



A Development Method for User Interfaces of Rich Internet Applications

By Francisco Javier Martínez Ruiz

A thesis submitted in fulfillment of the requirements for the degree of

Diploma of Extended Studies (Diplôme d'Études Approfondies)
in Management Sciences
Option "Information Systems"

of the Université catholique de Louvain

Examination committee:

Prof. Jean Vanderdonckt, Advisor
Prof. Monique Noirhomme-Fraiture, Examiner
Prof. Jaime Muñoz Arteaga, Examiner

Acknowledgement

I would like to express my thanks to:

- My advisor, Professor Jean Vanderdonckt, for his constant support and tremendous patience that make me focus in the goal.
- Professor Jaime Muñoz Arteaga for his ideas and academic discussions.
- Professors Monique Noirhomme-Fraiture for accepting to participate to the jury of this thesis.
- My parents: Juanita and Francisco (without their love, cares and support I couldn't do this).
- My colleagues from IAG school of management at Université catholique de Louvain. Special thanks to Josefina Guerrero, Adrian Stanciulescu and Juan Manuel Gonzalez Calleros.

This thesis was realized thanks to the support of:

- The Programme ALBan for their grant to support my work (*the European Union Programme of High Level Scholarships for Latin America, scholarship No. (E06D101371MX)*).
- The BHCI laboratory at UCL/IAG/ISYS.
- The University of Zacatecas (Specially, the PhD. H. Rene Vega Carrillo).
- The UsiXML Consortium¹.

¹ <http://www.usixml.org> for User Interface eXtensible Markup Language.

Table of Contents

ACKNOWLEDGEMENT	II
TABLE OF CONTENTS.....	1
TABLE OF FIGURES	4
TABLE OF TABLES	6
CHAPTER 1 INTRODUCTION.....	7
1.1 Motivation: Why Modelling Graphical User Interfaces for Rich Internet Applications based on model-driven engineering?.....	7
1.2 Thesis	9
1.2.1 Thesis statement	9
1.2.2 Definitions.....	9
1.2.3 Focus	11
1.3 Reading Map.....	11
CHAPTER 2 STATE OF THE ART.....	13
2.1 Introduction to Web Technology	13
2.1.1 The Web Platform	13
2.1.2 The Standard Web Application Architecture	15
2.1.3 The general RIA Architecture	16
2.1.4 Dialogue architectures	16
2.1.5 Design frameworks and tools	18
2.2 Classification of User Interface Models and design approaches.....	19
2.2.1 CAMELEON Framework	20
2.3 User Interface Description Languages	22
2.3.1 XML-based User Interface Description Languages	22
2.4 User Interface Modelling in Web development	22
2.4.1 HTML.....	22
2.4.2 XHTML.....	23
2.4.3 DISL	23

2.4.4	XAML	23
2.4.5	Open Laszlo (LZX)	23
2.4.6	XFORMS.....	23
2.4.7	MXML.....	24
2.4.8	XUL.....	24
2.4.9	XBL	24
2.4.10	SVG.....	24
2.4.11	AJAX.....	24
2.5	Comparison between UIDLS.....	24
2.6	Comparison between Client-Server Dialogues	26
2.7	Comparison between normal webapps and RIAs	26
2.8	Conclusions	28
CHAPTER 3 TASK AND DOMAIN MODELLING.....		29
3.1	Task modelling.....	29
3.1.1	Introduction of case study	30
3.1.2	Task model of the Mobalpa Running Example (MRE).....	30
3.1.3	Task types: Are the RIA tasks different from other tasks?	32
3.2	Domain modelling.....	35
3.3	Derivation of RIA level from Task and Domain models.....	36
3.3.1	Categorization of RIAs.....	36
3.3.2	Dimensions of RIAs	38
3.3.3	Derivation of RIA level from Task and Domain models	39
3.3.4	Applying to the Mobalpa Running Example.....	39
3.4	The Role of XLST in the overall process of reification.....	40
3.4.1	The XLST language	41
CHAPTER 4 BUILDING THE ABSTRACT USER INTERFACE.....		42
4.1	Identification of Abstract Container Hierarchy	42
4.1.1	Containers and webapps	43
4.1.2	The Problem of creating the clusters and their associated containers	43
4.1.3	Applying to MRE	51
4.2	Selection of Individual Components	53
4.3	Designing the menu	56
4.3.1	Steps in menu selection	57
4.3.2	Algorithm to generate menu objects.....	58
4.3.3	Proposing a taxonomy of menu objects.....	60

4.3.4	Menus on RIA applications	64
4.3.5	Implantation of the menu in the AUI	65
CHAPTER 5 CONCRETE USER INTERFACE REPRESENTATION.....		66
5.1	Selection of target platform	66
5.2	Transformation of AUI in CUI	67
5.2.1	Selection of Concrete Interface Components	67
5.2.2	Defining CICs spatial position	68
5.2.3	Defining Navigation	68
5.2.4	Resulting CUI UsiXML specification	68
5.3	Refining presentation and behavior for CUI	69
5.3.1	Behavior a basic introduction	69
5.3.2	Behavior of the CUI representation	70
CHAPTER 6 GENERATION OF FINAL USER INTERFACE		71
6.1	Processing the CUI to generate the Final User Interface	71
CHAPTER 7 CONCLUSION		74
7.1	Summary of contributions	74
7.2	Brief discussion of future work	75
REFERENCES		76
ANNEX A. TASK TYPE TAXONOMY		88
ANNEX B. COMPARING STANDARD WEB APPLICATIONS AND RIAS		89

Table of Figures

Figure 1-1: XSLT transformation.....	11
Figure 1-2: Reading paths.....	12
Figure 2-1: Web Architecture components.....	13
Figure 2-2: Web Architecture relationships (based on [Jaco04]).....	14
Figure 2-3: Basic web Architecture.....	14
Figure 2-4: Web Application Architecture components.....	15
Figure 2-5: Typical architecture of RIAs.....	16
Figure 2-6: Dialogue architectures of Web applications.....	17
Figure 2-7: General Dialogue architecture of RIA applications.....	18
Figure 2-8: Classification of User interface models and design approaches.....	20
Figure 2-9: The CAMELEON Reference framework.....	21
Figure 3-1: In step 1 the task and domain models are created as well as the interaction of the User Interface.....	29
Figure 3-2: Mobalpa site.....	30
Figure 3-3: MCS Task model.....	31
Figure 3-4: Task types in a RIA application.....	32
Figure 3-5: The Star field visualization of the price range (unselected).....	33
Figure 3-6: The Star field visualization of the price range (selected).....	33
Figure 3-7: MCS Domain model.....	35
Figure 3-8: The Data/Complexity continuum in Web Applications.....	37
Figure 3-9: a Rich Internet Applications Categorization.....	38
Figure 3-10: Decision tree for derivate the RIA level.....	40
Figure 4-1: Activities to create an Abstract User Interface.....	42
Figure 4-2: a The Basic layers of a web application.....	45
Figure 4-3: example of a Java GUI and its hierarchy of containers.....	46
Figure 4-4: this minimal task tree shows the calculated values of x and y, 2 and 4 respectively.....	47
Figure 4-5: The identification of Abstract Containers (v containers) and their components	48
Figure 4-6: The possible configurations of the containers for three leave tasks affected by concurrent operators.....	50
Figure 4-7: Possible configuration of containers.....	51
Figure 4-8: The levels and containers of MRE.....	51
Figure 4-9: The Containers X_8 and X_7 that are generated.....	52
Figure 4-10: The Container X_6 with detail of navigation marker.....	52
Figure 4-11: The AUI with the selection of facets of containers X_8	54
Figure 4-12: The AUI with the selection of facets of containers X_7	54
Figure 4-13: The AUI with the selection of facets of containers X_6	54
Figure 4-14: The steps of interaction with a menu.....	57
Figure 4-15: Menu in the task tree of MRE.....	59
Figure 5-1: Sub steps of the transformation from AUI to CUI.....	66

Figure 5-2: Elements of graphic modality	67
Figure 5-3: Fragment of a CUI UsiXML file	69
Figure 6-1: The final step: delivering a RUI	71
Figure 6-2: The final step: delivering FUI code for the interpreters or compilers.	72
Figure 6-3: XSL transformation document	73
Figure 6-4: XAML resultant file	73
Figure 7-1: a development method for User Interfaces of RIAs	75

Table of tables

Table 2-1 Global comparison of UIDLs	25
Table 2-2 Evaluated features in the UIDLs	25
Table 2-3 Global comparison of dialogue features.....	26
Table 2-4 Description of abbreviations.....	26
Table 2-5 Comparison of features between SWA and RIAs	27
Table 2-6 Available frameworks for RIA platforms.....	27
Table 3-1 extension of Taxonomy for including RIA task types.....	34
Table 3-2 Task types proposed in [Gonz07].....	35
Table 4-1 UsiXML code of the three containers shown in the previous section.....	56
Table 4-2 Menu generation Algorithm	58
Table 4-3 Resultant menu structure for MRE.....	60
Table 4-4 Taxonomy of Menu elements.....	61
Table 4-5 Taxonomy of Menu elements extended to RIA	64
Table 5-1 Fragment of AUI (some AIOs) from MRE and their equivalent CIOs.....	67
Table 5-2 Dimensions of Behaviour Modelling (modified from [Bock99])	70
Table A-1: Task types.....	88
Table B-1 Features and Weights needed to categorize a RIA	89
Table B-2 Comparison between e-commerce applications	90
Table B-3 Comparison between weather web applications.....	91
Table B-4 Comparison between web mail applications	92
Table B-5 Comparison between map dispatcher applications.....	94
Table B-6 Comparison between online reservation systems	95

Chapter 1 Introduction

1.1 Motivation: Why Modelling Graphical User Interfaces for Rich Internet Applications based on model-driven engineering?

Web applications are increasing their integration to our life. First, the document-based web was substituted by applications using the same channel but trapped in the same structure: the page metaphor. Now the classical web application is yielding her position to more complex and intuitive applications: Rich internet applications (RIA).

This new kind of applications is emerging with the help of a plethora of toolkits and frameworks that help developers to create the sophisticated user interface that it's required by users since the beginning of web apps because of the natural comparison between desktop and web applications.

The Design of Rich User Interfaces for internet applications (RUIs) remains in the domain of experts who learned their craft over years. Some researchers [Prec05] are aware that there is a need of developing specific methodologies because the existing methodologies do not fulfil the challenges imposed by the RUIs among them we can list the followings:

- *Lack of methodology*: as we stated before there are some methodologies but they were proposed to endorse the development of classical web applications, for instance: methods for creating a RUI are proprietary solutions [Open06], [Flex07] or they are in initial states as [Lina07].
- *Lack of experience*: Every year, there are new emerging technologies, frameworks and tools for web applications. Each one requires a learning curve that is more pronounced by the lack of information about its

integration, capabilities and usability considerations to the existing technologies.

- *Lack of knowledge:* Developing a Web application requires gathering a vast knowledge in an important set of technologies which need to be carefully tuned to create in front of the user the illusion of unity. Most of developers couldn't achieve a full understanding of all the components and this is not even advisable because within development teams it's a better solution to specialize our personal.
- *Lack of Model-based tool support:* Proprietary and free tools are still scarce for designing or coding of RUIs. Most of the tools are drawing or template oriented user development environments.
- *Reaching known levels of consistency:* A user relies in his/her past experience with similar systems when using a new one (this is consistency, a well known principle of user interface design). Since the beginning of web apps the user's experience is behind the user expectations in comparison to desktop apps but now RUIs offers the possibility of reaching consistency of Web apps to known levels. Nevertheless, this implies a careful design, time and expertise of the developing tool.
- *Usability Considerations:* developing a web user interface is a demanding task because we have to deal with browsers that don't follow W3C standards and usability is sometimes compromised to limitations imposed for creating a User Interface intended to a wide audience (usually this implies a design with simpler interfaces with fixed features).
- *Definition of a general framework:* introducing a general model that allow developers to specify abstract user interfaces without being attached to any particular platform would reduce the costs of development and maintenance of RUIs besides to support reusability of components and adaptation of others to fulfil the requirements of alternative platforms.

1.2 Thesis

1.2.1 Thesis statement

The research questions that guide our research are intimately related with the spirit of Model Driven Architecture and standardization, there is a need to develop a proper model for these emerging technologies to reduce developing costs and to produce flexible and adaptable interfaces for the next technological leap. Therefore, we inquire the following:

1. What are the elements that make different RUIs from Traditional Web apps User Interfaces?
2. Is it possible to extract from the RUI UIDLs the common essential elements to model it in a neutral language?
3. What are the extensions needed in UsiXML to model today RUI frameworks features, such as: delivery of the interaction level, cinematic experience and multimedia elements?

Therefore, we will defend the following thesis:

The introduction of a meta-model of user interfaces for *Rich Internet Applications* establishes a common ground to standardize their design and development through a *model-based* and *neutral representation* which could be ported by a *transformational schema* to various web development environments.

The concepts introduced above are succinctly defined in the next section.

1.2.2 Definitions

1.2.2.a Rich Internet Applications

The Rich Internet Applications or RIAs are Web applications that transfer the load of processing the User Interface to the Web client while the control and business data is managed on the application server. A more complete description is devoted to them in the next chapter.

1.2.2.b Model-based approach

This approach uses models as development tool to specify a UI independent from a specific implementation [Flor06]. The models can be iteratively refined to finally deliver platform specific models [OMG07]. Usually, the UI specification is conformed by an interrelated set of models, each one describing an aspect of the general model. The main advantages of model-based development include: User-centered development (which allows designers manipulate tasks, users and domain abstractions instead of implementation specific details) and portability between platforms. Also in [Schm06] is highly pondered how model-based development can deal with complex platforms and domain integration into models.

1.2.2.c Neutral representation

This feature is important in three aspects: First, A neutral representation is intimately related with consistency because with a neutral representation is possible to use a consistent representation since early development steps to almost the final implementation. Second, independent description as discussed before allows portability between different environments. And third, the use of open standards instead of proprietary solutions reduces the risk of technological dependence of companies.

1.2.2.d Transformation schema

The Extensible Style sheet Language Transformation (XSLT) [Kay03] is the transformational approach used in this work. XSLT is indeed, a Turing complete language [Keps04] designed to transform a XML document into another XML document with a different structure. The source XML file remains the same and the result of the application of the transformation rules is deposited in a new file (Fig. 1-2).

