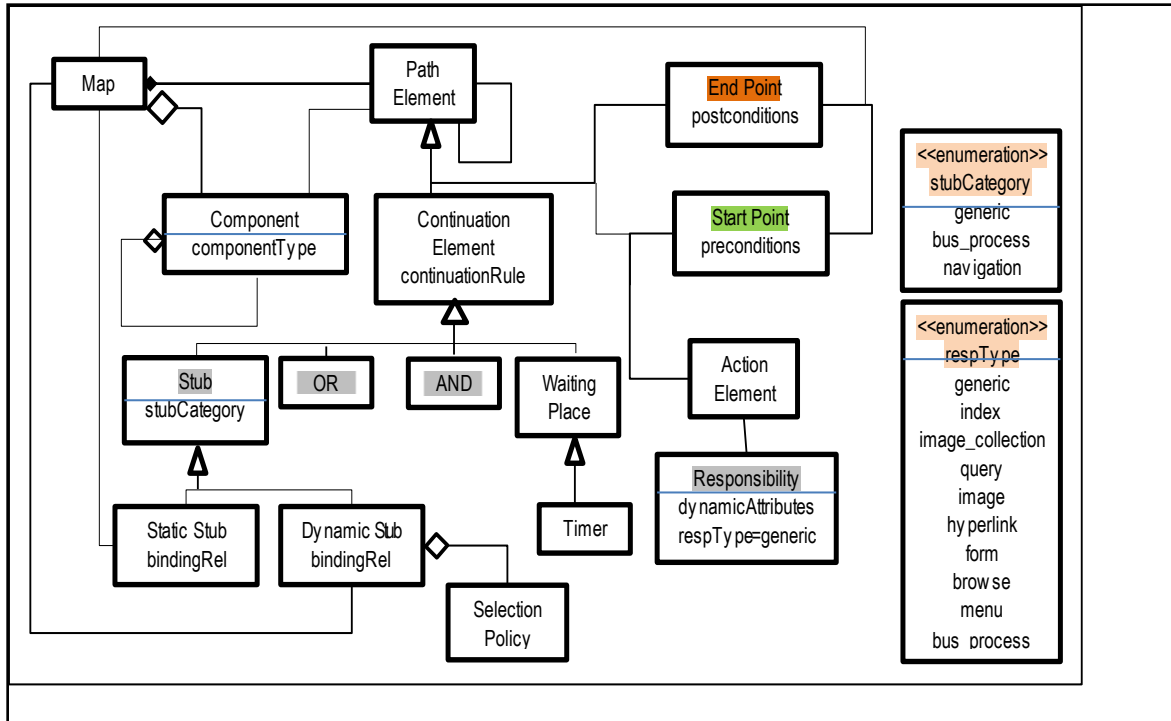


Figure 3.3. WebUCM metamodel









The enhancements in WebUCM metamodel over UCM metamodel are listed below:

1. The WebUCM notation incorporates web design elements and artefacts in the form of responsibilities. The responsibilities are the places of activity in a walkthrough or a scenario. For web application scenario modelling, the responsibilities have been enhanced as follows: generic, index, hyperlink, image, image-collection, query, form, browse and menu. For doing this an attribute of responsibility type has been added for identification of type of responsibility. The responsibility mentioned here are essentially web design artefacts that if identified in early requirements analysis stage help the requirements engineer and the designer for a complete and detailed requirements model.

2. The UCM metamodel consist of stubs that refer to sub-maps. These are like the plugins that can be reused and are detailed enough to be shown as a separate scenario. In web applications, business process requirements can be very detailed and can consist of sub modules. The stubs have been categorized in WebUCM metamodel to generic or business process stubs.

"WebUCM notation has been derived from the WebUCM metamodel as shown in table 3.2 explained in (Chawla et al 2015)".

Table 3.2. WebUCM notation

Element	Notation	Responsibility	Notation
Start Point		Generic	✕
End point		Index	✕
AND Fork		Image collection	✕
AND join		Image	✕
OR fork		Query	✕
OR join		Hyperlink	✕
Generic Stub		Form	✕
Business process Stub		Menu	✕

The advantages of web specific User Requirements Notation are listed below:

- The stakeholders get very clear idea of the system-to-be in early requirements analysis phase.
- The web specific notation helps the requirements engineer and designer to visualize the system and assist in designing the architectural constructs. The WebURN notation links the high level ambiguous business goals to precise requirements.
- "Web specific functional and non-functional requirements" are modeled in an integrated, precise and detailed manner.
- WebURN is a combination of WebGRL and WebUCM notation i.e. "goal and scenario notation". The coupling of "goals and scenarios" brings out better requirements model with in depth understanding of working of the system.
- WebURN notation supports evaluation, reasoning and conflict resolution.

It also helps in modeling the web specific scenarios that depict the walkthrough of the web applications for different goals. This helps in easy architectural design as the notation is very close to the web design elements. In order to move to the next phase of web engineering we need to transform the web specific GRL diagrams into the content, navigation and presentation design models. We use these specific WebGRL diagrams as an input representing the requirements engineering phase of the web engineering process and transform this using our transformation technology into the "content", "navigation" and "presentation" EA00-H design models of this MDWE approach. The A-00H approach is based on the "Objected Oriented Hypermedia approach (00-H) approach" for web engineering therefore both the 00-H and the A00-H approaches are explained in the section below.

3.3 The Object Oriented Hypermedia (00-H) Method

The "00-H (Object-Oriented Hypermedia) method of (Gomez, Cachero, & Pastor, 2001), centers on the product development phase. The 00-H design process of (Gomez et al, 2000; Gomez et al, 2001) is based on the Unified Software Development Process (USDP), also

known as Unified Process or UP (Jacobson, 1999) and on a popular refinement of it, the Rational Unified Process (Kruchten, 2004). The UP is a popular Utterance and incremental software development process framework which specifies how to develop software using UML. The UP is not simply a process, but rather an extensible framework which can and should be customized for specific organizations and projects". The most important characteristics of the UP are described next:

- Use-Case and Risk Driven - The use case based and risk driven methodology utilizes the use cases for the entire span of the software development procedure and identifies and analyses risks in the early phase of the project life cycle.
- Architecture-Centric - The most important static and dynamic aspects of software architecture are identified. The architecture is dependent on the need for users and used for core Use Cases.
- Iterative and Incremental - The development of software products is defined into series of timebox repetitions. Each repetition results in an incremental release of the system with improved functionality in contrast to the previous release. An iteration may include all workflow in process. Use cases define the iterations.

As a result performing activities of each of the design process phases, in OO-H we get a set of models, reflecting views of the interface to be generated. A model can evolve along the different phases over time. The OO-H method extends the typical software methodology (based on UML) with two new models that use web oriented characteristics which are different from the traditional software methodology and rest on the following hypothesis: (1) web application behavior is no longer a simple information update functioning, but it needs to address the issue in a thorough manner.(2) similarly domain behavior and context information in case of a web based application should be tackled with already tested "object oriented software engineering methods, as in any other distributed applications", and (3) in spite of the same logic, the behavior of a web application is very different from standard software development applications.

In comparison to the traditional UML Compliant conceptual modeling approaches the "OO-H Method" design process includes the development of two extra perspectives, namely,(a) a navigation model comprising of the "Navigational Access Diagram (NAD), that defines a navigation view, and (b) a presentation model consisting of an Abstract Presentation Diagram (APD) (Gomez 2001)", that assembles the perception identified with presentation further partitioned into a unique presentation layer and a layout layer. Both the "NAD and the APD catch the interface-related design information with the help of patterns, defined in an Interface Pattern Catalog which is incorporated in the OO-H Method (Gomez 2001)" proposition. The arrangement of notations, procedures and tools that together provide a sound way to deal with the web product modeling stage form a part of the OO-H Method. The "Pattern Catalog, a NAD and an APD bolstered by a CASE tool form the design process that automates the development of web applications with the help of OOH Method (Gomez 2001)".

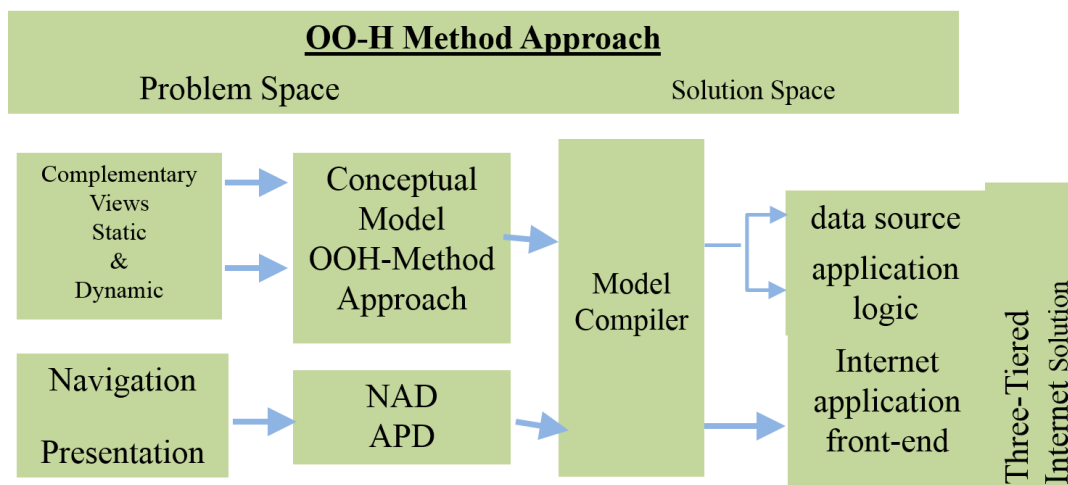


Figure 3.4. The OO-H Overview

The "OO-H method" uses a navigation model, and Use cases. Use cases furnish OO-H with the user centered point of view which is vital for Web based applications, while the "class diagram" is a used to represent the "domain conceptual model" as well as to uncover the interfaces that allow the association with prior business rationale modules, despite their existence or the interior nature like Web administrations, dll's, javaBeans etc. In the execution

stage, "tagged values" connected with the interfaces establish their character and an appropriate access method for invoking the "services and get return values". The procedure is "iterative and incremental", and each repetition results in a refined variant of the application production. Unique components, specific to any sort of Web based application such as "ecommerce applications", are tended to by "frameworks and patterns that are assembled in an OO-H design pattern catalogue".

"Patterns" are useful as they encapsulate the design knowledge and permit reuse, these patterns can be added dynamically. The use of patterns reduces the effort involved in the design process and separates the implementation issues which change dynamically and might confuse the designer. Along these lines, the designer is permitted to concentrate on more unique components, for example, service chains, client profiles, usability, et cetera. Similarly a "presentation model" can be attained in an automated way. The "abstract presentation model" consists of "XML templates" that accumulates the holistic views. The designer has the option of modifying the existing "XML documents or applying patterns from the pattern catalog on this default specification" to get the much needed output. In the sections below description of each of these concepts is presented.

3.4 Design Process

The design process "defines the phases the designer has to cover in order to build a functional interface that fulfills the user requirements (Cachero et al 2002)". As shown in Fig. 3.5, the "OO-H Method design process" is different from a "UML-compliant class diagram" that stores the domain data. Every user type is represented with different "NAD instances". Each instance of NAD is used for the characterization of data, services and the paths to be followed by the user within the system for navigation. After the "NAD" has been built, a default web interface can be created using a set of mapping steps. With a specific end goal to enhance the interface, the "OO-H Method" presents a second diagram, the "APD, based on the concept of templates (Atzeni et al 1998), (Fernandez et al 1998), (Fraternali 1998), (Mecca et al 1999) and whose default structure is likewise extracted from the NAD". So as to help the designer to

improve the structure while keeping up its value, the "Pattern Catalog" has a group of solution patterns based on the problems encountered till that time. This methodology encourages the reprocess of the existing design knowledge for the development of different interface modules as well as application interfaces.

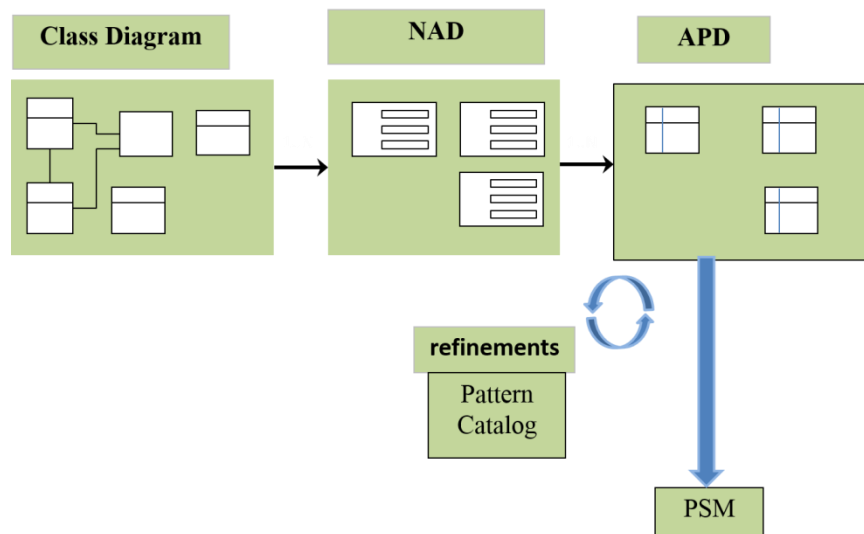


Figure 3.5. OOH Method - The design process

After refining the "APD", we can produce both static and dynamic front end for the web based application for the desired domain. This autonomy from final execution issues is an absolute necessity where new languages for web access are always developing. Both the "NAD and the APD" make broad utilization of a " Pattern Catalog", which is described in the section below.

3.4.1. Pattern Catalog

"The OO-H Method Pattern Catalog provides a Hypermedia Interface Pattern Language. This language can be seen as a partially ordered collection of related patterns that work together in the context of Hypermedia Interfaces (Rossi et al 1997). They help to capture the abstract interaction model between the user and the application. The pattern style defined in (Buschmann et al 1996) is chosen for the specification of the patterns. This style, largely based on the Alexandrian style (Alexander 1979) is best suited for abstract patterns with several possible ways of

implementation, which are common characteristics of those found in hypermedia environments. The implementation specification is enriched in order to allow them to drive the evolution of the diagrams: each pattern has one of its possible implementations set to 'default', in order to help the designer to obtain the desirable interface features with minimum effort. Also, each implementation has an associated Transformation Rule, expressed in OCL-like (Warmer & Kleppe 1998) syntax. These transformation rules can be instantiated and applied at page level and schema level, and they drive the changes on the diagram where the patterns are applied. These changes include the creation of new pages, the redirection of links among pages or the creation of new page dependencies, among others. Furthermore, in order to improve its usability, the catalog contains a Sample Usage section that points to working web examples where it has been successfully applied. One of the main features of our catalog is that it is, as the rest of the model, 'User-Centered', that is, the granularity at which the patterns are described is such as to provide the designer with additional mechanisms to fulfill the user requirements. The patterns included in the catalog offer alternative solutions to well-known hypermedia problems, considered from the user point of view. Furthermore, its use allows the designer to choose the most suitable among a set of alternative implementations, depending on the target application domain and on the designer's experience. The main categories are:

1. Information Patterns: they provide the user with useful application context information. One of the most relevant examples is the 'Location Pattern' whose definition will be presented later.
2. Interaction Patterns: its use involves user-interface communication issues regarding both functionality and navigation .
3. User Schema Evolution Patterns: they cover structural advanced features. As an example we can cite the Multiview Pattern, which allows the designer to present two or more simultaneous views of the information .

In OO-H Method, different sets of patterns can be applied to the two different diagrams of the modeling process, which are (1) the NAD diagram and (2) the APD diagram. At the NAD level we can apply patterns related to user information selection and navigation behavior. At the APD level, on the contrary, we can apply patterns that provide the interface with non-mandatory additional features, aimed at improving its usability. Next, we are going to introduce the main structural and semantic aspects of the NAD diagram, whose definition constitutes the first step in the interface modeling process .

3.4.2. Navigational Access Diagram

As stated above, the "navigation model is captured by means of one or more Navigational Access Diagram. The designer should construct as many NAD's as different perspective of the organization are required, and he should provide at least a different NAD for each user-type who is allowed to navigate through the system. This diagram is based on four eccentric of constructs: (1) Navigational division, (2) Navigational Quarry, (3) Navigational tie-in and (4) Ingathering. Also, when defining the navigation structure, the designer must take into account some orthogonal aspects such as the desired navigation demeanour, the target population selection, the parliamentary procedure in which objects should be navigated or the cardinality of the memory access. These characteristic are captured by means of different kinds of navigation patterns and filters associated with links and collections".

- Navigational Classes (NC): they are improved version of function family whose measurement and strategy profile has been confined by client access grant and navigation requirements. An example is the distinction between the three instances of measurement: "V-Attributes (Visible Attributes)", "R-Attributes (Referenced Attributes)", which are shown after a client interest and "H-Attributes (Hidden Attributes)", just showed when a thorough framework populace perspective is required. An example of the same is shown in the figure below:
V-Attribute: Like the Paper List class has a solitary trait, the

"Paper nameList" attribute, the explanation behind this observability is that the paper name uniquely determines every give and take subject inside the interface, thus it should dependably be available. R-Attribute: The "Paper" class has, other than its recognizing property "Questions", a field named "Answers", displayed as a R-Attribute. This infers the client needs to unequivocally tap on its reference to peruse the substance. They assemble the components of the model that gather up in the scope of every client navigational need. Again taking a look at our illustration (see Fig. 3.6), we can see a simple singular navigational necessity: "Participation inPaperlist" that has both the data and the services required for the user for exploring the system.

- Navigational Links (NL): are the navigation routes that a client or a user can follow within the web application. They have associated with them both a "Navigation Pattern and a group of Navigation Filters", which combine to give the additional data for the development of the "navigation model". There are four types of links : "I-Links (Internal Links)" define the navigation route inside the limits of a given "NT"; "T-Links (Traversal Links)" are defined between "navigation classes" between different "NT"; "R-Links (Demand Links)" point at the beginning navigation point inside every "NT"; and "S-Links (Service Links)" demonstrate the services accessible to a user related to that "NAD".

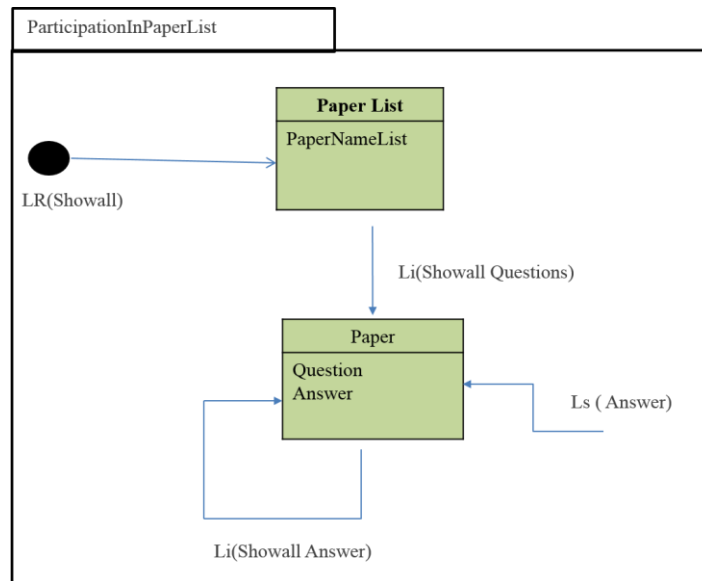


Figure 3.6. Sample NAD Diagram for Participation inPaperlist

3.4.3. Abstract Presentation Diagram

The Abstract Presentation diagram (APD) is used for the specification of the display and organization of the page structure of the web. "In order to separate the different expression that contribute to the final port appearance and behavior, we have defined five eccentric of templates, expressed as XML (eXtensible Markup Language) written documents (XML), (McGrath 1998)". In order to define the tags and the structure of the document a "Document case Definition (DTD)" is associated with a template type, whose description is given in the following subdivision.

Template Case The five template types defined in OO-H Method acting are:

- 1." tStruct: its example define the entropy that has to appear in the precise page".
2. "tStyle:its instances define characteristic such as physical locating of elements, typography or colouration palette".
- 3." tForm: its instances define the data items required from the user in order to interact with the system of rules".

4. "tFunction: its instances capture node functionality. They are based on the DOM (Document Object Model) specification(XML),which aims at giving a platform and nomenclature interface to the structure and content of XML and Hypertext Markup nomenclature text file".An attempt has also been made to standardize an "interface to these objects for navigation and document processing". The functions which have been described in the template are platform independant and can be easily mapped to a target language.

5." tWindow: its instances define a set of simultaneous views available to the user".

The basic structure of the template can be obtained from the data stored in the NAD alongwith the defaults used for the undefined values. This default production of the template will be explained next. Default "APD OO-H Method defines a set of steps for the mapping from the different elements contained in the NAD diagram into the equivalent elements in the APD". The resultant "APD" is a very simple interface that is further improved to be added to the final application however it can be used as a prototype for validation of requirements.

The main mapping steps that drive this default APD generation are described below.

1. "V-Attributes, I-Links, T-Links and R-Links: they appear as elements of the tStruct page" (Cachero 2002).

2." C-Collections, S-Collections: they are static trees, and are defined by means of a tStruct single abstract page that contains a tree-like structure made up of link elements pointing to other tStruct elements. An S-Collection also might imply a new tForm template to be created, if the filter associated with it involves any user-dependent value (Cachero 2002)".

3. "S-Links: they may generate a previous tForm abstract page that contains the parameters to be introduced by the user for all the commands involved in the service. They must contain a link element that points to a command". If the command returns

anything, another "tStruct page will be generated with the structure defined by this return value either an object or a set of objects (Cachero 2002)".

4."R-Attributes: they cause a new tStruct abstract page to appear on the diagram, and a new link element pointing to it on the original one. The new template will contain all the R-Attributes to be shown plus a meaningful reference to the original object (Cachero 2002)".

5."NAD Navigation Patterns: all indexes, guided tours, indexed guided tours and showall patterns are represented in the APD with the help of a set of link elements defined on a single tStruct page".

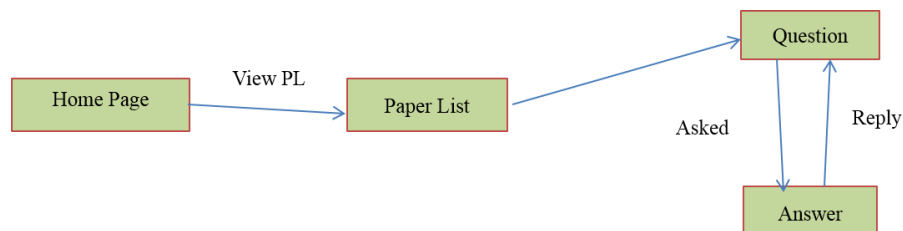


Figure 3.7. Sample APD for Paper list

6. All objects to be demonstrated are encased in collections, and there are default client defined information validations for the fundamental attribute or the main attribute . These approvals are actualized by method with the help of call components with "onChange" event functions connected with "validation" works that rely on upon the type of attribute. After having refined the APD, the model compiler is given as an input the device-independent modeled interface features, that has the target-environment knowledge for creating an operational web interface. In the section below we briefly give the description of the AOO-H method of (Garrigos et al 2010) based on OO-H methodology.

3.5 The Overview of AOO-H Method

The "Enhanced AOO-H method (EAOO-H)" is an enhancement of the "AOO-H (Adaptive Object Oriented Hypermedia)" approach of (Garrigos et al 2010) and the OO-H approach of (Gómez et al, 2000; Gómez et al, 2001) to support the modelling of GOREWebapplication. It retains the basic features of the AOO-H, but changes and extends the existing A-OOH approach to support the WebGRL modeling to capture the specific WebGRL in entire especially the "non-functional requirements". The development process in both OO-H and AOO-H starts with a workflow specifying the functional requirements similar to most we can find in methodologies for the design of other types of software applications. This activity of requirement gathering for every user type manages the rest of the activities of the development process. The functional requirements considered can be requirements related to the content, to the structure or to the presentation. The adaptation (personalization) requirements can also relate to the content, structure or presentation of the Website. "Non-functional requirements" like the system properties, performance etc. are not taken into account.

Both the "AOO-H approach of (Garrigos et al 2010) and the OO-H (Object-Oriented Hypermedia) method (Gómez, Cachero, & Pastor, 2001)" focus on the product development phase as stated above. The USDP consists of repetitive cycles over the lifetime of a system. There are four phases in a cycle namely: "Inception phase, Elaboration phase, Construction phase and the Transition phase (Gómez, Cachero, & Pastor, 2001)". Each cycle ends with a product output, there are also releases within a cycle. A-OOH design procedure considers an extra phase:

Maintenance. This extension was considered in the UWE approach (Koch, 2001) due to the importance of the maintenance in the life cycle of a hypermedia application. Maintenance implies changes in the content, in the hypermedia structure and in the layout. The five stages in a complete round are the following:

- Inception stage - In this stage product vision is achieved by core development. This phase ascertains the product practicability and sets the limits of the project scope. The milestone for this stage is defining the life-cycle objectives.
- Elaboration Phase - In the elaboration stage most of the "Use Cases" are determined in detail and the architecture of the system is designed. This stage concentrates on the "Do-Ability" of the task. We distinguish critical risk factors and set up a timeframe, staff and cost summary for the complete project. The milestone for this stage is defining the life-cycle architecture.
- Construction Phase – This stage focuses on the development of the system. A fully operative and efficient product is elaborated. This stage is finished when all the use cases are implemented. The milestone of this phase is the initial operational capability.
- Transition Phase - This stage is frequently started with a beta release of the application. Different exercises incorporate isbuild action, and defect identification and redressal. The turning point of the move stage is the release of the product. The main objective of this stage is to guarantee that the requirements stated meet the general approval of the partners. A beta release of the application usually forms the beginning of this. Different exercises incorporate website grounding, and defect identification and redressal. The turning point of the move stage is the release of the product
- Maintenance Phase – This stage begins after the completion and delivery of the system to the final users, and lasts during the life of the system. An adaptive Website needs continuously refresh and update, the designer can add new adaptivityrequirements, update the content of the Website, adapt the Website to new technologies, etc.

The "Unified Process identifies five core workflows that occur during the software development process namely Requirements, Analysis, Design, Implementation and Testing". The workflows are repeated based on the feedback and are not sequential. "Each workflow is defined by a set of activities that are performed by a set of workers with the goal to produce some artifacts (it can be a model, a model element or a document), which are measurable results of the workflow(Visser et al 2008)". Even though the workflows are described

separately, however they run simultaneously, and are interrelated and utilize artifacts amongst themselves. The AOO-H case considers the following workflows:

1. Requirements: In this stage the requirements for each type of user are gathered, including the personalization (adaptation) requirements.

2. Analysis and Design: In this stage all the activities related to the analysis and design of the software product are included:

a. Domain Analysis: From the user requirements and the designer knowledge of the domain, the relevant concepts for the application are gathered.

b. Domain Design: The domain analysis model has to be refined in consecutive Utterances with new helper classes, attribute types, parameters in the methods... etc.

c. Navigation Design: The domain information is the main input for the design navigation activity, where the navigational paths are defined to fulfil the different functional requirements and the organization of that information in abstract pages.

d. Presentation Design: Once the logic structure of the interface is defined, OO-H allows to specify the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages.

e. Adaptation Design: In parallel to the other sub-phases an adaptation design phase is performed, which allows to specify the adaptation (or personalization) strategies to be performed.

3. Implementation: Implementation is the following workflow considered in A-OOH where the final application is generated.

4. Test: This process verifies that the system execution is as intended.

As a result of performing the activities of each of the design process phases, in A-OOH we get a set of models, reflecting views of the interface to be generated. A model can evolve

along the different phases over time. In the sections below we present these workflows in more detail, as well as the diagrams used in each workflow.

3.5.1. Requirements

The development process in A-OOH starts with a workflow specifying the functional requirements as well as the personalization requirements, similar to most we can find in methodologies for the design of other types of software applications. This activity of requirement gathering for every user type manages the remaining activities of the development process. The functional requirements considered can be requirements related to the content, to the structure or to the presentation. The adaptation requirements can also relate to the content, structure or presentation of the Website. Non-functional requirements are not considered in AOO-H. Therefore, we enhance the AOO-H method to the EAOO-H method to incorporate non-functional requirements.

3.5.2. Domain Analysis and Design

As the result of the domain analysis and domain design phases the domain model (DM) is defined by (Garrigos 2009). It specifies the structure of the Web application domain data. It is expressed in AOO-H as an UML compliant class diagram. It encapsulates the structure and functionality required of the relevant concepts of the application and reflects the static part of the system (Garrigos 2009). In this moment the issues related to the navigation, presentation, workflow or interaction are not taken into account yet, which simplifies and gives a bigger reuse capability to the model.

The main modeling elements of a class diagram are the classes (with their attributes and operations) and their relationships (association, aggregation, composition and generalization) . Its construction follows well-known object oriented modeling techniques:

- Identify the classes.
- Determine the association relationships among them.
- Define hierarchical relationships.
- Specify the most relevant attributes and operations.
-

Identify the cardinalities. • Include as classes the actors identified in the requirements workflow, if it is needed to store some kind of information.

3.5.3. Navigation Design

Once the Domain Model (DM) has been specified the navigation structure is defined by means of the Navigation Access Diagram (i.e. NAD) as in (Garrigos 2009). This diagram enriches the DM with navigation and interaction features.

In A-OOH instead of using the notation of the NAD presented in OO-H, an extension of UML by means of a UML profile is used. This profile is defined by a set of stereotypes and tagged values to represent the NAD concepts in UML notation. The main advantage of using a standard language (UML) is that the learning curve for the designer is smaller. In this section first we present the elements of the Navigational Model to understand the presented approach and then we present the UML MOF Metamodel and UML profile.

3.5.3.1. Navigation Access Diagram

A "Navigational Model (NM)" presents the navigation view on the data captured by the "DM". In "A-OOH the NM" is captured by one or more "Navigation Access Diagrams (i.e. NADs)". The designer is required to build a number of NADs corresponding to the different views of the system with minimum one for each identified user. The NAD is composed of Navigational Nodes and Navigation links where the navigation nodes present a constrained view of the domain concepts and the navigation links present relationships of the domain model which act as navigation routes for the user. Each Node is coupled with an owner Root Concept from the DM attached to it by the notation: "Node:DM.RootConcept". Three different types of navigational Nodes are as follows:

- Navigational Classes (NC): These represent the "domain classes of the domain model containing operations and attributes" with access authorization depending on the navigational needs. There are three types of attributes as in

the case of domain classes namely the "V-attributes, R-attributes and H-attributes" It is represented by a "UML class with the stereotype <<NavigationalClass>>".

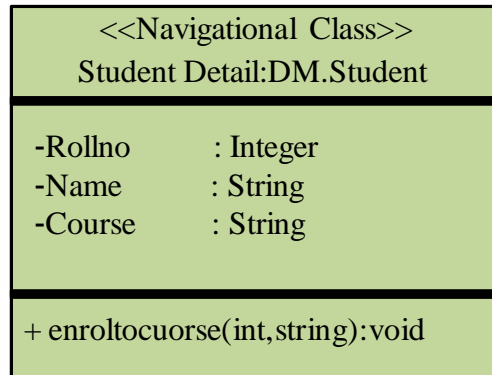


Figure 3.8. Navigational Class

- Navigational Targets (NT): They represent a set of model elements which work towards the realization of the navigational needs of the user. NTs are represented by UML packages with the stereotype <<NavigationalTarget>>.

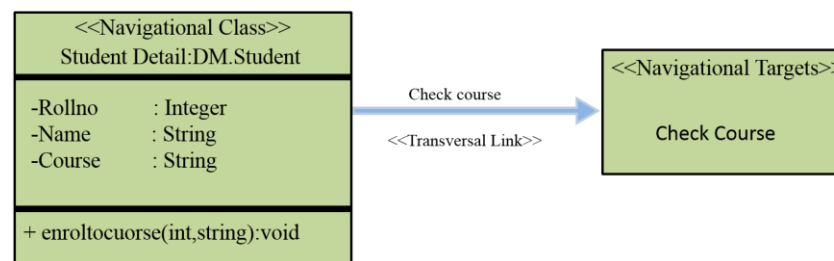


Figure 3.9. Navigational Target

- Collections: They represent the probable hierarchical composition defined in Navigational Classes or Navigational Targets. They present an alternate method of data access. The most frequently used collection type is the C-collection or the Classifier collection, an example of the same is the concept of menu that uses generalization for representing a set of navigation links. Another important collection is the S-collection (Selector collection) with

which we can represent a selection mechanism (for example the concept of form). It is represented by a "UML class with the stereotypes <<NavigationalC-Collection>> or <<NavigationalS-Collection>>".

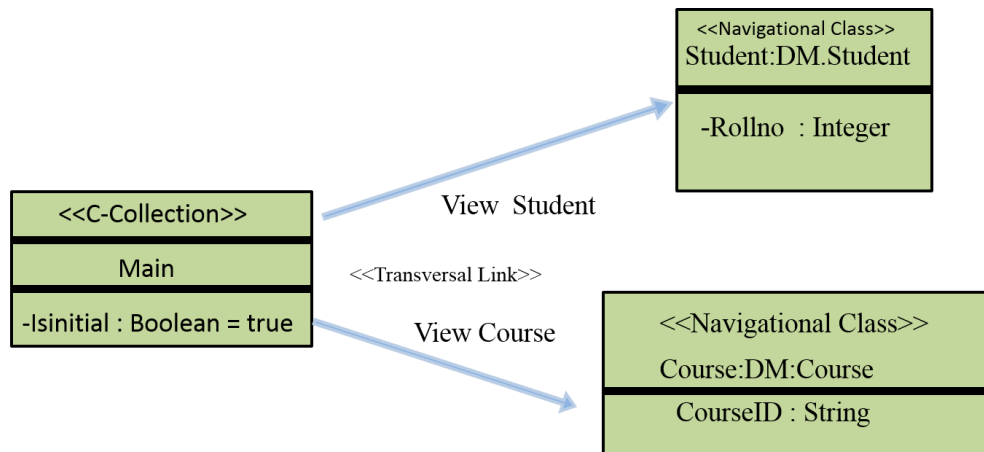


Figure 3.10. C-Collection

Every node has a boolean tagged value “IsInitial”, if its value is set to true indicates that the node is the entry point of the NAD or of a NT. By default its value is set to false.

Navigational Links (NL): They denote the navigational routes that a user follows within the web based application. A-OOH has two main types of links:

- T-Links (Transversal Links): They denote a link for moving between two types of navigation nodes which can be a combination of navigational classes, collections or navigational target. The navigation performed is done to show information through the user interface, without modifying the business logic. This type of links is represented by the "stereotype <<TransversalLink>>".

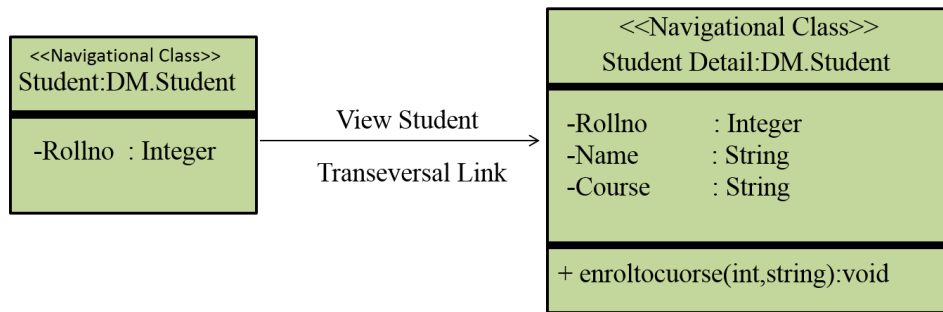


Figure 3.11. Transversal Link

- S-Links (Service Links): They denote the triggering of an operation that results in the modification of the business logic leading to navigation node where the execution result is presented. It is established when a service of the navigational class is activated. This type of links is represented by the stereotype <<ServiceLink>> and has associated the name of the invoked service.