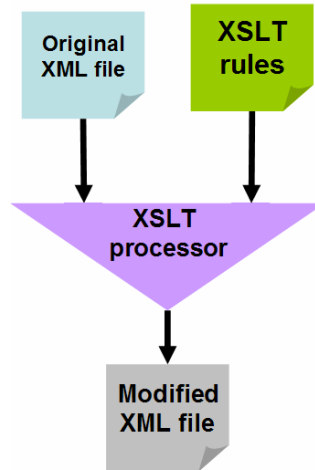


Figure 1-1: XSLT transformation.

1.2.3 Focus

- **What is the focus of this thesis?** Web applications, defined as: a distributed software system which uses a web browser as its deployment environment where the transit of data is endeavor with an HTTP gateway. The request parameters are treated by the Web server in order to generate a dynamic HTTP response [15]. Common examples are: Portals, e-commerce and search engines web apps.
- **Which Type of User Interfaces is studied?** Graphical User Interfaces (GUIs), because its preponderancy in the web apps arena.
- **To whom is directed this work?** As the presented tool remains a proof-of-concept prototype, the target audience is the CHI research community as well as academics interested in web applications and state-of-the-art development.

1.3 Reading Map

The rest of the sections are organized as follows:

Chapter 2 presents a state-of-the-art review of User Interface Description Languages (UIDLs) that are used for developing RUIs and the UI design in general. The following Chapter 3 introduces the process to generate the Task & domain models. Then, in chapter 4 discuss the generation of the Abstract User Interface. Next, chapter 5 explains the steps needed to obtain the Concrete User Interface. After, Chapter 6 describes our proposal to generate the Final User Interface. Finally, Chapter 7 provides the Author's conclusions, a summary of the

contributions and a brief discussion of future work. This whole methodology is supported by the CAMELEON [Calv03] framework and a XSLT transformation schema [Kay03]. In addition, for the different readers are provided alternative paths for guiding the lecture in the document. Two path of readings are considered (see fig. 1-2), the first one is recommended for the people with a basic level of knowledge in web technologies and the second should be selected by experts who already know about UI design.

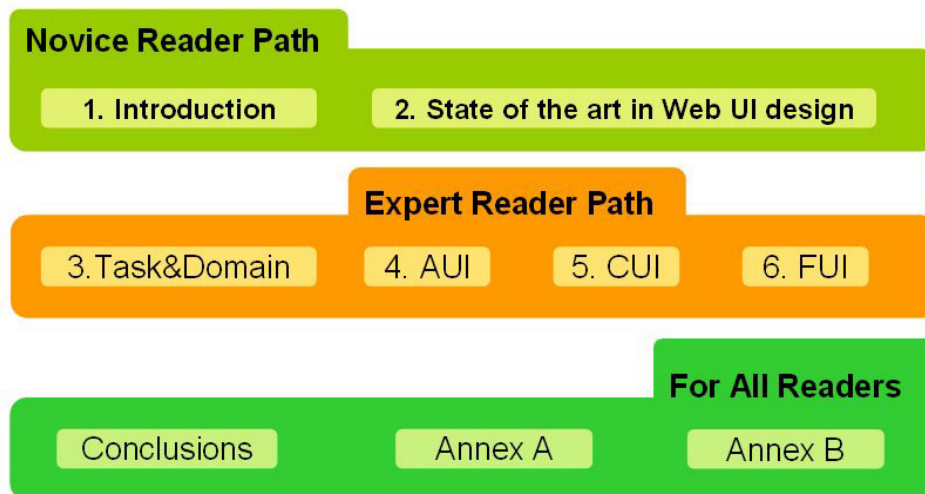


Figure 1-2: Reading paths.

Chapter 2 State of the Art

This chapter is divided into two sections: The first one includes a brief introduction of the Web platform, the Standard Web application architecture² beside the general RIA architecture. Then, the second part presents a review of the state of the art of User Interfaces within the context of the Web.

2.1 Introduction to Web Technology

2.1.1 The Web Platform

The World Wide Web (or simply web) is based upon three core components: The Uniform Resource identifier, the interaction protocols used by the agents to get the resources and a representation of the data contained by the resource [Jaco04]. The basic components are depicted in figure 2-1.

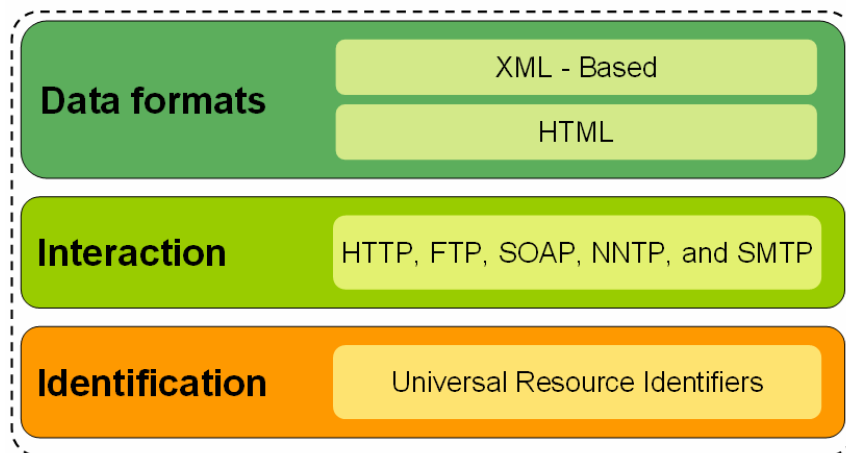


Figure 2-1: Web Architecture components.

All resource available through the Web uses a common representation system in order to assure the identification of the resource by all the agents (e.g., a Web browser). This marker is called: The Uniform Resource Identifier (URI). The syntax of the URIs is expressed using different schemes, for instance to represent an email we have to include “mailto” followed by a colon symbol “:” and then the email, to produce something like this:

mailto:joanna@myexample.org

² A brief definition of the concept of Web applications was included in the previous chapter but in this section we review them in deep for establishing their general architecture. Then, we use this description to build a comparison framework that is employed to analyse the structure of RIAs and extract their essential features.

The interaction between the Web agents is endeavour by standard protocols (e.g., HTTP, FTP, SOAP, NNTP, and SMTP) that allow the exchange of messages among them. For instance, if we have to retrieve some information from a FTP service, the procedure includes a command to request the data (e.g., a FTP GET request over the port 21 in the TCP/IP protocol) that will cause that the FTP server response with the transmission of data. Finally, the Web exchange of information requires a body of standardized data format specifications (e.g., XHTML, RDF/XML and CSS among others) for providing an adequate interpretation of the data. The relationship between the three elements of the architecture can be succinctly described with the following figure 2-2.

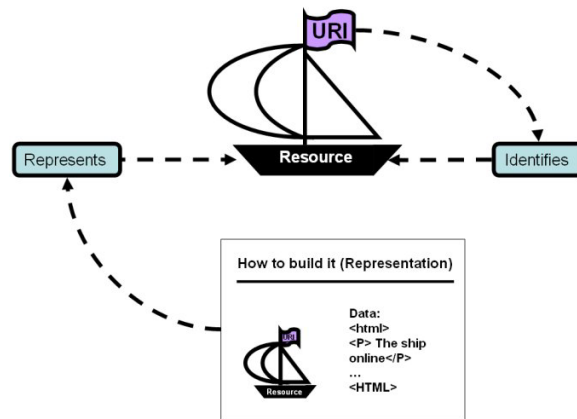


Figure 2-2: Web Architecture relationships (based on [Jaco04]).

The original function of the web was to deliver plain hypertext documents over a client-server architecture (see figure 2-3) but its success has caused a natural evolution from static and passive contents to dynamic and interactive ones. This offspring is discussed in the next section.

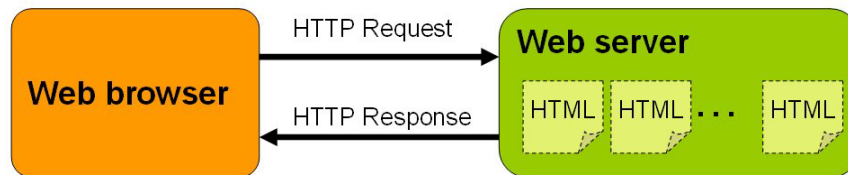


Figure 2-3: Basic web Architecture.

2.1.2 The Standard Web Application Architecture

We are going to work with Web applications, so it's very important to provide a proper definition:

“A Web application (or webapp) is a distributed software system that uses a Web browser as client to be accessed with minimal or zero installation procedures over a network using a XML dialect to build the User Interface, store and exploit data mainly through a HTTP gateway”.

This definition is based on [Jab04] and it settles the common ground to introduce the architecture of webapps. The standard architecture of webapps is based on the three-tier architecture [Ade95]. The web browser remains as the immovable and universal client (however some little variations are presented in the next section where this now standard model is expanded and compared with other common alternatives). The application server is the most important element in the second-tier (the server one) because it supplies the environment to execute the components of the application. Generally, these servers include a framework (APIs, interfaces and structures) to provide a way to interact with others systems and the user. Typically, the application server supplies an environment to construct the GUI in some User Interface Description Language (UIDL) e.g., HTML, or some XML dialect. Finally, in the third-tier is included a repository of data that is available through a data access interface from the second-tier. All these components are described in the figure 2-4.

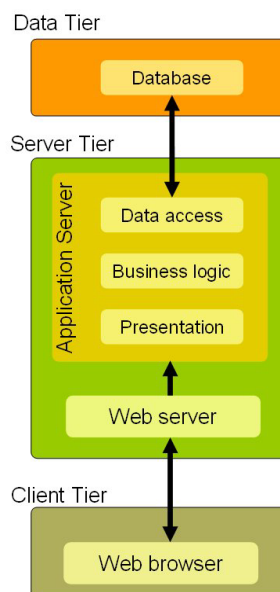


Figure 2-4: Web Application Architecture components.

2.1.3 The general RIA Architecture

The RIAs are webapps that take into account the power of the client to increase the responsiveness of the Web UI while the management of the application and data remains on the server. RIAs offer similar functionalities as the ones exhibit on desktop applications. A standard RIA architecture (Fig. 2-5) includes an application controller, an application server in charge of Web services calls that use some XML dialect to send data and layout information and a client rendering engine which is downloaded the first execution to process locally the presentation [9].

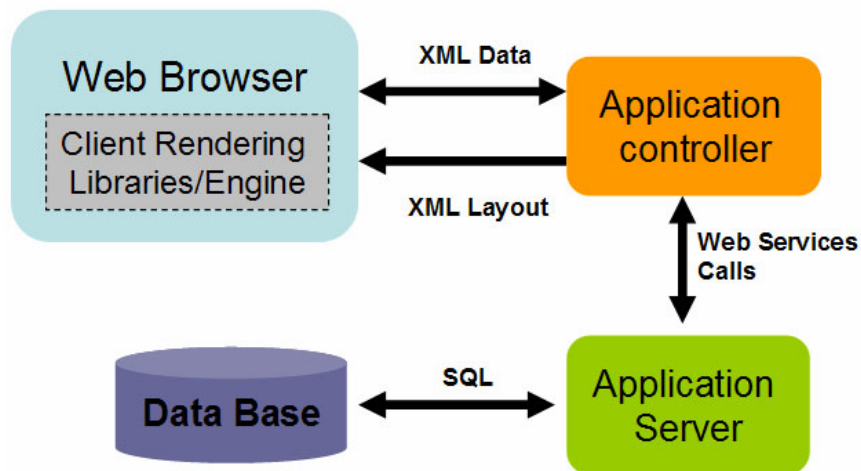


Figure 2-5: Typical architecture of RIAs.

2.1.4 Dialogue architectures

The Dialogue scheme is one of the most important aspects of the webapps and could be called navigation [Pont04]. Here we are going to analyse it in two different realms: First, the dialogue between the elements that support a webapp (e.g., browser or web server) using sequence diagrams [Booc05] to understand the interaction process among them. Second, we are going to review the proposed methodologies or tools to model the navigation inside a specific webapp.

2.1.4.a Dialogue : the big picture

This section presents a review of the three alternative dialogue architectures that are used in the design of webapps. This recapitalization is not exhaustive merely representative of the most general models. In fig. 2-6 we have depicted the dialogue of the typical webapps, in the left is the basic dialogue model of original web sites (fig. 2-1) where the client-server architecture is very clear but the persistence of the web browser is very limited since every http request/response event will cause a refresh and loss of the actual interface representation while in fig. 2-6 at right, we have the dialogue structure of standard webapps (fig. 2-4) where the most remarkable detail is the inclusion of the application server that handles the server scripts requested by the client in a safe environment (the server). In fact, this kind of configuration is preferred by business-oriented developers because of security and control reasons. In the other hand, The RIA dialogue architecture is different from the previous ones in many aspects: First, The inclusion of a new element, the **client engine** that provides all the functionality that the browser doesn't support or the one that is difficult to deliver without supplementary tools or frameworks. Second, the asynchronous communication process between the client and the server sides in the background without refreshing the UI, this includes a continue exchange of XML formatted streams to describe the interface, as well as the application data. And finally, most of the RIAs have associated to them a UI description language.

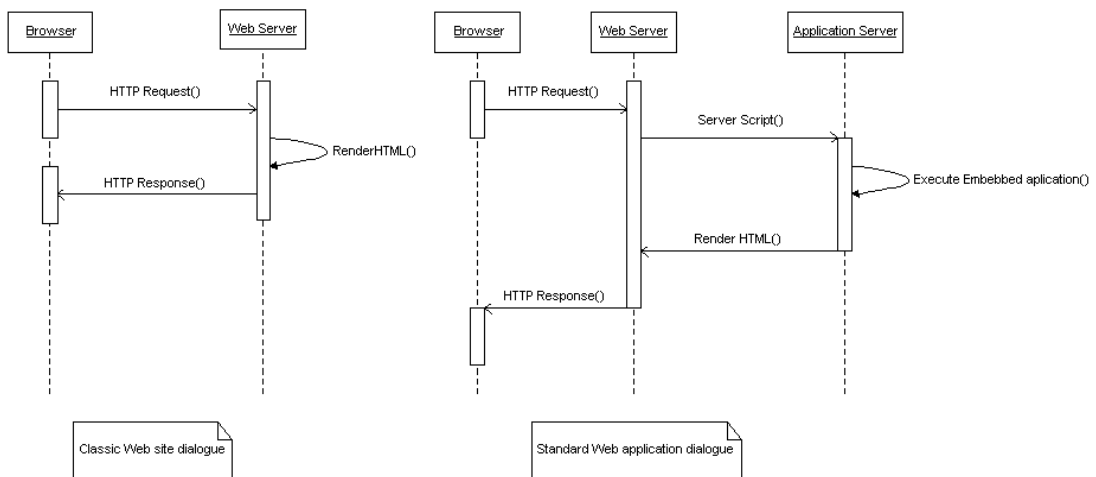


Figure 2-6: Dialogue architectures of Web applications

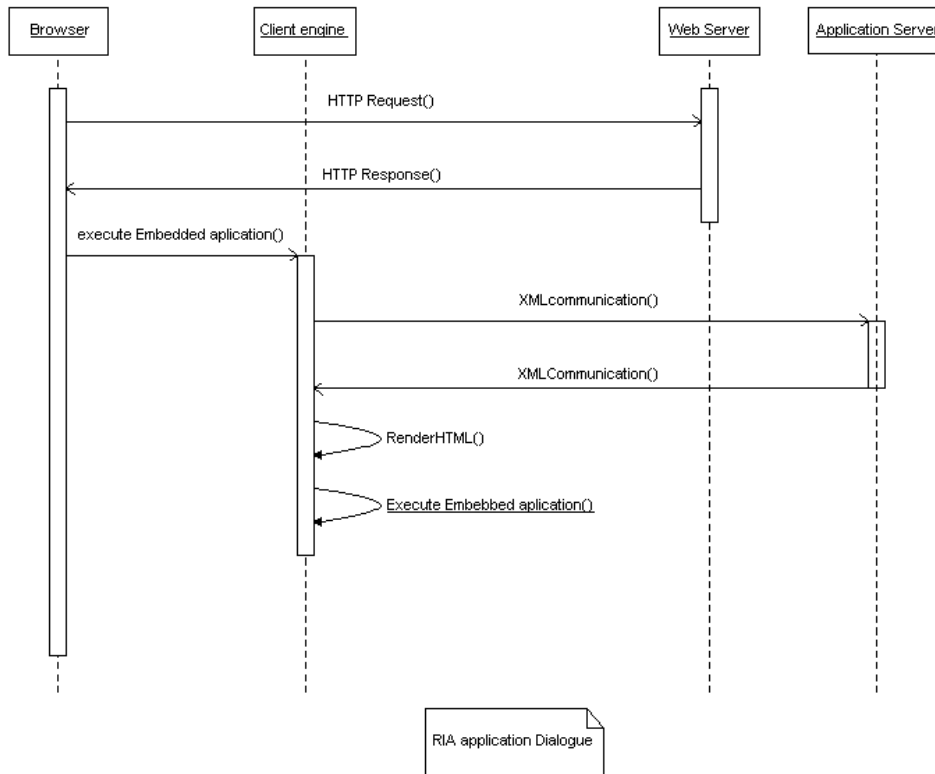


Figure 2-7: General Dialogue architecture of RIA applications

2.1.5 Design frameworks and tools

There are diverse approaches for modelling the navigation which are immersed in web design methodologies.

The OO-HDM methodology proposes UML extensions (activity nodes) to model the navigation [Ross03]. Also, the UWE methodology applies UML extensions (in fact, extended UML class diagrams besides stereotypes) trying to capture the web features including the navigation [Koch02]. Other UML associated method is the one included in [Cona02] that proposed a Web Application Extension (WAE) to UML in this methodology the navigation model is not as explicit as in the other methodologies. Other option is WSDM that is a user-driven methodology based on the ConcurTaskTrees notation where the navigation is modelled within its conceptual design step [Detr03]. In the line of data-oriented methodologies we have first OO-H designed for data-intensive webapps which is also based on UML extensions: the navigation access diagrams (NAD) and abstract presentation diagrams (APD) [Cach02]. And Second, WebML methodology that provides a navigational model within its hypertext model [Ceri01].

2.2 Classification of User Interface Models and design approaches

This section presents a review of the state of the art in the field of User Interfaces with special interest on UIs at the Web environment. The research of User Interfaces design is a very wide area so this review instead of trying to be exhaustive is a summary of the most common models and approaches (see figure 2-1 for a panoramic view). According to [Honk07] there are three differentiable areas: first, the **interaction models** that deals with user/computer interaction. The most representative of these models is the WIMP model that is conformed by Windows, Icons, Menus and a Pointer device [Canf90]. The direct manipulation model [Schn83], another model proposed is more advanced than the WIMP because it imposes less constraints to the interaction with the objects, for instance, in graphics-design applications. Other alternative is the direct combination model [Holl99] that is in fact, a specialization of the direct manipulation model which adds to element overlapping a semantic meaning to cause the execution of some operation, e.g., if a magnification glass object is superposed to a map image, the result should be a zoom operation.

Second, **architectural models** propose alternative structures to coordinate presentation, dialogue and data. In this section we included the models associated to the web environment. The MVC model was proposed for desktop applications but in web applications it's popularly named model-2 [Sesh99] and it's one of the most adapted in modern web applications (struts [Holm06], tapestry [Lewi04] among others). The Presentation-Abstraction-control is based on a hierarchical structure of agents that includes in each agent the three elements [Cout87] in order to be capable of running in different threads. Service-oriented architecture organizes resources in a network without knowledge of implementation details. Multi-tier architecture is an extension of the client-server architecture, in Web applications a typical implementation of this architecture is the 3-tier version [Ecke95]. And third, the **implementation models** that are all the languages and toolkits which produce a UI implemented in a specific platform.

Nevertheless, these models even if they could provide a guide for designing our UI, they have to be supported by a development approach; here we discuss the most prominent ones:

- The *exploratory approach* is based on developing mock-ups of the application interfaces for user evaluation. This approach takes advantage of the visual programming development environments where it is possible to construct easily a mock-up of the interface.

- The *programmatic approach* produces the UI representation by means of coding in a procedural, object oriented or declarative language that includes a toolkit with a set of common widgets in order to simplify the design task [Limb04].
- The *model-based approach* pretends build upon abstractions an iterative, quality-based and reproducible process, a systematic method for developing UIs [Limb04]. In the next section we are going to discuss in deep the elements of one of the most used frameworks in the model-based approach: The Camaleon Framework.

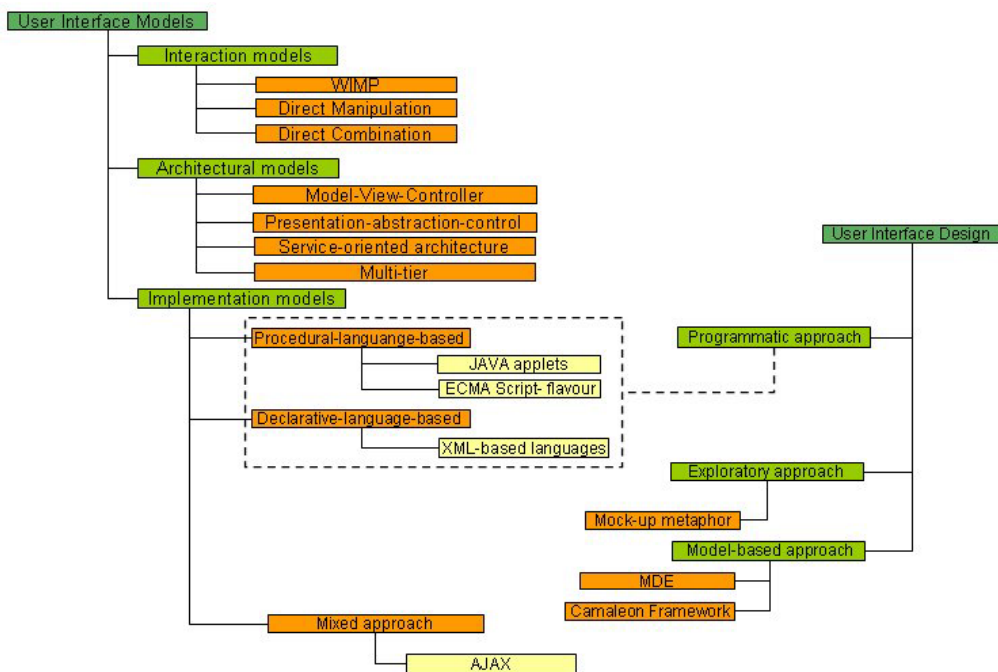


Figure 2-8: Classification of User interface models and design approaches

2.2.1 CAMELEON Framework

Building a model based application requires a framework to define the design steps needed for describe our computer system, including the features: Multi-level abstraction, Modality independence, among others [Boui05]. The Cameleon Reference framework [Calv03] expresses these features to describe an application. This framework structures the development process within four levels of abstraction: Task and concepts, Abstract User Interface (AUI), Concrete User

Interface (CUI) and Final User Interface (FUI), as shown in Fig. 2, the arrows pointing to a lower position in a hierarchy represent reification steps (forward engineering) from abstract to a real world interface. Meanwhile, arrows pointing to upper positions reflect the process of inference abstract descriptions from the run-time code (reverse engineering).

To denote a UI at any level of abstraction, it's required a User Interface Description Language (UIDL) [Flor06]. One of these description model based languages is UsiXML (UsiXML which stands for User Interface eXtensible Markup Language). This language incorporates the four abstraction levels of Fig. 2 as described in [Limb04].

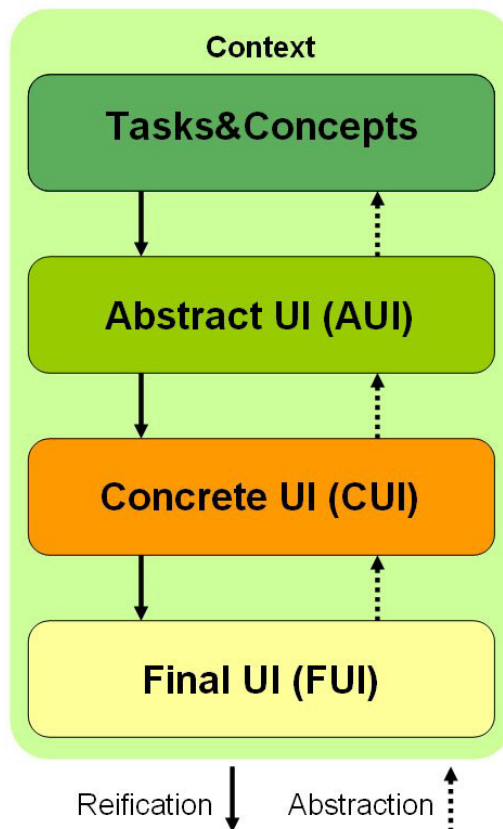


Figure 2-9: The CAMELEON Reference framework

2.3 User Interface Description Languages

A User Interface Description Language is a XML-based declarative language that is used to describe a UI. There is a plethora of User Interface Description Languages (UIDLs) here we are going to focus in the ones used in Web design.

2.3.1 XML-based User Interface Description Languages

Some of the reviewed UIDLs have already been analysed in [Souc03]. And can be, as well, used in desktop or in web environments.

2.3.1.a XIML

The eXtensible Interface Markup Language (XIML) is a XML-based language for developing UIs that include various models: task, domain, user, presentation and dialogue [Puer02].

2.3.1.b UIML

User Interface Mark-up Language (UIML) is a XML-based language for describing a User Interface with some levels of abstraction, for instance, in the definition of UIML there is a very important element the <peers> tag that provides the needed mapped parameters to produce a final user interface according to some specific toolkit and the UI logic [Abra99]. It's one of the oldest attempts to produce a neutral representation of the UI.

2.3.1.c UsiXML

The USer Interface eXtensible Markup Language (UsiXML) is a XML-based language created to define the UI over multiple contexts of use. The UsiXML language covers all the layers of the Camaleon framework and includes a set of interrelated models to support a model-based approach for developing UIs (for instance, a task& domain model, an abstract User Interface Model, a Concrete User Interface model and a interesting addition is its transformation model that can be used to translate instances of the model from abstract levels to concrete ones in a forward engineering process or the other way around to recover the abstract user interface in a reverse engineering process) [Limb04].

2.4 User Interface Modelling in Web development

2.4.1 HTML

HTML (Hypertext Mark-up Language) remains the principal mark-up language on the web. This language describes the layout, appearance and content in a

nested structure of tags and human readable [Ragge99]. The combination of features besides the low learning curve are its major strength, this last feature has produced hordes of authors with no programming skills.

2.4.2 XHTML

The Extensible Hypertext Mark-up language (XHTML) is the successor of HTML, in fact is a XML version of HTML, this implies that a XHTML document must have a well-formed structure. Also describes the distribution of the elements in the page, content and the structure [Pemb02].

2.4.3 DISL

It's a XML-dialect created to design UI for Mobil devices, (especially in this derivation SDML). This version of DISL/UIDL was created to model UIs that must be transmitted in a low band network where data structures have to be stored in some buffer and then transmitted in series of bytes. As stated by the authors for a specific scenario Mobil phones applications. The main distinction with the other version (UIDL) is the definition of all the elements as lists where each element could be tracked using node-list attributes (for instance, next-widget or child-widget ids) [Muel04].

2.4.4 XAML

The Extensible Application Mark-up Language (XAML) is another XML-based language used for definition of UIs and their properties and components (UI elements) and also their interactions (events and data binding). The MS Windows Standard Development Kit for the new operating system Vista contains a Representation subsystem called Avalon which integrates XAML [XAML06].

2.4.5 Open Laszlo (LZX)

LZX is a XML-based language for describing UIs based on a declarative language very similar to HTML that after the creation of the UI. The interface is converted into FLASH or DHTML [Open06].

2.4.6 XFORMS

XFORMS is a XML-based language for data processing over web pages. It's the next generation of the standard web form. Some of its features are the inclusion of layers to deal in an independent way with the data, its structure, submission methods and form controls [Boye06].

2.4.7 MXML

MXML is a XML-based language to produce User Interfaces within the Adobe Flash [Flas06] environment to produce UIs and also to control the logic and behaviour of the defined elements [Kazo07].

2.4.8 XUL

XUL is a XML-based language intended to create UI structures for the Mozilla Browser, include a set of general widgets. XUL is built upon existing web standards, including CSS, EMAScript and DOM elements [Hyat01].