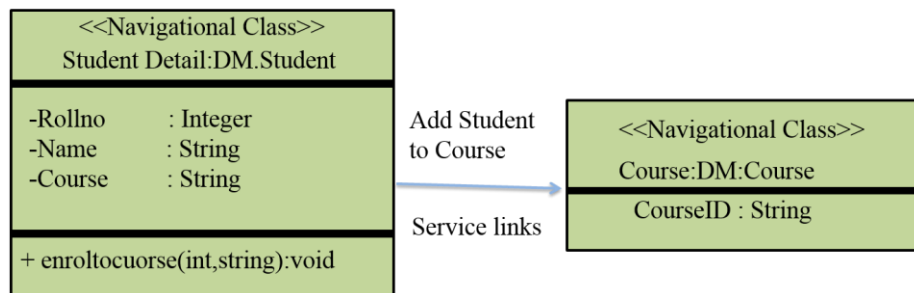


Figure 3.12. Service Link

Design Patterns: A number of factors need to be considered while defining the structure of navigation like the statics and dynamics associated with navigation, selection and priority of access for object navigation and the access cardinality. These features are collected together and form the basis for different types of navigation patterns and the related filters to "links and collections". Links can have associated one of the three navigational patterns: Guided Tour, Showall and Index. Navigational patterns are characterized by two properties: indexed (yes/no and, if the value is true, number of elements in each page, to allow the pagination with

indexes) and navigation (yes/no, and in case of “yes”, number of elements by page, to decrease the size of guided tours). Next we briefly describe the design navigation patterns supported by A-OOH:

- Index pattern: it causes an indexed navigation of a set of objects which are presented in an index page. In this pattern internal navigation amongst the target objects doesn't exist, the only way to access the objects of the collection is by means of the index.
- Guided tour pattern: it causes a non-indexed navigation in which internal navigation exists and the objects of the navigation are presented one by one.
- ShowAll pattern: it implies navigation without indexing and without internal navigation, all the objects are shown in the same abstract page. This is the default pattern in a Navigation Class.

In the NAD they are represented with an stereotype with the name of the design pattern (i.e <<Index>>,<<GuidedTour>>) excepting for the Showall pattern which is set by default in a Navigation Class. An example of the representation of the Index pattern is shown in the Figure 3-13.

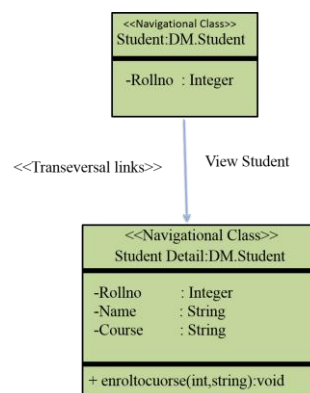


Figure 3.13 Index Design Pattern

The stereotypes for representing the different concepts described in this section are defined in the UML profile presented in Section 3.3.2.3. This UML profile extends the concepts defined in the NAD metamodel presented next.

3.5.3.2. The NAD Metamodel

In Figure 3-14 the MOF metamodel defined for the NAD is shown. The main elements of the NAD metamodel are the "Navigational Node" and the "Navigational Link".

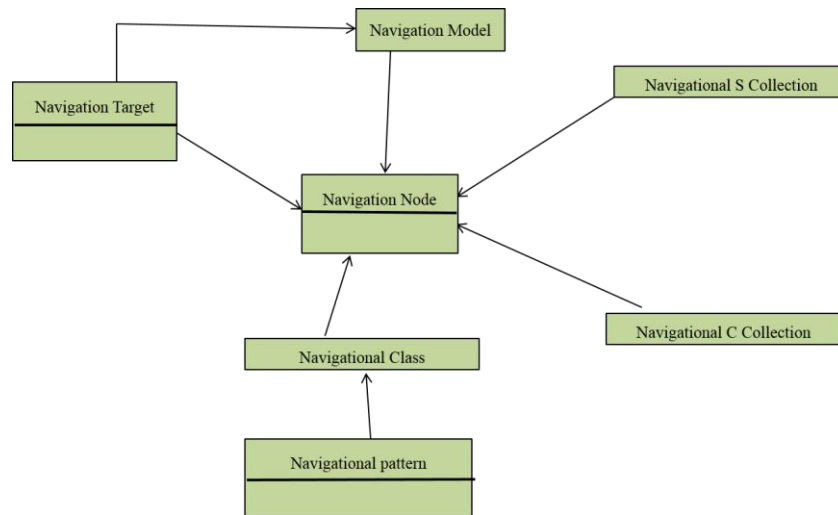


Figure 3.14. NAD Metamodel

In the metaclass representing the Navigational Link we can see that several attributes are defined (see Figure 3-15):

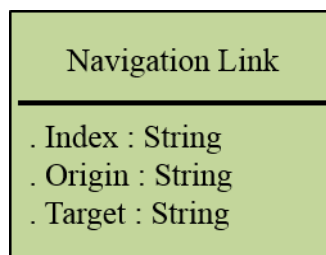


Figure 3.15 Navigational Link metaclass

- indexedBy -This attribute indicates, in the case of an indexed navigation, the attribute by which the node instances are indexed.
- filterOrigin, filterTarget -To define navigation constraints, OO-H utilizes the "Object Constraint Language (OCL)" of (Warmer & Kleppe, 1998), a subset of the standard UML that permits programming engineers to compose

constraints over object models increasing the model precision. OO-H relates these constraints to the navigation model filters described on links. Filters can be defined over the origin or the target Navigational Class restricting the set of objects. In the NAD they are represented in the target or source role value, as it is shown in the Figure 3-16. The link Product Detail must be active only for the products with a price greater than 50.

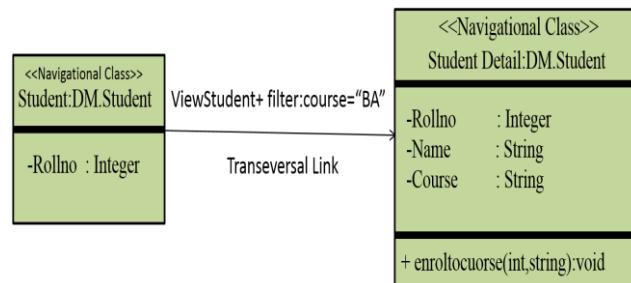


Figure 3.16. Filter defined in a Navigational Link

- User interaction (manual/ automatic):- The user at times prefers to get the information without clicking on a link (i.e. the navigation is activated by the system). This feature is encapsulated in the attribute of user interactions in the metamodel that will take the value automatically instead of the manual traditional value (by default). The automatic property is represented by a composition as it is shown in the Figure 3.17, meanwhile the manual property is represented by means of an association

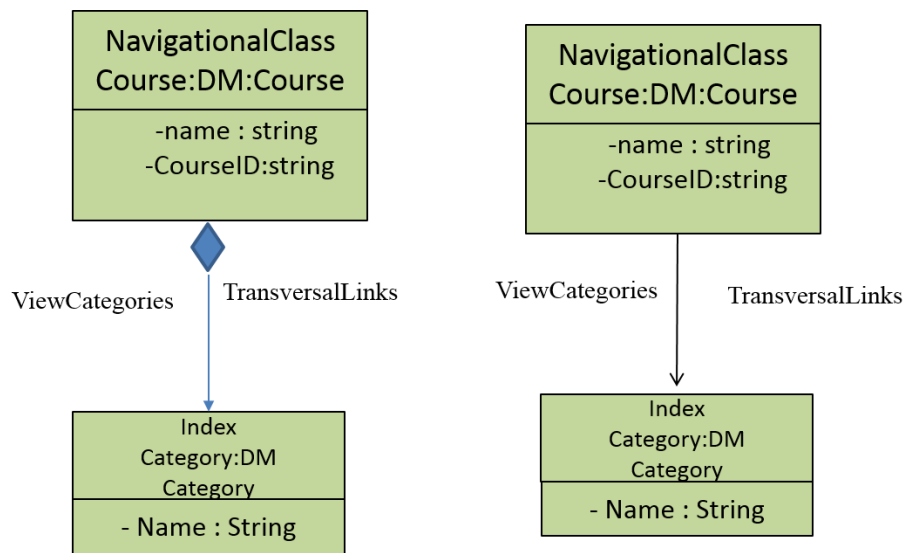


Figure 3.17. Automatic and Manual Links

- scope Of Application This attribute is also defined in the NavigationalLink metaclass. The Navigational Links can have additional information to specify the application field /scope of application (simple / multiple / universal). In the case of the links that transfer information about the origin of the navigation three different cases can be given:
 - o Simple: the object from which the navigation departs is a single object. This is the default value in the Transversal Links.
 - o Multiple: the origin page shows a set of objects from which the user selects a subset at execution time.
 - o Universal: the link has to transfer all the information referent to the whole set of objects in the origin page. This is the default value in the Service Links associated with a method of a class. The attribute scope Of Application can then have one of these three values (i.e. simple, multiple, universal).

In the case of the Navigational Node metaclass, we can see the four types of Navigational Nodes defined in Figure 3-18. The Navigational Target metaclass has the attribute url which contains the url of the target page in the case of being an external page (i.e. not defined in the system).

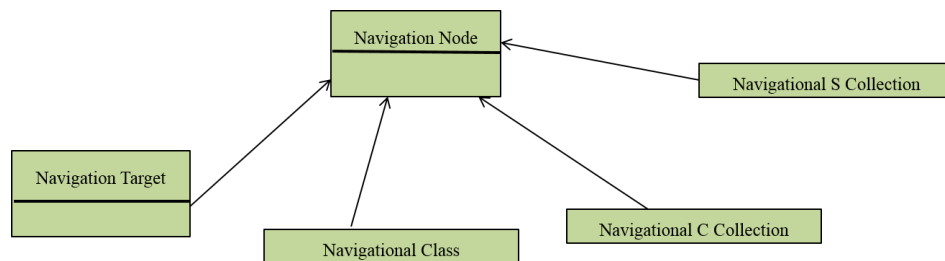


Figure 3.18 Navigational Node metaclass

Personalization can be added to the NAD by means of personalization rules. These rules can be attached either to Navigational Nodes or to Navigational Links. The rules attached are stored in the PersonalizationRules tagged value. In the section below we describe the Presentation Design for the A-OOH method.

3.6 Presentation Design

During the presentation design, the concepts related with the abstract structure of the site and the specific details of presentations are gathered. The Presentation Model is defined in this activity. It is captured by one or more Design Presentation Diagrams (i.e. DPDs). There should be one DPD for each NAD defined in the system. This diagram enriches the Navigation Action Diagram described in previous section. The "DPD" uses UML notation. In this section the DPD is presented describing its main elements, then the MOF metamodel in which the DPD is based is described. Finally the UML profile defined to extend the UML concepts in order to represent the DPD elements is presented.

3.6.1. Design Presentation Diagram

The DPD is used to depict the presentation of the navigation elements to the user. The DPD is used with an objectives to provide the page structure of the Website, by gathering Navigational Nodes into Presentation Pages. The Presentation pages are depicted at the conceptual level which can be converted into concrete pages at the implementation level. The designer also has the option of adding static pages directly on the DPD. This forms the level 0 of the DPD. The second objective is to portray the format and style of every page of the interface by detailing the abstract pages defined

in level 0. This forms the level 1 of the DPD. In this level the designer decides the positioning and the components to be represented on this page. As in level 0 the designer also has the option of modification of the structure of the page as well as addition of the static pages directly on the DPD.

The definition of the refined DPD leads to a web application front-end, either static or dynamic, using the desired implementation technology of the target platform such as HTML, WML, ASP's, PHP's etc. The main modeling elements of the DPD are described next. We classify them into level zero or level one depending on where they can be defined.

As mentioned above the main element defined in the level zero of the DPD is the structure of the presentation page formed by gathering the Navigational Nodes into presentation pages to which new static pages can also be added.

- Presentation page:-The presentation page is an abstract page containing the components exhibited on this page related to a Presentation model. It is represented as a UML Package with the stereotype <<PresentationPage>> (see Figure 3-19).

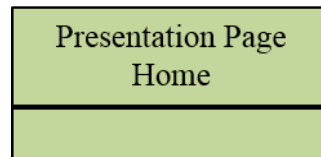


Figure 3.19. Presentation Page

- Page Chunk: - The "Page Chunk" is basically a piece of an abstract page related to a "presentation model" where the components that appear in this section are characterized. This section can be reused in the distinctive pages that form the Web application and therefore do not repeat these common parts of more than one page. It is represented as a UML Package with the stereotype <<PageChunk>> (see Figure 3-20).

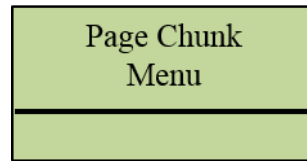


Figure 3.20. Page Chunk

- Window This element represents the window of the Web browser that is being used. This element has been defined to give the chance to the modelled Web application of using several windows of the browser during the use of the application. The notation is a UML class with the stereotype <<Window>>.
- FrameSet, Frame The FrameSet element represents a set of frames in which the browser window can be divided. In this way we provide the designer the possibility of using a frame based design for his application. Notation is a UML class with the stereotype <<FrameSet>>. The Frame element represents a frame that is part of a FrameSet. Notation is a UML class with the stereotype <<Frame>>. FrameSet and Frame elements are associated with the <<includeFrame>> association.

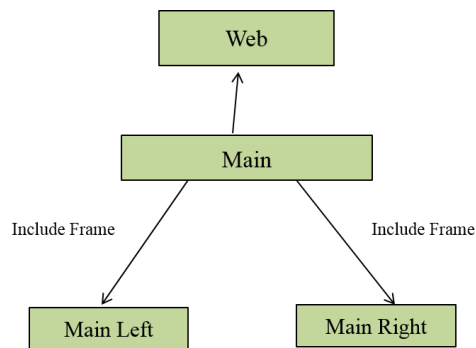


Figure 3.21. Window, FrameSet and Frame elements

Main elements defined in the level one of the DPD: The main objective of this level is to give a detailed description of the "style and layout" of each page defined in level one. Static elements can be added to the page (e.g. static text).

- Layout Instead of using frames, layout can be defined for defining the disposition of the objective visualized in the Web page. These layout can be of three different

eccentric: BorderLayout, BorderLayout and GridBagLayout; each of them provides a concrete distribution for its cubicle. The layout are translated, in last case, to Hypertext markup language tables. The different type of layouts considered can be nested.

- **BoxLayout:-** The BoxLayout places the boxes on top of each other. It spreads them in a row as per preference of the designer.

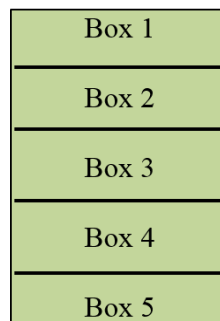


Figure3.22. BoxLayout

- **BorderLayout:-** As the Figure 3-23 shows, a BorderLayout has five areas in which we can place the different elements of a Web page.

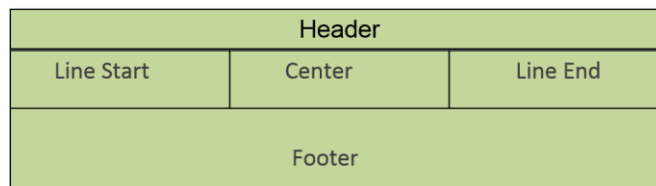


Figure 3.23. BorderLayout

- **GridBagLayout:-** GridBagLayout is the most adaptable layout. In case of a gridbaglayout, the components of the page are placed in rows and columns with no restriction on the span. They may cover multiple rows and columns where each component has rows and columns of different heights and widths. The figure below shows a gridbaglayout enveloping three rows and three columns. The fourth button is spread across all the columns, whereas the fifth button envelopes only the two right columns.

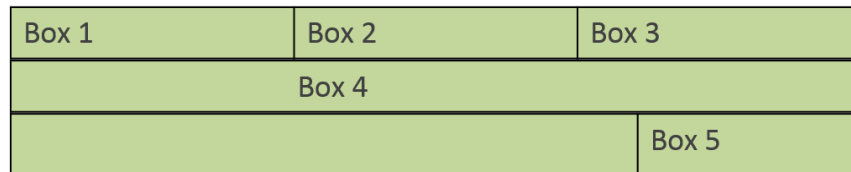


Figure 3.24. GridBagLayout

- Cell- A cell represents a component which is part of a layout. The cells act as containers for the interface components.
- InterfaceComponent An interface component presents the visualization of an object (text, images, links). The interface components considered are:
 - o Composed Interface Component *f* Anchor is a composed interface component which contains a simple interface component.
 - o Simple Interface Component
 The simple interface components considered are: *f* Image, Text , FormElement (e.g. InputButton, InputSubmit...etc).

The stereotypes used in the DPD notation in our approach are defined in the UML profile presented by us in Chapter 7. This UML profile extends the concepts defined in the DPD metamodel presented in the section below.

3.6.2. DPD Metamodel

The DPD MOF metamodel has been defined to formalize the elements of the DPD and the existing associations among them. A metamodel defines the language to express the model. The main elements of a DPD are the Presentation Nodes and the relationships among them (i.e. Presentation Links).

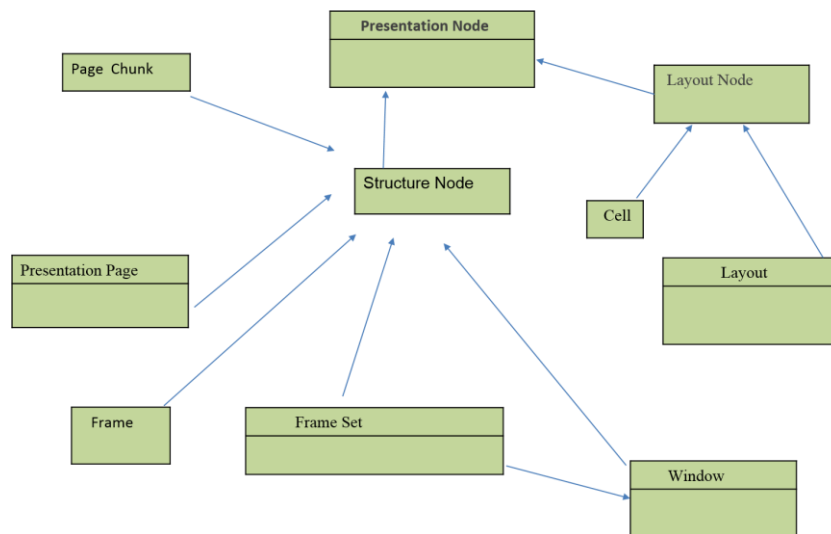


Figure 3.25. Presentation Node Subtypes

The different types of Structure Nodes considered are: Presentation Page, Window, Frame, Frameset and Page Chunk (level zero of the DPD). The types of Layout Nodes considered are: Layout and Cell (level 1 of the DPD).

In a Presentation Model five types of Presentation Links can be defined (see Figure 3-26):

- Navigates: This relationship can be defined between Presentation Page elements.
- Contains: This relationship is defined between Presentation Page elements and Page Chunk elements to indicate a presentation chunk is contained (and shown) in one or several Presentation pages.
- Presents: This relationship is characterized between Frame elements and Presentation Page elements.
- IncludeFrame: This relationship is represented between FrameSet and Frame elements.
- IncludeCell; This relationship is defined between Layout and Cell elements.

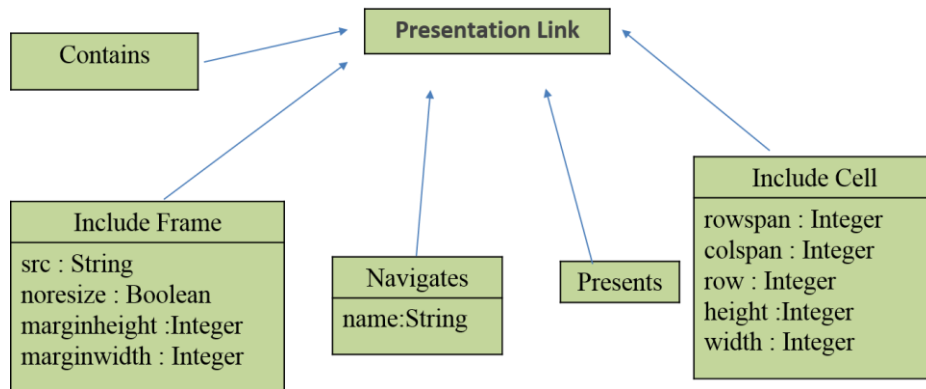


Figure 3.26. Presentation Link types

The Layout elements have associated information like the alignment, width, height...They can have associated one or more cells. In Figure 3-27 the metaclasses representing the different layout types are shown. In this figure we can also see that the association “IncludeCell” (defined between Layout and Cell elements) contains information about the positioning and size of the elements inside the cells.

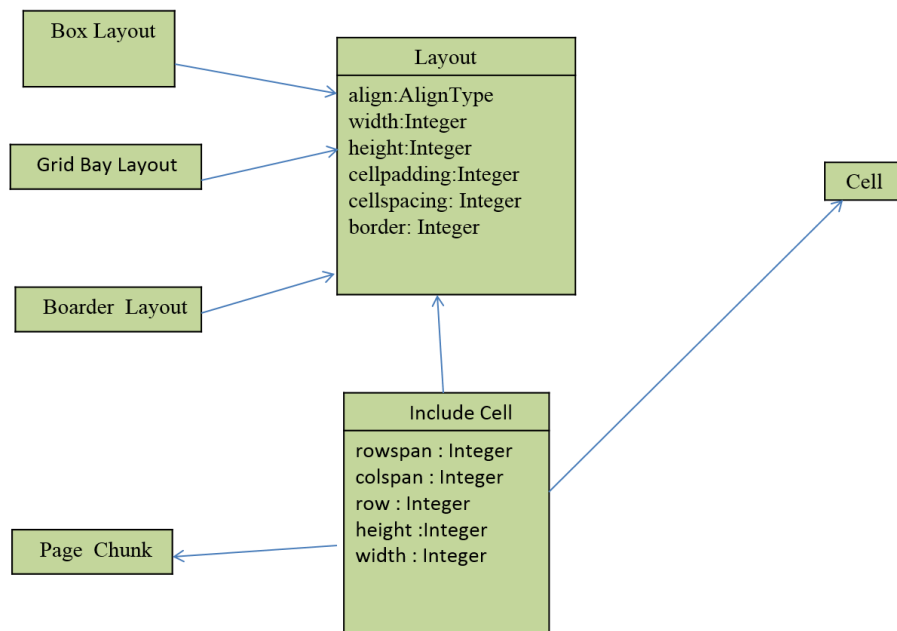


Figure 3.27 Layout and IncludeCell metaclasses

Using the specific WebGRL diagrams as input we transform the specific WebGRL diagram input into the "EAOO-H design models" which can be further implemented based on their

UML Profile. These updates and extensions are explained in the chapters 4, 5 and 6. The same as "AOO-H and OO-H, EAOO-H" is also user-motivated approach based on the "object oriented paradigm and partially based on standards (XML, UML, OCL...). The enhanced EAOO-H approach provides the designer the semantics and notation needed for the development of adaptive Web-based interfaces and their connection with pre-existing application module (Srivastava 2014)".

It is important to note that "OO-H, A-OOH and EA-OOH "is not a methodology that supports the full development cycle of the application. However, the presented approach is focused on the authoring of the application: requirements, design, implementation and maintenance and evaluation. Our reason for choosing the AOO-H approach for the transformation from the requirements phase to the design phase are:-

- The "AOO-H approach is based on an Object Oriented paradigm" hence it is "UML Compliant".
- Being UML Compliant it can be easily transformed into its UML Profile and easily implemented using any object oriented programming language.
- Also, the WebGRL approach being goals and scenarios based the AOO-H approach offers a compatible platform for transition from the requirements to the design phase without any loss of information.
- The WebGRL specific diagrams need to extend the existing AOO-H approach to incorporate all the features of the WebGRL model for seamless transformation from requirements to design model which can be easily done with the existing AOO-H approach.
- A seamless transformation also provides easy traceability from the design to the requirements phase.

Using this as the basis we have chosen the AOO-H approach for the design phase of web engineering. In the next chapter 4, we present the Enhancement of the design process of

the AOO-H method i.e the EAOO-H DomainModel, EAOO-H Navigation Model and the EAOO-H Presentation Model respectively. After Enhancing the AOO-H model we use the web specific WebGRL diagrams which incorporate both "functional and non-functional requirements" as input for the transformation process from the requirements phase to the design phase of Web engineering. Using the specific WebGRL diagrams as a base we present a seamless transformation process for the transformation from the Content WebGRL diagram , Navigation WebGRL and Presentation WebGRL of the Requirements phase to the "Domain Model, Navigation Model and the Presentation Model" of the EAOO-H based design phases in Chapters 5,6 and 7 respectively.

Chapter 4

Transformation Approach for Content WebGRL to EA00-H Domain Model

As emphasized earlier we need to capture and model the different facets of web applications both in the requirements phase and the design phase. In order to ensure a comprehensive description of the requirements we capture them in different web specific models namely the content, navigation and the presentation models. These web specific models form a very important part of structured modelling of web applications. Therefore we need to ensure that the transformation of each of these web specific models is taken care of while defining the conversion strategy from the requirements stage to the design stage. In the previous chapter we have described the WebGRL and the A-OOH design approach used as a base for our transformation strategy. In this chapter we present the EA00-H model and the transformation strategy for translation from the WebGRL content model to the EA00-H domain model.

4.1 INTRODUCTION

Web applications have become integral to our lives however, their development lacks a systematic development approach. The different phases of the Software Development Life Cycle like the Requirements engineering and the design phases have been either skipped or done in a hurry as such we get a web application which is not well structured as in the case of conventional information systems. As such both the Requirements analysis and the design

based on such an approach is important. "Web application has multiple stakeholders, and the size and purpose of the applications is large (Chawla et al 2010)"and in order to ensure proper implementation of the goals, they need to be dealt with comprehensively and thereafter seamlessly transformed in to the design phase in order to get a good quality with high stakeholder satisfaction level web application. In the previous chapters we have discussed the various existing web engineering approaches and the issues faced by them. As a result we have come to the conclusion that a good design approach is one that is based on a more detailed requirements engineering approach for web applications as in traditional information systems, the quality of output of web engineering approaches would increase.

"In order to support the requirements engineering activities involved in web application development we have the GOREWEB framework: It is a Goal Oriented Requirements Engineering for Web applications (Chawla et al 2010)". In the GOREWEB model the User Requirements Notation (URN) has been extended to Web URN for web based application requirements. Therefore, using the GORE approach of Requirements Engineering "the WebGRL Metamodel of GOREWEB framework for Web applications offers goal oriented requirement analysis of web applications (Chawla & Srivastava 2010)". The same has been described in Chapter 3. Further we have chosen the AOO-H design model of (Garrigos 2010) as a base and enhanced it for smooth transformation from the requirements stage to the design stage for web engineering or web application development.

Presently, the Web engineering approaches spotlight is on the web application development directly therefore the requirements phase is completely overlooked and the goals of the users are comprehended by the designer. This results in creating misunderstanding and confusion for the user, not the "real" user requirements. This causes both the development and implementation problems for designers and increases the initial project budget. A good requirements approach for the web application would "give the designer the ability to take decisions from the very beginning of the development phase. These decisions could affect the design of the website for meeting the real goal requirements and preferences of user type (Chawla et al 2014)".

In our approach we use the GOREWEBpart forGoal Oriented Requirements tomodel statics and the dynamics especially where behavior can change in time of web application development systems. "The use of goal driven requirements analysis helps in capturing stakeholders' goals and the requirement clarification and the conflicts between requirements can be evaluated and the best design option selected to suit the requirements (Srivastava& Chawla 2009)". Also there are specific WebGRL diagrams developed from the base WebGRL diagram in the requirements phase which helps in giving a detailed picture from different perspectives related to the web applications in the design phase. Further, we enhance the domain, navigation and presentation design models of AOO-H into the Enhanced Adaptive Object Oriented Hypermedia (EAOO-H) design models. Thereafter, we define the transformation strategy for the specific WebGRL models to derive the domain, navigation and presentation design models of EAOO-H that are supported by a UML profile.

In all these approaches explained in Chapter 2 they lack the seamless transition in the requirements phase to the design phase due emphasis has not been given to the requirements phase, especially the non-functional requirements. The design models from the Base WebGRL diagram in the requirements phase helps in giving a detailed picture from different perspectives related to the web applications in the design phase. Moreover the transition of the design models to a UML compliant UML Profile helps in platform independent development of the product.

In this chapter we begin by giving the details of the Enhanced A-OOH approach for the domain model and discuss the extensions to the "A-OOH approach" required to meet the smooth transition of content WebGRL. We further present the transformation rules to derive the EA-OOH content model and its UML Profile using the transformation algorithm for the transformation of WebGRL content model as input into the EAOO-H domain model. Thereafter, we present the UML Profile for the transformation of the WebGRL Content model into EAOO-H based Domain Model and in the last section we illustrate the transformation strategy with the help of the Case Study of Online Bookstore which we will be using for

illustration of the transformation strategy for the navigation and presentation models in Chapters 5 and Chapter 6.

4.2 The EA00-H model

The basics of the EA00-H approach is based on the "A-OOH (*Adaptive Object Oriented Hypermedia method*) of(Garrigos et al 2010)" it follows the same workflow as the A-OOH as it is an expansion of the "OO-H modeling method (Gomez, Cachero & Pastor 2001)". This approach makes use of "UML-profiles" therefore all the models are "UML-compliant".

EA00-H model considers the following workflows:

1. **Requirements:** In this stage the requirements for each type of user are gathered, including the nonfunctional requirements. This phase is done using the GOREWEB framework. The WebGRL and the webucm are developed to capture the "functional and non-functional requirements" of the web application under development. This forms the "base WebGRL diagram". In the second phase in order to get a more holistic view of the requirements with respect to the web the base WebGRL diagram has been extended into the "specific WebGRL diagrams" namely the content, navigation, presentation and other WebGRL diagrams. We use these diagrams as the input to our design phase. The basis of using these specific WebGRL diagrams is their ability to capture both the "functional and non-functional requirements". While most of the existing approaches focus on the functional requirement only. Also the WebGRL diagram generated by the GOREWEB framework generates a set of specific WebGRL diagrams that solve the conflict resolutions and provide alternative solutions.

2. **Analysis and Design:** In this stage all the activities related to the analysis and design of the software product are included:

- a. Domain Analysis:* The user requirements captured by the content WebGRL diagram are Goal, Softgoal, Task, and Resource and the intentional links are namely the decomposition links, contribution links, means end links and dependency links, the relevant concepts for the application are gathered besides the designer knowledge of the domain,.

b. Domain Design: The content WebGRL model of domain analysis has to be refined in consecutive iterations with new goals, softgoals etc. stated above into the final WebGRL content diagram with the help of the GOREWEB framework. This refined content WebGRL diagram is used in the model transformation strategy for smooth transition from the content WebGRL diagram into the EAOO-H domain model explained below.

c. Navigation Design: The domain information along with the navigation WebGRL diagram is the main input for the design navigation activity, which is done using the transformation strategy for the transition of navigation WebGRL to navigation model. The "navigation model is captured by means of multiple Navigation Access Diagrams representing the different navigational views where the navigational paths are defined to fulfill both the different functional and the non-functional requirements as in (Garrigos 2009)" and the organization of that information in abstract pages.

d. Presentation Design: Once the logic structure of the interface is defined, the presentation WebGRL diagram is used as an input to the transformation process for transforming the Presentation diagrams that allows specifying the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages.

3. Implementation: Implementation is the final workflow considered in the EAOO-H where the final application is generated.

4. Test: The goal of this workflow is verifying that the implementation work as intended.

The Steps 1 and 2 are done using WebGRL diagrams and the domain analysis results in the Web Specific GRL diagrams. With the help of EAOO-H we map the refined analysis models to their respective design models. The considered Web engineering approach EAOO-H is expressed as a UML-compliant class diagram. We have extend this profile in order to adapt EAOO-H design models for the Web specific domain terminology. This has led to the enhancement and development of our own UML profile later described in chapter 7.

4.2.1. Discussion on the Extension of the A-OOH approach to EA00-H approach

The reasons and the steps taken to enhance the A-OOH approach into the EA00-H approach are:-

- The A-OOH approach is requirement based whereas our work is goal oriented therefore in place of task we extend the goal as well as softgoals to the stereotypes defined in the A-OOH approach into navigation and presentation stereotypes.

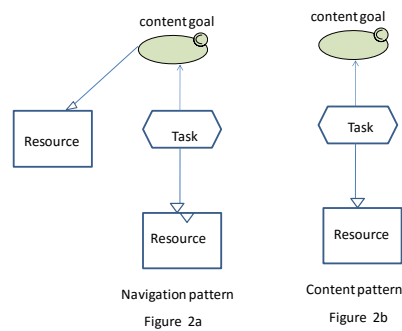


Figure4.1.Showing Content Goals, tasks and Resources

- The A-OOH model uses the adaptive OOH approach to define the domain and the navigation model from the use case diagrams using domain analysis. We differ here by gathering the requirements and using gore approach to develop the specific WebGRL diagrams using grl approach for web applications.
- We do the requirements specification and analysis using the WebGRL approach and use EA00-H only for the design phase to generate the domain model, navigation model and the presentation model based on the specific WebGRLmodel.
- We extend the UML profile of the A-OOH approach by defining new stereotypes into the UML Profile for the EA00-H approach to support these design models.

Once the requirements have been defined using the Web GRL diagram a transformation approach can be used to obtain the design models for the website.

1. "Domain model (DM)": The transformation strategy uses a set of rule to transform these web specific diagrams into the Domain model (DM), for defining the structure of the domain data.
2. "Navigation model (NM)": NM is used to represent the "structure and behavior of the navigation view over the domain data".
3. "Presentation model (PM)": PM represents the "layout of the generated hypermedia presentation".

All of which are expressed using a UML compliant UML profile for the WebGRL. Before explaining of the derivations, A-OOH DM has been explained for the reader to easily follow the derivation of them.

4.3 The EA00-H "Domain model (DM)"

As the result of the domain analysis and domain design phases the domain model (DM) is defined. "It specifies the structure of the Web application domain data. The EA00-H DM is a UML compliant class diagram. It give main points of the structure and functionality required of the relevant concepts of the application and reflects the static part of the system. The main models elements of a class diagram are the classes with their attributes, conditions and operations and their relationships" (Srivastava 2014). The EA00-H domain model is also UML compliant class diagram. Though the main concepts are same as in A-OOH domain model however, it varies slightly in the description and the semantics of the domain model concepts. In the next section we present the domain model concepts and their description.

4.3.1. The EA00-H Domain Model Concepts

The EA00-H domain model essentially consists of the basic modeling elements of a class diagram to capture the structure and the functionality of the Content WebGRL diagram and ensure a smooth transformation from the WebGRL content Model to the EA00-H domain model.

The Domain Model is composed of the following concepts:-

- 1 Domain Classes (DC): The Domain model consists of the domain class embodying the main objects, interactions in the application and the classes to be customized with an extensible program-code-format for object creation, giving preliminary values to state i.e. part variables and executions of behavior using member functions or methods. The class stores the information content such as fields, attributes etc. The behavior of class is characterized utilizing methods. Operations are basically the subroutines for operating on objects or classes. These operations may lead to the alteration of the state of an object or basically give style of retrieving it. It is represented by a UML class with the stereotype << Domain Class>>. The class diagram is the main building block of domain modeling. The domainclasses in a domain model consists of three sections shown in the Figure of domain class below.

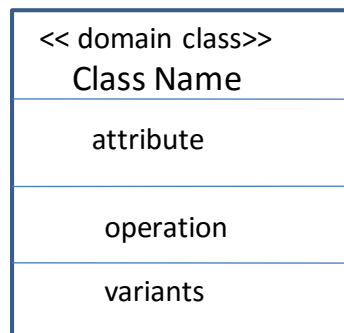


Figure 4.2.Domain Class

A class is made up of three parts as represented in the diagram above by a box with three segments:

- a) The uppermost segment is used to represent the name of the class. The name of the class has a capitalized first letter and is printed in bold and centered.
- b) The central segment represents the information content of the class and contains the attributes of the class. The attributes are written in lowercase with left-alignment.

c) The lowermost segment is used to represent the executable methods with the class and are written in lowercase with left-alignment.

An example of a domain class is shown below for the class name "Author". The author domain class is composed of attributes author name, author email and operation to display author list. A boolean condition can be imposed on the domain class for it to be valid.

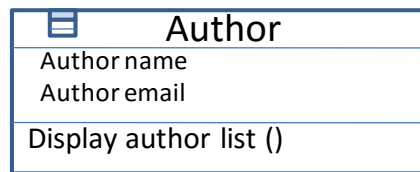


Figure 4.3. Example of the Author Domain Class

In the domain model design of the system, we pinpoint the classes and group them together for determining the static relations between those objects. As we proceed with the detailed modeling, the classes within the domain model are further divided into subclasses an example of the same is shown below. As shown in the figure below there are two domain classes namely author and category and the line shows the link or the association between them.