2.4.9 XBL

XML Binding Language (XBL) is a XML-based language created to bind the behaviour and look of XUL UIs, this language is in the process of being standardized by the W3C [Hyat00].

2.4.10 SVG

Scalable Vector Graphics (SVG) is a XML-based markup language proposed and created by the W3C to define the structure of 2D vector graphics. It also covers static as well as animated graphics. Since is a text representation, the internal search and accessibility is possible in SVG definitions. Includes features like: nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility [Ferr03].

2.4.11 AJAX

Asynchronous JavaScript and XML (Ajax) is the recycling of already known technologies for developing interactive web applications with data recuperation avoiding the refreshment of web pages, better speed, usability and functions. Ajax uses a combination of: XHTML (or HTML) and CSS, for marking up and styling information. A key element is the XMLHttpRequest object that is used to exchange data asynchronously with the web server. [Garr05].

2.5 Comparison between UIDLS

The result of our study is summarized in the following tables, the strengths and drawbacks of each of the languages make us conclude that the idea behind the

recycle already known technologies is the best path because none of the UIDLs as far as the ones compared have/include all the features needed to cover the full spectrum of webapps.

#	Name	A	B	C	D	E	F	G
1	HTML		S		+	++	-	++
2	XHTML		G		+	+		++
3	DISL	-	S		+	-		+
4	XAML		S		++	+		+
5	LZX		S		++	+	-	++
6	XFORMS		G		+	-	-	++
7	MXML		S		+	-		++
8	XUL		S			-	-	++
9	XBL		S		++	-	-	++
10	SVG		G		++	-	-	+
11	XIML		G		+		+	-
12	UIML		G		++	+	++	+
13	UsiXML	++	G	++	++	+	++	+

Table 2-1 Global comparison of UIDLs

Code	Name
A	Extensibility
B	Purpose
C	Transformation Model
D	Behaviour Integration
E	Ease implementation
F	Modality independence
G	Web integration

Table 2-2 Evaluated features in the UIDLs

Note: Purpose = {General, Specific}
 Good = ++
 Medium = +
 Low = -

2.6 Comparison between Client-Server Dialogues

#	Name	A	B	C	D	E
1	HTML	S	C, O	Y	N	Y
2	DHTML	S	C, O	Y	N	Y
3	XHTML	A	C, O	Y	N	Y
4	AJAX	A	C, O	Y	Y	Y
5	LZX	A	C, O	Y	Y	Y
6	XFORMS	A	C, O	Y	N	Y
7	MXML	A	C, O	Y	Y	Y
8	XUL/XBL	S	C, O	Y	Y	Y
9	JAVA applets	A	C, O	Y	Y	N

Table 2-3 Global comparison of dialogue features

Code	Name
A	Communication between client/server {SYN, ASYN}
B	Server technology {CGI-bin/JAVA technology, Own}
C	CSS {Y,N}
D	Client engine {Y,N}
E	ECMAScript support {Y,N}

Table 2-4 Description of abbreviations

2.7 Comparison between normal webapps and RIAs

In this section we include some features to establish a comparison point between normal/classical/typical webapps and RIAs. This is not an exhaustive study because the idea was to create a starting point and over this light classification in iterative cycles create new comparisons that could be contrasted to this one in order to include each time more details.

The comparison of frameworks is very restrictive because some of the so proclaimed RIA frameworks are only APIS for JavaScript with the inclusion of the XMLHttpRequest object. This list (while I'm writing this) and at the time of reading as well is getting old.

	SWA	Comments	RIA
Partial screen updates	✘	Using frames it's possible to mimic this behaviour	✓
Asynchronous communication	✘		✓
Widgets supporting direct manipulation	✘		✓
Multiple coordinated windows	✓	Partial	✓
Modal dialogs	✓	Partial	✓
Menus	✘		✓
Keyboard navigation	✓		✓

Table 2-5 Comparison of features between SWA and RIAs

Development language	AJAX	Mixed	AJAX	Java	Flash	
Framework	Rico	GWT	TIBCO		flash	Open Laszlo
Maintainability	-	+/-	-	+	+/-	+
Reliability	+/-	+	+/-	+	+	+/-
Availability	+	+	+	+/-	+/-	+/-
Scalability	-	+/-	-	+	+	+/-
Performance	+/-	+/-	+/-	-	+	+/-
Security	-	-	-	+	+/-	+/-
IDE available	-	Eclipse (-)	own	Eclipse netbeans (+)	Flash suite (+)	Eclipse (-)

Table 2-6 Available frameworks for RIA platforms

[+] = good, [] = null, [-] = Low, [+/-] = medium

2.8 Conclusions

The reviewing of the literature show us an increasing interest in the generation of frameworks and re recycling of known technologies to avoid one of the biggest problems of the Web: the compatibility and standardization. Also we present a very brief description of concepts, advantages and shortcomings of some of the most promising frameworks and languages in the Web field. The result of this process is the validation of UsiXML as one of the most versatile tools. Nevertheless, it is imperative the inclusion of improvements to model with ease the RIAs and with this work we begin this task.

Chapter 3 Task and Domain modelling

3.1 Task modelling

The task model is the definition of all the assignments and sub-assignments needed to fulfil a job. Meanwhile, the Domain model defines the type and scope of the elements involved in a task and we can add extra details not present in task model. In this chapter we are going to review in deep these models in order to build the “Germ of a User Interface” that is a hierarchy of tasks and relationships upon methods and data variables which are the basis of the UI. The graphical summarization of this chapter could be seen in figure 3-1.

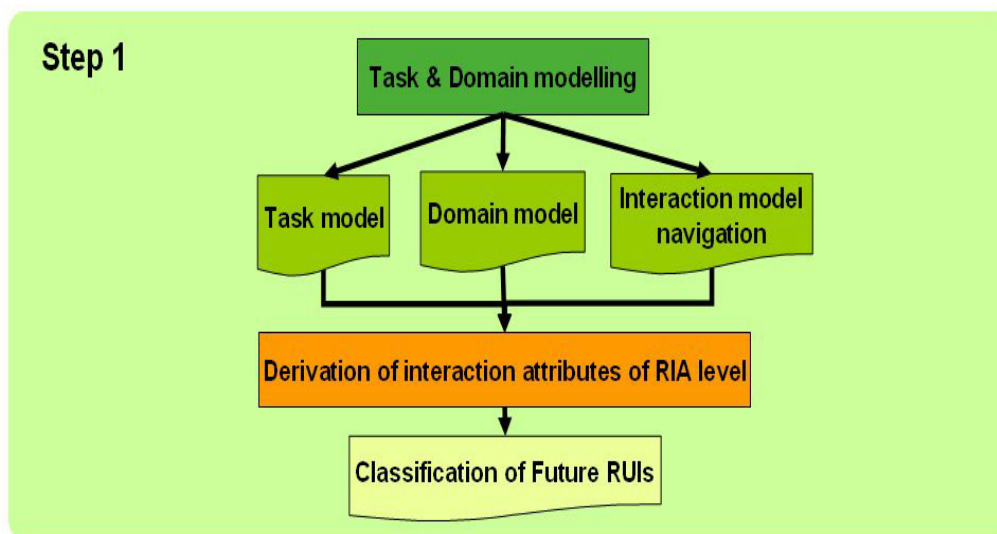


Figure 3-1: In step 1 the task and domain models are created as well as the interaction of the User Interface

Since we are working with UsiXML the selected tool for the task modelling is an extended version of the ConcurTaskTree notation [Pate99]. Such representation is capable of deal with the logical and temporal ordering of the user tasks and its benefits are: software engineering orientation, formal description based on LOTOS notation and easy communication (through a simple graphical notation) [Limb03]. Note: A bigger description is included in annex C.

3.1.1 Introduction of case study

Here we introduce the case study that is used in the whole document. It's the web site of the company of Mobalpa [Moba07] see figure 3-2, which has been selected because the combination of elements/amenities besides its design and use of the SPA approach. For the sake of simplicity in order to present a clear example we are going to model only the kitchens section, big enough to show the features of a RIA application.



Figure 3-2: Mobalpa site

3.1.2 Task model of the Mobalpa Running Example (MRE)

The initial step of designing a UI is the creation of a task model for MRE. a possible CTT tree is shown in figure 3-3, where the selection for expanding some of the tasks tries to present a general idea of the webapp.

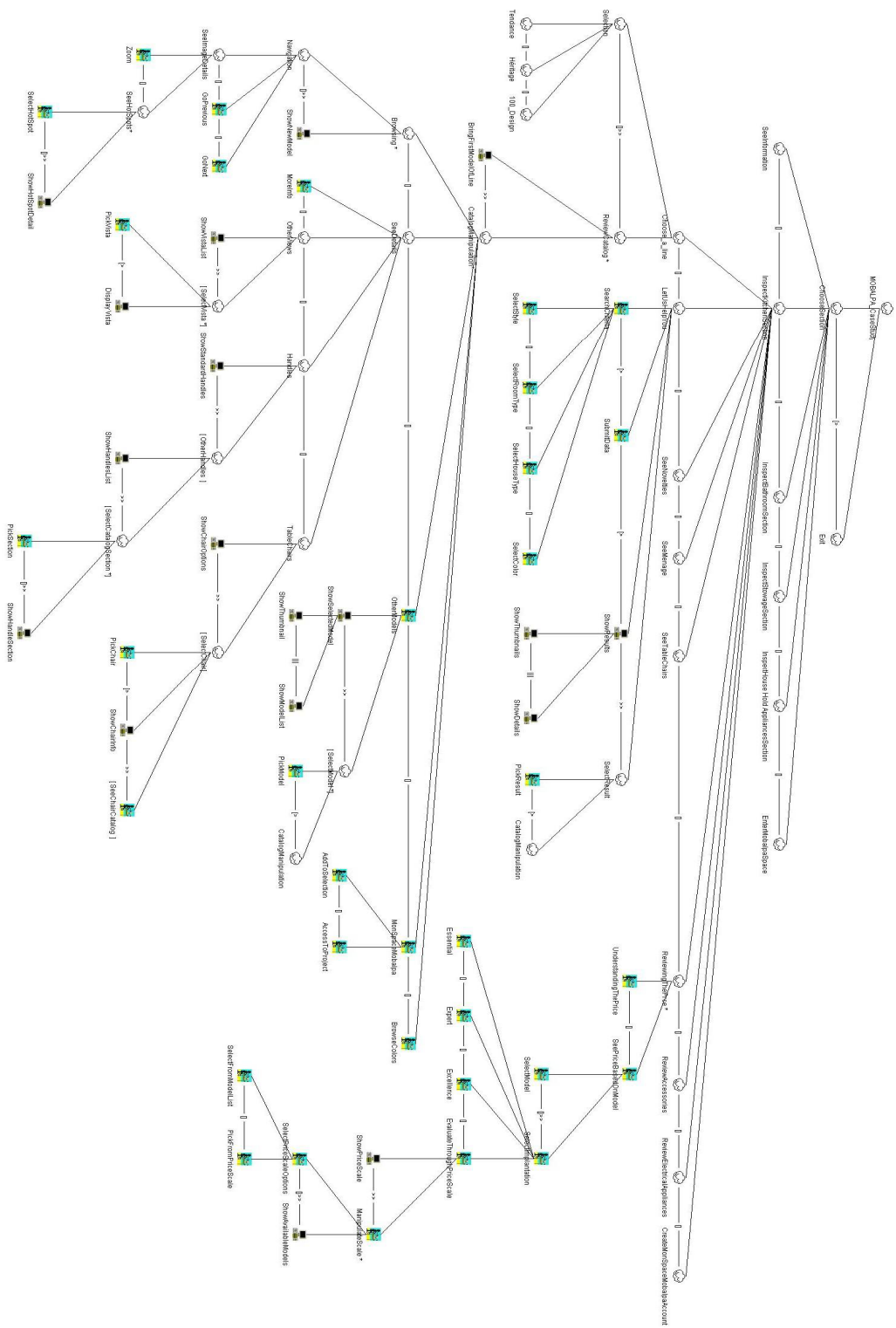


Figure 3-3: MCS Task model

3.1.3 Task types: Are the RIA tasks different from other tasks?

The tasks performed in a software system have been studied in many works: [Leno84], [Fole84], [Calh84], [Gree88], [Bles90] and [USIX07]. According to [Leno84] tasks are divided in three types:

- a) Goal-oriented: related with the user objective,
- b) Manipulation-oriented: related to the operation of the application,
- c) System-oriented: related to the execution of the internal logic of the application.

For instance, in a RIA application where the user is pursuing the goal of login to the system (see figure 3-4): there, the root level (a) and maybe some inner nodes of the task tree denote the goal level, the task that is the reason of the user interaction with the application. The operation of the system is shown in (b) and these activities are the manipulation of the application in order to fulfil the goal (these activities are composed by leave nodes; this is based on [Prib02]). Finally, The System tasks (c) also leave nodes are tasks that involve the running of the internal logic of the application but at two separated places: some logic is executed in the client realm (d) and other in the server realm (e). Then, the previous categorization should divide the last type to cover the notion of distributed execution.

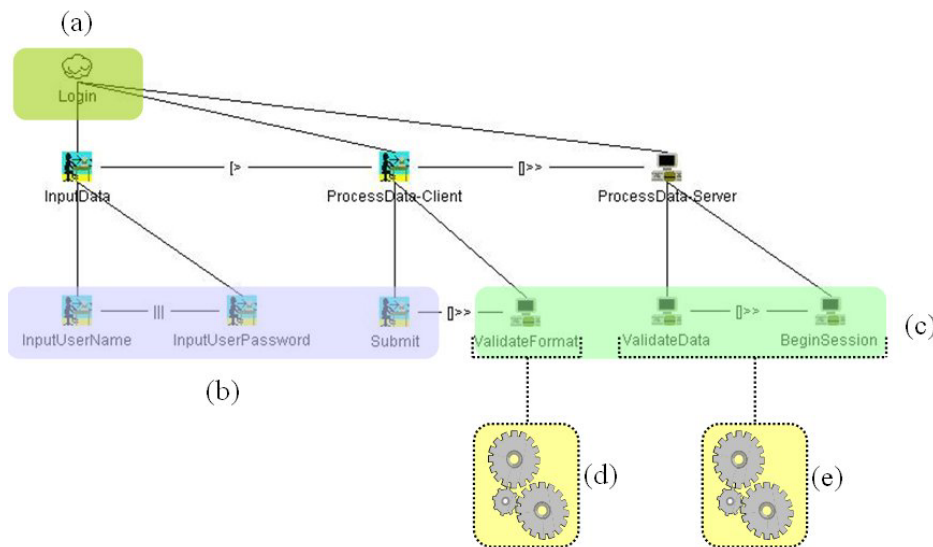


Figure 3-4: Task types in a RIA application

In table 3-1 from [Gonz07] there is a proposal of a task type classification that could be used for our purposes with some modifications. The main difference with the RIA realm (and Webapps in general) is the possibility of a range of hues. For instance, the **stop** task should not be integrate as the same or synonym other terms as **complete** since this concept could imply a different state of the task and second we have to include the modification of the task in the client or server sides. Third, we propose here a modification based in the integration of hybrid task types to model a kind of visualization that is included in the MRE: the star field visualization [Casn91]. Also in [Casn91] the author suggests that the success of applying certain graphic is how well support a specific task. Here, the selection of prices according to series of dimensions that are connected to create this dispersion of elements (that's why the name of star field) where the user could see in a very compact way the price range (see figures 3-5 and 3-6).