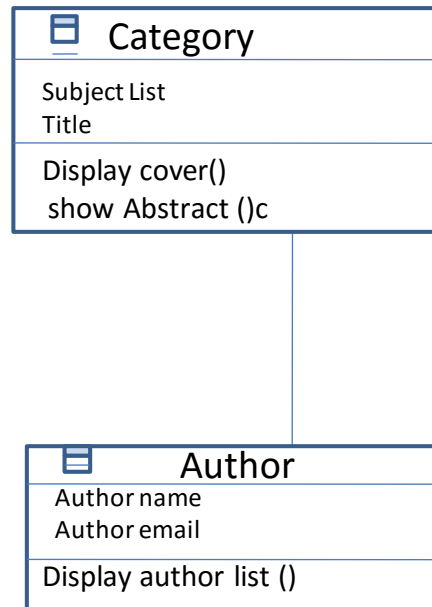


Figure 4.4.Example of relationships between domain classes.

- 2 Attributes- defines"the structure of the class". "A class consists of information or datafield explanations or properties, fields, data members, or attributes. These are usually field types and names that will be associated with state variables at program run time. These state variables either belong to the class or specific instances of the class". An example of the same is Author name: string.
- 3 Operations- The "object class or its instance behavior is defined using operations. These aresubroutines with the ability to operate on objects or classes. These operations may alter the state of an object or simply provide ways of accessing it. Many kinds of operations exist, but support for them varies across languages. Some types of methods are created and called by programmer code, while other special methods—such as constructors, destructors, and conversion operators—are created and called by compiler-generated code". An example of the same is the operation name +add to cart(int,float):void with input of the type int, float and initialization as void.
- 4 Relationship- Referencing between one or more related elements is represented using relationships. There is no general notation for a relationship. In most cases

the notation is a line connecting related elements. Subclasses of relationship are association and directed relationship. A directed relationship is a relationship between a collection of source elements and a collection of target elements. There is no general notation for a directed relationship. In most cases the notation is some kind of line drawn from the source to the target. Specific subclasses of the directed relationship define their own notation. Subclasses of the directed relationship are generalization and dependency. Generalization is a directed relationship between a superclass and a subclass. Dependency is a directed relationship which is used to demonstrate that some UML element or a set of elements requires, needs or depends on other model elements for specification or implementation. Dependency is a relationship between named elements as given in (uml-diagrams.org). An association represents a family of links. A binary association is normally represented as a line. An association can link any number of classes to depict a relationship between them. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility and other properties. There are four different types of association: bi-directional, uni-directional, Aggregation and Reflexive. For instance, an author class is associated with a category class bi-directionally. "Association represents the static relationship shared among the objects of two classes" (Wikipedia.org). Refer to figure 4.4 above.

- 5 Check-"A constraint is a packageable element representing some check condition, restriction or assertion related to some element that owns the constraint or several elements". "Constraint is usually specified by a Boolean expression which must evaluate to a true or false. Constraint must be satisfied by a correct design of the system. Constraints are commonly used for various elements on class diagrams. In general there are many possible kinds of owners of a constraint. Owning element must have access to the constrained elements to verify constraint. The owner of the constraint will determine when the constraint is to be evaluated". For example, an

operation can have pre-condition and/or a post-condition constraints. A Constraint is of the form '{ (name:.)boolean expression }'. Some of the languages used to define constraints are OCL, Java or any other constraint language. For example a constraint on a class attribute can be `+author: String {author->notEmpty()}`

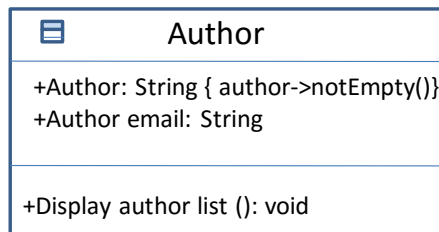


Figure 4.5.Constraint on Attribute

A constraint can be applied to an association or a class and is represented by a constraint string near the name or the symbol for the element. If a constraint applies to two elements, for example, two associations or to two classes then it is represented by a dashed line with an arrowhead between the elements with the constraint string labelling in curly braces. The direction of the arrow represents important information as the tail element of the arrow is mapped to the first position and the head element is mapped to the second position in the constrained Elements collection.

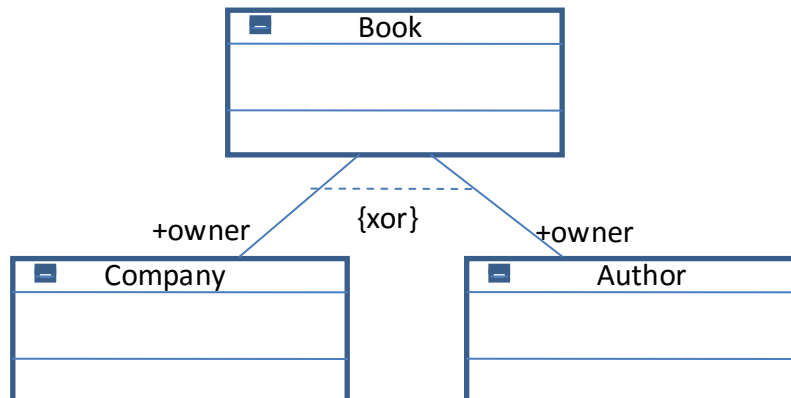


Figure 4.6.Constraint on association between classes

4.3.2. Deriving the EA00-H Domain Model

In this moment the issues related to the navigation, presentation, workflow or interaction are not taken into account yet, which simplifies and gives a bigger reuse capability to the model. Its construction follows well-known "object oriented modeling techniques":

- Identify the "domain classes".
- Include as domain classes the actors identified in the requirements workflow, if it is needed to store some kind of information. Determine the "relationships" between the domain classes.
- Define hierarchical relationships between the classes.
- Specify the most relevant "attributes and operations for a given domain class".
- Identify the cardinalities.
- In order to ensure that this domain model is a complete representation of the content WebGRL diagram after transformation, a set of transformation rules have been defined below.

4.3.3. The Transformation Rules to derive the EA00-H Domain Model from Content WebGRL Diagram.

The Transformation Rules defined to derive the Domain Model(DM) of EA00-H approach from the content WebGRL diagram of GOREWEB framework are as follows:-

- 1 *Content 2DomainClass*- This transformation rule is used to transform all Resources required to satisfy the content goals represented in the WebGRL diagram to derive a domain class of the DM with the same name as the Resource. In case of the decomposition of the goal into subgoals, the subgoals are used to derive domain classes for the resource required to satisfy them and the main goal is used to derive an aggregate class or a generalization class with decomposition link as an association between them.

Table 4.1. Transformation of Content web GRL diagram to Domain model

Content WebGRL Model Element	A-OOH Domain Model Element
Content Goal with dependency link to Resource	Domain Class
Content goals with and decomposition link to subgoals	Domain Classes with Aggregation Relationship between domain classes
Content goals with or decomposition link to subgoals	Domain Classes with Generalisation Relationship between domain classes
Content Goal with contribution link	Association between the domain classes
Tasks with Means-End link to Resource	Operation of that Domain class
Decomposition Link	Association between the source and target
Dependency Link	Dependency between domain Class and Resource
Contribution Link	Directed relationship between Source and target
Navigation goals/tasks	Relationship between domain classes

Content pattern of the Content Goal	Attribute of that domain class
SoftGoals	Constraints of that domain class or its elements

- 2 *Navigation Goals&Tasks2Relationship*- All navigation goals or tasks that represent a navigation pattern are used to derive associations in the DM. Preliminary relations into classes are derived from the relations among goals/tasks with attached resources by applying this rule . A *navigational pattern* consists of a task that requires navigation between resources or is represented by a navigational goal as shown in figure 4.1 (2a).
- 3 *Task2Operation* This transformation rule detects a means end link to a task attached to a content goal. In this case each task is transformed into one operation of the corresponding domain class .
- 4 *Content2Attribute*- In this rule content pattern attached to a content goal is translated into attribute of the corresponding domain class. Content pattern represents a set of attributes or content expressed in the task to achieve the content goal as shown in Fig 4.1 (2b).
- 5 *SoftGoals2Conditions*- This transformation rule is used to transform all softgoals attached to content goals as conditions defined on the model element which contains the satisfaction level value stored as a result of that condition defined on one or more model elements for that content goal.

4.3.4. The Transformation Method for transformation from Content WebGRL diagram to EA00-H Domain Model

All content Goals in the WebGRL diagrams are represented as Domain Classes for the resource used by them to satisfy that goal. The goal always states a function which is to be carried out. That function may be a service task or a navigation task. If it leads to

navigation to another content class then it is a navigation task. However, if it requires performing an operation on the attributes within the domain class defined for that content goal then it is a service task.

The operation within the domain class would be performed on some content variables or patterns within the domain class itself. These variables or content pattern are to be represented as the attributes of the domain class with operations in the domain class defined on them. If the operation to satisfy this goal needs a navigation to obtain information from another class then a relationship is defined between the two domain classes. Softgoals are represented as conditions of that domain class defined as member conditions on the model elements, i.e. domain class, association or conditions on the attribute of the domain class with satisfaction levels.

4.3.5. The Transformation Algorithm of Content WebGRL diagram to EAOO-H Domain Model

Based on the method shown above we have proposed a transformation algorithm for the transformation of the Content WebGRL diagram to the EAOO-H Domain Model. The Input to this algorithm is the Content WebGRL diagram to which the transformation algorithm steps are applied as per the transformation rules stated above in section 4.3.2.1 to generate the EAOO-H domain model as an output. This transformation algorithm identifies the concepts of the Content WebGRL model to be transformed into the contents of the EAOO-H Domain Model. Further, it specifies the links between the domain classes. Thereafter it identifies the goals which are going to be represented as attributes and the goals or means end task to represent the operations of a domain class. The final result is the complete EAOO-H domain Model for the given Content WebGRL diagram given as input. The pseudo code and the terminology used for defining the content WebGRL diagrams is as follows:-

A is a parent content goal

X_i 's are subgoals $i=1$ to n

S_i are softgoals $i=1$ to n

R_i are resource $i=1$ to n with resource name **R**

AX(L_i) are the links attached from **A** to **X** for $i=1$ to n ;

A(C_i)/ X(C_i) is the content information desired by a goal **A** or the subgoal **X** which will be stored as the attribute **C_i** for $i=1$ to n for the content parent or subgoal transformed into a domain class.

M_i are means end task attached to a subgoal or a parent goal which will be stored as the operation **O** for $i=1$ to n for the content parent or subgoal transformed into a domain class.

DC_i are the domain classes for $i=1$ to n .

DC_i (R_i) are the domain classes for goals attached to a resource **R** for $i=1$ to n .

We use the top down approach and start with the main goal as the parent goal. Main goal represents the vision of the website. The subgoals attached to the parent goal are checked for their link to a resource for their content extraction.

4.3.6. # EA00-H Domain Model Transformation Algorithm

(Rule: Check for content goals requiring a resource for the content extraction at each level while moving from top to bottom. Type: Transformation)

Input=Content WebGRL Diagram

Do for each level till done for levels $i=1$ to n

Foreach **A_i** attached to **aR_i** with a dependency link in (WebGRL.Content.Contentgoal) do

create **DC_i=" R_i "**

if **A_i** is not directly attached to a Resource but subdivides into subgoals by a decomposition or a contribution link which are attached to the Resource. Check the links between the parent goals and subgoals

if the link association is of type generalized

then **A=Aggregate DC_i** and **X_i= DC_i (R_i)** with **AXL_i** as the association between **A** and **X**.

```

if the link association is of type aggregate then
then A=Generalised DCi and Xi= DCi(Ri) and with AXLi as the association
between A and X.
if A/Xi is a goal that decomposes into means end task Mi then the
A/Xi=Dci(Ri).operation(Mi)
if A/Xi is a navigation goal or a navigation task between A and Xi then AXLi is
represented as a relationship between A and Xi.
foreach Si attached to a goal A/Xi =Constraint on the DC(Ri) or its members.
endif
endif
endif
endif
endif
endforeach
endforeach
enddo;
Output= EAOO-H Domain Model

```

4.4 Example of Transformation from WebGRL Content diagram to EAOO-H Domain Model for an Online BookStore

Now we present an example of the transformation from the WebGRL Content Diagram to the EAOO-H Domain model for an online web application that sells books. In this case study, a company wants to establish its sale of books by using the online bookstore on the web by attracting as many clients as possible.

4.4.1. Case Study Of Online Bookstore -Requirements Specification

The actors detected in this case study are namely “*User*”, “*Administrator*”, and “*Online Bookstore*” which depend on each other. The main goal of this “online bookstore” is to “*sell books*”, “*provide info about books*”, “*facilitate payment*” and “*maintain customer details*”, as shown in the base WebGRL figure 4.7 below. To fulfill the “*provide info about books*” goal the base WebGRL diagram shows its decomposition into four subgoals “*maintain reviews*” (which is a business process goal), “*provide search ability*” (which is a navigation goal), “*maintain subject list*” (which is a navigation goal) and “*present info systematically*” (which is a presentation goal). The adaptation goal “*provide personalize recommendations*” is related to the content requirement from the resource “*book*” and the browsing history of the customer. The navigation goal to “*provide searchability*” is decomposed into a couple of subgoals through “*enable browse of books*” into “*search book by title*” and “*search book by author*”, which are also related to the content requirement to resource “*book*”. In the same way, as goal to “*provide a cart*” is decomposed into two tasks: “*add item*” and “*remove item*” etc. These tasks are related to the content goal to resource “*cart*”. Finally, the goal to “*maintain customer details*” leads to the subgoals of “*maintain customer details*”, and “*maintain transaction and browsing history*”. These goals are represented in the base web GRL diagram shown in figure 4.7 below, which is refined and validated by our tool to give the web specific WebGRL diagrams.

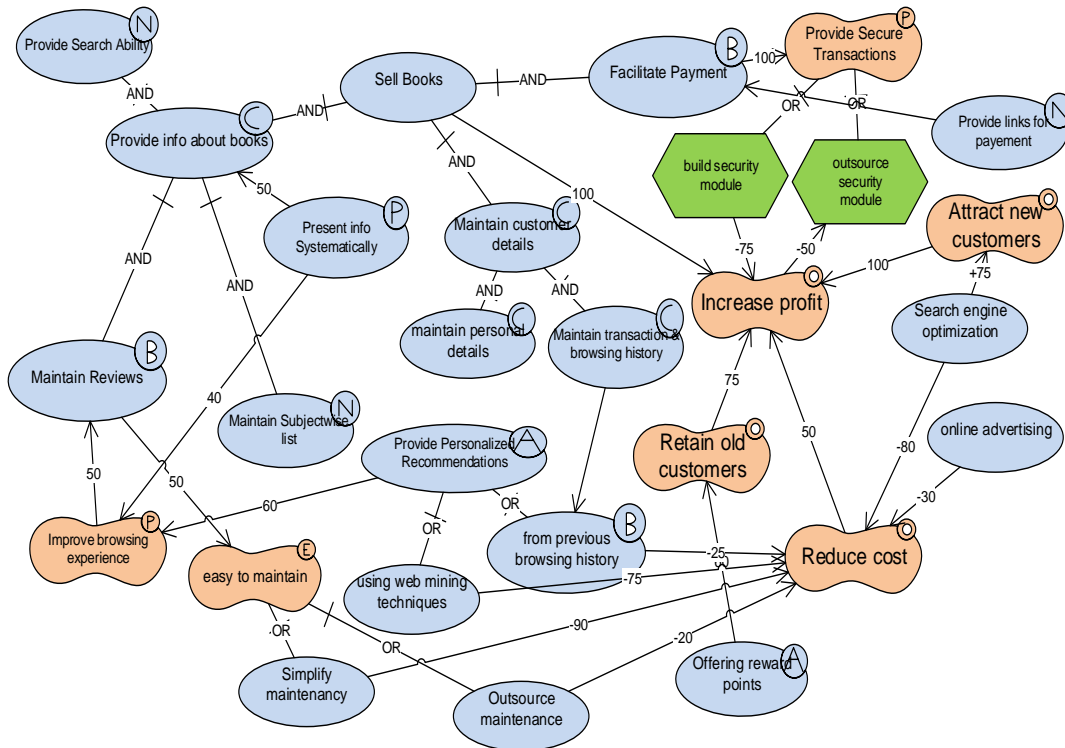


Figure 4.7. The BaseWebGRL diagram for Online Bookstore

From here we move to the refined web specific WebGRL diagrams namely the content WebGRL diagram of figure 4.8 shown in the section below.

4.4.2. The Content WebGRL Diagram

The main content goals in the content WebGRL diagram are to “provide info about books” and “maintain customer details”. The first content goal “provide info about books” is satisfied by the contribution from the task to achieve the same by display of book information like book cover, abstract, toc etc. and its decomposition into other content goals of “maintain author information” to resource author and to “enable browse of books” we need the content goal to “list categories” which is satisfied by the resource category. Similarly the “maintain order details” requires the resource order. The goal to “maintain customer details” is further decomposed into subgoals of “maintain personal details” and “maintain transaction and browsing history” which are fulfilled by the respective tasks attached to them

used to satisfy the content goal specified in the Content WebGRL model. Moreover, we detect one generalized resource for transaction and books browsed namely the “*transaction browse*” domain class with association between transaction, cart, book and customer classes.

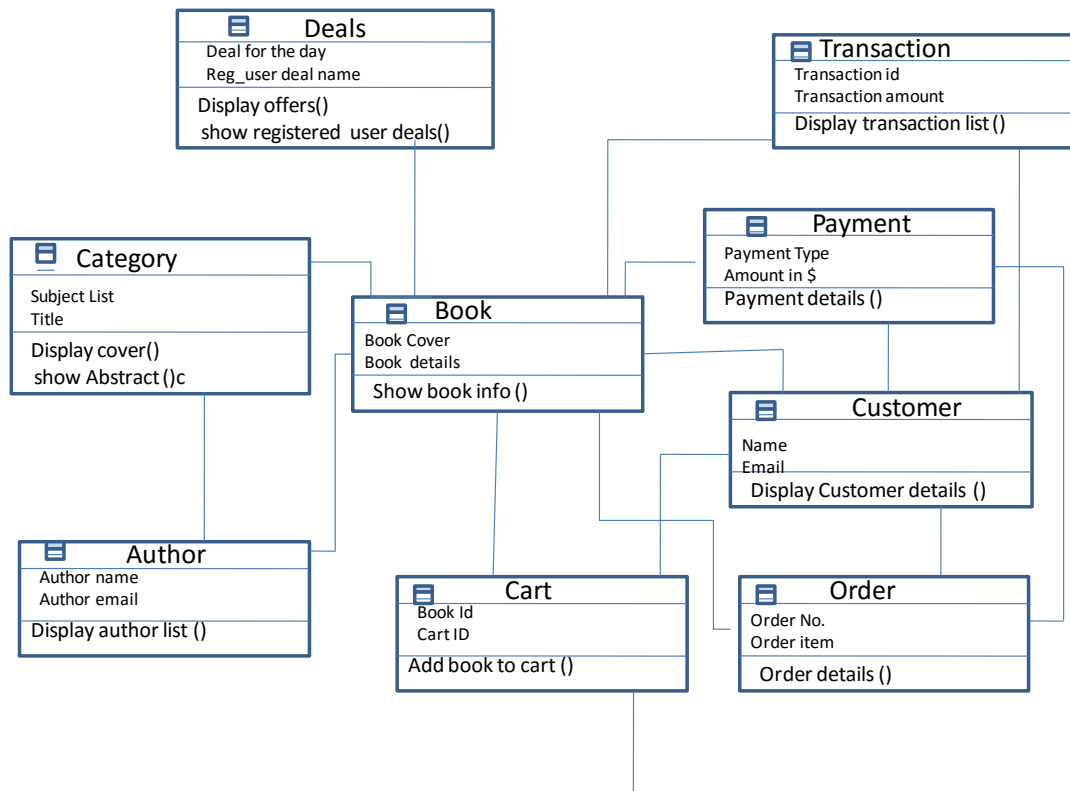


Figure4.9.The Domain Model for Online Bookstore

Similarly browsing history adds association between book and customer .Further four tasks are added as operations to the classes “*customer*”, “*cart*”and “*book*”by using the *Task2Operation* rule. The task of “*maintain personal details*” which represents a content pattern is used to store attributes by applying the *Content2Attribute* rule to add attributes of customer id, username, password etc. in the domain class of customer. The content WebGRL diagram has a content goal to“*Provide Book Info*” that follows the navigational pattern. The *Navigation2Relationship* rule results in the addition of associations among all the resources found in this pattern. Similarly *Softgoal2Condition*represents the softgoal of “*information and collection up to date*”

as a condition of the domain class customer. The generated Domain model is shown in Figure 4.9 above.

In this chapter we have presented the Enhanced EAOO-H Domain Model and its concepts. Further we have discussed the process of transformation of a WebGRL content diagram to an EAOO-H domain model based on the transformation rules and the Domain model Transformation Algorithm and illustrated it with the example of online bookstore in the next section. We will use this example throughout the thesis for better understanding of the all the phases of the whole transformation process. In the next chapter we propose the Navigation EAOO-H Model, its concepts and the transformation of the Navigation WebGRL diagram into the EAOO-H Navigation model in the design phase of web engineering.

Chapter 5

Transformation of WebGRL Navigation Model to EA00-H Navigation Design Model

We have discussed the EA00-H Domain Model and the transformation strategy for the translation of the content WebGRL model into the EA00-H Domain design model in the previous chapter. Web based design models, are used to specify structure, behavior, navigation, and presentation aspects. Navigation forms an important aspect of web application and dominates the movement within the web application. Therefore we need a strong conversion strategy to capture all the navigation goals and its aspects in the requirements phase for flawless and lossless presentation in the design phase. In this chapter we present a different facet of the web application design namely the Enhanced EA00-H Navigation Model and its components for the presenting the navigation characteristic of the design of a web applications as well as the transformation strategy for converting the WebGRL navigation model into the EA00-H Navigation model thus moving a step further in the design phase by capturing the navigation facet of the web application.

5.1 Introduction

The web specific requirements are unique and distinctive from the generic information systems requirements. The functional requirements consist of specific requirements like navigation, adaptation and the non-functional characteristics are also exclusive for web applications. The requirements analysis process used for generic systems cannot be applied to

web applications. To properly understand and capture the web application requirements holistically we need a notation that is tuned for web applications for modeling web specific requirements. Different web engineering approaches use different notations for modeling web specific requirements. WebGRL metamodel discussed in Chapter 3 forms the basis of the requirements engineering phase in our approach. The Requirements engineering phase results in web specific WebGRL models i.e. a set of content, navigation, presentation, adaptation and business process WebGRL diagrams. Web specific diagrams are used in the transformation scheme to generate the following as an output:

1. “Domain model (DM):It defines the structure of the domain data”.
2. “Navigation model (NM): in which the structure and behaviour of the navigation view over the domain data is defined”.
3. “Presentation model (PM):It defines the layout of the generated hypermedia presentation.In order to model the personalisation we need two more models the adaptation and the business process model”.
 - “Adaptation model:It defines the personalization strategies and the structure of information needed for personalization”.
 - “*Business Process model*: It defines the business processes related to the business of the web application which are expressed using a UML compliant UML profile for the WebGRL”.

In our approach we deal with the three most important factors of the “functional and non-functional requirements” captured mainly by the navigation, presentation and the content model. The adaption model is tuned for personalization of the website as per different user requirements, similarly the business organization process model represent the business strategy which is captured majorly by the soft destination. The business process model is business deal with majorly in the specific WebGRL diagrams through the softgoals. Therefore, the version and the business process model are not discussed in detail in our access

stated in this thesis. They are beyond the scope of the approach discussed here and will be quite a little in future study

As is clear from the discussion the WebGRL metamodel in chapter 3, the generation of specific WebGRL diagrams ensures the complete encapsulation of the functional and non-functional necessary, in a manner that is more suitable, more effective and closer to the real world web application development problem, as the different views of web engineering are captured separately and the focus is more on navigation, presentation and adaptivity of the web site. The transformation of these web oriented specific WebGRL diagrams thus results in a more holistic approach of transformation from the requirements to the design stage of web engineering. Generation of the EA00-H domain, navigation and presentation models of the design phase from the specific WebGRL models ensures transparency, validity and the ability to trace backwards. The seamless transformation also reduces the work of the design engineer as the entire transformation process is a validated, traceable and automated process. This also reduces the number of iterations, improves the scalability and maintainability of the software development process. Thus at the end of the transformation process we have a conflict resolved and a high quality output with alternatives. In the previous chapter we have discussed the Enhanced A-OOH approach for the domain model and the transformation of the content WebGRL diagrams to the EA00-H domain model of the design phase. In this chapter we will discuss the EA00-H Navigation model and the transformation of the WebGRL navigation diagram to the EA00-H navigation model.

5.2 The Enhanced A-OOH (EA00-H) Navigation Model

In this section, we present the Enhanced AOO-H navigation model. This navigation model is an enhanced version of the “A-OOH Navigation Model” (Garrigos et al 2010). The extensions and tuning of the A-OOH model has been done to ensure that the specific WebGRL diagram is captured in totality and there is no loss of information while transformation.

The EA00-H case considers the analysis and design phase in depth.

Analysis and Design: In this stage all the activities related to the analysis and design of the software product are included:

- a. Domain Analysis:* From the user requirements and the designer knowledge of the domain, the relevant concepts for the application are gathered.
- b. Domain Design:* The domain analysis model has to be refined in consecutive iterations with new helper classes, attribute types, parameters in the methods... etc.
- c. Navigation Design:* The domain information is the main input for the design navigation activity, where the navigational paths are defined to fulfill the different functional requirements and the organization of that information in abstract pages.
- d. Presentation Design:* Once the logic structure of the interface is defined, OO-H allows specifying the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages.
- e. Adaptation Design:* In parallel to the other sub-phases an adaptation phase is performed, which allows specific revision and individual customization strategy.

The first step is done using base WebGRL diagram and the domain analysis results in the Web Specific GRL diagrams. With the help of EAOO-H we map the refined analysis models to their respective design models. This approach is very close to the web specific diagrams generated in the analysis phase in our previous work so we have adopted this approach and enhanced as well as modified it to later on define the web specific models of WebGRL in the design phase into their respective design models with traceability. This has led to enhancement and development of our own UML profile later.

Once the requirements engineering of the web site with the help of the WebGRL has been done we have a good design alternative with conflicts resolved represented by web specific GRL diagrams. Once the requirements have been defined using the Web GRL diagram a transformation strategy can be used to derive the design models for the internet site. The transformation strategy uses a set of rules to transform these web specific diagrams into the

Navigation model in which as in the case of “A-OOH Navigation Model the structure and behavior of the navigation view over the domain data is defined (Garrigos et al 2010)” .

5.2.1. The Navigation Metamodel

The main activity of the navigation model is to capture the navigation requirements specified by the stakeholder. In order to do so we need to capture where the navigational paths are defined to fulfill the different functional requirements and the organization of that information in abstract pages. We present the navigation metamodel which forms the basis for the EAOO-H Navigation model design.

The Navigation metamodel is made up of two basic concepts:- the navigation node and the navigation link. The navigation node is used to present the source and target destinations for a navigation path. The navigation path is represented with the help of the navigation link. A navigation model can have any number of navigation nodes and navigation links. The navigation node is further specialised into three types of nodes namely the navigation class, menu and access primitives. The navigation path can be between any three of these. i.e. a navigation path maybe between a navigation class and a menu or a navigation class and an access primitive or vice versa. All possible combinations of navigation between the three types of classes are permitted. However in case of a menu having multiple navigation paths to different navigation class, each navigation path between the menu and a navigation class will be represented separately by as many navigation links as the navigation paths from the menu to the different navigation classes.

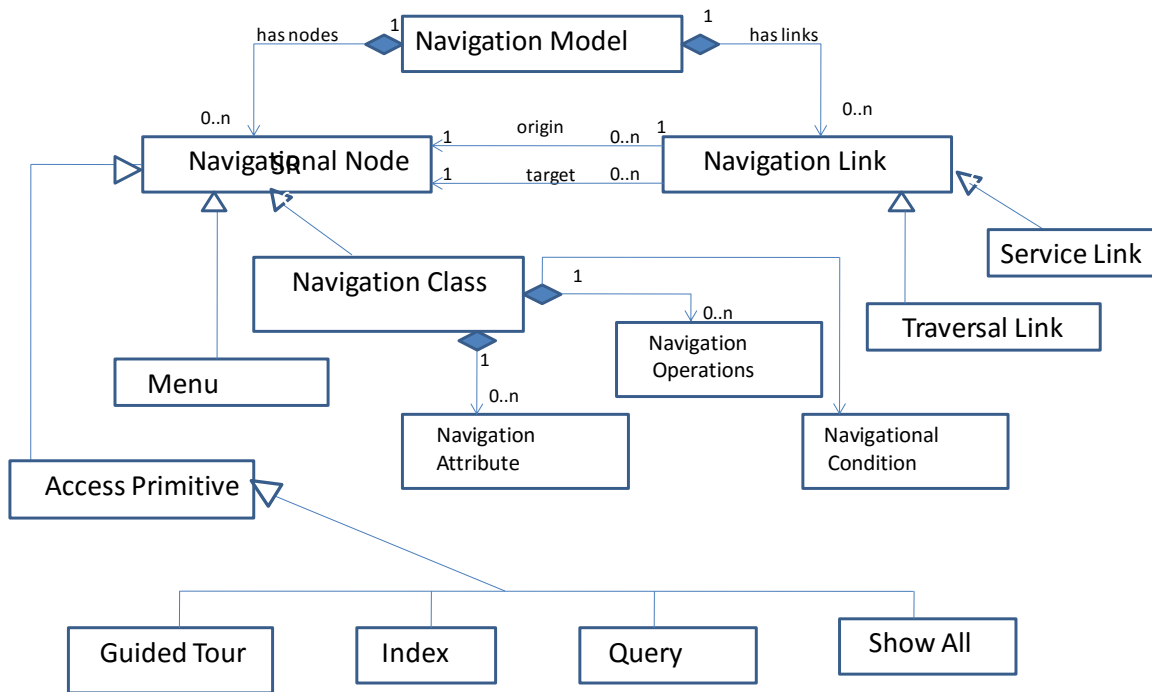


Figure 5.1. The Navigation Metamodel

Each navigation class is composed of attributes and navigation operations. A navigation class may have one or more navigation operations. Navigation attributes are used to store the properties of the navigation class and navigation operations are used for display or to carry out a procedure for that navigation class. Access primitive can be an Index, a Guided Tour, Show all or Queries which collaborate in the fulfilment of every navigation requirement of the user. Further there is a Menu or a Collection of hierarchical structures defined in Navigational Classes (Srivastava 2014). The most common important collection types is the concept of main grouping Navigational Links in various navigation course. The navigation tie-in is used to represent relationship between two navigation classes. Navigation links are of two types namely the traversal link and the service link. In case of a navigation link is used to navigate from the source to target navigation class for traversal or simple process like display the target navigation class it is a traversal link. However, if the navigation link results in an update of the logic while traversing from the source to the target navigation class in service link.

5.2.2. Navigation Access Diagram (NAD)

An EAOO-H Navigational Model (NM) describes a navigation view on data specified by the Domain Model. In AOO-H and OO-H the NM is captured by one or more Navigation Access Diagrams (i.e. NADs). The designer should construct as many NADs as different views of the system are needed, and provide at least one different NAD for each identified (static) user role. The AOO-H Navigation model the NAD is composed of *Navigational Nodes*, which represents a restricted view of the domain concepts, and their relationships indicating the navigation paths. The A-OOH Navigational model has been enhanced to transform the WebGRL navigation diagram. The EAOO-H Navigation model used by us is composed of Navigational Nodes, and their relationships indicating the navigation paths the user can follow in the final website Navigational Links. There are three types of Nodes: (a) Navigational Classes (which are view of the domain classes), (b) Access primitive which can be Index, Guided Tours, Showall or Queries which collaborate in the fulfillment of every navigation requirement of the user and (c) Menus or Collections (which are (possible) hierarchical structures defined in Navigational Classes. The most common collection type is the concept of menu grouping Navigational Links. Navigational Links (NL) define the navigational paths that the user can follow through the system. A-OOH defines two main types of links: Transversal links (which are defined between two navigational nodes) and Service Links or the Means End Link (in this case navigation is performed to activate an operation which modifies the business logic and moreover implies the navigation to a node showing information when the execution of the service is finished. We further enhance the navigation links to represent the contribution link and decomposition links of the navigation WebGRL diagram. The contribution link of the navigation WebGRL diagram will be represented as a traversal link or a service link depending on the contribution provided by the task and a decomposition link between a parent goal into subgoals is represented in the EAOO-H navigation model by a navigation target with service link which is represented by a link to an access primitive or simply by an access primitive alone. The various

navigation model concepts used by us for the transformation of the navigation WebGRL diagram into the EAOO-H navigation model are explained below.

5.2.2.1. The Navigation Node

Every navigation node is associated to an (owner) Root Concept from the DM attached to it by the notation: "Node:DM.RootConcept". "Navigation nodes" are of three kinds: namely the Navigational Classes, Access primitive and Menus or Collections.

- ***Navigational Classes (NC):***

These are domain classes consisting of attributes and operations whose the visibility is dependent on the permission for access to the class members.

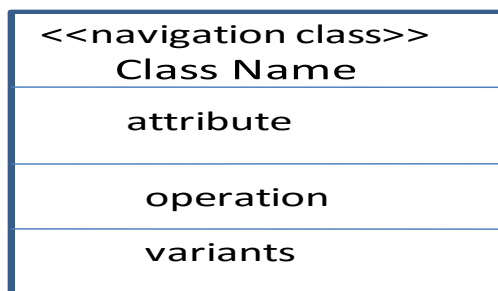


Figure 5.2. Navigation Class

It is denoted by a UML class embodied by a "stereotype the << Navigation Class>>" as shown in Figure 5.2 above.

- **Access Primitives:**

Access primitives is required to access navigation objects to satisfy the navigational requirements of the user. So, it is defined by the UML stereotypes:

Index, Guided tour, Showall and Query. These stereotypes and icons stem from Isakowitz, Stohr and Balasubramanian are defined below:

- ❖ **Index** - Index is a compound object, with one or more "index items". Each "index item" is a named object, with a link to the "instance of a navigation class". "Each index is a member of some index class, which

is stereotyped" by «index» with a corresponding icon. "An index class must be built to conform to the composition structure of classes (Srivastava 2014)" shown in Figure 5.2 above. With the help of an index the instances of a navigation class are permitted to have a direct access. In

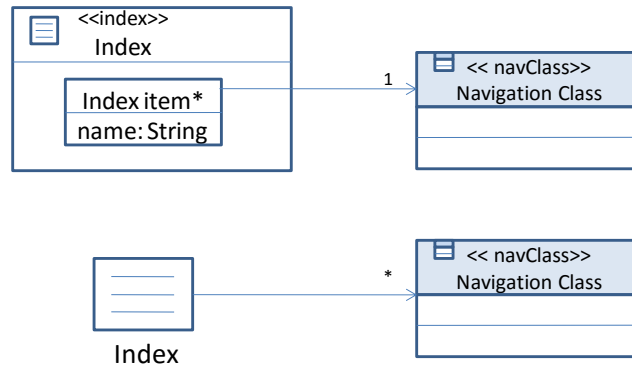


Figure 5.3. Index Class and shorthand for index

"the short form the association between Index and Navigation Class is derived from the index composition and the association between IndexItem and NavigationClass" (Srivastava 2014)

- ❖ **Guided tour** - A guided tour is an ordered access to "instances of a navigation class". "For classes, which contain guided tour objects we use the stereotype" «guidedTour» and its corresponding picture depicted in Figure 5.4. "Any guided tour class must be built to conform to the composition structure of classes (Srivastava 2014)" shown in the Figure below. Each NextItem must be connected to a "navigation class". Guided tours may be controlled by the designer or by the system. Figure 5.4 shows the shorthandnotation for a guided tour class.