Figure 3-5: The Star field visualization of the price range (unselected).



Figure 3-6: The Star field visualization of the price range (selected)

The problem with this presentation is that the task required to model it is a combination of the tasks: **perceive** and **select** then we propose a modification for this new task (see table 3-1). This new hybrid task implies relations of dependency between the tasks, especially in the modification of their state. For instance, the last task proposed **Communication-navigation** implies that we are navigating and generating a communication for instance if the whole application is a map the menu is the application itself then there is a flow of info caused by the navigation³.

³ <http://maps.google.com>

Task Type	Client or Server effects	Definition
Stop Suspend Exit Cancel Terminate	Waiting for acknowledge from the user (Client/server sides)	Specifies the end of an action (abnormal or anticipate action)
End Finish Complete	Sending acknowledge from system (client/server)	Specifies the end of an action (normal or fulfilled action)
Perceive-Select	Modifying presentation (client)	The selection updates the visualization
Communication- Navigation	The visualization is updated (client)	The Navigation involves communication

Table 3-1 extension of Taxonomy for including RIA task types

Task Type	Synonyms/sub-task types	Definition
Communicate	Convey, Transmit, call, acknowledge, respond/answer, suggest, direct, instruct, request	The action to exchange information
Create	Input/Encode/Enter Associate, name, group, introduce, insert, (new), assemble, aggregate, overlay (cover), add	Specifies the creation of an item instance
Delete	Eliminate, Remove/cut, ungroup, disassociate, ungroup	The action of deleting an item
Duplicate	Copy	Specifies the copy of an item
Filter	Segregate, set aside	The action of filtering an item
Mediate	Analyze, synthesize, compare, evaluate, decide	The action of intercede task items
Modify	Change Alter, transform, tuning, rename, segregate, resize, and collapse/expand?	An action of modifying an item
Move	Relocate, Hide,show? position? Orient? Path or travel? X	the action to change the location of an item
Navigation	Go/To	the action to find the way through containers
Perceive	Acquire/detect/search for/scan/extract, identify / discriminate / recognize, Locate, Examine, monitor, scan, detect,	The action of identifying items and/or information from the items
Reinitialize	Wipe out, Clear, Erase	The action of cleaning an item
Select/choose	Pick	selection between items
Start	Initiate/Trigger, Play, Search, active, execute, function, record, purchase	Specifies the beginning of an operation

Stop	End / finish/exit/suspend?/complete? /Terminate/Cancel	Specifies the end of an action
Toggle	activate/ deactivate, /switch	The existence of two different states of an item

Table 3-2 Task types proposed in [Gonz07]

3.2 Domain modelling

Now, we have to define the domain of the elements used in our MRE, this can be achieved using IDEALXML tool [Mont06]. The domain model diagram presented is generated manually as the one produced by a software engineer (Figure 3-5). Next, we have to define the relations between the task and domain model. Some of the tasks in the task model are mapped to methods and attributes of the domain model. For instance, **showPriceScale** and **showAvailableModelsInsert** methods are included in the **Catalog** class. The **kitchen** class include all the attributes that are required to present a kitchen in the webapp.

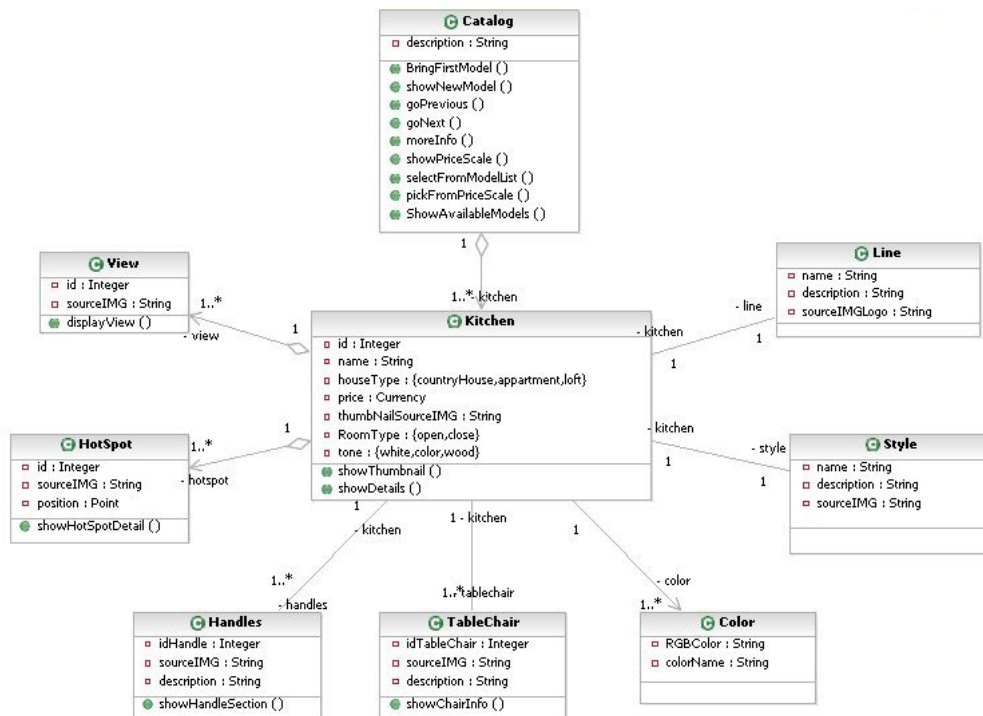


Figure 3-7: MCS Domain model

This model is a typical UML class diagram powerful enough to include the relationship between data and functionality of the user activities but that lacks the internal description of the multimedia elements that are described in [Lina07] this department is going to be expanded in UsiXML. For the moment we are considering the inclusion of the recommendation from W3C, the Synchronized Multimedia Integration Language (SMIL) language for the manipulation and integration of multimedia content [W3C05].

3.3 Derivation of RIA level from Task and Domain models

We are going to use a categorization of RIAs proposed in [Muño06] to infer the level that should have a RIA from its task and domain models. The purpose of this operation is to gather more information for guiding the selection of the best UI for our application. This is not a simple task; that's why the techniques and dimensions are explained in the following sections and finally the procedure to derivate the RIA level from the task and domain models.

3.3.1 Categorization of RIAs

The main problem behind the proposal of a RIA classification comes from the fact that they are a compendium of known Web technologies (JavaScript, CSS, XML and Java, among others) [Diaz97] that are used in novel ways including the integration of new features as the XMLHttpRequest [W3C06]. The process of classification could be seen as simple as tracking forward or backward in the Data/Complexity continuum of Web Applications (see figure 3-6) but in fact it's more difficult.

The major dilemma is that the utilization of RIA technology does not imply in all the cases that the application would exhibit an overwhelming set of multimedia capabilities we have to be cautious of fast categorizations. For instance, it's possible to create discrete examples with frameworks as Open Laszlo [Open06] that is one of the most popular and well established RIA threads nowadays and if the user does not know about the technology behind could assume a classical Web Application. Also the design and creation of a RIA is more complex than in a classical web application but the result is closer to the desktop application in features such as: changing shape cursors, Drag and Drop capabilities, embedded plug-ins to control video and audio streaming. These characteristics were used in [Prec05] to compare Web methodologies and in [Muño06] are used (among others) to propose a categorisation of RIAs.

This categorization of RIA applications is based on their extension and application domain (see figure 3-7) and defines three levels:

- Complementary applications (Level I). These applications are mini gadgets that are part of a bigger and usually more complex web application. They are very specialised and limited, e.g. calculators for particular fields or stream displays.
- Utilitarian Applications (Level II). These applications are activated by short time periods, e.g. a Web search engine service.
- Dominant Apps (Level III). The utilization period of these applications is very long also they interact with other applications without the user intervention from Web sources [Cran05]. An example could be a dashboard application⁴.

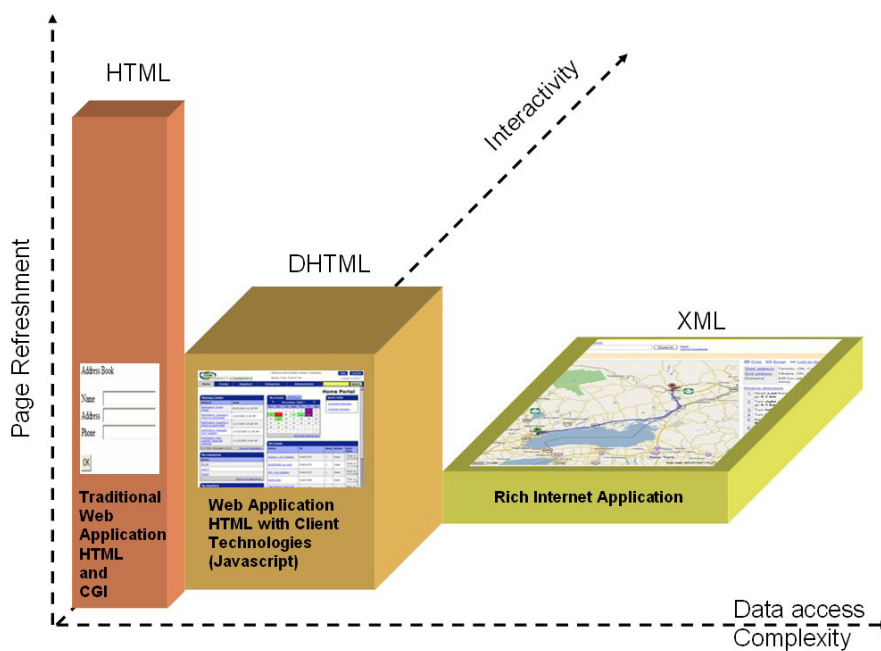


Figure 3-8: The Data/Complexity continuum in Web Applications

⁴ An example of a dashboard application is discussed in [Paye02]

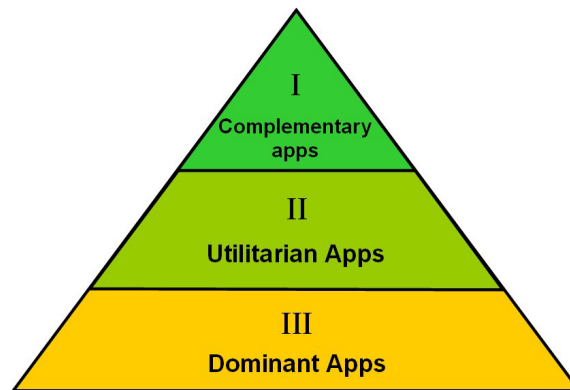


Figure 3-9: a Rich Internet Applications Categorization

3.3.2 Dimensions of RIAs

The dimensions are the features that were used to identify a RIA. These features are based on [Cran05], [Prec05] and [Bozz06]:

- Dynamic data retrieval i.e. streams of data from client to server and vice versa at running time.
- Perceptive continuity that is the reduction of page refreshments and freeze conditions.
- Adaptability that is the faculty of responding in autonomous way to the user necessities.
- Multimedia. Here understood as the capability of managing embedded graphics, video, audio and streaming.
- Collaborative faculties. That is the capability of cooperation between users to resolve a share problem or task.
- User Interface Description Language. Most of RIA applications include/propose a language for designing the UI.
- Push technology is the faculty of acquiring data that has not been requested by the user in order to update the current information [Fran98].
- The Use of Browser area. The RIA application breaks the structure of the classical web application and sometimes uses the whole window and eliminates the navigation bar (to break the web page model in the user's mind).

Note: The dimensions and their evaluation values as they were presented in [Muño06] are included in annex B (see table B-1).

3.3.3 Derivation of RIA level from Task and Domain models

The procedure that we propose for the derivation of the RIA level is based on the information recollected from the task and domain models and simple heuristics that we are going to describe below. For this we propose the use of a decision tree [Bres97] that could give us a rough estimate to infer the level of the RIA which is in front of us. The root of the decision tree deals with the **size** here understood as the amount of levels in the task tree that we get from a Breadth-first search i.e., aligning the siblings and counting the line as one level (The importance of the levels is discussed in the next chapter inside the container problematic). The amount of levels gives us the first cut: applications of **level I** could not include big arborescence structures so we presume/accept a maximum of three levels. The alternative path filter information from the domain model the **classes** and their **methods**. The procedure is straightforward having a major degree of complexity implies be more near to the complex types: **Level II and III**. At the end an extra consideration is pointed out, the appearance of **Concurrent operators**. This last filter is related to adaptability and collaborative dimensions because a complex application should include concurrent tasks. Nevertheless, this is a first approximation that we have to improve with the inclusion of more details and evaluation over other examples. The full decision tree is shown in figure 3-8.

But what is the purpose of this? For the moment is enough to say that the knowledge of the level a priori of the construction of the UI could help us in the selection of a better structure of containers/widgets. This will be discussed in other sections (chapter 4, 5 and 6).

3.3.4 Applying to the Mobalpa Running Example

The task and domain models from the webapp of the Mobalpa Company are available from previous sections so we can begin the analysis of the application: the amount of levels in the MRE is eleven. Then we have to verify how many classes contain this application in this example we found nine classes and a total of fifteen methods. The last decision is the based on the number of concurrent operators. The reason is that such operators are the “shadow” of tasks that at concrete levels are going to become tasks related to collaborative, multimedia and adaptability features of the application. In MRE we have found two in consequence our example is a **Level III application**. The implications of such result are for example that the design of our application is complex and probably it would necessary to create a complex hierarchy of containers.