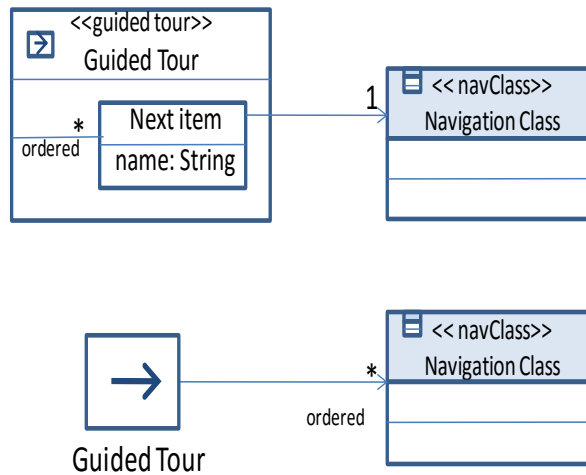


Figure 5.4. Guided Tour class and shorthand for guided tour

- ❖ **Query-** A "query is a class with query string as an attribute. This string may be given, for instance, by an OCL select mathematicaloperation". For query classes we use the stereotype «query» and the icon depicted in Figure 5.5 below. As shown in

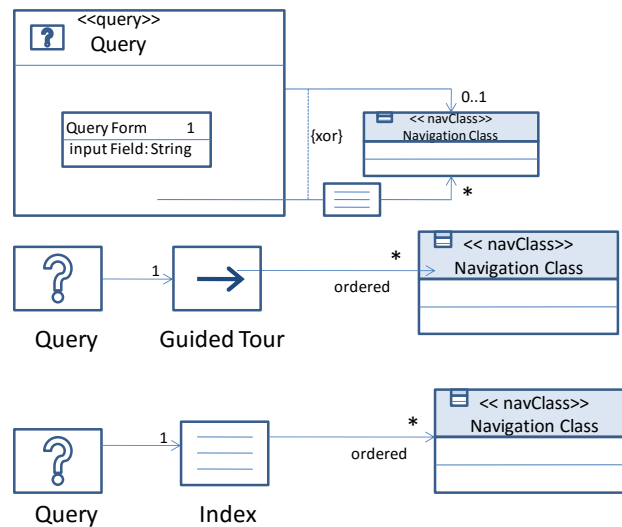


Figure 5.5. Query Class and Shorthand for Query

Figure 5.5 above, any "query class" is part of two directed associations related by the constraint {xor} . Query with several result objects is

modelled to leading first to an index supporting the natural selection of a particular instance of a navigation class . The "query" result can alternatively be used as an input for a guided tour . Figure 5.5 above also shows the shorthand annotation for a query class in combination with an index class or with a guided tour.

- ❖ **Show All** -: A show all provides navigation without indexing and without internal navigation, all the objects are shown in the same abstract page. This is modeled by the stereotype <<Showall>> and its corresponding icon is depicted in the Figure below.

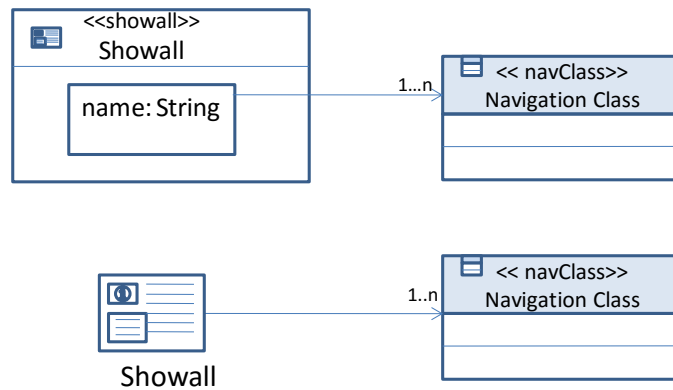


Figure 5.6. Showall Class and shorthand for showall

- **Menu**:- "A menu is a composite object" which contains a collection of navigation classes and navigation links represented by a fixed number of menu items. Each menu item has a constant name and owns a link either to an instance of a navigational class or to an index, guided tour or query . Another most common collection type is the concept of menu grouping navigation links.

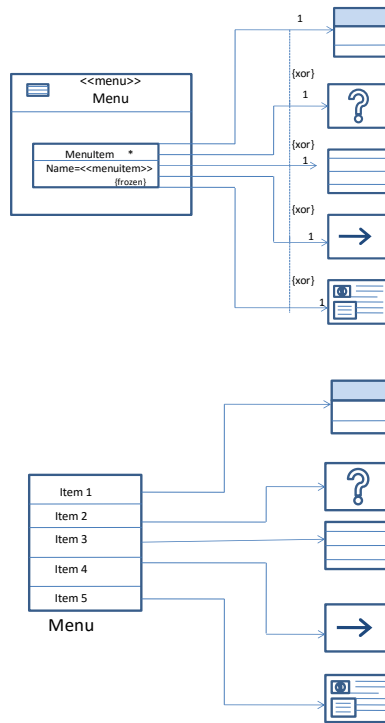


Figure 5.7. Menu class and shorthand for Menu

A menu class which is stereotyped by «menu» with a corresponding icon as shown in Fig. 5.7. A menu class should conform to the composition structure of classes described earlier.

5.2.2.2. Navigational Links (NL)

The Navigation Link is used to define the navigational route which user can follow through the system of rules. "A-OOH defines two main types of links:

- Transversal link: It is defined between two navigational nodes and Service Links for example navigational class, collection or access primitives (Srivastava 2014).
- Service or Means End Link: Navigation is performed to activate an operation which modifies the business logic and moreover implies the navigation to a node that displays data when the execution of the service is finished (Srivastava 2014)". We further enhance the navigation links to represent the

contribution link and decomposition links which will be represented as associations in the navigation model.

T-Links (Transversal Links)

"They are defined between two navigational nodes (navigational classes, collections or navigational targets). The navigation performed is done to show information through the user interface, without modifying the business logic. This type of links is represented by the stereotype <<TransversalLink>>(Garrigos 2009)".

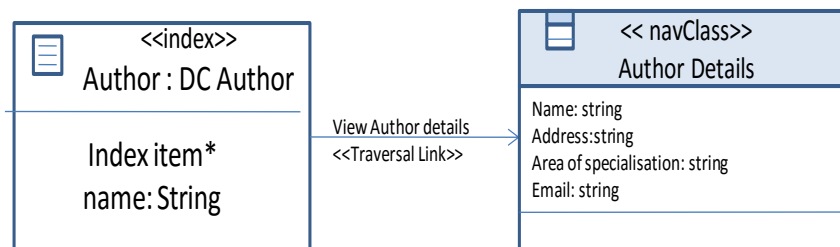


Figure 5.8. Transversal Link

S-Links (Service Links)

Service Links (S-Links);- "Navigation is an operation which modifies the business logic and moreover implies the navigation to a node showing information when the execution of the service is finished. It is established when a service of the navigational class is activated. This type of links is represented by the stereotype <<ServiceLink>> and has associated the name of the invoked service" (Garrigos 2009)".

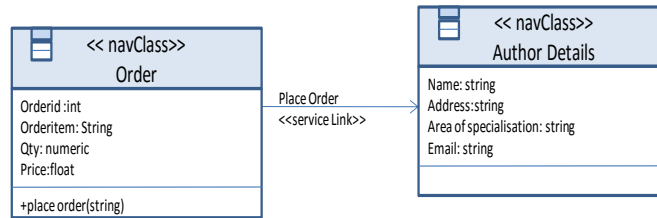


Figure 5.9. Service Link

In case the navigation performed is done to show information through the user interface, without modifying the business logic, then the navigation link is represented by a traversal link. Further, if a navigation link is also a decomposition link then as many traversal links between NCs are added as the number of decompositions of the navigation goal into its navigation sub goals represented by access primitives. However, if the navigation link is a contribution link then it is represented by a traversal or service link depending on the goal requirements.

The Navigation nodes and the Navigation link types have been described above. Now we proceed to describe the Navigation Metamodel.

5.3 Deriving the EA00-H Navigation Model

The navigation model has a one to one correspondence with the Domain model. all the information represented by the domain model should be captured by navigation between the source and target nodes. Its construction is as follows:

- Identify the navigation classes. Corresponding to each domain class there is a derived navigation class. Therefore each navigation class also represents the domain class from which it is derived.
- Identify the menu which is represented in general by a collection of navigation links to different navigation classes or access primitives.
- Identify the access primitives represented in general by the subgoals needed to satisfy the parent goal requirements.
- Determine the relationships between the navigation classes, menus and access primitives. These are the navigation paths between the different types of navigation nodes.
- Identify the appropriate attributes and operations of the navigation class.
- Find

the associated cardinalities. In order to ensure that this navigation model is a complete representation of the content WebGRL diagram after transformation, a set of transformation rules have been defined below.

5.3.1. The Transformation Rules to derive the EAOO-H Navigation Model from Navigation WebGRL Diagram.

Now we will discuss the rules for transformation from the Navigation WebGRL to Navigation EAOO-H Navigation model in this section. The Transformation Rules defined to derive the Navigation Model(NM) of EAOO-H approach from the navigation WebGRL diagram of GOREWEB framework are as follows:-

1. *Navigation Goals2Navigation Class*: This rule adds a “home navigational class to the model”. The home or the main menu is a “collection of navigational links”. This results in a "main menu" with a transversal link from the “home Navigation Class" to each of the "Navigation Classes" thus generated. From each "navigational goal" with an associated "content goal" or a "resource a navigational class (NC)" is derived. This navigation class must be derived from the domain classes represented in the domain model.
2. *Navigation Soft Goals2Navigation Class conditions*- This rule transforms the navigational softgoals with satisfaction level value expressed as conditions or constraints of the Navigation Class.
3. *Navigation2Traversal Link*: This rule inspects the navigation between the goals, if navigation exists between two associated classes a “transversal link is added from the NC which represents the root navigational goal to each of the NCs representing the associated navigational goals" (Sangeeta 2014)". If the traversal link is a contribution link the contribution value is stored as the link attribute.
4. *Contribution2Traversal / Service Link*: This rule checks navigation between one or more goals, if it is a contribution link then it is represented by a navigation link to its contributing goal and the contribution value is stored as the link attribute.

5. *Decomposition2Traversal Link with Access primitive*: This rule checks navigation between one or more goals, "if it is a decomposition link, then a transversal link is added from the NC that represents the supernavigational goal to each of the NCs representing the sub navigational goals represented by an access primitive" (Sangeeta 2014). In case of complete decomposition of the supergoal into subgoals it can also be represented by just an access primitive with no supergoal and its traversal link. For e.g. Provide Searchability goal is decomposed into search books by category and search books by author the supergoal is not represented in the EA00-H navigation model and only the subgoals are represented as an access primitive or as a part of menu. The supergoal if on decomposition is completely represented by its subgoals then the supergoal becomes superfluous.
6. *Task2Service Links*: Tasks linked to navigation goals represent navigation between class objects which provide access to the instances of the navigation class. In case of navigational means end tasks of WebGRL diagram required to fulfill a goal is represented by an access primitive then the same is represented by a service link to the navigational class which is required to carry out the operation required for that task. However, if it is a navigational goal with means end links directly attached to it then it is represented as the operation of that NC.

5.3.2. The Transformational Method for transformation from the Navigation WebGRL Diagram to the Navigation EA00-H Model

The Navigation Model is derived from the navigation goals. All navigation goals in the WebGRL diagrams are represented as Navigation Classes for the resource used by them to satisfy that goal. They are derived from the domain classes of the Domain model. The navigation goal always states a navigation task that leads to a navigation path to another navigation class. The parent navigation goal is always satisfied by a set of navigation goals at the next level. Therefore a menu is added in the EA00-H navigation model to represent the main navigation goal. The menu is further associated to contributing navigation goals. These navigation goals are identified. They may be navigation classes or access primitives depending on the goal satisfaction requirements. This navigation path

between the menu and access primitives or navigation classes is represented by traversal links between the two. Further these goals may be decomposed into subgoals which may be another navigation class or access primitive with a means end task. For each navigation path to another navigation class or access primitive a traversal link is added between the source and target classes. In case of a means end task a service link is added from the "source class to the target class" defining the operation to satisfy the task. Softgoals are represented as conditions of that navigation class or defined as member conditions on the model elements, i.e. navigation class, association or conditions on the attribute of the navigation class with satisfaction levels.

**Table 5.1. The Transformation of Navigation WebGRL diagram to
NM**

Navigation WebGRL Element	Enhanced A-OOH Navigation Model
Navigation Goals & Navigation Goals with dependency link to resources	Navigation Class
Navigation Soft Goals	Conditions of the Navigation Class
Navigation between Goals	Navigation links/Traversal links
Contribution Link/	Traversal link/ Service Link
Decomposition Link	Access Primitive like Index , Query , Showall and Guided Tour with or without a traversal link
Tasks with Means End link	Service Link to an Operation of an NC or an Operation of NC

5.3.3. The Transformation Algorithm of Navigation WebGRL diagram to EAOO-H Navigation Model

Based on the method shown above we have proposed a transformation algorithm for the transformation of the navigation WebGRL diagram to the EAOO-H Navigation Model. The Input to this algorithm is the Navigation WebGRL diagram to which the transformation algorithm steps are applied as per the transformation rules stated above in section 5.3.1 to generate the EAOO-H navigation model as an output. This transformation algorithm identifies the concepts of the navigation WebGRL model to be transformed into the navigation nodes of the EAOO-H Navigation Model. Further, it specifies the navigation links between the navigation nodes. Thereafter it identifies the goals which are going to be represented as attributes and the goals or means end task to represent the operations of a navigation class. The final result is the complete EAOO-H navigation Model for the given Navigation WebGRL diagram given as input. The pseudo code and the terminology used for defining the navigation WebGRL diagrams is as follows:-

A is a parent goal

X_i 'sare subgoals $i=1$ to n

S_i are softgoals $i=1$ to n

R_iare resource $i=1$ to n with resource name **R**

AX(L_i) are the navigation links attached from **A** to **X** for $i=1$ to n ;

AX(T_i) are the traversal links attached from **A** to **X** for $i=1$ to n ;

AX(S_i) are the service links attached from **A** to **X** for $i=1$ to n ;

M_{ni} are the menus for a navigation goals with a collection of links for $i=1$ to n ;

AP_iare the Access Primitives for navigation goals with some operation referring an existing **NC_i**

I_i are the Index access primitives for $i=1$ to n

SA_i are the Showall access primitives for $i=1$ to n

GT_i are the guided tour access primitives for $i=1$ to n

Q_i are the Query access primitives for $i=1$ to n

NC_i(A_i) are the attributes for the Navigation Class for $i=1$ to n

NC_i(O_i) are the operations for the Navigation Class for $i=1$ to n

M_i are means end task attached to a subgoal or a parent goal which will be stored as the operation O_i for $i=1$ to n for the parent or subgoal transformed into a navigation class.

NC_i are the navigation classes for each domain classes for $i=1$ to n .

DC_i (R_i) are the domain classes for goals attached to a resource R for $i=1$ to n .

5.3.4. # EA00-H Navigation Model Transformation Algorithm

(Rule: Check for parent goal and its subgoals for defining the menu and the navigation paths between the navigation nodes of the website at each level while moving from top to bottom.
Type: Transformation)

Input= Navigation WebGRL Diagram

Create **NC_i** for each corresponding **DC_i** in the Domain Model;

For A₁ create a homepage with a collection of link i.e. Main Menu M₁;

Do for each level till done for levels $i=1$ to n

Foreach navigation link from A_i to its subgoals check the type of link

if AX(L_i) is a simple navigation link or a contribution link

then Ax(L_i)= AX(T_i) between A_i and X_i where X_i=some NC_i corresponding to a DC_i

if AX_i ia a decomposition link

then AX (L_i) =AX(T_i) between A_i and X_i where X_i is not a NC_i but accesses a NC_i at other level through some other AX(L_i) then X_i=A_{Pi}

create X_i=A_{Pi} with an AX(T_i) between A_i and A_{Pi}

if AX(L_i) is a simple navigation link or a contribution link

then Ax(L_i)= AX(T_i) between A_i and X_i where A_i=some A_{Pi} and X_i= some NC_i corresponding to a DC_i

if AX(L_i) between A_i and X=M_i where A maybe an A_{Pi} and X_i is some NC_i then AX(L_i)=AX(S_i) and M_i=X_i(O_i)

if AX(L_i) between A_i and X= M_i where A_i maybe an NC_i and M_i is some means end task then M_i=A_i(O_i)

5.4 Example of Transformation from WebGRL Content diagram to EA00-H Navigation Model for an Online BookStore

In this section, we use the same example of Online Bookstore described in the example of Chapter 4 to maintain continuity and show the transformation of the Navigation WebGRL diagram to the EA00-H navigation model. The example of our approach is based on a company that sells books on-line. In this case study, a company would like to manage book sales via an online bookstore, thus attracting as many clients as possible. Also there is an administrator of the Web to manage clients .

5.4.1. Case Study Of Online Bookstore -Requirements Specification

Three actors are detected that depend on each other, namely “*User*”, “*Administrator*”, and “*Online Bookstore*”. The main goal of online bookstore is to “*sell books*”, “*provide info about books*”, “*facilitate payment*” and “*maintain customer details*”, as shown in the base WebGRL figure below. To fulfill the “*provide info about books*” goal the base WebGRL diagram shows its decomposition into four subgoals “*maintain reviews*” (which is a business process goal), “*provide search ability*” (which is a navigation goal), “*maintain subject list*” (which is a navigation goal) and “*present info systematically*” (which is a presentation goal). The adaptation goal “*provide personalize recommendations*” is related to the content requirement from the resource “*book*” and the browsing history of the customer. The navigation goal to “*provide searchability*” is decomposed into a couple of subgoals through “*enable browse of books*” into “*search book by title*” and “*search book by author*”, which are also related to the content requirement to resource “*book*”. In the same way, as goal to “*provide a cart*” is decomposed into two tasks: “*add item*” and “*remove item*” etc. These tasks are related to the content goal to resource “*cart*”. Finally, the goal to “*maintain customer details*” leads to the subgoals of “*maintain customer details*”, and “*maintain transaction and browsing*”

history". These goals are represented in the base web GRL diagram which is refined and validated by our tool to give the web specific WebGRL diagrams.

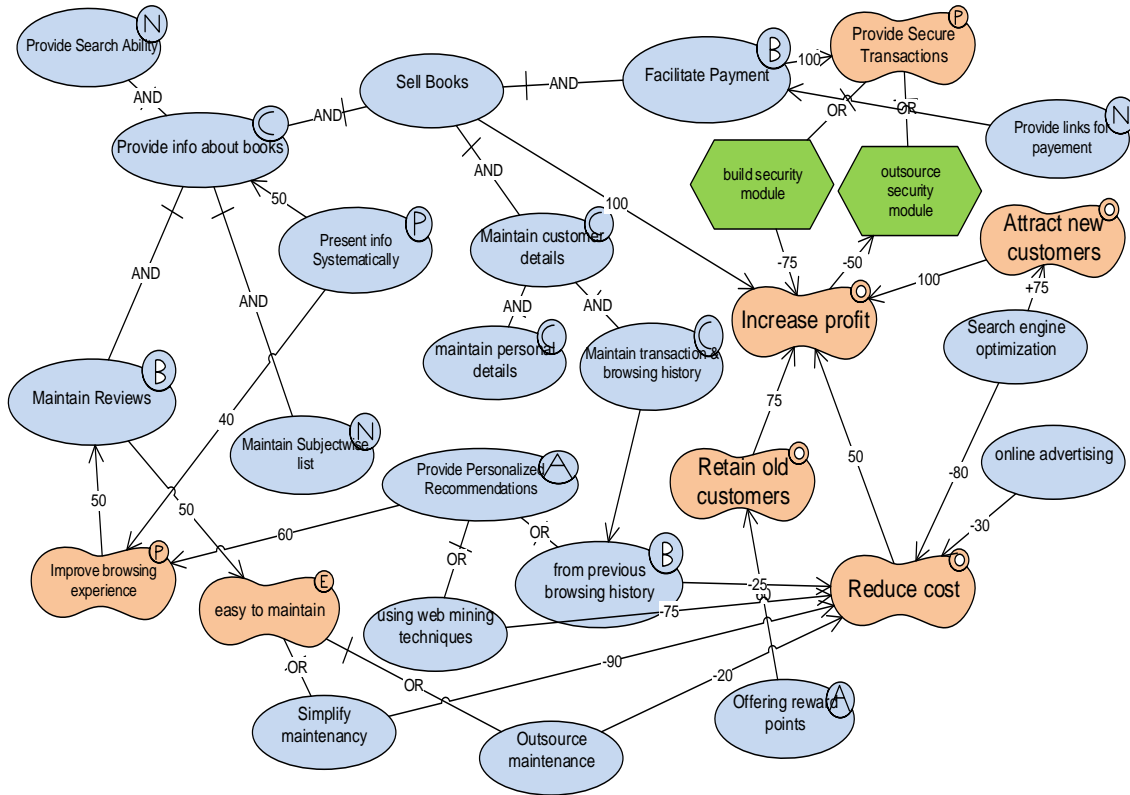


Figure 5.10. The BaseWebGRL diagram for Online Bookstore

From here we move to the refined webspecific WebGRL diagrams namely the navigation WebGRL diagrams of figure.

5.4.2. The Navigation WebGRL Diagram for Online Bookstore

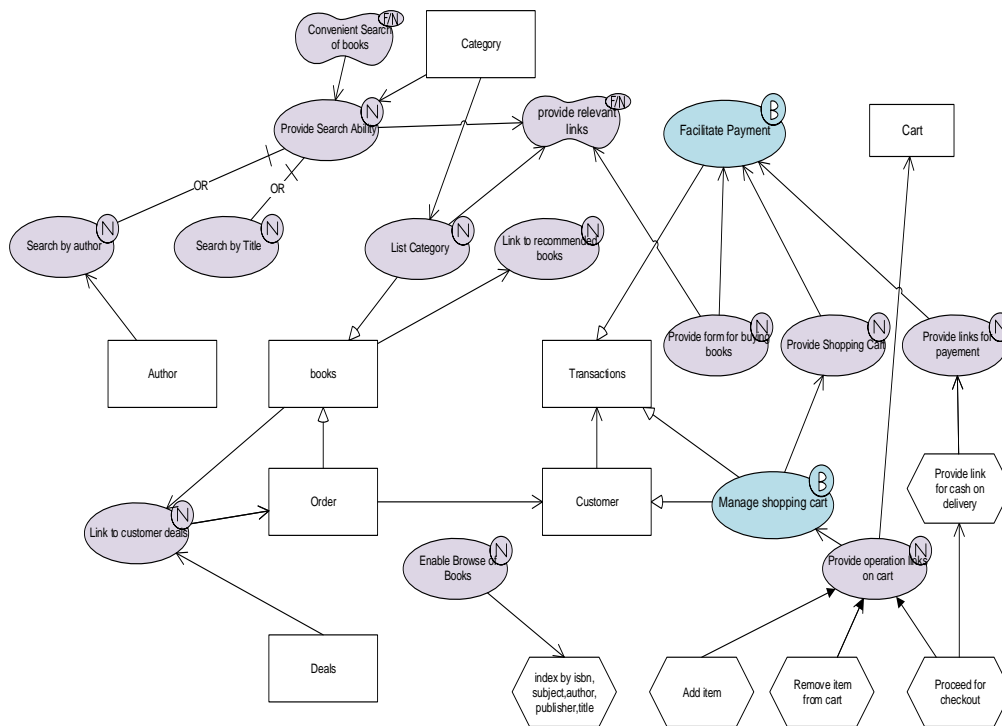


Figure 5.11. Navigation WebGRL diagram

From the refined navigation WebGRL diagram we have the main navigation goal to “*provide searchability*” is decomposed into subgoals into “*search book by title*” and “*search book by author*”, which are also related to the content requirement to resource “*book*”. In the same way, we have the navigation goal to “*provide operation links on cart*” is decomposed into two tasks: “*add item*”, “*proceed for checkout*” and “*remove item from cart*”. These tasks are related to the content goal to resource “*cart*”. There is also a navigation link to “*customer deals*”. Finally, the goal “*facilitate payment*” is used to “*place an order*” for sale of book, “*provide link for payment*”, “*provide link to shopping cart*” and “*provide form for buying books*”.

5.4.3. The Method for deriving the EA00-HNavigation Model m

To derive the NM we take into account the navigation goals. All navigation Goals in the WebGRL diagrams are represented as Navigation Classes for the resource used by them to satisfy that goal. They are derived from the domain classes of the Domain model. The

navigation goal always states a navigation task that leads to another navigation class for this a traversal link is added between the two navigation classes that is supported by the access primitives depending on the requirement stated in the goal. Softgoals are represented as conditions of that navigation class defined as member conditions on the model elements, i.e. navigation class, association or conditions on the attribute of the navigation class with satisfaction levels.

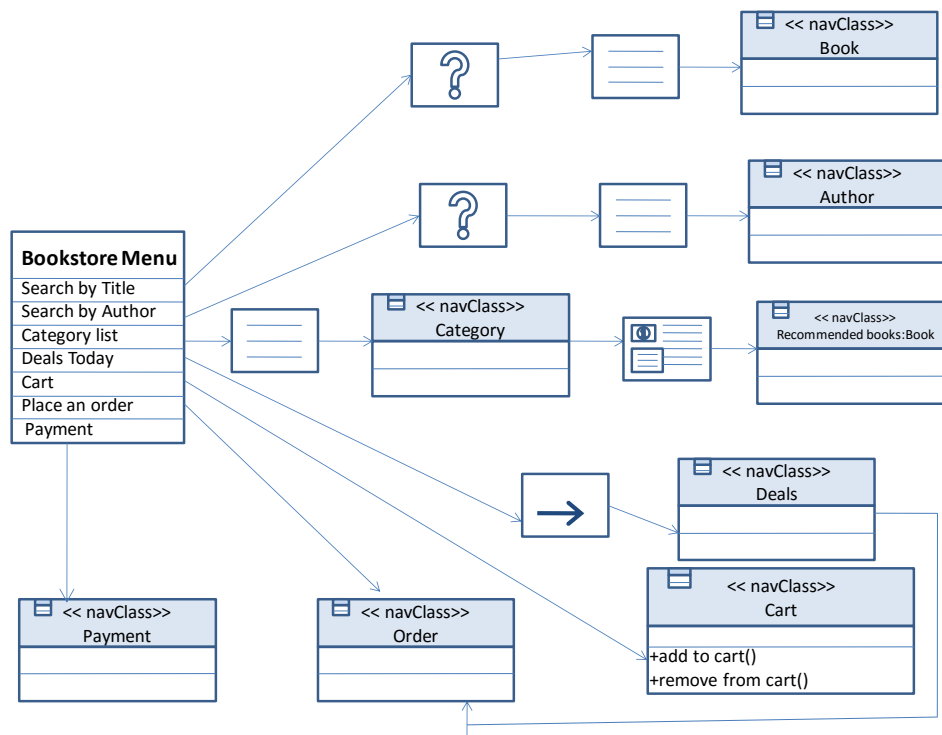


Figure 5.12. The Navigation Model

In the case of the Navigational WebGRL model shown above we proceed as follows to transform the navigation WebGRL diagram to its corresponding EA00-H navigation Model. The rule *Navigation2NavigationClass* is performed adding a home page with a collection of links (i.e. menu). Afterwards, one NC is created for each navigational goal with an attached resource, in this case we have seven "NC created from navigational goals". From the menu, a transversal link to each of the created NCs is added. The rule *NavigationSoftgoal2Navigationcondition* represents the navigation softgoal to "provide relevant links" as a constraint on the traversal links by checking source and target

navigation classes. The *Navigation2Traversal links* adds traversal links to cart, order and payment to satisfy the goals of facilitate payment, place an order and provide shopping cart. In addition to this to satisfy the searchability goals and browse of books by subject list is provided by using the rule of *Task2AccessPrimitives* to add search query of search by title and search by author are satisfied by query NCs and a category index list NC further linked to a showall of all the recommended books in that category. A guided tour NC of the deals is also linked to the main menu. Provide operations on cart goal is attached to means end tasks of add to cart, remove from cart etc. These means end tasks are represented by operations of the NC they access in this case the Cart. Finally, as we are referring to the website stakeholder, we find a guided tour to the deals NC for attracting new customers. The transformed navigation model is shown above in Figure 5.12.

In this chapter we have presented the Enhanced EAOO-H Navigation Model and its concepts. Further we have discussed the process of transformation of a WebGRL navigation diagram to an EAOO-H navigation model based on the transformation rules and the Navigation model Transformation Algorithm and illustrated it with the example of online bookstore in the next section. We will use this example throughout the thesis for better understanding of the all the phases of the whole transformation process. In the next chapter we propose the Presentation EAOO-H Model, its concepts and the transformation of the Presentation WebGRL diagram into the EAOO-H Presentation model in the design phase of web engineering.

Chapter 6

Transformation Approach for Presentation WebGRL to EAOO-H Presentation Model

In this chapter, we take another important step from the requirements phase to the design phase. Web specific requirements for different facets of web applications have been captured in the requirements phase as described in chapter 3. In the previous chapters 4 and 5 we have described in detail the EAOO-H domain and navigation models respectively. Web applications require a holistic representation of the requirements captured in the requirement phase into the design phase. Here, we present another feature of Web based applications namely the presentation design model. In this chapter we focus on the EAOO-H design model and the transformation strategy for conversion from the presentation WebGRL requirements stage into the EAOO-H presentation design model. This completes the automatic transformation strategy for translation of some very important facets of web application from the repeatedly refined requirements into the design stage for web based applications. This presents a better design and high quality of product development for web applications which captures the stakeholders' goals very closely.

6.1 Introduction

We have discussed the need for web oriented requirements engineering in the development of Web applications. "Web based applications" might have multiple stakeholders, and the size

and purpose of the applications may be different for each of these web applications (Srivastava & Chawla 2010) . "Many approaches have been developed for generic systems (Rolland et al 1999),(Castro et al 2002). However, the notations and models developed for generic applications do not address very important issues of web applications like navigation, presentation etc. Some work has been done by researchers (Bolchini & Paolini 2004), (Jaap et al 2006), (Azam et al 2007),(Chawla & Srivastava 2010) on web engineering approaches taking into account the Goal driven analysis, but many concepts of goal driven analysis like design rationale, conflict resolution, goal prioritization have been surpassed and not taken in totality". An extended form of GOREWEB framework called WebGRE framework is being used as the basis for this work. We use the refinedWebGRL diagram of the WebGRE approach as an input to the model transformation approach to transform seamlessly from the requirements engineering phase to the design phase. The advantage of using the WebGRE approach is that most of the problems have already been tackled in the requirements engineering detailed analysis phase and we have a very refined input to our transformation approach.

Web based application design has been attempted by a number of approaches, namely the Object-Oriented Hypermedia Design Method (OOHDM) (Schwabe & Rossi 19950) and its successor. Semantic Hypermedia Design Method (Schwabe & Rossi 2001),(Schwabe et al 2004) that allow the concise specification and implementation of web applications. Hypermedia design approach of WebML (Web Modeling Language) (Ceri et al 2002), (Ceri et al 1997) it is a visual language for specifying the content structure of a Web application and the organization and display of contents in one or more hypertexts. There is also theUWE approach (Koch et al 1999) this is an object oriented approach which has as a unique feature i.e. it's Unified Modelling Language compliance as UWE is defined in part of UML profile and is an extension of the UML metamodel .

Our approach is different from the design applications stated above as we use the goal driven requirements analysis helps in capturing stakeholders' goals very finely, they enhance the requirements analysis in many ways, as the requirement clarification and the

conflicts between requirements can be detected early and design alternatives can be evaluated and selected to suit the requirements (Chawla et al 2014). The output of the WebGRE Framework is used as the transformation input to derive the design models supported by a UML profile. In all these approaches listed above they lack the flawless transition from the requirements phase to the design phase where due emphasis has not been given to the requirements phase. Presently, "effort for requirement analysis in Web engineering is rather focused on the system and the needs of the users are figured out by the designer. This scenario leads us to websites that do not assure *real* user requirements and goals, thus producing user disorientation and comprehension problems. There may appear development and maintenance problems for designers, since costly, time-consuming and rather non-realistic mechanisms (e.g. surveys among visitors) should be developed to improve the already implemented website, thus increasing the initial project budget()". The "main benefit of our point of view is that the designer will be able to make decisions from the very beginning of the development phase. These decisions could affect the structure of the envisioned website in order to satisfy needs, goals, interests and preferences of each user or user type (Garrigos 2009)". Also we develop three design models from the BaseWebGRL diagram in the requirements phase which helps in giving a detailed picture from different perspectives related to the web applications in the design phase. Further, the transition of the design models to a UML compliant UML Profile aids in platform independent development of the product.

The "WebGRE Framework for Goal Oriented User Requirements of (Chawla and Srivastava 2011) has been used as the goal driven requirements analysis helps in capturing stakeholders' goals comprehensively and the requirement clarification and the conflicts between requirements can be detected early with evaluation of the design alternatives based on the suitability of the requirements". The output of the WebGRE Framework is used as the transformation input to derive the design models supported by a UML profile. In all these approaches listed above they lack the flawless transition from the requirements phase to the design phase where due emphasis has not been given to the requirements phase. Web engineering is mostly focused on the presentation aspect of the website as such many user

goals are overlooked by the designer. "This scenario leads us to websites that do not assure *real* user requirements and goals, thus producing user disorientation and comprehension problems. There may appear development and maintenance problems for designers, since costly, time-consuming and rather non-realistic mechanisms e.g. surveys among visitors should be developed to improve the already implemented website, thus increasing the initial project budget. The designer will now be able to make decisions from the very beginning of the development phase. These decisions could affect the structure of the website visualized in order to satisfy needs, goals, interests and preferences of each user or user type"(Garrigos 2009).Also we develop three design models from the Base WebGRL diagram in the requirements phase which helps in giving a detailed picture from different perspectives related to the web applications in the design phase. Further, the transition of the design models to a UML compliant UML Profile aids in platform independent development of the product.

We present the EAOO-H Presentation model in this chapter. We first present the concepts used by us in the Presentation approach used by us for design models. In section 3 we present the Enhanced A-OOH based Design Presentation Model and its elements. In Section 4 the transformation rules and the method to derive the EAOO-H presentation model is given, Further, in section 5 we present a case study of transformation of an online bookstore presentation WebGRL diagram to the EAOO-H Design Presentation diagram.

6.2 The EAOO-H Presentation Model

In this part, we briefly present the EAOO-H model in order to ensure that the transformation of specific WebGRL models is flawless and all the elements of the WebGRL model are well represented in the enhanced AOO-H design models. EAOO-H case considers the following workflows of Analysis and Design:-

Analysis and Design: In this stage all the activities related to the analysis and design of the software product are included:

a. Domain Analysis: From the user requirements using the WebGRE framework and the designer knowledge of the domain, the relevant concepts for the application are gathered.

b. Domain Design: The domain analysis model has to be refined in consecutive iterations with new helper classes, attribute types, parameters in the methods. This is done using the WebGRL Content Diagram transformation to the Domain Model

c. Navigation Design: The domain information and the Navigation WebGRL diagram is the main input for the design navigation activity, where the navigational paths are defined to fulfil the different functional requirements and the organization of that information in abstract pages with the help of the Navigation Model.

d. Presentation Design: Once the logic structure of the interface is defined, EAOO-H allows specifying the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages. This is presented with the help of a Design Presentation Diagram consisting of two different levels.

e. Adaptation Design: In parallel to the other sub-phases an adaptation design phase is performed, which allows special adaptation strategies.

Requirements Engineering part is done using WebGRL diagrams and the domain analysis results in the Web Specific GRL. With the help of EAOO-H we map the refined analysis models to their respective design models. As stated earlier since the A-OOH approach is very close to the web specific diagrams generated in the analysis phase in our previous work so we have enhanced as well as modified this approach to the EAOO-H approach to define the web specific models of WebGRL in the design phase into their respective domain, navigation and presentation design models with traceability.

After refining the base WebGRL diagram we get the content, navigation, presentation etc. web specific diagrams. We have already discussed the Content and the Navigation WebGRL models and their transformation to EAOO-H Domain and Navigation Design Models in Chapters 4 and 5. In this chapter we first present the EAOO-H Presentation Model, its

concepts and the transformation of the Presentation WebGRL Diagram into the Presentation Design Model.

6.3 The Presentation Design Model

After analyzing and modeling the requirements of the website with the help of the WebURN tool presented in the paper (Chawla et al 2011) we have a good design alternative with conflicts resolved represented by specific WebGRL diagrams. Once the requirements have been defined using the WebGRL diagram a transformation strategy can be used to derive the conceptual, navigational and presentation role model for the website. The transformation strategy uses a set of rules to transform these web specific diagrams: the Domain model (DM) for defining the structure of the domain data, Navigation model in which the structure and behavior of the navigation view over the domain data and the Presentation model in which the layout of the generated hypermedia is presented. In this chapter we focus on the seamless transformation of the Presentation WebGRL Diagram to the Presentation Design model. The Domain and Navigation models and their transformations have already been discussed in detail in Chapters 4 and 5.

During the presentation design, the concepts related with the abstract structure of the site and the specific details of presentations are gathered. The Presentation Model is defined in this activity. It can be captured by one or more "Design Presentation Diagrams (DPDs), for every NAD in the system there is a corresponding DPD". In the next section we present the Enhanced Design Presentation Diagram (EDPD) Metamodel and the description of its main elements.

6.4 The EDPD Metamodel

The DPD MOF metamodel has been defined to formalize the elements of the DPD and the existing associations among them as shown in fig. 6.1. A metamodel defines the language to express the model. There are two main basic elements of a DPD Metamodel namely the Presentation Nodes and the relationships among them represented by the Presentation Links.

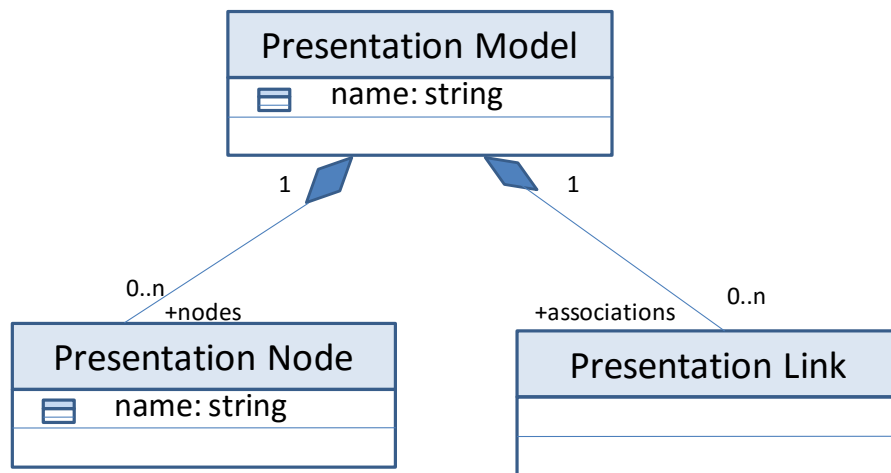


Figure. 6.1. DPD Main Elements

There are two types of presentation nodes, the structure node and the layout node. The structure and the presentation nodes are considered at two different levels of the DPD. The structure nodes are used to define the structure of the presentation of the website and are considered at level zero of the DPD. The Layout of the presentation model for the web application is considered at level one. The different types of structure and layout nodes are described briefly below.

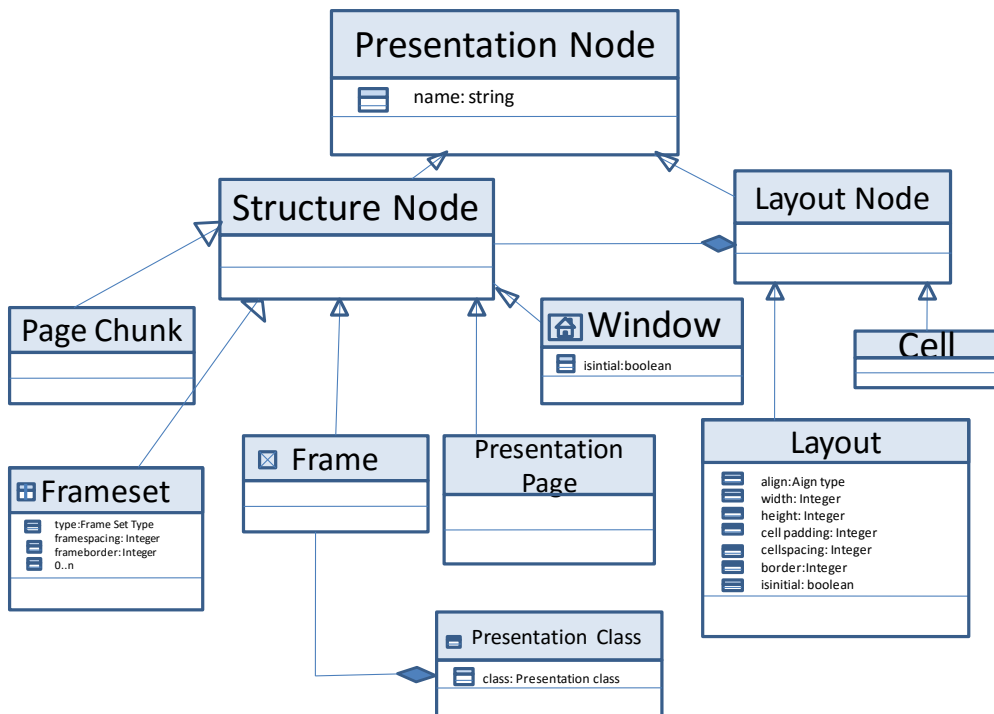


Figure 6.2.Presentation Node Subtypes

The different types of Structure Nodes are: Presentation Page, Window, Frame, Frameset, Presentation Class and Page Chunk (level zero of the DPD) .Similarly the different types of Layout Nodes considered are: Layout and Cell (level 1 of the DPD) as shown in fig. 6.2.

Just like the presentation nodes there are a variety of presentation links to represent different types of relationships between the presentation nodes. In a Presentation Model five types of Presentation Links can be defined as shown in fig.6.3:

- *Navigates*: This relationship can be defined between Presentation Page elements.
- *Builds*- This relationship is defined between server page and client page where a server script is used to build the client page.
- *Submit*: It is defined as the relationship between the form and the server page where the input content of the form is submitted to the server page.

- *Contains*: This relationship is defined between Presentation Page elements and Page Chunk elements to indicate a presentation chunk is contained (and shown) in one or several Presentation pages.
- *Redirects*:- This relationship can be defined for a Presentation Page that redirects to itself.
- *Displays or Presents*:-This relationship can be defined between a Presentation Page and its frame elements or a window and frame on which the presentation page is displayed.
- *IncludeFrame*: This relationship can be defined between FrameSet and Frame elements.
- *IncludeCell*; This relationship is defined between Layout and Cell elements.

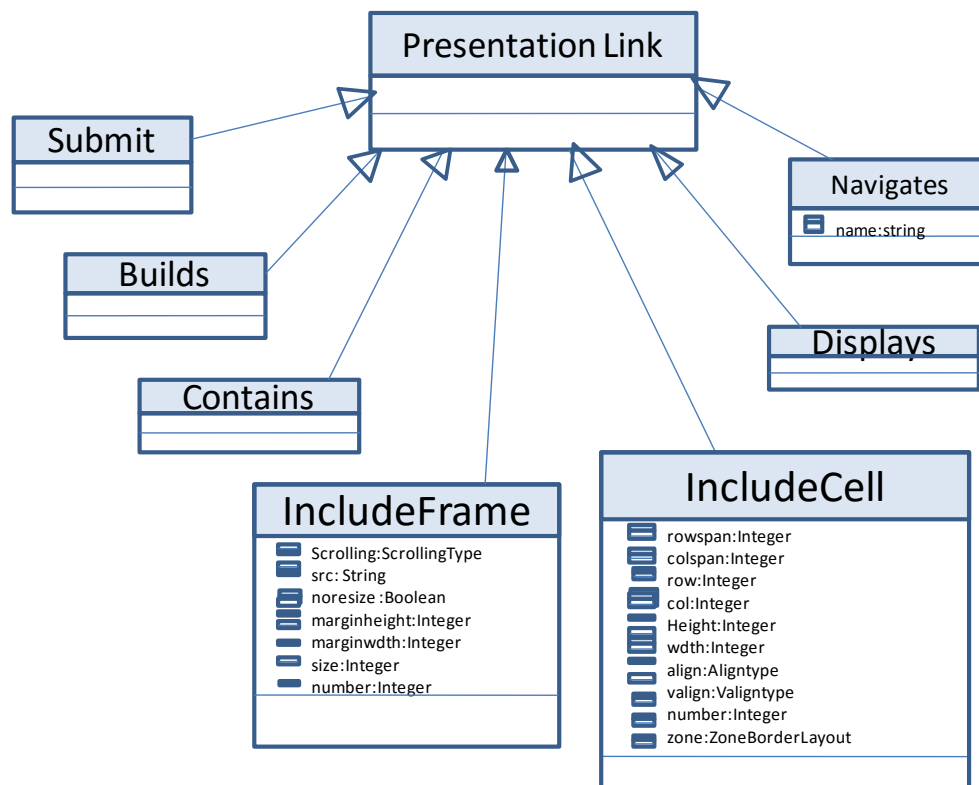


Figure 6.3.Presentation Link types

The Enhanced DPD (EDPD) Metamodel and its elements have been described in this section in brief we now present the elements of the EDPD in detail in the next section.

6.4.1. The Enhanced Design Presentation Diagram

The basis of the enhanced DPD described below is the A-OOH DPD. However as per the requirements of the WebGRL diagram and to ensure faultless transformation from WebGRL diagrams to EAOO-H based models we have enhanced the existing A-OOH DPD to the Enhanced Design Presentation diagram (EDPD). The enhanced version EDPD and its modelling elements are described below. The main goals of the EDPD are as follows:

- The first goal is to describe the organisation of pages within the website where navigation nodes of the NAD are assembled into the presentation pages. Presentation Pages are abstract pages which can be represented by converting them into one or more concrete pages. The designer can use additional static pages to the EDPD also. This represents the level 0 of the EDPD.
- The second goal is used to represent the layout and style of each page of the interface. The layout and style are used by the designer to describe the components, their style and their positioning on the page. The designer can later on change the individual page structure and add static elements to the EDPD. This shapes the level 1 of the EDPD formed by expanding the abstract pages defined in the level 0 earlier. After the refinement of the EDPD the front end for the web application, static or dynamic can be generated depending on the constraints of the target platform. The main modelling elements of the EDPD are described next. We classify them into level zero or level one depending on where they can be defined.

6.4.2. Main Elements in the Level Zero of the EDPD

As aforementioned this level provides the assembling of navigational nodes into presentation pages that form the page structure of the website. Moreover new static pages can be added to this level. We enhance the existing DPD of A-OOH to represent the following:-

- **Presentation page**

This element corresponds to an abstract page which can be a Server or a Client Page with an associated Presentation model where we define all the components shown in that page. It is represented as a UML Package with the stereotype <<Presentation Page>> as in fig.6.4. Inside this package the Presentation Model attached to the Presentation Page is shown.

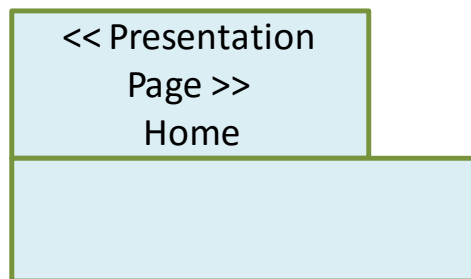


Figure 6.4.Presentation Page

- **Server Page** is used to specify content generated using a server script.
- **Client Page** is used to specify content generated using a client script which is presented or displayed in a target window. The position of the content on the presentation page is given by the window, frame or framesets.
- **Target** is an abstract class used to generalize the concept of window and frame on which the presentation page is displayed.

- **Window** is the part of the user interface where presentational interface components are displayed. The notation is a UML class with the stereotype <<Window>> as we can see in Fig. 6.5 below.

All the elements described above are presented in the fig 6.5 below.

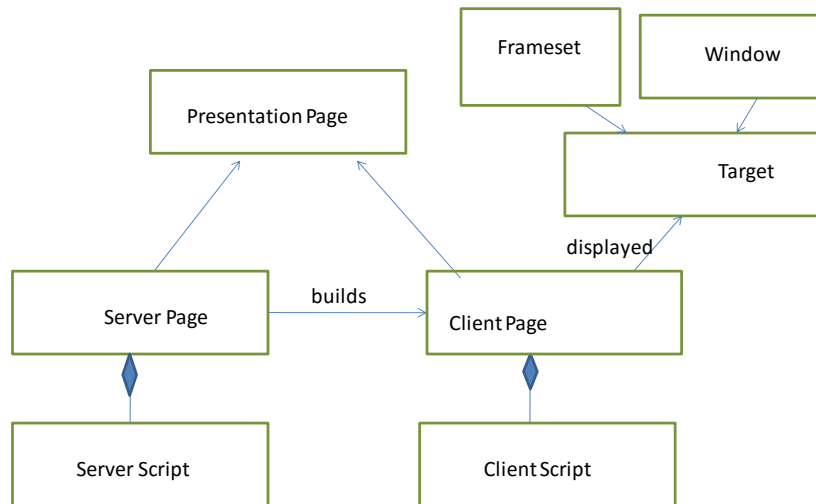


Figure 6.5.Main Elements in the Level 0 of the EDPD

• Page Chunk

The Page Chunk element defines a section of an abstract page. The constituents of the page chunk are defined in the associated Presentation model. This section can be used again in other pages of the Web application as such repetition is avoided. It is symbolized by a UML Package with the stereotype <<Page Chunk>> in fig. 6.6. The Presentation Model attached to the Page Chunk is shown inside it.

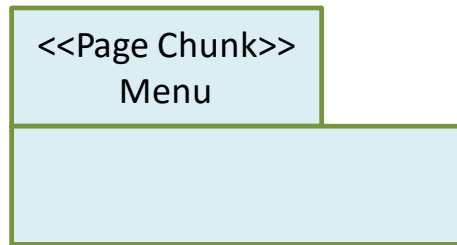


Figure 6.6:Page Chunk

• **FrameSet, Frame**

The FrameSet element represents a set of frames in which the browser window can be divided. In this way we provide the designer the possibility of using a frame based design for his application. It is symbolised by a UML class with the stereotype <<FrameSet>> as seen in fig 6.7. The Frame element represents a frame that is part of a FrameSet. It is symbolised by a UML class with the stereotype <<Frame>>. FrameSet and Frame elements are associated with the <<includeFrame>> association.

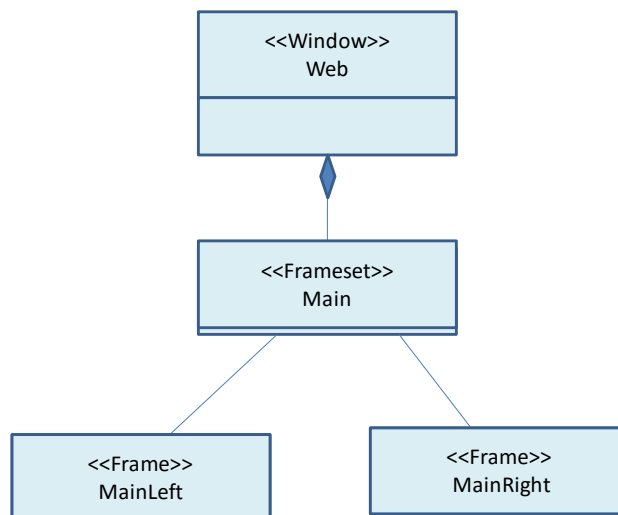


Figure 6.7.Window, FrameSet and Frame elements

6.4.3. Main Elements in the Level One of the EDPD

As already explained, the goal of level one is to describe the layout and style of each page of the interface defined in the level 0. Static elements can be added to the pages like static text. If we use frames and framesets to present the content on a presentation page then each frame or a frameset has a presentation class associated with it.

- **Presentation Class**

The Presentation class is a structural unit which contains the user Interface Components shown in fig. 6.8 which are to be represented in that frame or frameset.

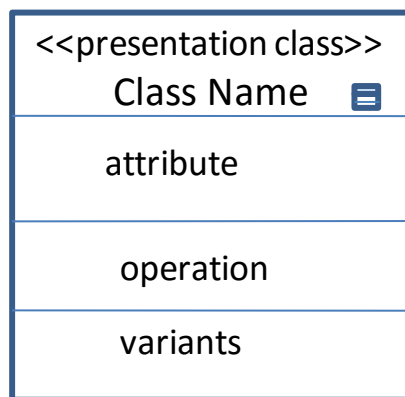


Figure 6.8.Presentation Class

- **Interface Components are text image video audio anchor and forms.**

- ❖ *Composed Interface Component*

The composed interface components considered are Collection, Anchored Collection and anchor

- Anchor- Anchor is a composed interface component which contains a simple interface component.
- Collection- Collection is a collection of simple interface components image, text, video and audio.
- Anchored Collection-Anchored Collection is a collection of anchors.

- ❖ *Simple Interface Component*

The simple interface components considered are:

- Image, Text, Audio, Video and Form Elements (e.g. Input Button, Input Submit...etc.) as in fig. 6.9.

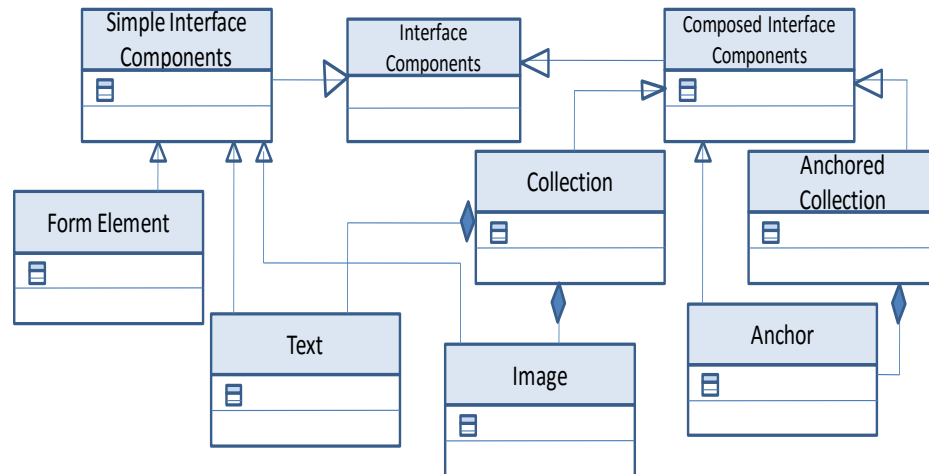


Figure 6.9. Presentation Class and the Interface Components

- **Layout**

Instead of using frames, layouts can be defined for defining the disposition of the objects visualized in the Web page. These layouts can be of three different types: BorderLayout, BorderLayout and GridBagLayout; each of them provides a concrete distribution for its cells. The layouts are translated, in last instance, to HTML tables. The different types of layouts considered can be nested.

- ❖ **Box Layout**

Box Layout arranges components on top of each other or row of your choice a typical box layout is shown in fig. 6.10.

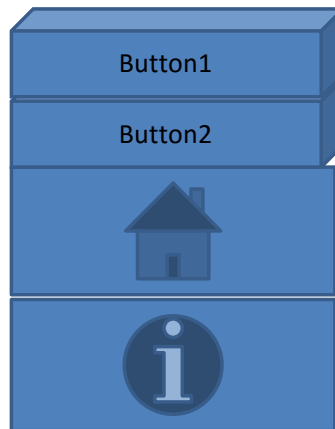


Figure 6.10.BoxLayout

❖ **BorderLayout**

As the Fig. 6.11 shows, a BorderLayout has five areas in which we can place the different elements of a Web page.

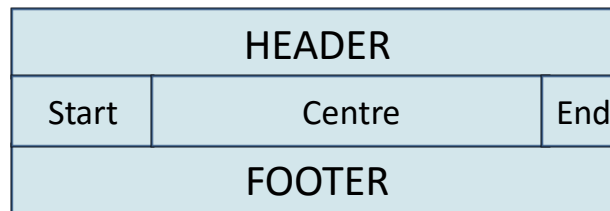


Figure 6.11.BorderLayout

❖ **GridBagLayout**

GridBagLayout is one of the most flexible layout. A GridBagLayout places components in a grid of rows and columns, allowing specified components to span multiple rows or columns. Not all rows necessarily have the same height. Similarly, not all columns necessarily have the same width. Essentially, GridBagLayout places components in rectangles (cells) in a grid, and then uses the components' preferred sizes to determine how big the cells should be. The following fig. 6.12 shows an example of a gridbaglayout. As you can see, the grid has three rows and three

columns. The button in the second row covers all the columns, whereas the third row button covers only the two right columns.

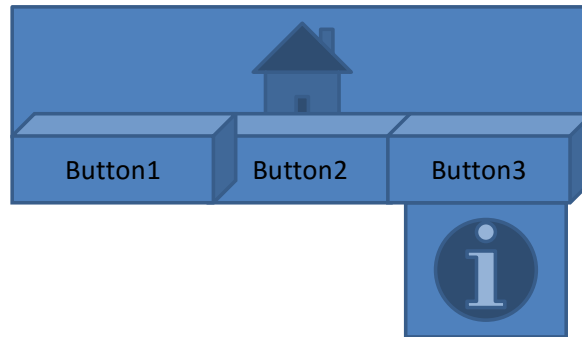


Figure 6.12.GridBagLayout

- **Cell**

A cell represents a component which is part of a layout. The cells act as containers for the interface components.

6.5 Deriving the EA00-H Presentation Model

The presentation model has a one to one correspondence with the navigation model. All the information represented by the navigation model should be presented in the presentation model. Its construction is as follows:

- Identify the presentation class.
- Each navigation node is represented by a presentation node in the EA00-H presentation model. The structure of that presentation node is to be determined based on the information to be displayed on that presentation page.
- This leads to the identification of structure nodes at level zero of the DPD into a page chunk, presentation page, frameset, frame and window.
- Further, identify the layout of the structure nodes to be presented at level one as cell, boxlayout, borderlayout and gridbaglayout.
- Next, determine the relationships between the presentation nodes, and divide them into navigation, builds, submit, contains and displays etc.
- Specify the most relevant attributes and operations of the

presentation class. For seamless transformation of the presentation WebGRL model into the EAOO-H Presentation design model a set of transformation rules have been defined below.

6.5.1. The Transformation Rules for deriving the EAOO-H Presentation Model from the Presentation WebGRL Model

Now we present the Transformation Rules to derive the Presentation Model (PM) of EAOO-H from the presentation WebGRL diagram of WEBGRE framework. The Transformation Rules defined to derive the Presentation Model of EAOO-H approach from the presentation WebGRL diagram are as follows:-

1. *Presentation Goals2Presentation Pages*: This rule results in adding a “home” presentation page to the model, which is an Interface Component or a collection of anchors representing a Menu grouping navigational links. From the “home” Presentation Class (PC) an anchor directs the navigational links to other Presentation Pages with a redirection link to the home page. From each presentational goal with an associated content goal or a resource a Presentational class (PC) is derived. A Presentation Class is required for each Navigation Class in The NAD.
2. *Presentation Soft Goals2Presentation Class conditions*- This rule transforms the presentational softgoals with satisfaction level value expressed as conditions or constraints of the Presentation Class.
3. *Navigation2Navigation Link*: This rule checks navigation between one or more goals, if it is detected, then a menu item or an anchor is used to navigate to other Presentation pages.
4. *Contribution2 Presentation Pages with Navigation Link*: This rule checks if the link is a contribution link then the contributing subgoals are presented as Presentation Pages. An anchor or menu item from the root Presentation Page (PP) navigates to each of the PPs representing the associated presentational goals.

5. *Decomposition2 Navigation Link* : In case of decomposition links the number of navigation links represented by anchor or menu item between the super goal PP and the subgoal PP is as many as the number of decomposed subgoals where each super or sub goal is represented by a PP.
6. *Task 2Interface Components with include cell or Include Frame link*: Tasks linked to presentational goals represent operations of their presentation class or maybe business process goal. They are represented by include frame include cell or an Interface Component. The following interface Components are defined as UML stereotypes: text, anchor, image, video, audio, form, collections and anchored collection. "Their stereotypes and associated icons are defined in (Koch & Mandel, 1999)".

6.5.2. The Transformation Method for deriving the EA00-H Presentation Model from the Presentation WebGRL Model

To derive the DPD we take into account the main presentational goal first. This is represented by the Main or Home page of the website with Frames, Frameset and Presentational Class with Interface Components to represent the content of the Home Page and have navigation links represented by Interface Component of anchored collection to other Presentation Goals represented by Presentation Pages. All Presentation Goals in the Presentation WebGRL diagrams are represented as Presentation pages with Presentation Classes. There is one Presentation Classes for each Navigation Class in The NAD and Index. Tasks associated to the main presentation goal are represented by the interface component on that main page itself using text, image etc.

We travel from the main presentational goal to other presentational goal by adding an Interface Component like anchor or menu item to navigate between the presentation goals now represented by Presentation Pages. The content for each Presentation Goal is

displayed by Presentation Class and its Interface Components on that Presentation Page. We move from top to bottom by creating a Presentational Page for each Presentation Goal with a navigation link to contain or navigate to other pages.

In case of a presentation Goal with requirements for form the Presentation Page must have a display link to the Interface Component Form with submit link. A list or an index is an anchored collection navigating to other Presentational pages. Presentation Goal with Search use build to generate the new Presentation Page each time a search is performed with the help of server pages. Soft goals are represented as conditions of that presentation class of the presentation page to which they are attached. They are defined as member conditions on the model elements, presentation class, association or conditions on the attribute of the presentation class with satisfaction levels.

TABLE 6.1. Transformation from Presentation WebGRL Diagram to Presentation Model or the DPD.

Presentation WebGRL Element	Enhanced A-OOH Presentation Model
Main presentation Goal	Home Presentation page with redirection link
Presentation Goals	Presentation Pages with display link to a window or frameset and associated to a presentation class with interface components
Presentation Goals with dependency link to resources or Content Goal	Presentation Pages displayed using a Window/ Frame associated to a Presentation Class with Interface Components to display

	the content.
Presentation Goals with link to Navigational Goal	Presentation Pages displayed using a Window/ Frame attached to a Presentation Class with Interface Component anchor to navigate from one presentation page to the other presentation page
Presentation Soft Goals	Conditions of the Presentation Class
Contribution Link	Presentation Page for the contributing goal and Interface component Anchor to that Presentation Page
Dependency Link	Builds, Submits
Decomposition Link	Navigation Links or Contains
Tasks with Means end link	Include Frame and Include Cell for Interface Components like text, image, anchor, audio, video etc.

The zero level of the DPD shows how the navigation nodes are grouped into abstract pages (and page chunks in the case) in the Website. In this case, each of the Navigational Nodes correspond with one Presentation Page, with the presentation class corresponding to its resource or the content class and the information on the page is represented with the help of Interface Components.

6.5.3. The Transformation Algorithm of deriving the Presentation EAOO-H model from Presentation WebGRL Diagram

Based on the method shown above we have proposed a transformation algorithm for the transformation of the presentation WebGRL diagram to the EAOO-H Presentation Model. The Input to this algorithm is the Presentation WebGRL diagram to which the transformation algorithm steps are applied as per the transformation rules stated above in section 6.5.1 to generate the EAOO-H presentation model as an output. This transformation algorithm identifies the concepts of the presentation WebGRL model to be transformed into the presentation nodes and presentation links of the EAOO-H Presentation Model. Further, it specifies the components required to present the level one and level zero of the DPD. The final result is the complete EAOO-H Presentation Model for the given Presentation WebGRL diagram given as input. The pseudo code and the terminology used for defining the Presentation WebGRL diagrams is as follows:-

A is a parent goal with collection of links

X_i 'sare subgoals $i=1$ to n

S_i are softgoals $i=1$ to n

R_iare resource $i=1$ to n with resource name **R**

AX(L_i) are the navigation links attached from **A** to **X** for $i=1$ to n ;

P is the main presentation goal used to represent an assortment of links for $i=1$ to n ;

P_i is the presentation goal for $i=1$ to n ;

PP_i is the page chunk or presentation page for a presentation goal **P_i** for $i= 1$ to n which may be displayed using a window or a frameset.

M_i are means end task attached to a subgoal or a parent goal which will be presented as Interface Components, Include cell or include frame of that Presentation Page for the parent or subgoal

PC_i are the presentation classes associated with a frameset to display content using Interface Components for each navigation classes for $i=1$ to n .

IC_i are the Interface Components to represent the content on a frameset or frame using image, text, audio, video, anchor or a form.

IF_i are the Include cell or include frame components for $i= 1$ to n .

AX (A_i) represent the Interface Components anchor or a collection of anchors for $i=1$ to n .

DC_i (R_i) are the domain classes for goals attached to a resource R for $i=1$ to n .

6.5.4. # EA00-H Presentation Model Transformation Algorithm

(Rule: Check for main presentation goal and its subgoals for defining the home page and their presentation pages for the subgoals attached to it by links now represented as anchors. Derive the components required at level zero and level one to present a website while moving from top to bottom. Type: Transformation)

Input= Presentation WebGRL Diagram

Do Create P_{Pi} associated to a P_{Ci} for each corresponding N_{Ci} in the Navigation Model;
For A create a homepage with a collection of anchors i.e. Main or Homepage P_i;

Do for each level till done for levels $i=1$ to n

Foreach navigation link from P_i to its subgoals check the type of link

if AX(L_i) is a simple navigation link

then AX(L_i)= AX(A_i) between A_i and X_i where A_i is the parent goal and X_i is the subgoal

if AX(L_i) is a simple contribution link between A and X_i

then AX(L_i)= AX(A_i) between A_i and X_i where A_i is the parent goal and X_i is the contributing presentation goal

create A_i and X_i= P_{Pi} associated to a P_{Ci}

if AX(L_i) is a decomposition link between A and X_i

then AX(L_i)= AX(A_i) between A_i and X_i where A_i is the parent goal and X_i is the sub presentation goal

create A_i and X_i= P_{Pi} associated to a P_{Ci}

if AX(L_i) between A_i and X=M_i where M_i is some means end task

then M_i= IF_i or IC_i

endif

endif

endif

endif

endforeach

enddo

endfor

enddo;

Output= EA00-H Navigation Model

6.6 Example of Online Bookstore

We extend the example of the Online Bookstore whose WebGRL Presentation diagram is shown below. We apply our transformation rules stated in the section 5 above to this Presentation WebGRL diagram in fig. 6.13 to generate the DPD for the Online Bookstore.

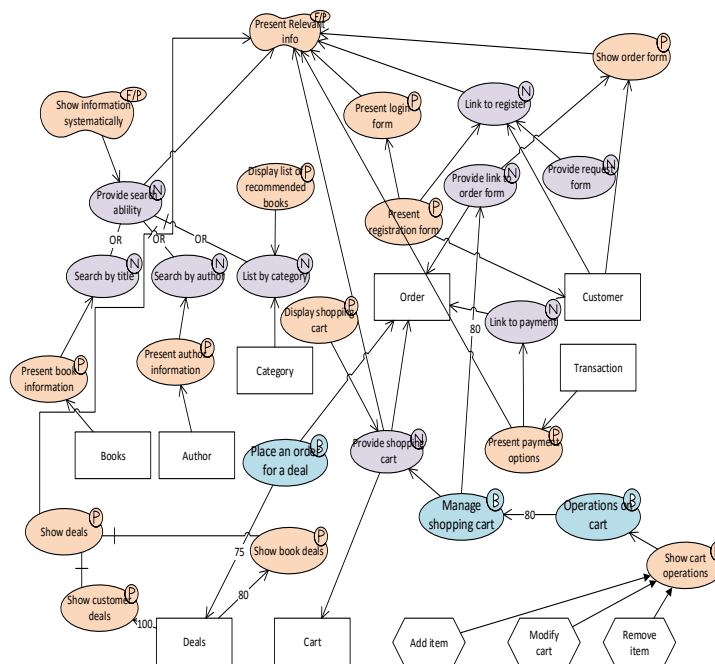


Figure 6.13. The Online BookStore Presentation WebGRL diagram

The main presentation goal of *Provide relevant links* is presented as per rule1 by the Main or *Home* page of the website with Frames, Frameset and Presentational Class with Interface Components to represent the content of the bookstore *Home* Page and have navigation links represented by Interface Component of anchored collection to other Presentation Goals represented by Presentation Pages as shown in fig. 6.13.

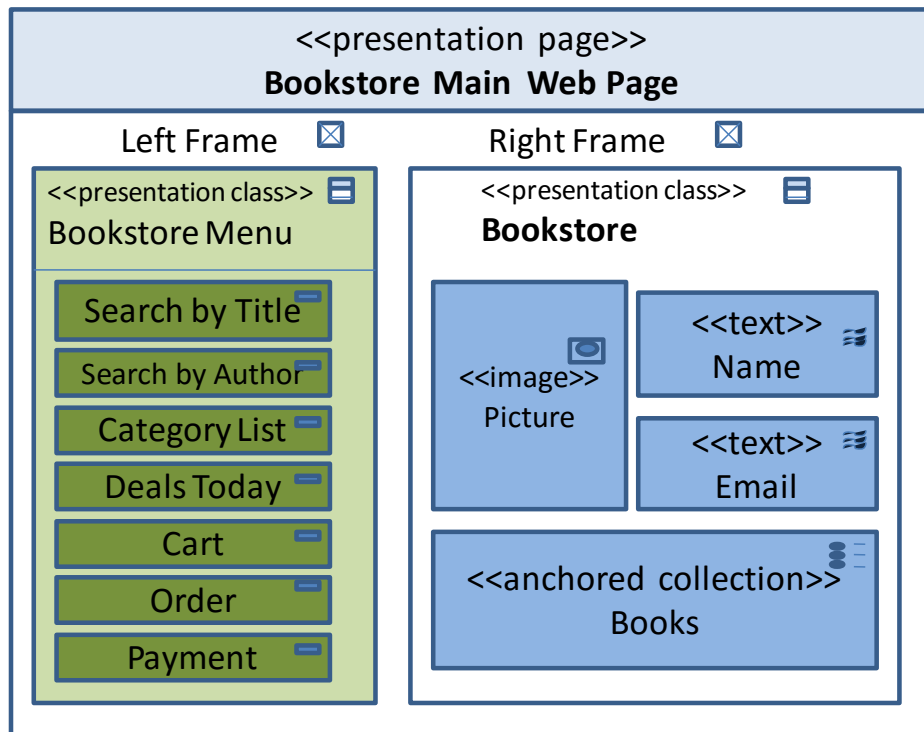


Figure 6.14.BookStore Home Presentation Page

In the second step we trace the navigation links and the presentation goals attached to them. The first Presentation goal to *provide search ability* is satisfied by contribution links to presentation goals of *Search by title*, *Search by author* and *List by category*. All these navigation and contribution links are represented on the *Home* page menu as anchors to the respective Presentation pages of *Search by title*, *Search by author* and *List by category*. The Presentation goals of *Provide Shopping Cart*, *Show Deals*, *Show Payment form*, *Provide Payment Link* are also represented on the *Home* page menu as anchors to the respective

Presentation pages of Provide *Shopping Cart*, *Show Deals*, *Show Payment form*, *Provide Payment Link*. This gives us the transformation of the first level of navigation links and presentation goals in the Presentation WebGRL diagram shown in Figure 6.15

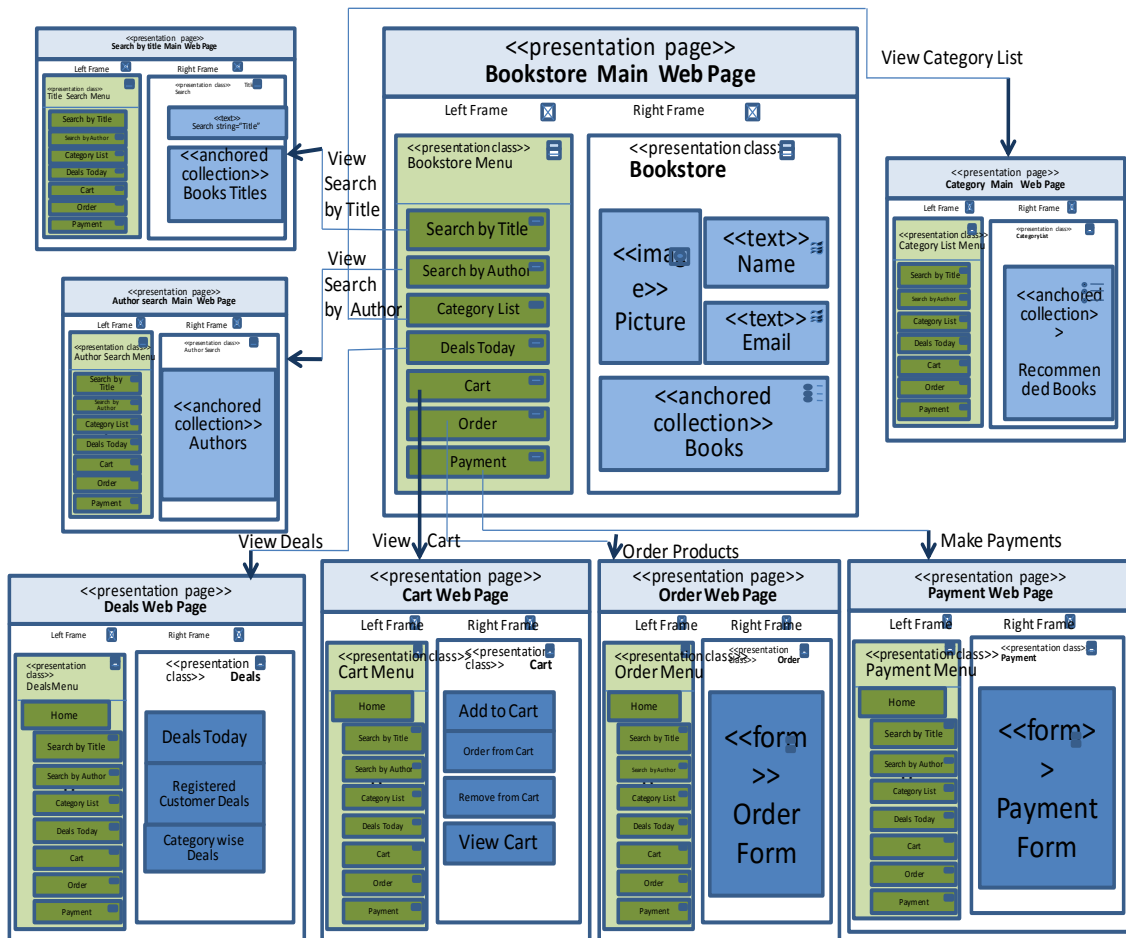


Figure 6.15.Level 1 of the DPD for the BookStore Home Presentation Page.