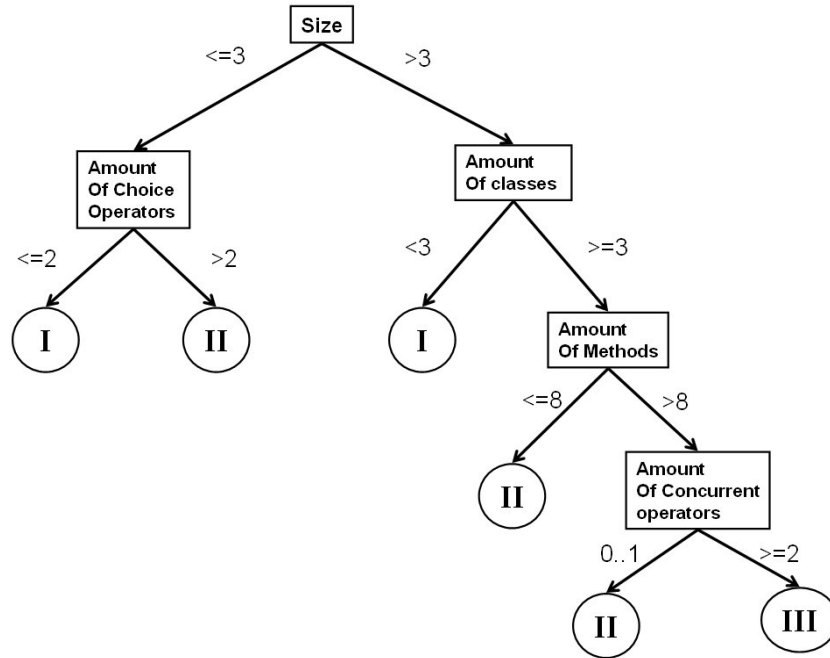


Figure 3-10: Decision tree for derive the RIA level.

3.4 The Role of XSLT in the overall process of reification

The reification process is the process to iterating from the abstract representation to more concrete levels and ultimately arrives to the Final User Interface. This overall procedure is called: forward engineering [Limb04]. The CAMALEON framework requires from each UI representation a process of transformation to deliver the next step.

We have suggested the Extensible Stylesheet Language Transformations (XSLT) language to deliver these transformations. Then in each step the next level requires the definition of a series of XSLT templates. In [Limb04] the transformation schema is based on Graph tree transformations adding lacking of abstraction and verbosity of XSLT. Nevertheless, the cost and losing of abstraction power is exceeded by the capacity of deliver automatic processing of the style sheets from a wide variety of XSL APIS besides the benefit of having an

XML representation and a XML transformation method in the nowadays standardization stage of the Web⁵.

3.4.1 The XSLT language

Extensible Stylesheet Language Transformations (XSLT) is an XML-based language that translates XML documents into documents with a different format and specification⁶ (most of the times other XML document). The transformation procedure includes three elements (see figure 1-2) and the mechanism for implementing the transformation is straightforward: we have to define a rule which is applied to a specific node that matches the imposed constraints from the Stylesheet the result will be the creation of nodes in the result tree [Kay03]. In [Mart06] we have used XSLT stylesheets to build some templates to transform the UI representation from CUI to FUI (specifically from UsiXML to XAML [XAML06]), this is a work in progress and the templates to cover the full set of elements of UsiXML are still under development.

⁵ The current version is XSLT 2.0, which reached W3C recommendation level at begin of 2007.

⁶ <http://www.w3.org/TR/xslt#section-Introduction>

Chapter 4 Building the Abstract User Interface

After gathering all the information of the germinal UI at Task & Domain levels we are going to use it for building an abstract representation of our UI called under CAMELEON framework: The Abstract User Interface (AUI). The construction of the AUI implies a series of challenges: The identification of Abstract Container Hierarchy, selection of Individual components and Menu design and finally the generation of the AUI. We are going to discuss them in the subsequent sections of this chapter.

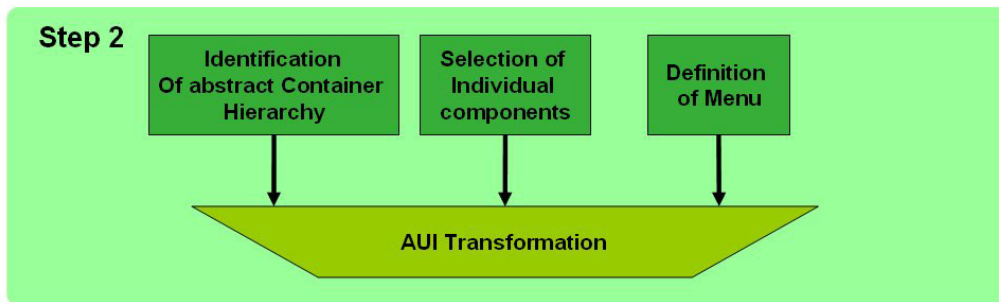


Figure 4-1: Activities to create an Abstract User Interface

4.1 Identification of Abstract Container Hierarchy

The UI design for Web applications is moving from the Web page metaphor to the Single Page Application (SPA) [Mahe06]. But this situation does not imply the extinction of the pagination understood as the problem of dividing a complex and big UI into a hierarchy of related widgets. As a matter of fact, it's the migration to a model that generates inner containers besides dynamic transposition of visible and hidden areas to divide our UI into sub UIs that are the visual representation of the task pursuit by the user. That's why we include as part of the study of RUIs the topic of container generation as well as a classification that would help us to understand more the RUIs and finally construct them. Before we go any further, the definition of container is provided:

The containers are widgets for grouping other widgets. They are equivalent to the Abstract Data Structures used to gather classes/objects and they share some of the same methods: add, remove and retrieve elements/widgets (based on [Klei04]).

4.1.1 Containers and webapps

In the realm of webapps the problem has been studied from many angles. In [Mand02] a run-time approach that evaluate the HTML code in order to identify specific tags and after collecting a fixed amount produce a “division mark” used to split the Web page and creates a new page. Other solution is the inclusion of a specialized mark-up language in order to define information units or sections within the UI (see [Spri03]). When the Web page is already created in [Chen05], the Document Object Model (DOM) is parsed looking for HTML tags e.g., DIV to define a division point. Using as base the HTML itself is acceptable but if you are looking for a more generic definition of the UI some researchers as [Göbe01] had used a User Interface Description Language (UIDL) in combination with another language to describe the dialog and navigation. For instance, the language XUL [Mozi07] used in [Ye04].

In [Chu04] the selected approach generates a widget hierarchy using a device-independent schema besides a bottom-up algorithm, a split or not attribute is applied to all the nodes of the tree of components the resultant sub trees are marked as a page. However, the process does not include temporal information from a task model and the division point is defined in a fixed schema. (Note: this approach is near to our own work that is going to be described in chapters 5 and 6).

The idea presented in [Flor06] is based precisely in a task model that is going to be explored. Here, the temporal operators play a capital role to provide information to define how to create the containers (indeed, the container concept fits barely in [Flor06]. Because, the process depicted in this work is based in the concept of interaction spaces). The task model is traversed in a Breadth-first search and using a set of principles begins a process of reduction from the most complex to the simplest platform. This process is known as “graceful degradation”. Also using as starting point the task model is [Prib02] where the information is extracted from a domain model besides the identification of the nature of the tasks that are divided in user goals and supplementary tasks. The last method does not worry about the space limitations because is looking for the relationships between tasks and subtasks in order to produce a general, device-independent UI, [Limb04].

4.1.2 The Problem of creating the clusters and their associated containers

In [Dyck90] the container problem is divided in two main threads: In one hand, the problem of trying to put all (or part) the elements into a single container with

the goal of maximize the container utilization in order to loose the less possible of container space. In the other hand, the problem is to put the elements (widgets) in one or more containers trying to do it in the minimal set of containers⁷. Here, we are dealing with the second instance. Then, we could subdivide the problem in the following aspects:

- a. What should be the better position for a container?
- b. How many containers should have a specific UI?
- c. How to cluster the components (widgets and other containers)?

Briefly, we are going to tackle each of these questions as preamble to the proposed solutions.

4.1.2.a What should be the better position for a container?

This question is related to spatial disposition or layout. The disposition inside the UI should deal with the screen positioning, dimensioning of components and arrangement [Boda94a] and at the same time each strategy requires to follow some ergonomic guidelines to produce a stable UI [Boda94b].

4.1.2.b How many containers should have a specific UI?

This problem is very complex because it combines the limit of the user cognitive load and the available space in the container father of the actual element or elements. The conventional approach to solve the container loading problem is by finite available space [Lim05]. Finally, the dimension of the solution space shows that it's not a trivial task that could be treated by brute force. For instance, if you want to test all the combinations of a set containing 10 containers the amount of variants is 115,975 [Weis07]. In this document we propose a solution that is included as part of the general method.

4.1.2.c How to cluster the components (widgets and other containers)?

The process of gathering is supported by libraries of containers e.g., the java SWING API [Java07] has a lot of components to support this task. An example of this container hierarchy is shown in figure 4-1. First, we have a frontier-element that interacts directly with the windows system of the operating system or that is contained by a general reader/launcher application (e.g., the Web browser). Its purpose is serving as background of the other containers of the application.

⁷ This is another variation of the well-known knapsack problem which is NP-hard.

It's the limit of our application. Second, we have internal windows. Third, container elements to hold up together the widgets for coherence and spatial reasons while the last one it's merely used for spatial distribution for a specific layout disposition. In this schema we have four layers but it could be extended since a window could include inner windows, dialogues or frames. This feature does not apply to all the levels, in the lowest one the elements could be restricted to hold up only widgets and no containers. This process of division is very important for the software applications, for instance it's better to divide than incorporating vertical scroll widgets that could become later a usability issue [Gill03].

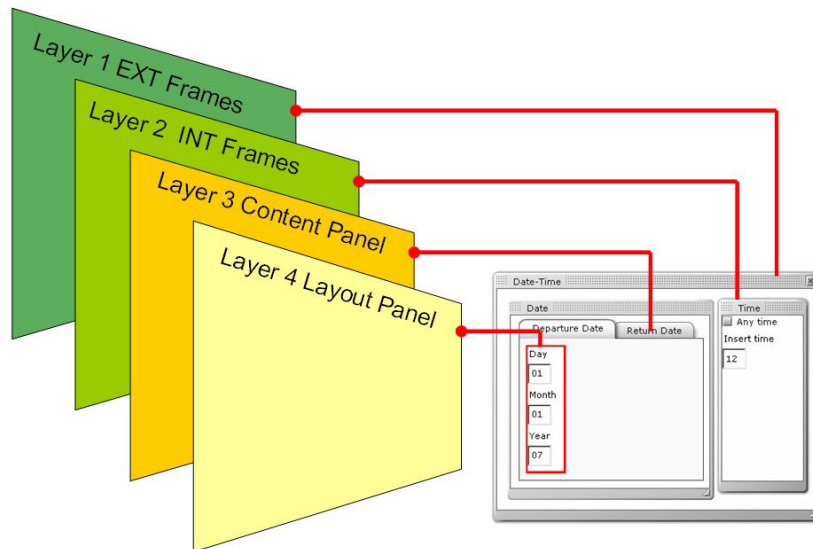


Figure 4-2: a The Basic layers of a web application

Now we are going to describe the proposed method steps in order to create a feasible set of containers that is conformed by four steps. Each following section is dedicated to their definition and discussion. At this point the designer has already built a task model. The gathering of the information as the sequential order and the disposition of the operators is essential to create acceptable container configurations.

4.1.2.d Normalization of task tree and definition of levels

In this step we have to normalize the whole tree to eliminate ambiguity but instead of using operator priority we are going to create dummy tasks (both possibilities were introduced in [Pate99]). The advantage of using the second

approach is simpler trees and the heuristic notion that these dummies sub trees give us extra information of the relationship between tasks. First, we have to determinate how many layers are acceptable for our platform. This means how many elements of containment could be embedded in one another. For instance, in the Java application of figure 4-2 we could count in the hierarchy of containers three layers. The next step in our method requires that the developer define the number of layers that are acceptable for the UI.

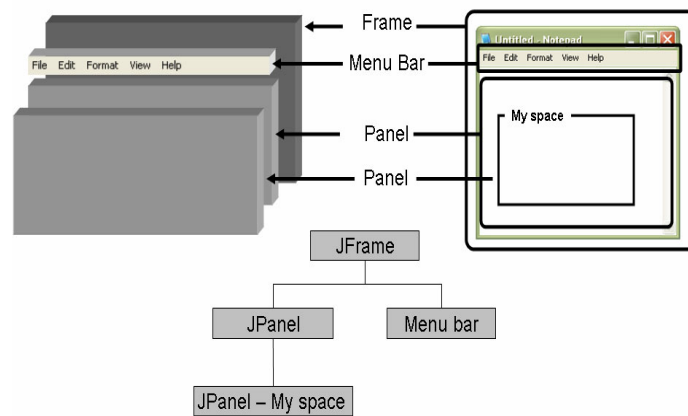


Figure 4-3: example of a Java GUI and its hierarchy of containers

4.1.2.e Division of the task tree into containers

Once the number of layers has been defined we have to follow this iterative procedure to analyze our task model:

- $x = 0$, this is the number of containers
- $y =$ number of layers
- Each node of the tree must be aligned with her siblings to generate a row that we call “level”.
- While the clustering of nodes does not cover the root,
- The node located in the position most to the left and in the lowest level is selected as **anchor**.
- Starting from the anchor we have to create a path to reach the father node which is in first layer (for instance, if we have a similar structure to figure 4-1, we have to jump four levels). The entire sub tree from the father node is going to be labeled as the next container (x is incremented).

At the end of this process we have x containers with y or fewer layers. A special situation could arise if we have three or less remaining levels before reaching the root, in that case we have a problem of fragmentation, similar to the problem of memory allocation for the process in Operating Systems [Stal01], a plausible solution is to agglutinate the root node to the last created container.