Further contribution links to presentation goals of *show book deals* and *show customer deals* leads to an anchor in the *deal* Presentation Page to new Presentation Page of *Show Deal details*.

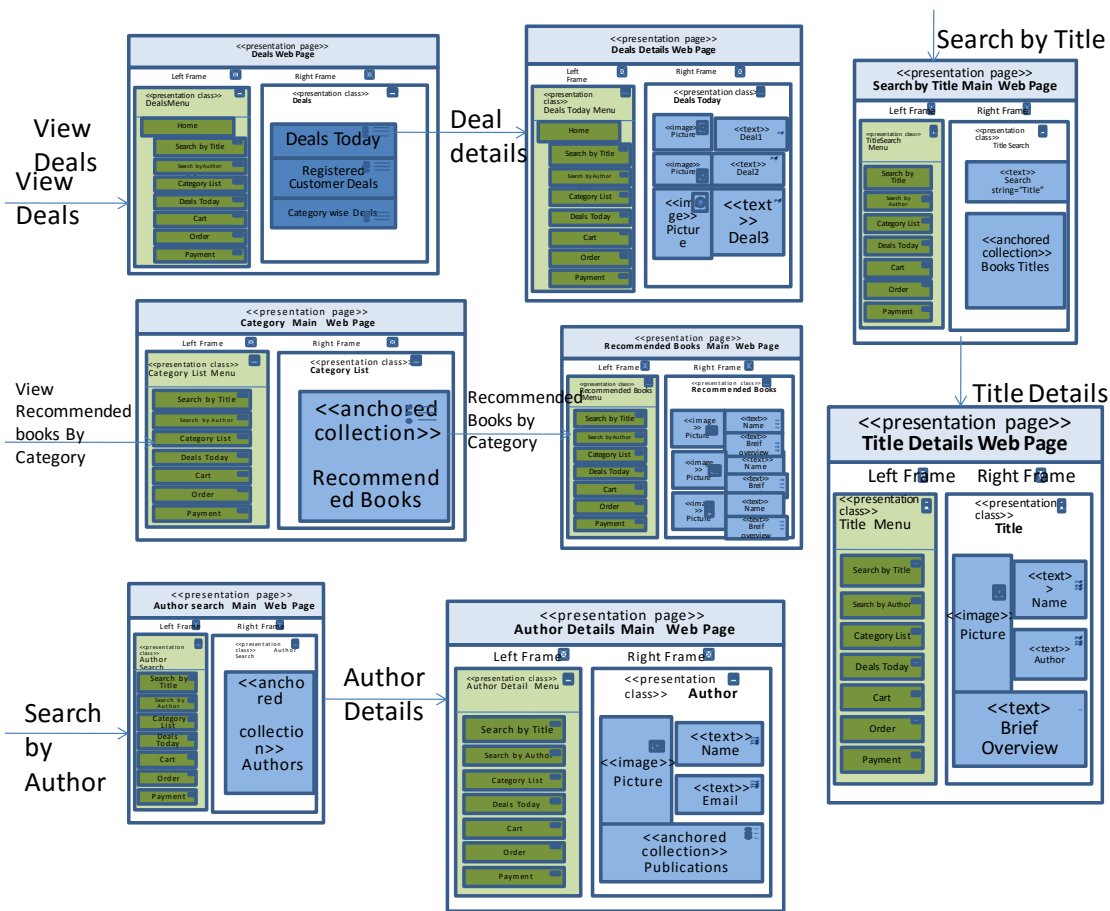


Figure 6.16. Expanded Level 1 of the DPD for the BookStore Home Presentation Page

Similarly following the navigation links from the presentation goals of *List by category*, *search by title* and *search by author* lead to new presentation pages showing the result of list and searches, which may be an anchored collection further leading to other new presentation pages which may be forms or search results. Thus detailed refinement of the presentation WebGRL diagram would give us the final DPD shown in Figures 6.15 and Figure 6.16 above.

In this chapter we have presented the Enhanced EAOO-H Presentation Model and its concepts. Further we have discussed the process of transformation of a WebGRL presentation diagram to an EAOO-H presentation model using the transformation rules. Transformation Algorithm which has been further explained with the example of online bookstore in a separate section. In the next chapter we propose the UML Profile for the three EAOO-H

Design Models namely the Domain, Navigation and the Presentation Models. By doing a web specific UML profiles for the content, navigation and presentation Web GRL models which are easily portable across different platforms and can be used easily by a developer for generating the code using any object oriented programming environment.

Chapter 7

UML Profile for the WebGRL Requirements Model and EA00-H Design Models

In this chapter, we move from the design stage towards the implementation stage. While adhering to the web based goal oriented requirements engineering we take a step from the requirements to the design stage and thereafter we take a small step towards the construction stage using the intermediary EA00-H design model. A UML Profile for the web specific content, navigation and presentation design models has been presented in this chapter, resulting in a platform independent output. UML Profile makes the job of the Software Engineer simpler by providing a UML Compliant, UML Profile which can easily be used for the coding purpose by using an object oriented design language supported by the OMG's UML Metamodel. This provides a straightforward method of adaptation to the UML Meta model in the construction phase. This forms the last leg of our thesis work. This entire transformation process has been automated and presented in the next chapter. In this chapter we explore the UML Profile for the EA00-H design Models.

7.1 Introduction

In the previous chapters we have described the Enhanced EA00-H model and the transformation of the web specific WebGRL model namely the content navigation and the

presentation models to their respective domain, navigation and presentation EAOO-H design models. This leads to a seamless transition necessary from the requirements to the design stage. The advantage of doing so is that we are using the GOREWEB framework that not only focuses on some important aspects of web applications like content, navigation, presentation etc. and also caters to the concepts of design decision, dispute resolution, and ordered goal primacy which provides a more holistic view of the web application being developed. The factors listed above stating the advantages of use of GOREWEB framework helps us in using a requirements engineering input base of a very high quality for the transformation to the design phase. The advantage of using the WebGRE approach is that most of the problems have already been tackled in the requirements engineering detailed analysis phase and we have a very refined input to our transformation approach.

In order to ensure that the quality of the requirement engineering input is maintained in the output design phase also we require a lossless transformation. Thus we have enhanced the chosen A-OOH design model into the EAOO-H design model which helps in faultless transformation of the web specific WebGRL models to their EAOO-H design counterpart models. This entire process of transformation of a web application necessary for design phase has been described in detail in the previous chapters 3, 4, 5 and 6.

The requirements analysis of the WebGRL results in the Web Specific GRL. A number of UML Profiles or UML compliant approaches exist for transformation from requirement phase to design phase like UWE, OOH, and OOHDM for web engineering. However, A-OOH modeling method is very close to our approach. With the help of A-OOH we map the refined analysis models to their respective design models. This approach is very close to the web specific diagrams generated in the analysis phase in our previous work so we have adopted this approach for our research work and enhanced it as well as modified it to EAOO-H approach to define the web specific models of WebGRL in the design phase into their respective design models with traceability. This has led to enhancement and development of our own UML profile.

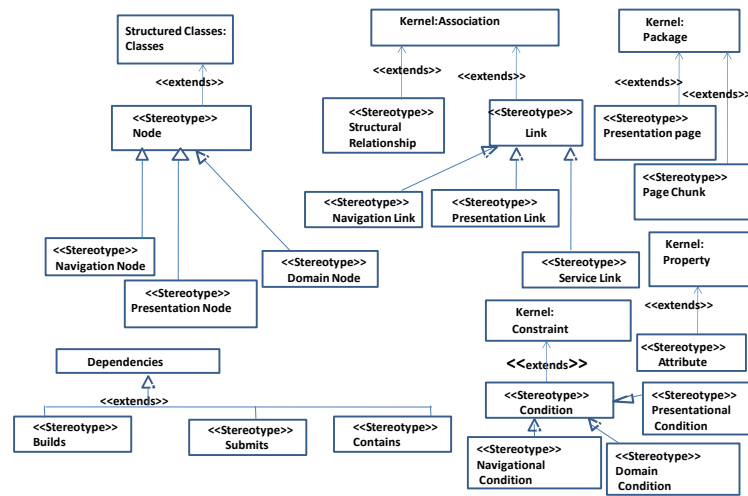
This step from the design stage towards the construction stage ensures that the EA00-H domain, navigation and presentation design models generated by the transformation process are UML Compliant i.e they are supported by a UML Profile. The advantage of ensuring that the EA00-H domain, navigation and presentation models are supported by a UML Profile makes the job of the Software Engineer simpler by providing a UML Compliant UML Profile which can easily be used for the coding purpose by using an object oriented design language supported by the OMG's UML Metamodel.

In this chapter the UML profiles defined for the EA00-H domain, navigation and design models are presented. The purpose of defining an UML profile is to provide an easy mechanism of adaptation to the UML meta model to elements that are specific to a particular domain, platform or method. In this sense, the particular profile for the DM,NAD and EDPD consists in adapting the elements defined in these design models to the UML metamodel. In this way the profile allows to specify the EA00-H design model elements in any commercial tool that supports UML.

7.2 UML Profile for the EA00-H Design Models

In this part we will shortly describe the extension mechanism of UML along with the UML Profile for the EA00-H design models in the figure 7.1 below. The UML Profile of the EA00-H Design models is based on the definitions and the diagrams explained in "Object management Group (OMG) Unified Modelling Language specifications (www.uml-diagrams.org). Profile Diagram is a structure diagram which describes lightweight extension mechanism to the UML by defining custom stereotypes, tagged values, and constraints in OMGs's Meta Object Facility (MOF), UML and UML superstructure specifications. Stereotypes, tagged values and constraints represent the built-in extension mechanisms of UML. UML represents a family of modeling languages, rather than a single language. By Lightweight we mean that it can be easily supported by tools and it does not impact the interchange of formats as stated above. Profiles allow adaptation of the UML metamodel for different platforms ".

The elements of a standard "UML Metamodel" can be specialised and represented in a "UML Profile". A "UML Profile" only permits modification of an existing metamodel by adding constructs for a particular domain, platform, or method. Existing restrictions will hold while new ones can be added for a particular profile as given in "OMG's UML Specifications". A profile is used to describe the modifications, which is then applied to a package. "Stereotypes are specific meta classes, tagged values are standard meta attributes, and profiles are specific kinds of packages (uml-diagrams.org)".



UML PROFILE FOR EA00-H DESIGN MODELS

Figure 7.1. UML Profile for the EA00-H Design Models

Now we will present the elements of the UML Profile Diagram used for defining the UML profile for the EA00-H Design model namely the stereotype, tagged value, constraints, package, association, class, property and dependencies. The OMG's UML profile Diagrams and UML profile diagram specifications have been used by us in defining the extensions to the UML Profile for the domain, navigation and the presentation design models.

7.2.1. Stereotypes

A UML stereotype is defined in "Object Management Group's (OMG) Unified Modelling Language (UML) as a new kind of model element defined within the model based on an existing kind of model element. *Stereotype* is a *profile class* which defines how an

existing metaclass may be extended as part of a profile. It enables the use of a platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass as defined in OMG's catalog of UML Profile specifications (uml-diagrams.org) ". A stereotype is used in conjunction with the metaclass it extends it cannot be used alone. Nor can it be by another stereotype. The notation for a stereotype is the same as that of a class, with the keyword «stereotype» There should not be a clash between the keyword names and the stereotype names. "As stereotype is a class, it may have properties which are called tag definitions. The values of the properties are referred to as tagged values whenever a stereotype is applied to a model element as specified in OMG's UML superstructure specifications".

There are both advantages and risks associated with this potent extension mechanism. The advantage is that by using clearly defined modeling elements it is possible to create modeling languages for special application domains. Some of the already defined examples of such extensions are the real-time OMG's UML Version 1.4, the business (Conallen 1999) and the Web domain (Koch et al 2000). Similarly the risk associated with this extension mechanism is the extreme use of stereotypes resulting in a complex modelling language. Some stereotypes append or characterize the restrictions on the metamodel elements, while others only change the notation of a modelling element serving as a comment.

7.2.2. Tagged Values

Stereotype is used in class diagrams, use case diagrams, line diagrams, etc. As per the OMG's UML Profile diagrams, "when a stereotype is used for model element then the values of its properties may be referred to as tagged values. UML 1.x defined tagged value as one of UML extensibility mechanisms permitting arbitrary information (which could not be expressed by UML) to be attached to UML Superstructure specifications. Tagged value is a keyword-value pair which may be connected to any kind of model element. A UML tagged value is a (tag, value) pair that allow any information to be attached to a model element. The keyword is called a tag (uml-diagrams.org)".

"Tag" presents a special "kind of property" suitable "to one or many kinds of model elements". The "tag" and its "value" are converted into a datatype however most of the time it is encoded as a string. "Tagged value specification in UML 1.x has the form name = value".

The tagged value name is specified as the name of a tag or model quality. The value is represented by a string which indicates its value. Tagged value or model quality is shown as a comma to fix sequence of properties inside a pair of curly braces "{" and "}". This data is shown in the form of text. It is generally used to store non-functional requirements or project management information. The interpretation of the value is a convention between the modeler and the modelling tool.

7.2.3. Constraints

According to "OMG's UML Profile Diagrams a constraint has been defined as a condition or restriction that allows new semantics to be specified linguistically for a model element. A constraint is a packageable element which represents some condition, restriction or assertion related to some element (that owns the constraint) or several elements. Constraint is usually specified by a Boolean expression which must evaluate to a true or false. Constraint must be satisfied (i.e. evaluated to true) by a correct design of the system (Uml-diagrams.org)". Various elements of class diagrams utilize constraints an example of the same is the pre and post conditions of an operation. The constraint may or may not have a name. "A constraint is shown as a text string in curly braces according to the following syntax from catalog of UML Profile Specifications:

Constraint: = '{ (name ':') Boolean-expression }'."

There are no restrictions on languages for expressing the constraints. Some of the constraint languages are OCL, Java, some machine readable language or a natural language. Uml has already defined the constraint language OCL however the UML tool can use any other constraint language also. "For an element whose notation is a text string

(such as a class attribute), the constraint string may follow the element text string in curly braces".

7.2.4. Package

A package in the Unified Modeling Language is defined in "OMG's UML Profile diagrams and is used to group elements, and to provide a namespace for the grouped elements. A package may contain other packages, thus providing for a hierarchical organization of packages. Pretty much all UML elements can be grouped into packages as per OMG's UML Profile diagram specifications (uml-diagrams.org)".

7.2.5. Association

Association has been defined by the "OMG's UML diagrams as a relationship between classifiers which is used to show that instances of classifiers could be either linked to each other or combined logically or physically into some aggregation of a UML Profile Diagram. UML specification categorizes association as semantic relationship (uml-diagrams.org). Some other UML sources define association as a structural relationship or a link as per OMG's UML Profile diagram specifications".

Links between objects are illustrated by structural relationships, there are labels associated with them to depict the number and the role of the links. Association could be used on different types of UML structure diagrams there are many other concepts related to association like Aggregation type, navigability, and end ownership.

7.2.6. Class

The definition of class in "UML Profile diagrams says that a class is a classifier which describes a set of objects that share the same features, constraints, or semantics (meaning). A class is shown as a solid-outline rectangle containing the class name, and optionally with compartments separated by horizontal lines containing features or other members of the classifier. Features of a class are attributes and operations. When class is shown with three compartments, the middle compartment holds a list of attributes and the bottom compartment holds a list of operations (uml-diagrams.org)". Attributes are used to

depict the properties of a class at times it may represent the navigable ends of binary associations. Every object of a class should have a value for every attribute of that class based on its characteristics. Attributes or operations may be grouped together based on their perception. A keyword or symbol can be attached for multiple features with the same visibility. Classes may have additional compartments to show other details, or for partitioning different features.

7.2.7. Property

Property is defined “as a structural feature which could represent an attribute of a classifier, or a member end of association, or a part of a structured classifier of UML profile diagrams. As a structural feature, property represents some named part of the structure of a classifier as per OMG's UML Profile diagram specifications (uml-diagrams.org)”.

7.2.8. Dependencies

Dependency has been described in "UML Profile Diagrams as a semantic relationship in which a change on one thing (the independent thing) may cause changes in the other thing (the dependent thing).It is a directed relationship which is used to show that some UML element or a set of elements requires, needs or depends on other model elements for specification or implementation uml-diagrams.org)". Dependency can be a relationship between a number of different elements like classes, interfaces, components, artifacts, packages etc.

After defining the UML Profile Diagram extension constructs used for extension mechanism to describe the UML profile for the EAOO-H design models we move further and use these extension constructs in defining the UML Profiles for both the web specific WebGRL diagrams and their corresponding design models. The UML profiles have been defined by us for both the web specific WebGRL diagrams and the EAOO-H design models. However, in the later sections though we present the correspondence between the WebGRL elements and the UML Profile we do not go into the depths of the same as we would require the UML Profile only after the design phase. Since we use the EAOO-H design model for the design

phase it is more appropriate to present the UML Profiles for the EAOO-H Design models which can be used in diverse application domains such as banking, internet, aerospace, healthcare as well as in implementation on different platforms like J2EE, .NET etc. In the later sections we present the UML Profile for the domain navigation and presentation EAOO-H design models.

7.3 UML Profile for the EAOO-H Domain Model

As aforementioned a profile for the Domain Model has been defined. Extending the UML concepts of class and association the Domain Model concepts are specified. The main purpose of defining an UML profile is to ensure simple transformation to the UML model elements for a particular domain or platform. In this sense, the particular profile for the Domain Model consists in adapting the elements defined in the Domain Model to the UML metamodel. In this way the profile allows to specify the Domain Model elements in any commercial tool that supports UML.

The EAOO-H Domain Model elements are the Domain Node, the Domain Class, Structural Relationships, Attributes, Operations, Links, and Conditions. The Domain model represents the content design of the web application. We further transform this using the Enhanced A-OOH design models as intermediary to map the WebGRL content diagrams into a UML Profile as explained below. The transformation of the content WebGRL diagram to the Domain EAOO-H model has been explained earlier by us in chapter 4.

- **Domain Node**-The "stereotype <<Domain Node>> in the EAOO-H Domain model is described as an expansion of the UML class concept with attributes and operations" which are again extensions of the UML concepts.
- **Domain Class**-Similarly the "stereotype <<Domain Class>> has been described using an expansion of the UML Class concept as above with attributes and operations". Regarding the Domain Class concept the following information is stored: The Domain Class has information about the Resource of the Content Model from which it is derived and stored in the tagged value rootConcept.

- **Link**-The "stereotype <<Link>> is defined by the extension of the UML Association concept". An association signifies a set of links. A binary association denotes a link between two domain classes. An association can also be used to link any number of classes. An association may have a name and the ends of an association may have properties like multiplicity, visibility, role names and ownership indicators. There are different types of association: bi-directional, uni-directional, and Aggregation.
- **Structural Relationship**- The "stereotype <<Structural Relationship>> is used to describe an extension of the UML Association concept". Association between one or more than one elements is depicted by the structural relationship. In case of a relationship between a set of source and a set of target elements it is called a directed relationship which has subclasses namely the generalization and dependency. Generalization is used to depict a directed relationship between a superclass and a subclass. Dependency is used to depict a directed relationship based on need or dependency by some UML model elements on other model elements, for example in the specification and implementation of a resource concept of the Content WebGRL diagram.
- **Condition**-A "stereotype <<condition>> is defined as an extension of UML constraint concept". Constraint represents some type of condition or restriction associated with one or more elements that own it. The constraint value is denoted by a boolean expression which has a value of true or false. A condition can be a navigational condition that applies a constraint on the navigation from the source to the domain class or it may be a domain condition which is a constraint associated with the domain class or the attributes or operations of a domain class.
- **Attribute**- A "stereotype <<attribute>> is defined as an extension of UML property concept". Property represents a structural characteristic that denotes an "attribute or a member end of association, or a part of a structured classifier of a domain class or a link".

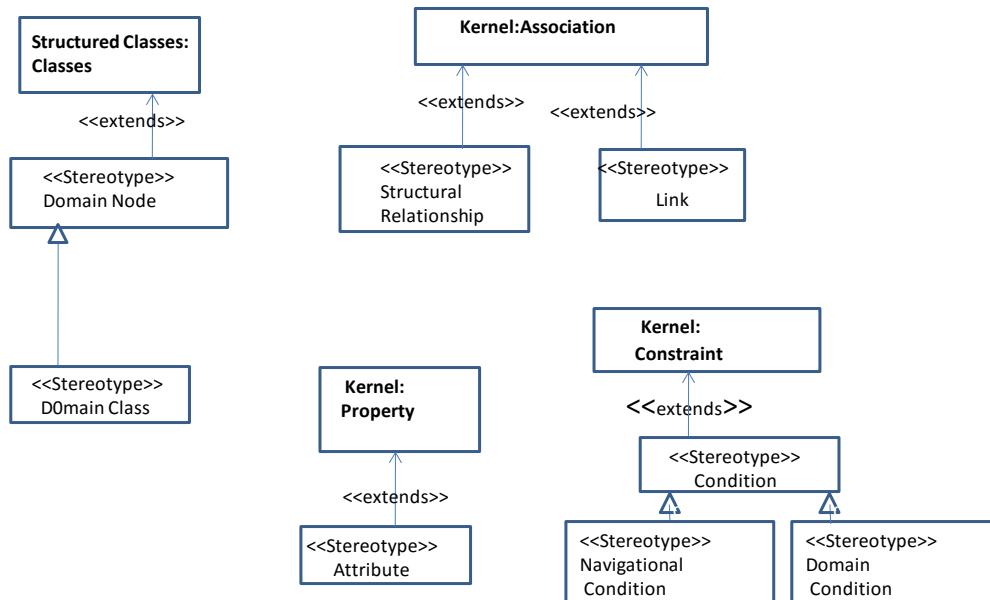


Figure 7.2. UML Profile for Domain Model

All content Goals in the WebGRL diagrams are represented as Domain Nodes represented as Domain Classes for the resource used by them to satisfy that goal of the EAOO-H Domain Model. The goal always states a function which is to be carried out. That function may be a service task or a navigation task. If it leads to navigation to another domain class then it is a navigation task. However, if it requires performing an operation on the attributes within the domain class defined for that content goal then it is a service task. The operation within the domain class would be performed on some content variables or patterns within the domain class itself. These variables or content pattern are to be represented as the attributes of the domain class with operations in the domain class defined on them. If the operation to satisfy this goal needs a navigation to obtain information from another class then a relationship is defined between the two domain classes. Softgoals are represented as conditions of that domain class defined as member conditions on the model elements, i.e. domain class, association or conditions on the attribute of the domain class with satisfaction levels. All these elements are represented in the UML Profile and UML Metamodel for the Domain Model in the figure 7.2 above.

**Table 7.1.UML METACLASS for WebGRL Content Model and
Domain Model**

WebGRL Content Model elements	Domain Model Stereotypes	Stereotyped UML Metaclass
Content GRL spec	Domain Model	Model
Content GRL model Element	Domain model Element	Named Element
Actor	Domain Class	Class
Intentional Element Content Goal & Resource	Domain Class	Class
Intentional Element Softgoal	Condition	Constraint
Intentional Element Task	Domain Class Operation	Operation
Element Link	Link	Association
Contribution	Relationship	Association
Contribution Type	Contribution type attribute	Enumeration
Decomposition	Structured Relationship	Association
Decomposition Type	Decomposition Type	Enumeration
Dependency	Association	Association
Intentional Element Content Goal & Resource	Domain Class Attributes	Property

7.4 UML Profile for the EAOO-H Navigational model

Now we present the UML Profile for the EAOO-H Navigation Model by extending the UML concepts of class, constraint, property, dependencies and association. The purpose of defining an UML profile is to provide an easy mechanism of adaptation to the UML metamodel to elements which are special for a particular area, platform or method. Therefore, the particular profile for the Navigation Model consists in adapting the elements defined in the Navigation to the UML metamodel, allowing the Navigation Model elements to be represented in any commercial tool that supports UML.

The EAOO-H Navigation Model elements are the Navigation Node, and the association between the nodes that denotes the "navigation paths the user can follow using the navigational links" in the final website. The EAOO-H NM has three kinds of nodes namely the Navigation Class, Menu, and Access Primitives like Query, Index, Showall and Guided Tour. Navigation Links denote the paths followed by the user for navigation within the system these are namely the "traversal link, means end task, decomposition and contribution links" (Srivastava 2014). There are also Navigation Attribute, Operations and Navigation Conditions. The Navigation model represents the navigation aspect of the web application. A UML Profile for using the transformed Enhanced A-OOH navigation design model is explained below. The transformation of the navigation WebGRL diagram to the Navigation EAOO-H model has been explained earlier by us in chapter 5. The two main concepts of the navigation design model are the navigation nodes and the navigation links.

- **Navigation Node-** The "stereotype << Navigation Node>> has been defined as an extension of the UML class concept which has attributes and operations (also extensions of the UML concepts) (uml-diagrams.org)". There are three types of navigation nodes namely the navigation class, menu and access primitives. "Navigational Classes" are a view of the domain classes. Access primitive are the

Index, Showall and Guided Tours or queries that are used in fulfilling navigation requirement of the client and Menus or Collections are the probable hierarchal structures defined in Navigational Classes.

- **Navigation Class**-The "stereotype <<Navigation Class>> has also been defined by the extension of the UML Class concept and has attributes and operations". Each Node has associated with it a root concept from the DM attached to it by the notation: "Node: DM.RootConcept". These are domain classes augmented with attributes and operations which are visible to the client based on the permissions granted to him as well as his navigational necessity.
- **Access Primitive**-The stereotype <<Access Primitive>> has also been defined by the extension of the UML Class concept. Besides the navigation nodes which need to access navigation objects to satisfy the navigational requirements of the user. Access primitives is as UML stereotypes by the extension of the UML Class concept: index, guided tour, showall and query.
- Third type of navigational nodes is denoted by access primitives. These are used to access navigation objects to fulfill the navigational needs of the user. The access primitives defined as UML stereotypes by the extension of the UML Class concept are as follows: index, guided tour, showall and query. The following modeling elements are used for describing indexes, guided tours and queries.
 - **Index** - The stereotype << index>> is a member of some index class with a corresponding icon. An index class is an extension of the UML Class concept. Index is used for directentrance to the instances of a navigation class for which it has been defined. The index is a composite object with one or more index items. Each index item is used to identify and relate to an instance of a navigation class. The index composition and the association between IndexItem and NavigationClass is used to determine the relationship between Index and Navigation Class.

- **Guided tour-** A guided tour is a sequential access to navigation class instances. For classes, which contain "guided tour objects we use the stereotype «guidedTour» an extension of the UML Class concept". Each "NextItem" must be connected to a navigation class.
- **Query-** A query class is denoted by a class which has a query string as an attribute. For "query classes we use the stereotype «query»". This string is given by the user.
- **Show All -:** A show all provides navigation without indexing and without internal navigation, all the objects are shown in the same abstract page. This is modeled by the stereotype«<<Showall>>».
- **Menu-**Menu is a composite object made up of named menu items corresponding to a group of navigation classes and links. A menu item points to an index, guided tour, query or an instance of navigation class. Another most common collection type is the concept of menu grouping navigation links. "Any menu is an instance of some menu class which is stereotyped by" «menu» which "must be built conform to the composition structure of classes" described earlier.
- **Navigation Links-** The navigation link defines the navigational paths that the user can follow through the system. The "stereotype <<Link>> is defined by the extension of the UML Association concept". An association is used to denote a group of links. An association can be unidirectional or bidirectional. Similarly it can be unary, binary or n-ary. A binary association is normally represented between two navigation nodes. An association can link any number of navigation nodes to depict a relationship between them. Associations have names and properties like role names, multiplicity and ownership details. There are four types of navigation links: Transversal links which are used to describe the movement from one navigation node to the other, the Means End Link are used to activate an operation that changes the business logic and points to the node where the service results are stored. The contribution link and decomposition links which will be represented as associations in the navigation model.

- **Traversal Links**- They are defined between two navigational nodes i.e. the navigational classes, menu or access primitives. The navigation performed is done to show information through the user interface, without modifying the business logic. This type of links is represented by the "stereotype <<TransversalLink>> and has associated with it the source and target node which it is used to link".
- **Service Links** - Navigation is used to activate an operation which changes the business logic and points to the node where the service results are stored. It is established when a service of the navigational class is activated. This type of link is represented by the "stereotype <<Service Link>> and has associated the name of the invoked service".
- **Decomposition Link**- If a navigation link decomposes into one or more navigation links to satisfy a navigational goal then it is a decomposition link. The decomposition link is represented by as many traversal links between Navigation Classes as the number of decompositions of the navigation goal into its navigation sub goals. This type of link is represented by the stereotype <<Decomposition Link>> and has associated the names of the supergoal and the subgoal into which it represents a decomposition link.
- **Contribution Link**- A contribution link is represented by a traversal or service link depending on the goal requirements. This type of link is represented by the stereotype <<Contribution Link>> and has associated the name of the source and target navigation nodes or the invoked service.
- **Condition**- A "stereotype <<condition>> is defined as an extension of UML constraint concept. Constraint is a packageable element representing some condition, restriction or assertion related to some element (that owns the constraint) or several elements (uml-diagrams.org)". The value of a constraint is denoted by a boolean expression whose value is either true or false.

- **Attribute**- A "stereotype <<attribute>> is defined as an extension of UML property concept". Property represents a structural characteristic that denotes an "attribute or a member end of association, or a part of a structured classifier of a navigation class or a link".

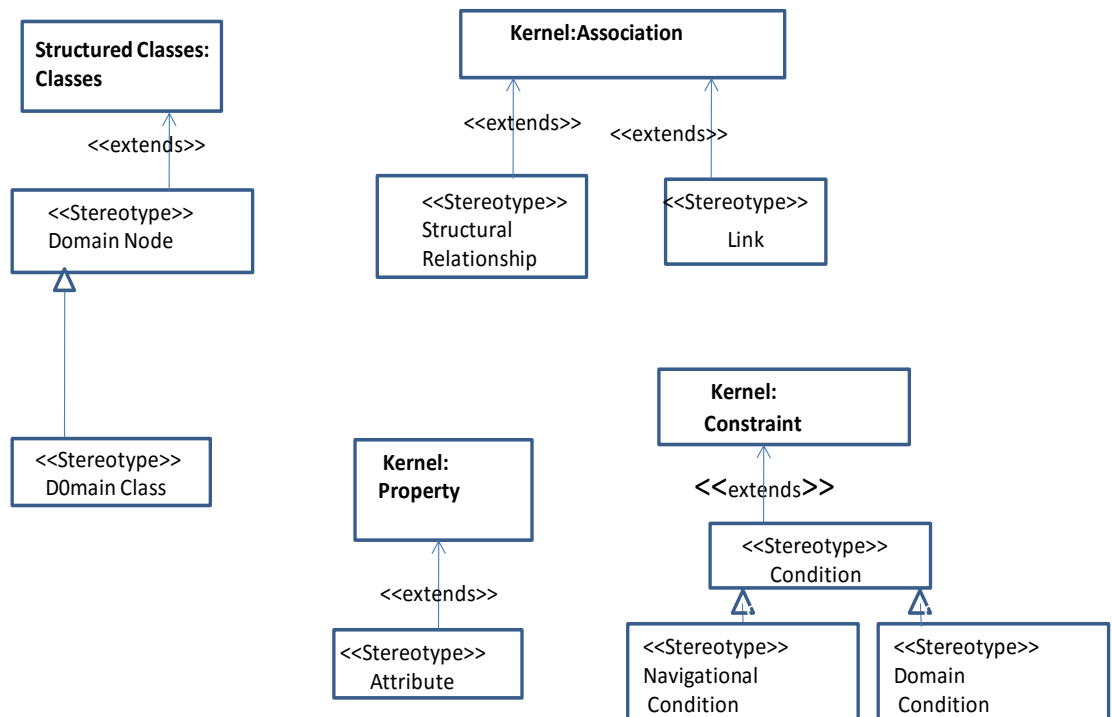


Figure 7.3.The Navigation Metamodel

In order to develop the NM we make use of the WebGRL navigation goals. All navigation Goals in the WebGRL diagrams are represented as Navigation Classes for the resource used by them to satisfy that goal. They are extracted from the domain classes of the Domain model. The navigation goal always states a navigation task that leads to navigation to another content class then a traversal link is added between the two navigation classes that is supported by the access primitives depending on the requirement stated in the goal. Softgoals are represented as conditions of that navigation class defined as member conditions on the model elements, i.e. navigation class, association or conditions on the attribute of the navigation class with satisfaction levels.

The NAD "Navigational Node is an extension of the UML class concept which has attributes and operations". The different types of navigation nodes are represented by different stereotypes i.e. <<NavigationalClass>>, <<Menu>>, <<Index>>, << Guided Tour>>,

<<Query>> and <<Showall>>. The Navigational Class concept has information about the name of the root concept in the Domain model stored in the tagged value rootConcept.

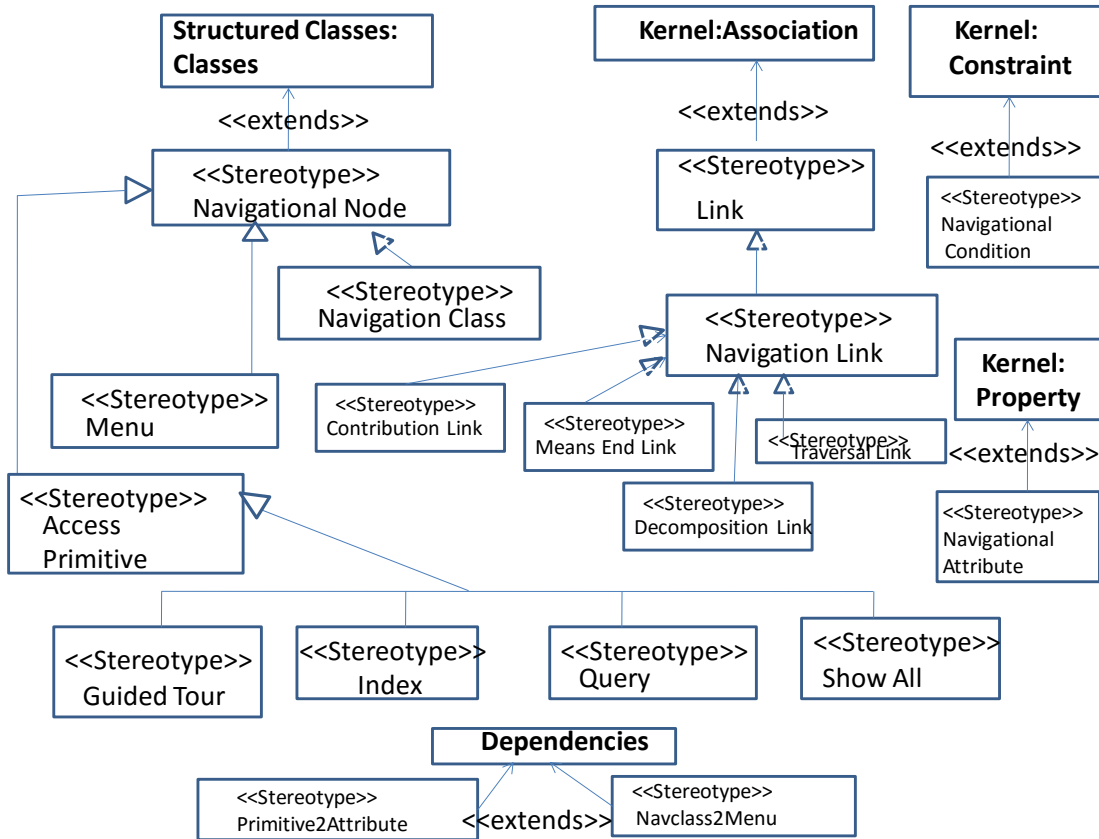


Figure 7.4.UML profile for NAD

All these elements are represented in the UML Profile and UML Metamodel for the Domain Model in the Table 7.2 below.

Table 7.2.UML Metaclass for Navigation WebGRL and Navigation Model

Navigation WebGRL Model Elements	Navigation Model Stereotype	Stereotyped UML Metaclass
Navigation WebGRL spec	Navigation Model	Model
Actor	Navigation Node	Class
Navigation Intentional Element Goal & Resource	Navigation Class	Class
Navigation Intentional Element Softgoal	Navigational Condition	Constraint
Navigation Element Link	Link	Association
Navigation Intentional Element Task	Service Link to an Operation of a Navigation Class	Link
Importance Type	Navigational Importance	Enumeration
Navigational Link	Navigational or Traversal Link	Association
Contribution	Navigational Service or Traversal Link	Association
Contribution Type	Contribution type	Enumeration
Decomposition	Access Primitive like Index , Query , Showall and Guided Tour with or without a	Navigation class with or without an Association

	traversal link	
DecompositionType	Decomposition type	Enumeration
Dependency	Primitive 2Attribute	Dependency

7.5 UML Profile for the EA00-H Presentation model

UML Profile is defined for the Domain Model and the Navigation Model in the previous section. Now we proceed to describe the UML Profile for the last of the three EA00-H design models i.e. the Presentation model. We extend the UML concepts of package, class, property and association to specify the Presentation Model concepts are specified. The purpose of defining an UML profile is to provide an easy mechanism of adaptation to the UML metamodel to elements for a given platform or method. Therefore, profile for the Presentation Model consists in adapting the elements described in the Presentation Model to the UML metamodel. In this way the profile allows to specify the Presentation Model elements in any commercial tool that supports UML.

The EA00-H Presentation Model elements are the Presentation Node, Presentation Page, Page Chunk, Presentation Class, Window, Frame, Frameset, Interface Component, Form, Cell, Layout, Presentation Links, Attributes and Dependencies. The Presentation model depicts the presentation layout of the web based application. The main concepts of the EA00-H Presentation model have been represented by a UML Profile as explained below:-

- **Presentation Node**-The "stereotype << Presentation Node>> is defined as an extension of the UML Class concept which has attributes and operations (also extensions of the UML concepts). The Structure Nodes stereotypes defined as an extension of the UML class concept" are <<Window>>, <<Frame>> and <<Frameset>>, <<Presentation Class>> which contains Interface Components, <<Interface Component>>, <<Form>>, <<Cell>>, and <<Layout>>.

- **Presentation Page and Page Chunk-** The other two Structure Nodes i.e. <<Presentation Page>> and <<Page chunk>> are defined as "an extension of the UML Package concept". Each of these concepts is denoted by means of a stereotyped class.
- **Interface Component-**The Presentation Class contains the Interface Components. The Interface Components are simple and composed interface components where composed interface components consists of more than one simple interface components. The composed Interface Components are (i) Collection which consists of collection of interface component concepts like, image, text, and Form elements , (ii) Anchored Collection is a collection of anchors and (iii) Anchor which are also "extensions of the UML class concept" and it is represented with the stereotype <<Collection>>, <<Anchored Collection>> and <<Anchor>>. The composed Interface components can contain Simple Interface Components like text, image and form elements which are represented by UML attributes.
- **Attribute-** A "stereotype <<attribute>> is defined as an extension of UML property concept". Property represents a structural characteristic that denotes an "attribute or a member end of association, or a part of a structured classifier of a presentation node or a link".The simple Interface Component stereotypes << Text>, <Image>>, <<Video>>, <<Audio>> and <<Form Element>>is explained as an extension of the UML property concept.
- **Presentation Links-** The presentation links Include Frame, Include Cell and navigation and display defines the navigation routes which can be utilized by a user for the display of the presentation node. The stereotype <<Link>> is defined by the extension of the UML Association concept. An association denotes a group of links. It can be unidirectional or bidirectional. Similarly it can be unary, binary or n-ary.The IncludeFrame link is represented with the stereotype <<IncludeFrame>> and the IncludeCell link with the stereotype <<IncludeCell>>. Similarly the Navigation Link, Builds, Display, Contains, Redirects and submit is represented by the stereotype << Navigation >>, <<Builds>>, <<Display>>, <<Contains>>, <<Redirects>> and <<Submit>>link as shown in Figure 7.3.

- **Dependencies**- The rest of the Presentation links (i.e. Builds, Submits and Contains) are defined as an "extension of the UML dependency metaclass and represented with the stereotype <<Builds>>, <<Contains>> and <<Submits>> respectively.
- **Condition**- A "stereotype <<condition>> is defined as an extension of UML constraint concept. Constraint is a packageable element representing some condition, restriction or assertion related to some element (that owns the constraint) or several elements (uml-diagrams.org)". The value of a constraint is denoted by a boolean expression whose value is either true or false.

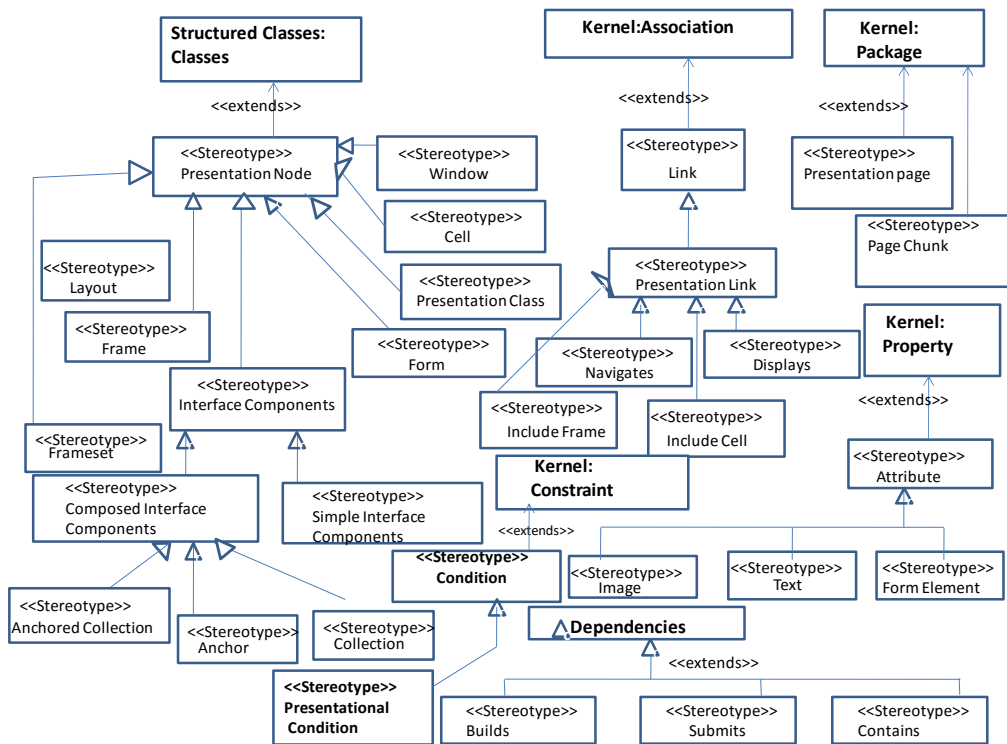


Figure 7.5. UML Profile for EDPD

To derive the EDPD we take into account the main presentational goal first. This is represented by the Main or Home page of the website with Frames, Frameset and Presentational Class with Interface Components to represent the content of the Home Page and have navigation links represented by Interface Component of anchored collection to other Presentation Goals represented by Presentation Pages. All Presentation Goals in the

Presentation WebGRL diagrams are represented as Presentation pages with Presentation Classes. There is one Presentation Classes for each Navigation Class in The NAD and Index.

We travel from the main presentational goal to other presentational goal by adding an Interface Component like anchor or menu item to navigate between the presentation goals now represented by Presentation Pages. The content for each Presentation Goal is displayed by Presentation Class and its Interface Components on that Presentation Page. We move from top to bottom by creating a Presentational Page for each Presentation Goal with a navigation link to contain or navigate to other pages. In case of a presentation Goal with requirements for form the Presentation Page must have a display link to the Interface Component Form with submit link. A list or an index is an anchored collection navigating to other Presentational pages. Presentation Goal with Search use build to generate the new Presentation Page each time a search is performed with the help of server pages. Soft goals are represented as presentational conditions of that presentation class of the presentation page to which they are attached. They are defined as member conditions on the model elements, presentation class, association or conditions on the attribute of the presentation class with satisfaction levels.

The zero level of the EDPD shows how the navigation nodes are grouped into abstract pages (and page chunks in the case) in the Website. In this case, each of the Navigational Nodes correspond with one Presentation Page, with the presentation class corresponding to its resource or the content class and the information on the page is represented with the help of Interface Components.

Once the DPD MOF metamodel has been defined, a UML profile is presented from the DPD model for expressing it in UML 2.0. We present the UML Profile for the EDPD below with their UML extensions in figure 7.3.

**Table 7.3. UML PROFILE FOR Presentation WebGRL and
Presentation Model**

WebGRL Model Elements	Presentation Model Stereotype	Stereotyped UML Metaclass
Presentation WebGRLspec	Presentation Model	Model
Actor	Presentation Page, Presentation Chunk	Package
Presentation Intentional Element Goal & Resource	Presentation Node	Class
Intentional Element Softgoal	Presentation Condition	Constraint
Intentional Element Attributes	Presentation Class- Interface Components	Class- Attributes
ImportanceType	Presentation Importance	Enumeration
Element Link	Presentation Link display	Association
Contribution	Navigational Link between Presentation Pages	Association
Contribution Type	Contribution type	Association
Decomposition	Navigational Link between Presentation	Association

	pages	
Decomposition Type	Decomposition type	Enumeration
Dependency	Builds, submits, contains	Dependency

A "UML Profile" for the three specific WebGRL diagrams to EA00-H design model conversion approach has been presented in this chapter. Both the "hard goals and the soft goals" have been considered in this transformation approach, therefore we need an approach that models the softgoals as well as web specific goals in the design phase. By applying the model transformation approach stated above we capture the goals as well as softgoals in the requirements phase and seamlessly transfer them to the design models suited for web applications along with a "UML compliant UML profile" to support them. The "UML Profiles is the Domain Model, Navigation Model and the Presentation Model" have been presented above. In the next chapter we briefly describe the WebGRL to EA00-H Transformation tool that automates the entire transformation step. Thus, providing an automatic leap from the requirements phase to the design phase".

Chapter 8

Implementation of the EA00-H Design Model with Web Application Design Tool

The EA00-H Design model has been implemented in the form of Web Application design (WAD) tool that serves as an editor for creating the three content, navigation and presentation design models by transformation from the specific WebGRL diagrams. The features of the Web Application design (WAD) tool and its working are explained in this chapter.

8.1 Introduction

The EA00-H Design model acts as an intermediary in transforming the user requirements into a web application design. The EA00-H design model has a UML Compliant UML Profile which further reduces the work of the software engineer. A UML Profile provides an easy adaption method from a platform independent model to a platform specific model. The transformation strategy used for converting user requirements captured by the specific WebGRL models into the specific EA00-H design models has been automated. In this chapter we present this automated transformation strategy for moving from the requirements stage to the design stage.

The Web specific Goal driven Requirements Engineering framework enables modeling and analysis of user and system requirements at a higher level of abstraction and also refines them for easy transition to design phase. This acts as an input to our WAD tool where the transformation strategy converts the requirements model into the design model. Thus presenting the three specific EA00-H design models namely the domain, navigation

and presentation models. These design models as emphasized earlier are UML compliant and the UML Profile for these design models has been presented in the previous chapter. This further eases the workload of the software engineer as it allows easy transformation from a platform independent UML Profile into any Platform dependent model.

8.2 Web Application Design Tool

There is a need to develop a tool support for the transformation from the requirements engineering to the design stage because none of the existing Web Engineering tools have the capability of using our transformation strategy for the conversion from WebGRL requirements model to EAOO-H design models. There exists a WebURN tool for capturing the requirements for the WebGRL framework which gives as an output the conflict resolved specific WebGRL diagram alternatives. The WebURN tool also provides the guidance to the user for generating the WebGRL diagrams. These specific WebGRL diagrams form an input to our WAD tool. Further, the transformation strategy ensures flawless conversion of these specific WebGRL diagrams into the three specific EAOO-H design models namely the domain, navigation and the presentation design models. Therefore we need a distinct tool to capture this transformation process. The implementation of the WAD tool is the actual manifestation of the use of transformation strategy for the WebGRL to the EAOO-H design approach for Web Design Engineering. This thesis is supported by the WAD tool that enables the transformation of WebGRL modeling elements into the enhanced notations for Web specific EAOO-H design models.

8.2.1. WAD Tool Technical details

This tool has been created in Visual Basic language has been used for implementing the transformation strategy for the WebGRL diagrams into the Enhanced AOO-H design model elements. The implementation has been done using Visual Basic because it easily supports the visual diagrams of the EAOO-H design models. WAD tool has been implemented as a software framework using Visual Basic which is an event based programming language. A software solution typically combines a custom program with

one or more packaged software applications. Rather than developing functionality from scratch, the solution developer uses functionality that is built into a packaged product. Programs in VBA code can help create the EA00-H modelling elements, and transfer information between the specific WebGRL diagrams to the specific EA00-H design models.

8.2.2. Features of WAD Tool

WAD tool is focuses on the following three factors: firstly the modeling elements of the specific WebGRL diagrams should be captured in totality, secondly the transformation strategy algorithm should easily be programmed for the automation purpose and thirdly the output of this transformation strategy should have the capability of representing all the three Enhanced AOO-H design modeling elements. It acts as transformer for converting the requirements model into the design model for the creation of the EA00-H design model It presents the EA00-H design model elements which are supported by a UML Profile thereby providing an easy adaptation mechanism to the UML models. The features of WAD tool are as follows:

a) Provides a base for defining the specific WebGRL diagram modeling elements as an input.

WebGRL has been enhanced for the "modeling of Web specific Requirements in a goal based approach. The enhanced Web specific Goal oriented Requirements Language (WebGRL) has similar basic elements but consists of enhanced Web specific notations like separate notation for Content Goal, Business Process Goal etc. and separate notation for softgoals or non-functional requirements like Product NFRs, Actor's NFRs, Organizational Objectives etc. In WAD tool provision has been made to enable formation of Web specific Goal oriented Requirements Language notation(Chawla et al 2015)". The three different WebGRL diagrams representing the content, navigation and the presentation requirements need to be captured and each of these follows a separate transformation strategy for conversion from the specific content, navigation or

presentation WebGRL diagram into their respective EAOO-H domain, navigation and presentation design models.

b) Provides the facility for taking the content WebGRL diagrams as an input and automates their transformation into the EAOO-H domain design model.

Based on the requirements specified by the stakeholders a content WebGRL diagram is produced which denotes the basic content or information requirements for the web based application being developed. This content WebGRL diagram from the Web URN tool is taken as an input for the transformation from the content WebGRL diagram into the EAOO-H domain model. The transformation algorithm presented by us in the Chapter 4 for the conversion from the WebGRL content model to the EAOO-H Domain design model is used for the automation of the transformation process. As stated earlier this acts as the first step towards the movement from the requirements stage of software development to the design stage while following a structured methodology as in the case of traditional information systems. This transformation step results in the EAOO-H Domain design model as an input.

c) Provides the facility for taking the navigation WebGRL diagrams as input and automates its transformation into the EAOO-H navigation design model.

The navigational requirements of the user are captured by the navigation WebGRL model. These specify the movement between a set of navigation nodes required by the user specified as navigational goals and softgoals within the navigation WebGRL diagram. This navigation WebGRL diagram acts as an input for the generation of the EAOO-H navigation design model.

In order to generate the navigation design model we turn to the transformation algorithm specified in Chapter 5 for the conversion of the WebGRL modeling elements into the navigation design model. This transformation algorithm is used for the automation of the navigational transformation process. This provides us the second step towards the movement from the requirements stage of software development to the design stage while

following a structured methodology as in the case of traditional information systems. This transformation step results in the EAOO-H Navigation design model as an input.

d) Provides the facility for taking the navigation WebGRL diagrams as input and automates its transformation into the EAOO-H navigation design model.

The presentation WebGRL diagram represent the presentation requirements of the stakeholders. They form the display and appeal of a web application for the ease of use by the client. The presentation requirements are captured with the help of presentation modeling elements primarily the presentation goals, soft goal and presentation links. Therefore, the presentation requirements and their representation in the design model form a very important part of the transformation strategy. This WebGRL presentation model is presented as an input to the transformation strategy for converting the presentation WebGRL diagram into the EAOO-H presentation design model.

In order to generate the presentation design model we turn to the transformation algorithm specified in Chapter 6 for the conversion of the presentation WebGRL modeling elements into the presentation design model. This transformation algorithm is used for the automation of the presentation transformation process. This transformation process emphasizes on the display and the style of presenting the contents to the user within the website as such it forms the front end to the user and like the other front ends in the traditional information systems they should have the appeal, attraction and usability as a prime feature of the front end This provides us the second step towards the movement from the requirements stage of software development to the design stage while following a structured methodology as in the case of traditional information systems. This transformation step results in the EAOO-H Navigation design model as an input.

The automation of these three transformation processes stated in b, c and d above results in a complete design of the web application. This can further be enhanced by the adaptability feature to incorporate the adaptability factor within the web application as per the requirements of the user. These features of WAD tool enable a user friendly and easy to use application for the transformation from the requirements engineering to the design

engineering phase. The transformation process has been closely followed in the WAD tool to produce a more complete and consistent Software Design models. In the next section, the working of WAD is explained in detail.

8.2.3. Design of the WAD tool

The WAD Tool has been designed to have three modules: The Domain, Navigation and Presentation transformation models. The first module is the first step towards the transformation from the requirements to the design model i.e. transformation of the ContentWebGRL diagrams into the EAOO-H Domain design model. This includes capturing the content WebGRL diagrams completely for lossless transformation into the domain design model. The content WebGRL diagrams is made up of content goals, softgoals, relationships and resources associated with these content goals. These content goals are used to represent the information to be stored within the web application and is used for transformation into the domain classes, relationships, cardinalities of these relationships etc. In order to transform the WebGRL content model we need to identify the domain classes, include as domain classes the actors identified in the requirements workflow, if it is needed to store some kind of information. Determine the relationships between the domain classes, define hierarchical relationships between the classes, specify the most relevant attributes and operations for a given domain class. In order to ensure that this domain model is a complete representation of the content WebGRL diagram after transformation, a set of transformation rules have been defined in Chapter 4 along with a transformation algorithm for the conversion of the WebGRL content model to EAOO-H Domain design model. The first module makes use of this transformation algorithm to construct the domain design model. The output of this module is the EAOO-H domain model which forms the first part of the complete EAOO-H design model to be generated at the end of the transformation process.

The second module is the second step towards the transformation from the requirements to the design model i.e. transformation of the NavigationWebGRL diagrams into the EAOO-H Navigation design model. This includes capturing the navigation WebGRL diagrams

completely for lossless transformation into the navigation design model. The navigation WebGRL diagram is made up of navigation goals, soft goals, navigation links and associations between these navigation goals. These navigation goals are used to represent the navigational information i.e. they present the navigation routes to be followed by the user for the movement within the website based on the access permissions granted to him. In order to transform the WebGRL navigation model we need to identify the navigation classes. Determine the associations between the navigation classes, and the navigation links that form the path to be followed by the user based on the conditions attached to these navigation classes and links. There are different types of navigation links which need to be modelled differently. Also the most relevant attributes and operations for a given navigation class also need to be identified. In order to ensure that this navigation model is a complete representation of the navigation WebGRL diagram after transformation, a set of transformation rules have been defined in Chapter 5 along with a transformation algorithm for the conversion of the WebGRL navigation model to EAOO-H navigation design model. The second module makes use of this transformation algorithm to construct the navigation design model. The output of this module is the EAOO-H navigation model which forms the second part of the complete EAOO-H design model to be generated at the end of the transformation process.

The third module is the third and final step towards the transformation from the requirements to the design model i.e. transformation of the Presentation WebGRL diagrams into the EAOO-HPresentation design model. This includes capturing the Presentation WebGRL diagrams completely for lossless transformation into the Presentation design model. The presentation WebGRL diagram is made up of presentation goals, soft goals, presentation links and associations between these presentation goals. These presentation goals are used to represent the presentation information i.e. they present the style and layout of the content on the website pages, frames and windows at two different levels. In order to transform the WebGRL presentation model we need to identify the presentation classes. Determine the associations between the navigation classes, and the navigation links that form the path to be followed by the user based on the

conditions attached to these navigation classes and links. There are different types of navigation links which need to be modeled differently. Also the most relevant attributes and operations for a given navigation class also need to be identified. In order to ensure that this presentation model is a complete representation of the presentationWebGRL diagram after transformation, a set of transformation rules have been defined in Chapter 6 alongwith a transformation algorithm for the conversion of the WebGRLpresentation model to EAOO-H presentation design model. The third module makes use of this transformation algorithm to construct the presentation design model. The output of this module is the EAOO-H presentation model which forms the last and final part of the complete EAOO-H design model to be generated at the end of the transformation process.

8.2.4. Working of WAD Tool

The WAD tool helps in automating the entire transformation process of identifying the modeling elements of the output model and also helps in the construction of the transformed design model based on the requirements captured in the requirements stage for all the three web specific WebGRL models into the EAOO-H design model. The WAD tool assists the Design Engineer in the construction of a detailed web oriented design model based on a structured process of software development where both the requirements and the design phase have been dealt with in detail. Further, the design output models are UML Compliant that helps in adapting the output easily into a UML metamodel thereby further reducing the workload of the software development engineers. The WAD tool has been developed in three main parts. In the first part, the content of the web application is captured by the requirements engineer and the same is presented as an input for the first module for transformation from the content WebGRL diagram into the EAOO-H Domain model. The entire process is automated therefore the domain design model is easily generated. Any changes desired by the Design Engineer can only be made at the input level. As we are assuming that the input WebGRL models have a conflict resolved alternative solutions from which the design engineer has already made his choice. The main objective of this transformation process is to ensure a lossless transformation to the design model and while transformation checks and verification has

been added in the transformation algorithm for a flawless transformation. The second and third part are used to generate the navigation and presentation models respectively.

The working of WAD is explained through an example where the Design Engineer has to transform the requirements of *an Online Education Web application System* to be developed. The Design engineer would carry out the following steps for constructing the three design models for this web application using the goal based WebGRL approach as an input and the EAOO-H design models as an output.

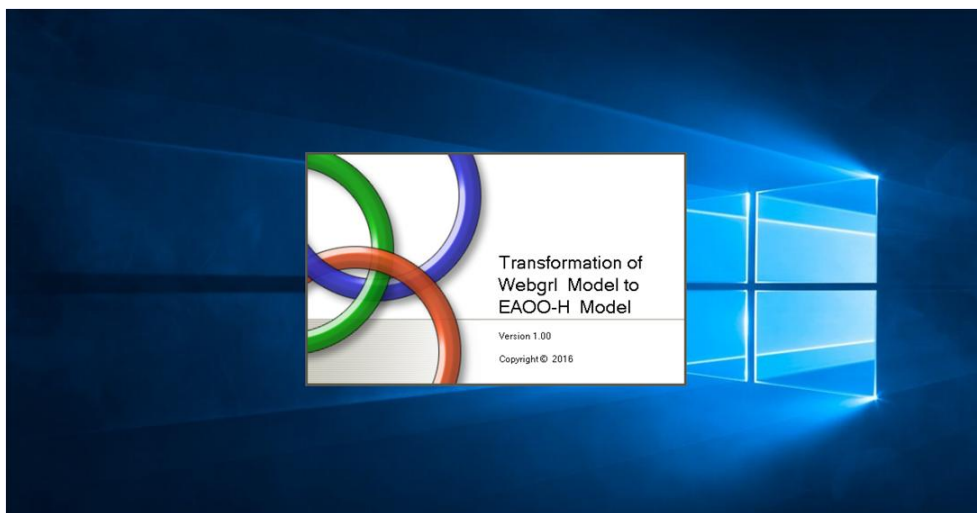


Figure 8.1.WAD Tool

a) Elicit the content goals, softgoals and other content WebGRL modelling elements to create the DomainEAOO-H design model.

First step is to gather the requirements from the stakeholders and understand their intention of creating the web application. The prime purpose or goal of the Web application is gathered and then it is further refined into different subgoals. This is further specialized into the three specific WebGRL models. The WAD tool consists of module which provides the three transformation options. For best results the three transformation processes need to be carried out in order therefore a wrong choice of transformation would lead to an error message. Therefore the first option of generating the domain design model from the content WebGRL diagrams used as an input is performed. This results in

seamlessly transformed Domain design models from the content WebGRL model. It appears as shown in Figure 8.2.

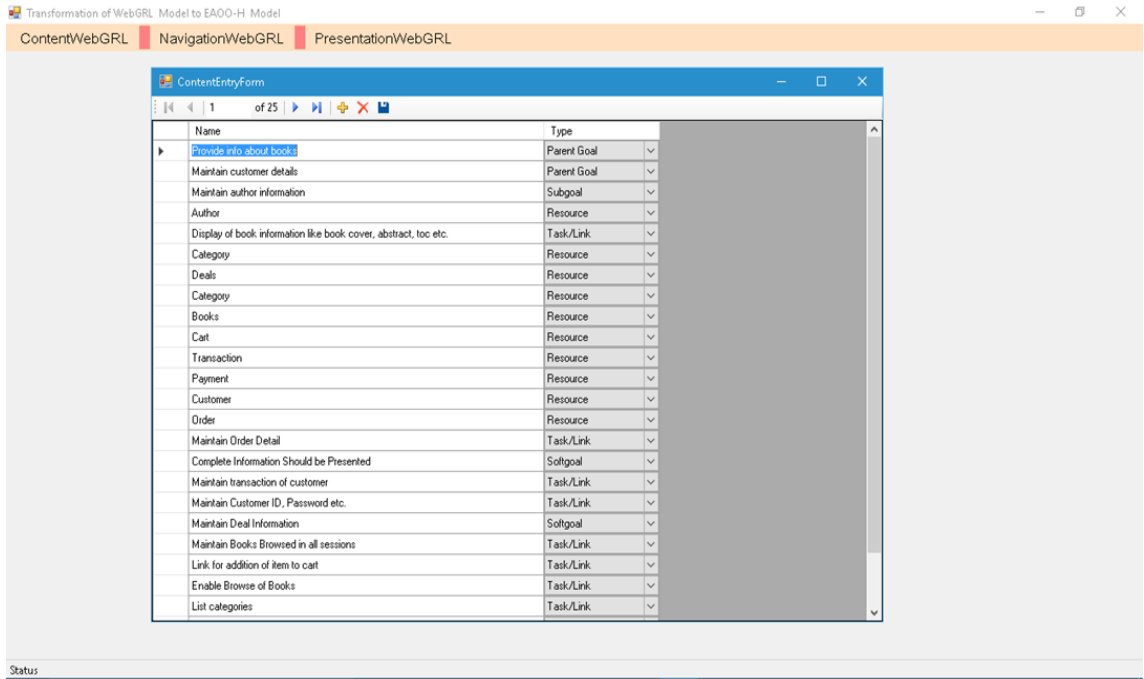


Figure 8.2.Input Information for the Content Model of WAD Tool

b) Elicit the navigation goals, softgoals and other navigation WebGRL modeling elements to create the NavigationEA00-H design model.

Second step deals with the generation of the navigation design model for the web application to be developed. For this we need to gather the navigational requirements from the stakeholders and understand their intention of creating the web application. The WAD tool presents the second option of generating the navigation design model from the navigation WebGRL diagrams used as an input is performed. As stated earlier the main objective is to ensure that the detailed requirements captured earlier are not lost while transformation. That is the reason of designing an automated transformation process so that the navigation WebGRL diagram is flawlessly transformed into the navigation design models. It appears as shown in Figure 8.3.

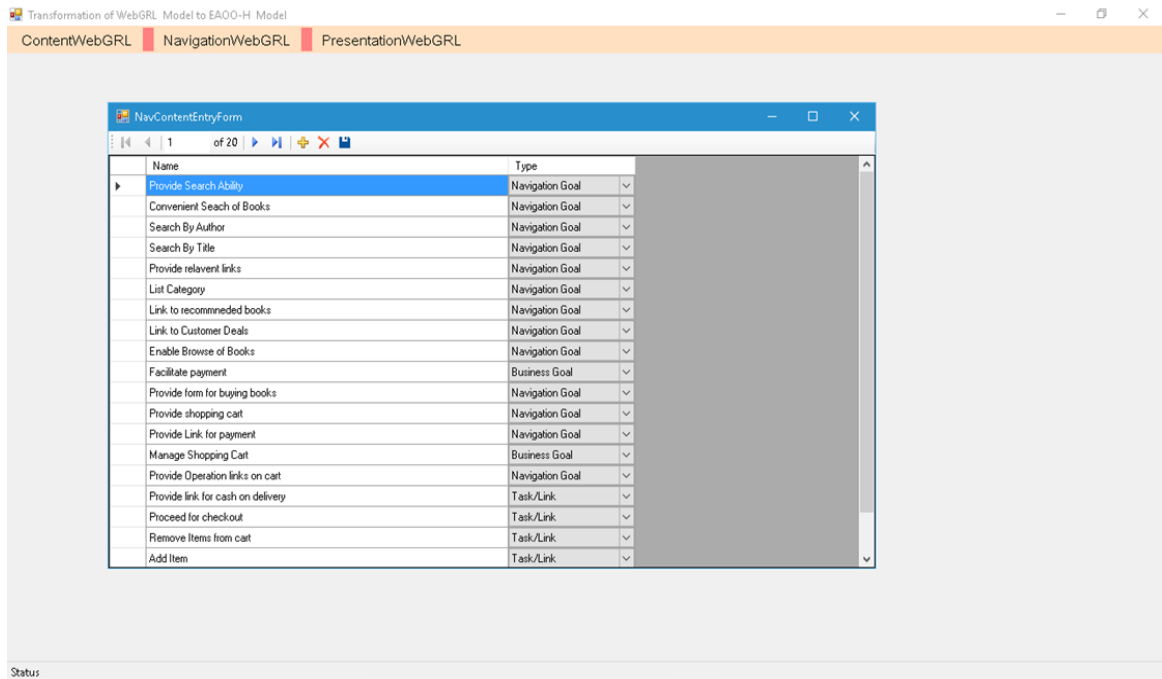


Figure 8.3.Input Information for the Navigation Model of WAD Tool

c) Elicit the presentation goals, softgoals and other presentation WebGRL modeling elements to create the PresentationEAOO-H design model.

Third and the final step leads to the construction presentation design model for the web application to be developed. In order to do so we need to gather the presentation requirements from the stakeholders and understand their intention of creating the web application. The WAD tool presents the third and final option of generating the presentation design model from the presentationWebGRLdiagrams used as an input is performed. As stated earlier the main objective is to ensure that the detailed requirements captured earlier are not lost while transformation. That is the reason of designing an automated transformation process so that the presentationWebGRL diagram is flawlessly transformed into the presentation design models.

8.3 Early Requirements Analysis of Web Based Education System

The WAD tool can be exhibited with the help of this example of Web Based Education System. The primary requirements are elicited from the stakeholders. They are as follows:

1. Create an Online Education System.
2. The system should provide tutorials and notes to the students on concerned subjects.
3. The system should support clarification of doubts of students.
4. The system should provide assignments and enable submission from students.
5. The assignments should be according to student's knowledge level.
6. The system should be able to conduct online examination.
7. The system is not for profit, but it should be able to cover the incumbent expenses.

Step 1.

First, the primary requirements gathered from the stakeholders are modelled by the requirements engineer with the help of the process guidance. The base WebGRL is prepared by the requirements engineer using WebURN tool and in the process, the goals are refined to subgoals and the interrelationship between different goals is also acknowledged. The base WebGRL diagram for web based education system is shown in figure 7.10.

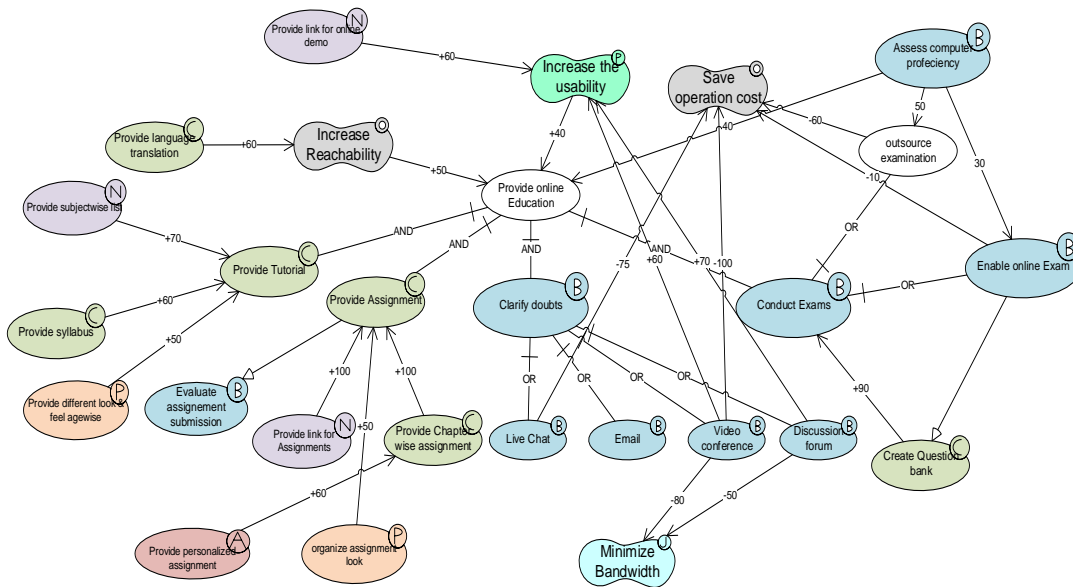


Figure 8.4.Base WebGRL diagram for Web based Education System

Step 2.

Refine Base WebGRL diagram into functionality specific diagrams in the predefined order i.e. Content, Navigation, Business Process, Adaptation and Presentation. At this stage, further elicitation is done with the stakeholders for functionality specific goals and requirements. This step produces WebGRL diagrams for specific functional requirements like content WebGRL (Figure 7.11), navigation WebGRL (Figure 7.12) and presentation WebGRL (Figure 7.15)

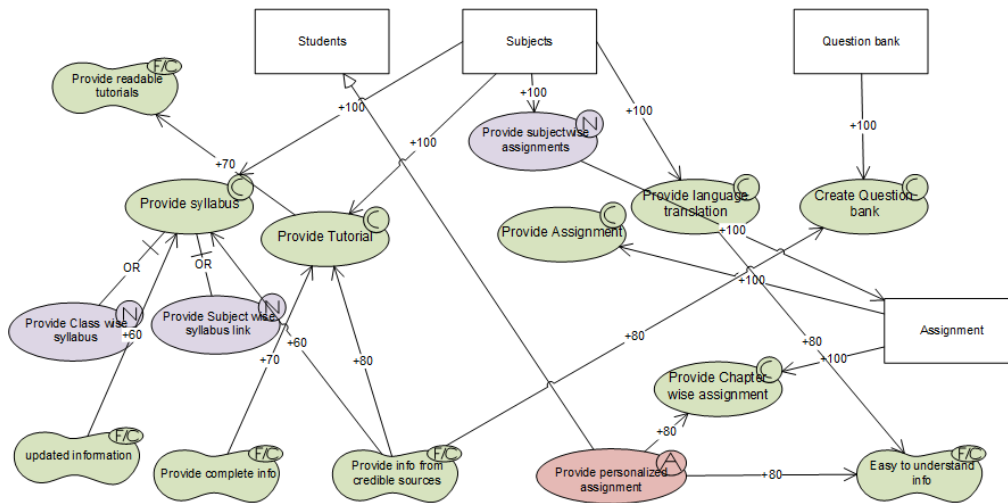


Figure 8.5.Content WebGRL diagram for Web based EducationSystem

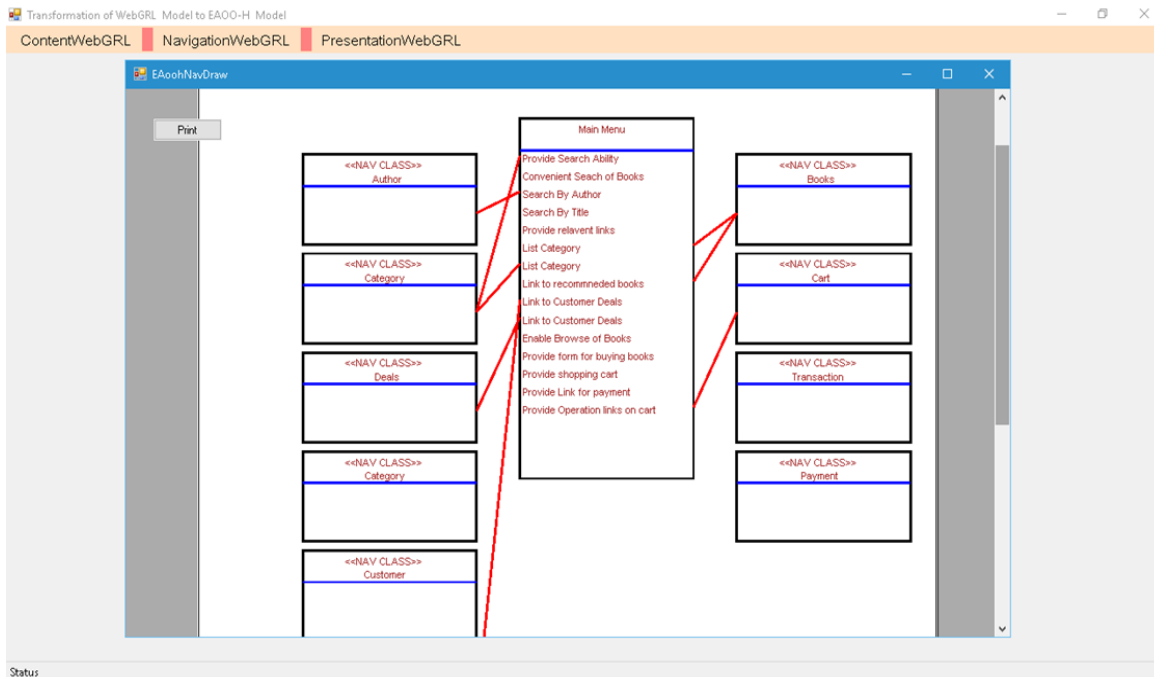


Figure 8.6.Output Content Design Model diagram for web based education system

Step 3.