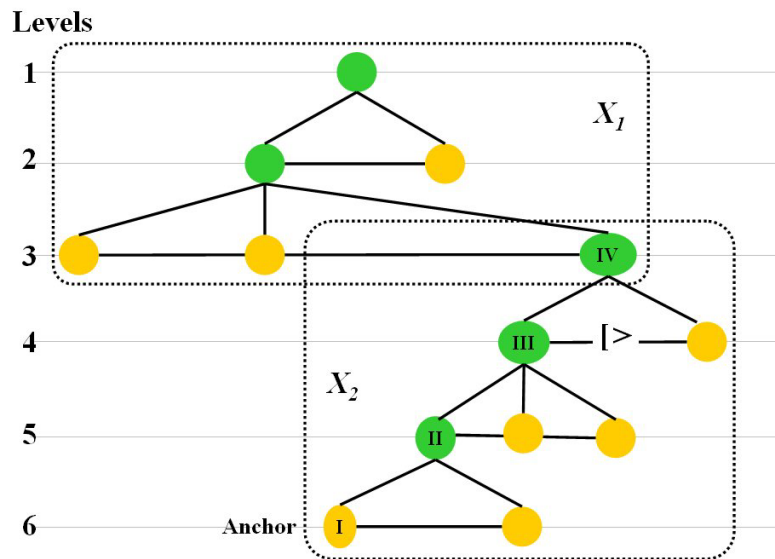


Figure 4-4: this minimal task tree shows the calculated values of x and y , 2 and 4 respectively

4.1.2.f Generation of the internal hierarchy of the containers

The previous step has provided us with an x set of containers. Each one should be evaluated in order to get the internal hierarchy of containers. The evaluation retakes part of the process used in [Limb04] to obtain an AUI and it could be roughly explained as the definition of two types of elements: the Abstract Containers (**AC**) and the Abstract Interface Components (**AIC**) (see figure 4-4). The ACs are elements of the tree which are inner nodes and the AICs are the leaf elements (Note: from this point the AC are called v (from virtual containers) to make the differentiation from real ACs that should be generated in the next step of reification of the CAMELEON/UsiXML method).

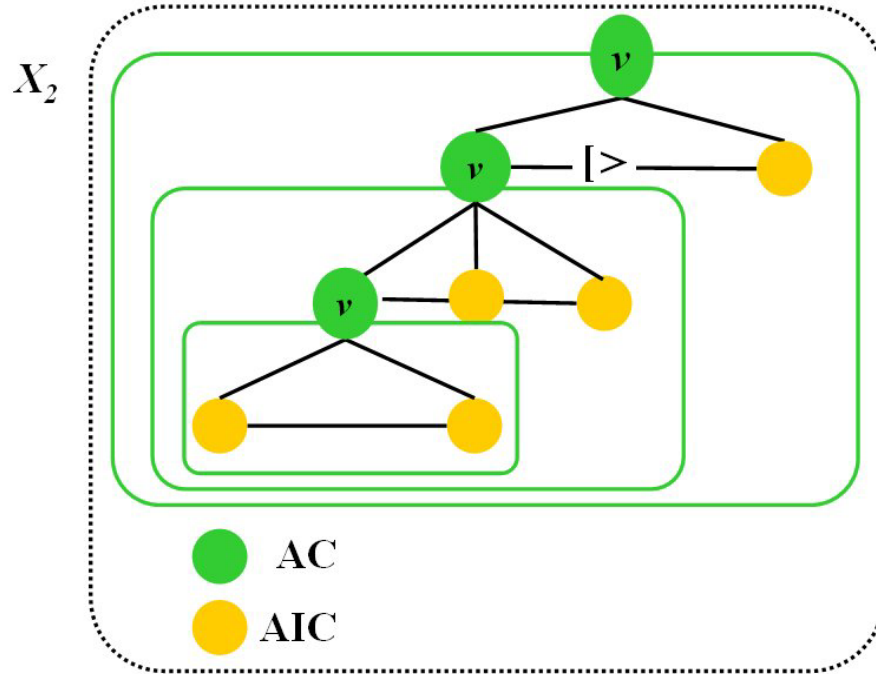


Figure 4-5: The identification of Abstract Containers (v containers) and their components

Again, using a Breadth-first walk over each container x we evaluate each sub container v with the following sub container Generation formula $G(n)$. Some assumptions are defined here in order to simplify the process. First, if a group of siblings contains a node that is also a father node then all are declared inner nodes P otherwise are leaves L and second all O operators are from the same group.

$$G(n) = \begin{cases} n & \leftarrow w \in P \wedge op \in C \\ B(n) & \leftarrow w \in Q \wedge op \in C \\ n & \leftarrow w \in P \wedge op \in F \\ 1 & \leftarrow w \in Q \wedge op \in F \\ BR(n) \cup 0 & \leftarrow w \in (P \vee Q) \wedge op \in S \end{cases} \quad (1)$$

Let

$w = \{\text{sons tasks which father is the root of } v\}$

$op = \text{operators interacting with the tasks}$

$P = \{\text{inner tasks}\}$
 $Q = \{\text{leave tasks}\}$
 $C = \{|||, |[]|, |=|\}$ all the concurrent operators
 $F = \{[]\}$ the Selection operator
 $S = \{>>, []>>, |>, [> \}$ is the set of sequential operators
 $n =$ the amount of son tasks (Beginning with zero)

The total number of container combinations when we don't have any restriction is equivalent to the problem of location of elements in a set of boxes [Weis07]. It could be calculated using the formula $\mathbf{B}(n)$.

The formula (2) stands for the Bell numbers [Weis07] which allows calculating the number of possible partitions of a set with n elements. The recursive definition of the Bell numbers is:

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k \quad (2)$$

Where n is the numbers of elements in the set and k the numbers of blocks. In our case we introduce also a constrained version of $\mathbf{B}(n)$ named $\mathbf{BR}(n)$ that includes the assumption of an integer order restriction that is the tasks in a sequence must follow the order depicted in the tree model, so:

For each task t associated to operators in S ,

$$R = \begin{cases} \text{valid} \leftarrow t_i \prec t_j : i, j \in N \wedge i \neq j \\ \emptyset \leftarrow \text{otherwise} \end{cases} \quad (3)$$

Then

$$BR_{n+1} = R(B_{n+1}) \quad (4)$$

The valid condition implies the inclusion of the configuration in the set of acceptable dispositions. \mathbf{N} is the set of all positive integers. The result of the application of (1) is a set of plausible scenarios of the UI. As in [Flor06] the selection of the preferred combination is leaved to the developer.

The function $\mathbf{G}(n)$ uses an operator based criteria besides the position of the nodes in the tree for offering possible container configurations of the tasks. For

instance, if we have a ν container (labelled F) as the one depicted in figure 4-5a since we have three concurrent tasks (A, B, C) we can choose between five different configurations for the containers (see figure 4-5b).

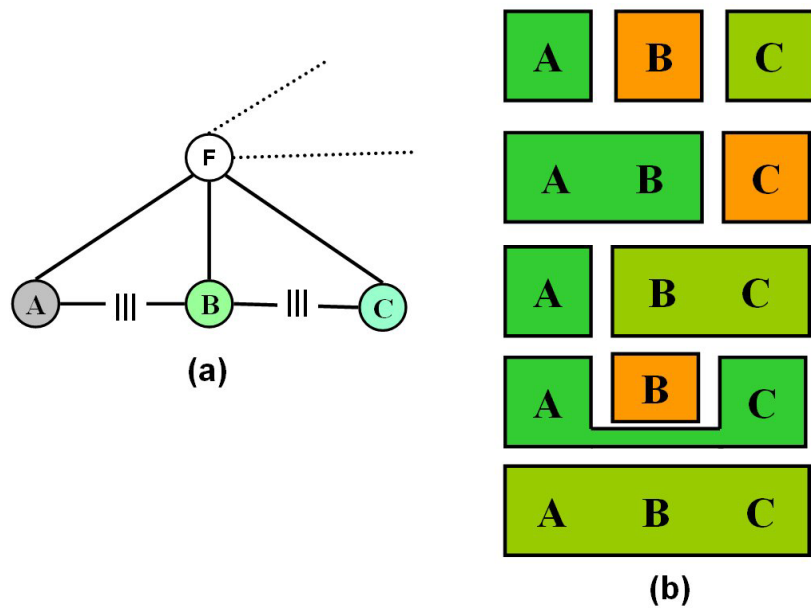


Figure 4-6: The possible configurations of the containers for three leaf tasks affected by concurrent operators

An example of the generation of the containers is shown in figure 4-6: the sons of node **F** would be derived in a couple of containers according to their sequential nature meanwhile the son tasks of **w12** because of the selection of the designer is gathered in a single compartment. All this information about the number and configuration of containers is stored to be used in the next step (that will be described in the following chapter). The election of zero containers derives in the elimination of the any AC sibling for example if **w11** and **w12** by designer's decision are together without labelling **w12** as AC the result would be the deletion of the last one.

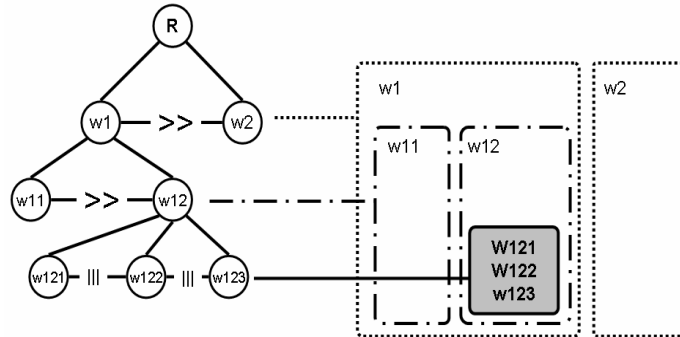


Figure 4-7: Possible configuration of containers

4.1.3 Applying to MRE

After the normalization we have to calculate the levels in the task tree. In our running example we have eleven levels. The levels could be recovered from a breadth-first walk over the tree. The result of this process is shown in Figure 4-7 also in the same figure we present the number of containers x and their sons v to be treated with $G(n)$.

The first anchor to be found, it's the task in charge of selecting the hot spot in the menu from this task we have to climb the tree to reach the father node which is located in the first layer (here we maintain the assumption again of 4 layers). This constitutes the first container x (figure 4-7.I). Now, looking at the same level we have found the next task to be marked as anchor (Figure 4.7.II). The process continues until we arrive to the last one in the first level which includes the root node so we have seven containers with 4 layers and one with 3 layers (figure 4-7).

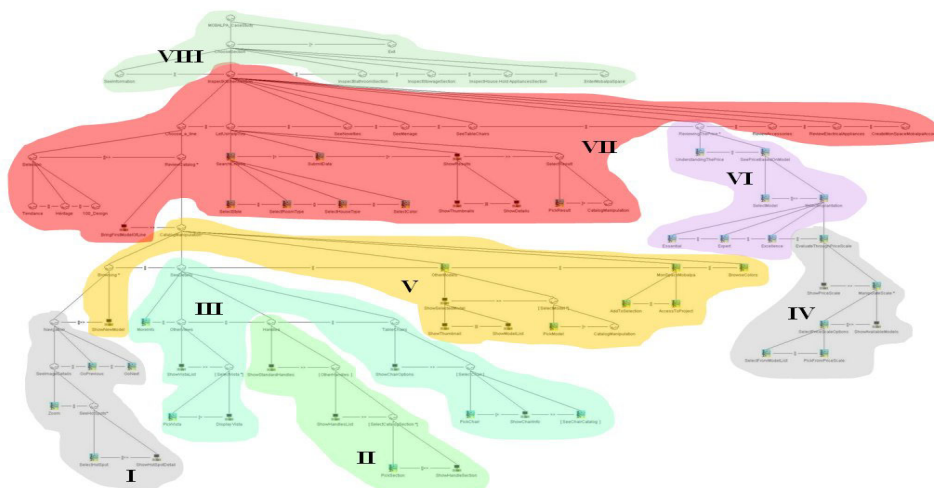


Figure 4-8: The levels and containers of MRE

The next step takes each container and creates the correspondent v sub containers (see figure 14). Then, we have to treat each v container in order to use them as input for the last step of the process and define their internal container configuration. The number of sub containers (in the layers) depends on the restrictions imposed by the formula (1) and the elections of the developer (A capital consideration is that our method is not searching –at this point- to produce the optimal neither the minimal configuration instead of that it’s looking to produce multiples valid scenarios to leave the developer with more options.

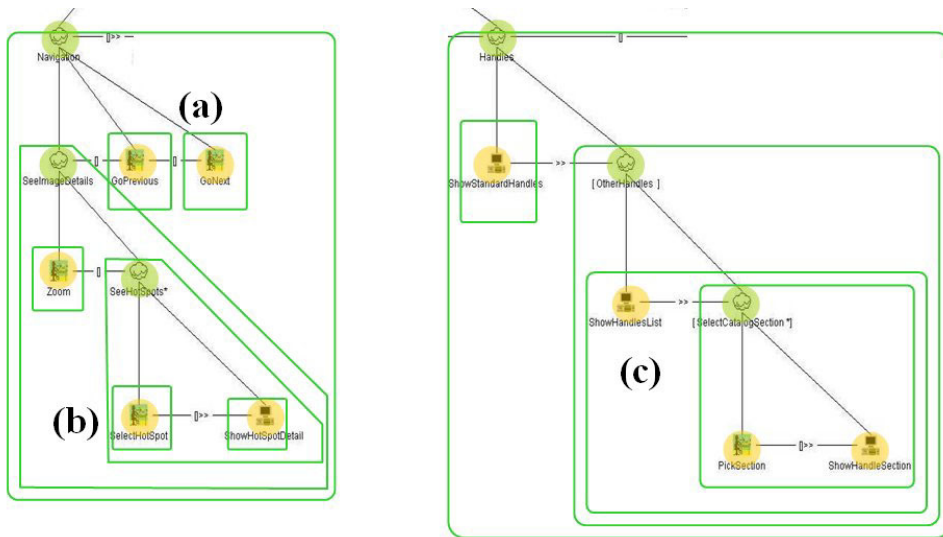


Figure 4-9: The Containers X8 and X7 that are generated

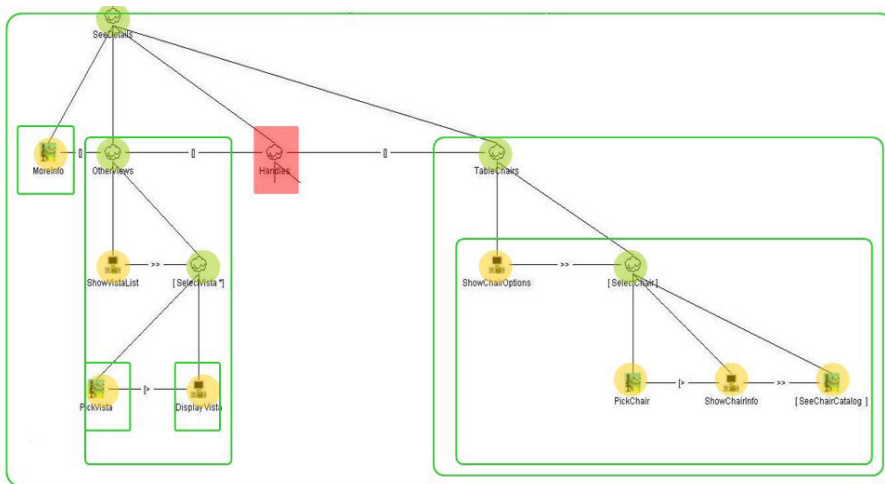


Figure 4-10: The Container X_6 with detail of navigation marker

The result of the application of $G(n)$ to the X_8 and X_7 containers is shown in the figure 4-9. It's important to remember that the selection of the structure is a “designer choice” e.g., the container that encapsulates the handle catalogue could be divided in more containers than the ones shown in the figure 4-9c. The function allow us to create all the combinations that do not break the sequence order beginning with zero then our preference was to create only one container. Meanwhile, in figure 4-9a because of the restrictions imposed by the choice operators the result is strictly three containers and finally in figure 4-9b the decision was to create two containers. Note: These selections were made to probe the algorithm not with a specific purpose in mind.