Refined functionality specific diagrams in the predefined order i.e. Content, Navigation, and Presentation WebGRL diagrams shown above are transformed resulting in the three outputs shown below. This step produces EAOO-H domain, navigation and presentation design models for specific functional requirements like content WebGRL (Figure 7.11), navigation WebGRL (Figure 7.12) and presentation WebGRL (Figure 7.15) diagrams respectively.

8.4 Conclusion

In this chapter, the implementation of WAD tool has been. The tool automates the process of transformation from the requirements engineering to the design engineering stage for web applications. It takes as input the WebGRL diagrams in the text form and outputs the three design models that are refined, validated for errors and evaluated for better alternative solutions by the WebGRL generation process. The WAD tool has the following salient features that assist the design engineer in the transformation process from the requirements to the design stage.

- WAD tool completely automates the creation of EAOO-H design models. The WebGRL modelling elements are input in the form and relevant information is stored in a file.
- The EAOO-H notation elements proposed in this thesis can be generated easily using this tool.
- The tool has easy input facility for specific WebGRL diagrams.
- The WAD tool provides transformation process support that acts as a helping guide for the design engineer.
- The EAOO-H design models created in the tool can be validated for losslessness and discrepancies.
- The WebGRL diagrams can be evaluated for choice of alternative solutions, by using the transformation process to generate alternate designs so that appropriate design

rationale is there and the satisfaction of the stakeholders from the provided solution is estimated.

- WAD tool is user friendly and very helpful tool for comprehensive transformation process from the WebGRL requirements model to the EA00-H design model that can be used in the web application development industry to achieve better applications and managing the project efficiently.

WAD tool summarizes and demonstrates the proposals made in this thesis for transforming a structured, goal driven and systematic requirements engineering approach for web application development into a web oriented EA00-H design model. In the next chapter, the contribution of the research work presented in this thesis is summarized and future directions of work are discussed.

Chapter 9

Conclusion and Future Work

***“Live as if you were to die tomorrow. Learn as if you were to live forever.”
– Mahatma Gandhi***

This thesis has presented several contributions that help in capturing the requirements in totality and further capture these requirements losslessly in the holistic design model suited for web applications. The enhancement of the existing A-OOH model to the EAOO-H model is done to capture the web specific requirements totally. Further, transformation rules for flawless conversion of the web specific requirements to model the WebGRL content, navigation and presentation model from the requirements to the design phase has been covered in the research. This automatic transformation is not only flawless but it also reduces the workload of the design engineer by automating the entire process. To enable real-world usage of the transformation process, a usable and extensible tool has been developed that supports the conversion of the Web specific Goal driven Requirements Engineering models into their respective design models. Then, future scope of work on the extension of this transformation model to capture and convert the adaptivity and the business process requirements has been discussed in this epilogue.

9.1. Main Contributions of the thesis

This thesis enhances an existing design model to capture the web requirements in its entirety and further defines a transformation strategy for transforming the requirements from the requirements phase to the design phase. The transformation into the design phase results in an output that is UML Compliant, as such it is easily convertible from the design phase

which is a platform independent output into the coding phase which is a platform specific output. Thereby resulting in a smooth transformation from the requirements phase to the coding phase without any loss of information. This structured technique ensures that the web application thus developed is a highly systematic, complete, consistent, reliable and a high quality product at lower cost with respect to the user requirements as direct transformation from the requirements phase to the coding phase is being done which covers some of the most important phases of the software development process. Our choice of input for the design process is the WebGRL diagrams. This is so, because we believe that the WebGRL requirements are an exhaustive method of capturing both the functional and non-functional requirements. The evaluation of requirements models is done in the WebGRE model that helps in reasoning of the requirements, selecting among various solutions and resolving conflicts. They also present design alternatives from which based on stakeholder reasoning the most suitable design alternative can be chosen for which the requirements can be tuned accordingly. The web design engineer gets complete guidance and support for modeling the requirements into the design phase and the output thus produced is a platform independent model that can be easily converted into a platform specific model in the coding phase. This results in smoother transition to web coding phase and less effort by the web designer. Thus, a complete and consistent Software Requirements Specification is transformed automatically into a web based design that is a platform independent model into the coding phase thereby reducing the work of the design and the coding engineer and yields a better quality product, besides reducing the number of iterations and the cost of the project.

The enhanced web design model and the requirements to design phase transformation method has attempted to address most of the problems faced in web application development as discussed in the beginning of this thesis. The proposed solution supports the design engineering of web application right from the requirements engineering phase to the coding phase. The various contributions of this thesis are listed below:

- 5. Enhancement of the AOO-H model into the EAOO-H Design Model to capture the GORE based WebGRL specific diagrams of requirements engineering***

The web specific "functional and non-functional requirements" (Shailey and Sangeeta 2010) are unique and different from requirements of generic software systems. The functional requirements are explicit requirements which can be easily identified whereas the non-functional requirements are hidden or implicit requirements. The "functional and the nonfunctional requirements specific for web applications have been captured by choosing the WebGRL notation of (Chawla et al 2015)" as an input to the design phase. In order to transform these web specific requirements presented by the specific WebGRL diagrams, into the design models for web applications, the "A-OOH design model of (Garrigos 2010)" has been enhanced. The Enhanced A-OOH model presents the augmented versions of the Content, Navigation and Presentation A-OOH design models. The Enhancement of each of these design models namely the content design model, the navigation design model and the presentation design model is done in a manner that it is able to capture the web specific requirements of the Specific Web GRL diagrams in totality without any loss. This improves the verifiability and reliability of the product as a whole. The existing work could not capture the WebGRL diagram flawlessly therefore the need for enhancement of the design models suited for web applications was felt. The web specific design models has been enriched in the following ways:

- The Domain design model is enhanced to capture the content requirements and the associated information structure using the enhanced version of the EA00-H Domain Design Model.
- The Navigation design model is enhanced to capture the navigation requirements and the associated navigation structure using the enhanced version of the EA00-H Navigation Design Model with the help of the navigation nodes consisting of navigation classes, menus, guided tours, showall and the navigation paths.
- The Presentation design model is enhanced to capture the presentation requirements and the associated layout structure. The enhanced version of the EA00-H Navigation Design Model consists of the presentation nodes, structure nodes and presentation links which are comprised of presentation classes, Presentation page, frame and window.

6. Transformation of Content WebGRL diagram to EA00-H domain model.

The user requirements captured by the content WebGRL diagram are Goal, Softgoal, Task, and Resource and the intentional links are namely the decomposition links, contribution links, means end links and dependency links, the relevant concepts for the application are gathered besides the designer knowledge of the domain. The contentWebGRL model of domain analysis has to be refined in consecutive iterations with new goals, softgoals etc. stated above into the final WebGRL content diagram with the help of the GOREWEB framework. This refined content WebGRL diagram is used in the model transformation strategy for smooth transition from the content WebGRL diagram into the EA00-H domain model explained below. The A-OOH design model is requirement based whereas the Enhanced AOOH approach is goal oriented therefore in place of task we extend the goal as well as softgoals to the stereotypes defined in the A-OOH approach into domain stereotypes. Transformation rules have been defined that automatically transfer the content WebGRL diagram for a web application into its corresponding domain model of the design phase. In order to ensure that this domain model is a complete representation of the content WebGRL diagram after transformation, a set of Transformation Rules have been defined that identify the domain classes, their attributes and operations as well as the hierarchical relationships between the classes.

7. Transformation of NavigationWebGRLdiagram to EA00-H Navigation model.

The domain information is the main input for the design navigation activity, where the navigational paths are defined to fulfill the different functional requirements and the organization of that information in abstract pages. The "Navigation metamodel" is made up of two basic concepts:- the navigation node and the navigation link. The navigation node is used to present the source and target destinations for a navigation path. The navigation path is represented with the help of the navigation link. A "navigation model" can have any number

of navigation nodes and navigation links. The navigation node is further specialized into three types of nodes namely the navigation class, menu and access primitives. The navigation path can be between any three of these. Transformation rules have been defined that automatically transfer the navigation WebGRL diagram for a web application into its corresponding navigation model of the design phase. The aim is to transform the navigation WebGRL diagram from the requirements phase into the design phase seamlessly. Therefore, a set of transformation rules help in the identification of navigation classes, menu and access primitives. Further, these rules also assist in defining the navigation paths between the different types of navigation nodes and the relationships between navigation classes and their attributes and operations.

8. Transformation of Presentation WebGRL diagram to EAOO-H Presentation model.

The WebGRL Presentation model, is used to define the layout of the generated hypermedia presentation. During the presentation design, the concepts related with the abstract structure of the site and the specific details of presentations are gathered. The "Presentation Model" is defined in this activity. It is captured by one or more "Design Presentation Diagrams". There are two main basic elements of a DPD Metamodel namely the Presentation Nodes and the relationships among them represented by the Presentation Links. There are two types of presentation nodes, the structure node and the layout node. The structure and the presentation nodes are considered at two different levels of the DPD. The structure nodes are used to define the structure of the presentation of the website and are considered at level zero of the DPD. The Layout of the presentation model for the web application is considered at level one. The different types of structure and layout nodes are described briefly below. The different types of Structure Nodes are: Presentation Page, Window, Frame, Frameset, Presentation Class and Page Chunk (level zero of the DPD). Similarly the different types of Layout Nodes considered are: Layout and Cell (level 1 of the DPD). Just like the presentation nodes there are a variety of presentation links to represent different types of relationships between the presentation nodes. In a Presentation Model five types of Presentation Links can be defined namely navigation, build, submit, contain, redirect, display, include frame and include cell.

Transformation rules have been defined that automatically transfer the presentation WebGRL diagram for a web application into its corresponding presentation model of the design phase. The aim is to transform the presentation WebGRL diagram from the requirements phase into the design phase seamlessly. Therefore, a set of transformation rules help in the identification of presentation pages, presentation classes, navigation links and layout for the display on the presentation pages, windows etc.

9. UML Profile for the WebGRL and EA00-H Design Models

Now we take a step from the design phase to the implementation phase by ensuring that the EA00-H domain, navigation and presentation design models generated by the transformation process from the requirements engineering phase to the design phase are UML Compliant i.e they are supported by a UML Profile. The advantage of ensuring that the EA00-H domain, navigation and presentation models are supported by a UML Profile makes the job of the Software Engineer simpler by providing a UML Compliant UML Profile which can easily be used for the coding purpose by using an object oriented design language supported by the OMG's UML Metamodel. The purpose of defining an UML profile is to provide an easy mechanism of adaptation to the UML metamodel to elements that are specific of a particular domain, platform or method. In this sense, the particular profile for the DM, NAD and EDPD consists in adapting the elements defined in these design models to the UML metamodel. In this way the profile allows to specify the EA00-H design model elements in any commercial tool that supports UML. The UML Profile is defined by using stereotypes for extension of the profile classes, associations, package and properties.

10. Execution support of the transformation with the help of transformation tool

WebURN tool has been developed for implementing the WebGRE framework. The tool provides complete support and guidance for all the requirements engineering tasks. The WebURN tool acts as an editor, validator, guide and analyzer of requirements model diagrams. The requirements specification produced after the conduct of WebGRE framework is correct, complete and consistent and much closer to the stakeholders' expectations and web designer's perception.

The advantages or the contribution of this thesis is as follows:-

1. This thesis presents a systematic approach for web application development as in the case of traditional Information systems.
2. Different phases of Information Systems development of proper requirements engineering and software design phases suitable for web application development are followed.
3. A suitable requirements approach which has already captured both functional and non-functional requirements in a comprehensive manner and is tuned specifically for web application development is used as an input to the design phase.
4. Similarly, a suitable design model for web application development is chosen that has the capability to capture different perspectives of the web application development and enhanced to define a new EAOO-H Design model which is more comprehensive.
5. A transformation strategy is defined for transforming the requirements from the requirements phase to the design phase that seamlessly transforms the requirements into design without any loss of information and traceability results in a high quality low cost better standard of designs of web applications.
6. An Automatic transformations method is followed from one phase of the web application development to the other phase as it reduces time of development and eases the work of the design engineer resulting in a structured, systematic, consistent and comprehensive web application development approach that has lower cost, lesser time and has better quality reliability and traceability.

9.2 Future Directions

Research work presented in this thesis can be extended along the following lines. The EAOO-H design model can be extended to capture adaptivity and business process requirements and transform the adaptivity and business process WebGRL model to web design and architectural constructs using the existing web engineering methodology. The transformation to a web design methodology would enable a seamless transition to Web architectural constructs giving web designers a clearer vision of the application to be built. The transformation to an existing web engineering methodology can help in automatic generation

of web application. This can help in creation of web application with minimum efforts with the help of tools.

Another area of future research is modeling of Adaptation goals using Aspect oriented techniques. The aspect oriented requirements engineering is a recent area of research wherein the focus is on using aspect oriented software development methods from early requirements analysis phase. Aspects are actually cross cutting concerns and aspect oriented software development helps in separation of concerns. Web applications are very dynamic and requirements like personalization and adaptation influence and are affected by other requirements thus, they need to be modified or changed often. Aspect oriented techniques would help in proper modularization of the cross cutting concerns so that the problem of scattering and tangling can be avoided. Many non-functional requirements are also cross cutting in nature, as they have impact on multiple concerns and are influenced by different types of requirements. "Aspect oriented approaches focus on systematically and modularly treating, reasoning about, composing and subsequently tracing crosscutting functional and non-functional concerns via suitable abstraction, representation and composition mechanisms tailored to the requirements engineering domain. Aspect oriented User Requirements Notation (AoURN) has already been proposed and it is lined up for standardization (Mussbacher, 2011)". The enhancement of AoURN for modeling the personalization and adaptation issues in web application can be explored further for providing a modular requirements model.

Bibliography

Aguilar, J.A., Garrigós I, Mazón, J. N., Trujillo, J. (2010). An MDA Approach for Goal-oriented Requirement Analysis in Web Engineering. *Journal of Universal Computer Science*, vol. 16, no. 17 (2010), 247, pp 2475-2494

Aguilar, J., Garrigós, I. and Mazón, J. (2011). A Goal-Oriented Approach for Optimizing Non-functional Requirements in Web Applications. *Advances in Conceptual Modeling. Recent Developments and New Directions*, pp.14-23.

Alexander, C. (1979). *The timeless way of building*. New York: Oxford University Press.

Amyot, D. (2003). Introduction to the User Requirements Notation: learning by example. *Computer Networks*, 42(3), pp.285-301.

Antoń, A. (1997). Goal identification and refinement in the specification of software-based information systems.

Atzeni, P., Mecca, G. and Merialdo, P. (1998). Design and Maintenance of Data-Intensive Web Sites. In: *Advances in Database Technology – EDBT’98*, pp.436-449.

Atzeni, P., Mecca, G. and Merialdo, P. (1998). Design and Maintenance of Data-Intensive Web Sites. In: *Advances in Database Technology - EDBT’98*. pp.436–449.

Azam, F., Li, Z. and Ahmad, R. (2007). Integrating Value-based Requirement Engineering Models to WebML using VIP Business Modeling Framework. In: *International World Wide Web Conference Committee (IW3C2)*. Banff, Alberta, Canada: ACM, pp.933-942.

Baresi, L., Garzotto, F. and Paolini, P. (2001). Extending UML for Modelling Web Applications. In: *Hawaii International Conference on System Sciences*. Miami, USA, pp.1285 - 1294.

- Baumeister, H., Koch, N. and Mandel, L. (1999).Towards a UML Extension for Hypermedia Design. In: UML'99, , LNCS, Vol. 1723. France: Springer-Verlag, pp.614-629.
- Beigbeder, S. and Castro, C. (2004).An MDA Approach for the Development of Web Applications. In: Web Engineering, 4th International Conference, ICWE 2004. Munich, Germany: Springer, VOL. 3140, pp.769.
- Bernstein, M. (1998).Patterns of hypertext. In: Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space---structure in hypermedia systems: links, objects, time and space---structure in hypermedia systems. Pittsburgh (USA): ACM, pp.21-29.
- Bolchini, D. and Paolini, P. (2004).Goal-driven requirements analysis for hypermedia-intensive Web applications.Requirements Engineering, 9(2), pp.85-103.
- Brambilla, M., Comai, S., Fraternali, P. and Matera, M. (2008).Designing Web Applications with Webml and Webratio. In: Web Engineering: Modelling and Implementing Web Applications, 1st ed. Vol. 4823, pp.221-261.
- Brambilla, M., Preciado, J., Linaje, M. and Sanchez, F. (2008).Business Process-Based Conceptual Design of Rich Internet Applications. In: International Conference on Web Engineering. Los Alamitos, California (USA): IEEE Computer Society, pp.155-161.
- Buschmann, F. (1996).Pattern-oriented software architecture. Chichester: Wiley.
- Cáceres, P., Castro, V., Vara, J. and Marcos, E. (2006). Model transformations for hypertext modeling on web information systems. In: Proceedings of the 2006 ACM symposium on Applied computing. Dijon (France): ACM, pp.1232-1239.
- Cachero, C. and Gomez, J. (2002). Advanced conceptual modeling of web applications: Embedding operation interfaces in navigation design. JISBD, pp.235–248.

- Cachero, C. and Gómez, J. (2002). Advanced conceptual modeling of Web applications: Embedding operation interfaces in navigation design. In: 21th International Conference on Conceptual Modeling (JISBD). Madrid, Spain, pp.235-248.
- Cadavid, J., Lopez, D., Hincapié, J. and Quintero, J. (2009). A Domain Specific Language to Generate Web Applications. In: Memorias de la XII Conferencia Iberoamericana de Software Engineering (CIBSE 2009). Medellin (Colombia), pp.139-144.
- Castro, J., Kolp, M. and Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: the Tropos project. *Information Systems*, 27(6), pp.365-389.
- Ceri, S., Daniel, F., Matera, M. and Facca, F. (2007). Model-driven development of context-aware Web applications. *ACM Trans. Inter. Tech.*, 7(1), pp.30-63.
- Ceri, S., Fraternali, P. and Bongio, A. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks*, 33(1-6), pp.137-157.
- Ceri, S., Fraternali, P., Bongio, A. and Maurino, A. (2000). Modeling data entry and operations in WebML. In: Third International Workshop WebDB2000 on The World Wide Web and Databases, Lecture Notes In Computer Science; Vol. 1997. Springer-Verlag, pp.201-214.
- Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S. and Matera, M. (2002). Designing data-intensive Web applications. San Francisco, Calif.: Morgan Kaufmann.
- Chawla, S., Srivastava, S. and Bedi, P. (2011). GOREWEB Framework for Goal Oriented Requirements Engineering of Web Applications. In: IC3, CCIS 168. Berlin: Springer-Verlag, S. Aluru et al, pp.229–241.
- Chawla, S., Srivastava, S. and Bedi, P. (2011). GOREWEB Framework for Goal Oriented Requirements Engineering of Web Applications. In: IC3 2011, CCIS 168. Berlin: Springer-Verlag, pp.229–241.

Chawla, S. and Srivastava, S. (2012). Goal oriented Requirement Analysis for Web Applications. IJMO, pp.192-196.

Chawla, S., Srivastava, S. and Bedi, P. (2015). Goal and Scenario based Web Requirements Engineering. in International Journal of Computer Systems Science & Engineering.

Chawla, S., Srivastava, S. and Bedi, P. (2015).Improving the quality of web applications with web specific goal driven requirements engineering.International Journal of System Assurance Engineering and Management.

De Troyer, O. (2005). Audience-driven web design. In: M. Khosrow-Pour, ed., Encyclopedia of Information Science and Technology, IDEA Group Publishing,ISBN: 1-59140-553-X, pp.184 - 187.

De Troyer, O. and Casteleyn, S. (2001). The Conference Review System with WSDM. In: First International Workshop on Web-Oriented Software Technology, IWOST'01. Oscar Pastor, Valencia (University of Technology), Spain.



De Troyer, O. and Casteleyn, S. (2004). Designing Localized Web Sites. In: 5th International Conference on Web Information Systems Engineering (WISE2004). Brisbane, Australia: Springer-Verlag, pp.547 - 558.

De Troyer, O. and Leune, C. (1998). WSDM: A User-Centered Design Method for Web Sites. In: Proceedings of the 7th International World Wide Web Conference, Computer Networks and ISDN systems. Brisbane, Australia: Elsevier, pp.85 - 94.

December, J. and Ginsburg, M. (1995). HTML and CGI unleashed. Indianapolis, IN: Sams Net, pp.194-258.

Frasincar, F., Houben, G., Barna, P. and Vdovjak, R. (2004).Engineering the presentation layer of adaptable web information systems. In: 4th International Conference, ICWE 2004, Web Engineering. Springer, pp.60-73.

- Fraternali, P. and Paolini, P. (1998).A Conceptual Model and a Tool Environment for Developing more Scalable, Dynamic, and Customizable Web Applications. In: Advances in Database Technology - EDBT'98. ACM, pp.421–435.
- Fraternali, P., Rossi, G. and Sánchez-Figueroa, F. (2010).Rich Internet Applications.IEEE Internet Computing, 14(3), pp.9-12.
- Garrigós, I., Mazón, J. and Trujillo, J. (2009).A Requirement Analysis Approach for Using i* in Web Engineering. In: 9th International Conference, ICWE 2009. Berlin Heidelberg: Springer-Verlag, pp.151-165.
- Garzotto, F., Paolini, P. and Schwabe, D. (1991).HDM - A Model for the Design of Hypertext Applications. In: Proceedings of Hypertext '91. ACM Press.
- Garzotto, F., Paolini, P. and Schwabe, D. (1993). HDM---a model-based approach to hypertext application design. ACM Transactions on Information Systems, 11(1), pp.1 -26.
- Gellersen, H. and Gaedke, M. (1999).Object-oriented Web application development.IEEE Internet Computing, 3(1), pp.60-68.
- Gómez, J., Cachero, C. and Pastor, O. (2000).Extending a Conceptual Modelling Approach to Web Application Design. In: Proceedings of the 12th International Conference on Advanced Information Systems Engineering. London, UK: Ed. Springer-Verlag, pp.79-93.
- Gómez, J., Cachero, C. and Pastor, O. (2000).Extending a Conceptual Modelling Approach to Web Application Design. Advanced Information Systems Engineering, pp.79-93.
- Gordijn, J., Yu, E. and van der Raadt, B. (2006).E-service design using i* and e/sup 3/ value modeling.IEEE Softw., 23(3), pp.26-33.
- Greenspun, P. (1997). Database backed Web sites. Emeryville, Calif.: Ziff-Davis Press, pp.214-251.

- Groenewegen, D., Hemel, Z., Kats, L. and Visser, E. (2008). WebDSL: a domain-specific language for dynamic web applications. In: 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications. TN (USA): ACM, pp.779-780.
- Harmelen, M. (2001). Interactive system design using Oo&hci methods. In: Object modeling and user interface design: designing interactive systems, Ed. Addison-Wesley Longman Publishing Co. Inc, pp.365-427.
- Hause, M. (2009). OMG Systems Modeling Language (OMG SysML™) Tutorial. INCOSE International Symposium, 19(1), pp.1840-1972.
- Hemel, Z., Verhaaf, R. and Visser, E. (2009). WebWorkFlow: An Object-Oriented Workflow Modeling Language for Web Applications. In: Model Driven Engineering Languages and Systems. Springer, vol. LNCS 5301 pp.113-127.
- Hennicker, R. and Koch, N. (2000). A UML-based methodology for hypermedia design. In: Proceedings of the 3rd international conference on The unified modeling language: advancing the standard. York (UK): Ed. Springer-Verlag, pp.410-424.
- Itu.int. (2012). Z.151 :  User Requirements Notation (URN) - Language definition. [online] Available at: <https://www.itu.int/rec/T-REC-Z.151/en>.
- Jacobson, I., Booch, G. and Rumbaugh, J. (1999). The unified software development process. Reading, Mass: Addison-Wesley, pp.1-512.
- Jacobson, I., Booch, G. and Rumbaugh, J. (1999). The unified software development process. Reading, Mass: Addison-Wesley.
- Kappel, G., Retschitzegger, W., Poll, W. and Schwinger, W. (2001). Modeling Ubiquitous Web Applications - The WUML Approach. In: Proceedings of the International Workshop on Data Semantics in Web Information Systems (DASWIS 2001). Yokohama, Japan.

- Kats, L., Bravenboer, M. and Visser, E. (2008). Mixing source and bytecode: a case for compilation by normalization. In: 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications. Nashville, TN (USA): ACM, pp.91-108.
- Koch, N. (2000). Hypermedia Systems Development based on the Unified Process. Ludwig-Maximilians-Universität München.
- Koch, N. and Kraus, A. (2002). The expressive power of uml-based web engineering. Málaga (Spain): Second International Workshop on Web-oriented Software Technology (IWWOST02), pp.105-120.
- Koch, N. and Wirsing, M. (2001). Software engineering for adaptive hypermedia applications. PhD. Thesis, Reihe Softwaretechnik, pp.145-289.
- Koch, N., Kraus, A. and Hennicker, R. (2001). The Authoring Process of the UML-based Web Engineering Approach. In: 1st International Workshop on Web-Oriented Software Technology. IEEE.
- Koch, N., Meliá-Beigbeder, S., Moreno-Vergara, N., Pelechano-Ferragud, V., Sánchez-Figueroa, F. and VaraMesa, J. (2008). Model-driven web engineering. Upgrade Novática Journal (English and Spanish), Council of European Professional Informatics Societies (CEPIS) IX, vol. 2, pp.40-45.
- Kraus, A. (2007). Model driven software engineering for web applications. München: Ludwig-Maximilians-Universität, pp.73-114.
- Kraus, A., Knapp, A. and Koch, N. (2007). Model-driven generation of web applications in UWE. In: Proceedings of the International Workshop on Model-Driven Web Engineering. Como (Italy), pp.23-38.
- Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In: 5th IEEE International Symposium on Requirements Engineering. IEEE, pp.249-262.

- Lamsweerde, A. (2004). Goal-oriented requirements engineering: a roundtrip from research to practice. In: 12th IEEE International Conference on Requirements Engineering. IEEE, pp.4-7.
- Linaje, M., Preciado, J. and Sanchez-Figueroa, F. (2007). Engineering Rich Internet Application User Interfaces over Legacy Web Models. IEEE Internet Computing, 11(6), pp.53-59.
- M. Fernández, F., Florescu, D., Kang, J., Levy, A. and Suciu, D. (1998). Catching the Boat with Strudel: Experiences with a Web-Site Management System. In: ACM SIGMOD International conference on Management of data. ACM, pp.414–425.
- McGrath, S. (1998).XML by example. Upper Saddle River, NJ: Prentice Hall PTR.
- Mecca, G., Merialdo, P., Atzeni, P. and Crescenzi, V. (1999).The ARANEUS Guide to Web Site Development.University of Rome.
- MeliáBeigbeder, S. (2007). WebSA: un método de desarrollo dirigido por modelos de arquitectura para aplicaciones web. Ph.D. Thesis, pp.85-223.
- Murugesan, S., Deshpande, Y., Hansen, S. and Ginige, A. (2001). Web Engineering: a New Discipline for Development of Web-Based Systems. In: Web Engineering, vol. 2016. pp.3-13.
- Nowack, P. (2000). Structures And Interactions - Characterizing Object-Oriented Software Architecture. Odense, Denmark: The Maersk Mc-Kinney Moeller Institute for Production Technology, University of Southern Denmark, pp.41-45.
- Parcus de Koch, N. (2001). Software engineering for adaptive hypermedia systems . München : Uni-Druck.
- Pastor, O., Fons, J., Pelechano, V. and Abrahão, S. (2006). Conceptual Modelling of Web Applications: The OOWS Approach. In: Web Engineering, Vol. 4143. Springer, pp.277-302.
- Pastor, O., Gómez, J., Insfrán, E. and Pelechano, V. (2001). The OO-method approach for information systems modeling: from object-oriented conceptual modeling to automated programming. Information Systems, 26(7), pp.507-534.

Pastor, O., Insfran, E., Pelechano, V., Romero, J. and Merseguer, J. (1997). OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods. In: International conference on advanced information systems. Barcelona (Spain): IN CAISE '97, pp.145-158.

Potts, C. (2016). Using schematic scenarios to understand user needs. In: 1st conference on Designing interactive systems: processes, practices, methods, & techniques. ACM, pp.247-256.

Rolland, C., Grosz, G. and Kla, R. (1999). Experience With Goal- Scenario Coupling In Requirements Engineering. In: IEEE International Symposium on Requirements Engineering 1998. Limerick, Ireland: IEEE.

Rolland, G., G.Grosz, C. and Kla, R. (1999). Experience With Goal-Scenario Coupling In Requirements Engineering. In: IEEE International Symposium on Requirements Engineering 1998. Limerick, Ireland: IEEE.

Rossi, G., Schwabe, D. and Garrido, A. (1997). Design Reuse in Hypermedia Applications Development. In: 8th ACM conference on HYPERTEXT '97. ACM, pp.57–66.

Schwabe, D. and Moura, S. (2003). Interface development for hypermedia applications in the semantic web. In: WebMedia&LAWeb 2004 Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress. IEEE Computer Society, pp.106-113.

Schwabe, D. and Rossi, G. (1995).Building hypermedia applications as navigational views of information models.california (USA): Hawaii International Conference on System Sciences. IEEE Computer Society, p.231.

Schwabe, D. and Rossi, G. (1995).The object-oriented hypermedia design model.Communications of the ACM, 38(8), pp.45-46.

Schwabe, D. and Rossi, G. (1995).The object-oriented hypermedia design model.Communications of the ACM, 38(8), pp.45-46.

Schwabe, D. and Rossi, G. (1998). An object oriented approach to web-based applications design. *Theory Pract.Obj. Syst.*, 4(4), pp.207-225.

Schwabe, D. and Rossi, G. (1998). An object oriented approach to web-based applications design. *Theory Pract.Obj. Syst.*, 4(4), pp.207-225.

Schwabe, D. and Rossi, G. (2001).A Conference Review System with OOHD. In: *First International Workshop on Web-Oriented Software Technology*.

Schwabe, D., MattosGuimaraes, R. and Rossi, G. (2002).Cohesive design of personalized Web applications.*IEEE Internet Computing*, 6(2), pp.34-43.

Schwabe, D., Szundy, G., Moura, S. and Lima, F. (2004).Design and implementation of semantic web applications.*WWW 2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web*, pp.1-7.

Selic, B. (2007). A Systematic Approach to Domain-Specific Language Design Using UML. *10th IEEE International Symposium on Object and Component-Oriented RealTime Distributed Computing (ISORC'07)*, Santorini island (Greece),pp.pp. 2-9.

Srivastava, S. (2013).A Repository of Software Requirement Patterns for Online Examination System.*International Journal of Computer Science*, 10(3), pp.247-255.

Srivastava, S. (2014). A Systematic Approach towards Transformation of Presentation Web Goal Oriented Requirements Language to Presentation Design. *International Journal of Scientific and Engineering Research*, 5(9), pp.7-17.

Srivastava, S. (2015).UML Profile for the WebGRL Requirements Model and EA00-H Design Models.*International Journal of Emerging Technology and Advanced Engineering*, 5(8), pp.313-322.

Srivastava, S. (2016). Model Transformation Approach for a Goal Oriented Requirements Engineering based WebGRL to Design Models. *International Journal of Soft Computing and Engineering (IJSCE)*, 3(6), pp.66-75.

Srivastava, S. and Chawla, S. (2010). Multifaceted Classification of Websites for Goal oriented Requirements Engineering. In: IC3 2010, Part I, CCIS 94. Berlin Heidelberg: S. Ranka et al, pp.479–485.

Stahl, T. and Völter , M. (2006).Model-driven software development. Chichester, England: John Wiley.

Valverde, F., Valderas, P., Fons, J. and Pastor, O. (2007). AMDABased Environment for Web Applications Development: From Conceptual Models to Code. In: 6th International Workshop on Web-Oriented Software Technologies. Bucharest (Romania), pp.164-178.

Visser, E. (2008). WebDSL: A Case Study in DomainSpecific Language Engineering. In: Generative and Transformational Techniques in Software Engineering II, 2nd ed. Springer, vol. 5235, pp.291-373.

www.omg.org. (2009). Unified Modeling Language™ (UML®). [online] Available at: <http://www.omg.org/spec/UML/2.2/Superstructure/PDF>

W3.org. (2004). Voice Extensible Markup Language (VoiceXML) Version2.0. [online] Available at: <http://www.w3.org/TR/voicexml20>.

W3.org. (2016).WAP Forum - W3C Cooperation White Paper. [online] Available at: <http://www.w3.org/TR/NOTE-WAP>.

W3.org. (n.d.).Extensible Markup Language (XML). [online] Available at: <http://www.w3.org/XML>.

Warmer, J. and Kleppe, A. (1999).The object constraint language. Reading [etc.]: Addison-Wesley.

CURRICULAM VITAE & LIST OF PUBLICATIONS

► SANGEETA SRIVASTAVA

B-203, Spring Valley Apartments,
Plot No. 3C, Sector 11, Dwarka, NewDelhi
Phone: 9873027777
E-mail: sangeeta.srivastava@gmail.com

Education

Ph.D in computer Engineering, Faculty of Technology, Delhi University, 2007

M.E in Computer Technology and Applications, DCE, Delhi University, 1997

B.Tech in Electrical Engineering, H.B.T.I, Kanpur University, 1987

- 2nd position holder and passed with honors in B.Tech.

Experience

Sr. Engineer, NHPC (1987 – 1991)

Dy.Manager, POWEGRID (1991-1999)

Associate Professor, BCAS, Delhi University (1999- till date)

My job responsibilities include designing the course structure, syllabus of both graduate and PG courses of Computer Sciences, conducting workshops related to the latest technology and Job Opportunities in the field of Computer Science , class room teaching, project work and research work in the field of Computer Sciences.

Skills

- My specialisation is in the field of Software Engineering and I have done my project and research works in this field. I have also published seven papers in conferences and 15 papers in international journals of repute.
- In all I have twenty two publications twenty in International Journals and Conferences and two in National Conferences.

PUBLICATIONS LIST FOR THE POSTDOCTORAL RESEARCH WORK FROM July 2012 to June 2016.

Srivastava, S. (2013).A Repository of Software Requirement Patterns for Online Examination System.International Journal of Computer Science, 10(3), pp.247-255.

Srivastava, S. (2014). A Systematic Approach towards Transformation of Presentation Web Goal Oriented Requirements Language to Presentation Design. International Journal of Scientific and Engineering Research, 5(9), pp.7-17.

Srivastava, S. (2015).UML Profile for the WebGRL Requirements Model and EA00H Design Models.International Journal of Emerging Technology and Advanced Engineering, 5(8), pp.313-322.

Srivastava, S. (2016). Model Transformation Approach for a Goal Oriented Requirements Engineering based WebGRL to Design Models. International Journal of Soft Computing and Engineering (IJSCE), 3(6), pp.66-75.

Srivastava, S., Saxena, R. R., Gupta, V. (2016). The EA00-H Design Model for Transformation of WebGRL based Web Applications. Accepted in the International Journal of Engineering Research and Applications (IJERA).

Srivastava, S., Saxena, R. R., Gupta, V. (2016). The Web Application Tool for Transformation of WebGRL Requirements Model into the EA00-H design Model based Web Applications. Accepted in the International Journal of Recent Trends in Engineering & Research (IJRTER).