In the next step we are going to create the AUI using IdealXML tool [Mont06] where each X is mapped to a AC as well as the v containers that are populated with AICs. Only three of the eight abstract containers are shown in the following figures (4-11 to 4-13) for the sake of simplicity and focus in the method not in the repetition of steps.

4.2 Selection of Individual Components

The purpose of this section is finding the right component for the right job besides to deal with presentation and behaviour features, this information is going to be used in the next chapter to construct a concrete model of the UI. Here, we use information from the task model and domain model to generate the correct AIC specification.

This specification is extracted from a set of characteristics that have to be defined from the previous step: action types, action items, and task types, data types of domain attributes, domain of value of domain concepts, enumerated domains, inheritance and aggregations: That is the mapping between the task and domain model. The process that is suggested here is based on the general UsiXML method from [Limb04] with the difference of being based in XSLT transformations. The facets (or purpose) of an AIC defined, could take one of these values: input, output, control or navigation. (The pencil icon indicates an input facet, the button a control facet, and the green arrow pointing south is for navigation indications and finally, the magnifying glass is an output facet). The result of the process depicted in Figures 4-11 through 4-13.

Note: the inclusion of the RIA task type for the identification is a depending task that requires the verification of the task defined against a review of a more extensive pool of RIA examples that right now is being collected.

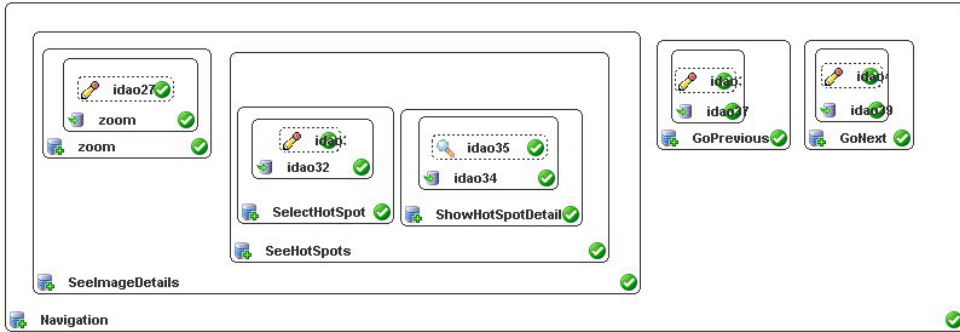


Figure 4-11: The AUI with the selection of facets of containers X_3

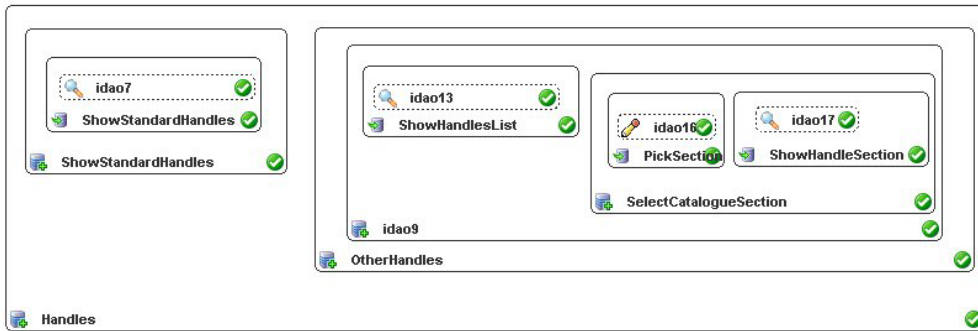


Figure 4-12: The AUI with the selection of facets of containers X_7

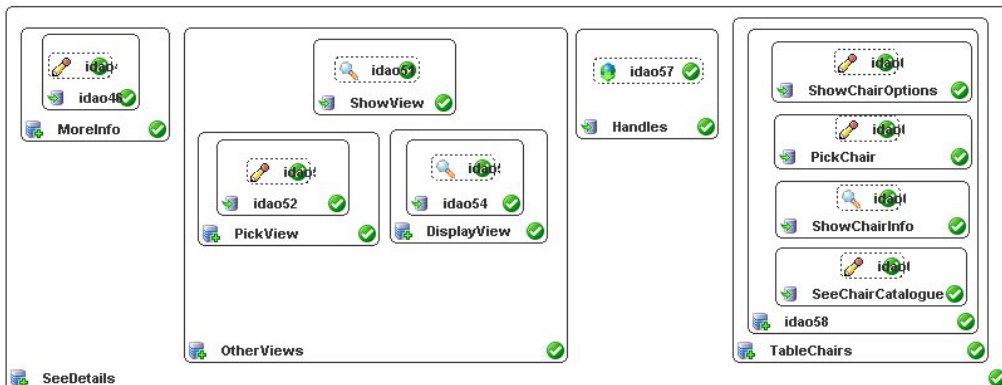


Figure 4-13: The AUI with the selection of facets of containers X_6

The next table is the resultant UsiXML specifications that were generated by IdealXML.

```
<abstractContainer id="idao0" name="Handles">
<abstractContainer id="idao3" name="OtherHandles">
<abstractContainer id="idao9" name="idao9">
  <abstractIndividualComponent id="idao11" name="ShowHandlesList">
    <output id="idao13" name="idao13" />
  </abstractIndividualComponent>
<abstractContainer id="idao12" name="SelectCatalogueSection" splittability="true">
  <abstractIndividualComponent id="idao14" name="PickSection">
    <input id="idao16" name="idao16" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao15" name="ShowHandleSection">
    <output id="idao17" name="idao17" />
  </abstractIndividualComponent>
</abstractContainer>
</abstractContainer>
</abstractContainer>
<abstractContainer id="idao4" name="ShowStandardHandles">
  <abstractIndividualComponent id="idao6" name="ShowStandardHandles">
    <output id="idao7" name="idao7" />
  </abstractIndividualComponent>
</abstractContainer>
</abstractContainer>
<abstractContainer id="idao20" name="Navigation">
<abstractContainer id="idao21" name="SeeImageDetails">
<abstractContainer id="idao24" name="zoom">
  <abstractIndividualComponent id="idao26" name="zoom">
    <input id="idao27" name="idao27" />
  </abstractIndividualComponent>
</abstractContainer>
<abstractContainer id="idao25" name="SeeHotSpots">
<abstractContainer id="idao29" name="SelectHotSpot">
  <abstractIndividualComponent id="idao32" name="idao32">
    <input id="idao33" name="idao33" />
  </abstractIndividualComponent>
</abstractContainer>
<abstractContainer id="idao31" name="ShowHotSpotDetail">
  <abstractIndividualComponent id="idao34" name="idao34">
    <output id="idao35" name="idao35" />
  </abstractIndividualComponent>
</abstractContainer>
</abstractContainer>
</abstractContainer>
</abstractContainer>
<abstractContainer id="idao22" name="GoPrevious">
  <abstractIndividualComponent id="idao37" name="idao37">
    <input id="idao38" name="idao38" />
  </abstractIndividualComponent>
</abstractContainer>
<abstractContainer id="idao23" name="GoNext">
  <abstractIndividualComponent id="idao39" name="idao39">
    <input id="idao40" name="idao40" />
  </abstractIndividualComponent>
</abstractContainer>
</abstractContainer>
</abstractContainer>
<abstractContainer id="idao42" name="SeeDetails">
<abstractContainer id="idao43" name="MoreInfo">
  <abstractIndividualComponent id="idao46" name="idao46">
    <input id="idao47" name="idao47" />
  </abstractIndividualComponent>
</abstractContainer>
</abstractContainer>
```

```
<abstractContainer id="idao44" name="OtherViews">
  <abstractIndividualComponent id="idao48" name="ShowView">
    <output id="idao51" name="idao51" />
  </abstractIndividualComponent>
</abstractContainer id="idao49" name="PickView">
  <abstractIndividualComponent id="idao52" name="idao52">
    <input id="idao53" name="idao53" />
  </abstractIndividualComponent>
</abstractContainer>
<abstractContainer id="idao50" name="DisplayView">
  <abstractIndividualComponent id="idao54" name="idao54">
    <output id="idao55" name="idao55" />
  </abstractIndividualComponent>
</abstractContainer>
</abstractContainer>
<abstractContainer id="idao45" name="TableChairs">
<abstractContainer id="idao58" name="idao58">
  <abstractIndividualComponent id="idao59" name="SeeChairCatalogue">
    <input id="idao66" name="idao66" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao60" name="ShowChairInfo">
    <output id="idao65" name="idao65" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao61" name="PickChair">
    <input id="idao64" name="idao64" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao62" name="ShowChairOptions">
    <input id="idao63" name="idao63" />
  </abstractIndividualComponent>
</abstractContainer>
</abstractContainer>
  <abstractIndividualComponent id="idao56" name="Handles">
    <navigation id="idao57" name="idao57" />
  </abstractIndividualComponent>
</abstractContainer>
```

Table 4-1 UsiXML code of the three containers shown in the previous section

4.3 Designing the menu

The RUI requires an especial attention in the designing of the menu because it's an exceptional element which contains the navigational and main control features of the UI. Then we need a definition of what is a menu:

The concept of menu is the activation of a group of actions. A menu is constituted by a name and a list of actions called menu items. Usually, a menu item is followed by an accelerator, i.e. a combination of keywords that allows selecting an item without mouse or keyboard selection. The advantage of menus is allowing the utilization of the options of the application without worrying about memorizing commands [Vand98].

4.3.1 Steps in menu selection

The steps that are required to interact with a menu are depicted in figure 4-8 this task tree shows a general model or pattern of the interaction with a menu and it allow us to divide the process in five sections that we are going to use to process a algorithm to model the menus in a webapp.

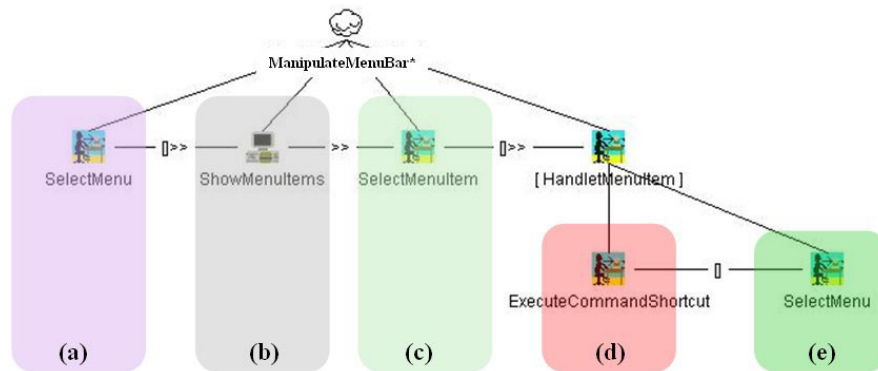


Figure 4-14: The steps of interaction with a menu

4.3.1.a Select menu

We could find inside the application some regions that show us the list of available options (menu) and we have to select one of these options. So in any region of a task tree with choice operators should be marked as a possible menu.

4.3.1.b Show menu items

The second step includes a task that could be explicit in the model (inclusion of a task node for it) or implicit (inside the logic of some interaction task).

4.3.1.c Select menu item

These elements of the menu could be seen as the task nodes in the sub tree of each task associated to a choice operator.

4.3.1.d Execution of command

This step implies that we have reached a leave node and we have an interactive or a system task in front of us. The work of a menu is leads to this point. Note depending of the structure of the menu the user could select this step or the next 4.3.1.e.

4.3.1.e Select (sub) menu

This step implies that we have more options to specialize our task, this inner menu use in a recursive way the task tree procedures of figure 4-8. But the node could include a behaviour called in table 4-2 level inflexion that could be different to the menu elements in upper levels.

4.3.2 Algorithm to generate menu objects

The algorithm that is presented here is a first approximation to the problem and should be tested against more task models to verify the feasibility for now remains as a probe of concept. Note: the menu structure that is generated in a semi-automatic procedure requires the correction of the designer but simplifies her/his work.

<pre>function Generate-Menu (CTT tree) returns menu-structure or failure initialize the search tree to root node loop do if there are no <i>candidate nodes</i> for expansion then return exit choose a <i>node</i> and expand its <i>sons</i> if the <i>sons</i> of <i>candidate node</i> include only <i>choice operators</i> then include it in <i>MenuList</i>[] and its sons as <i>menu items</i>. if <i>candidate node</i> previously marked as <i>menu item</i> then change the label to <i>submenu</i> associate to upper menu element if <i>candidate node</i> is unconnected to hierarchy of <i>MenuList</i> then mark it as flying menu and localized /*isolated*/</pre>

Table 4-2 Menu generation Algorithm

4.3.2.a Application of algorithm to MRE

In our example we got eight menus and the full structure is described in the table. The algorithm that is presented here is a first approximation to the problem and should be tested against

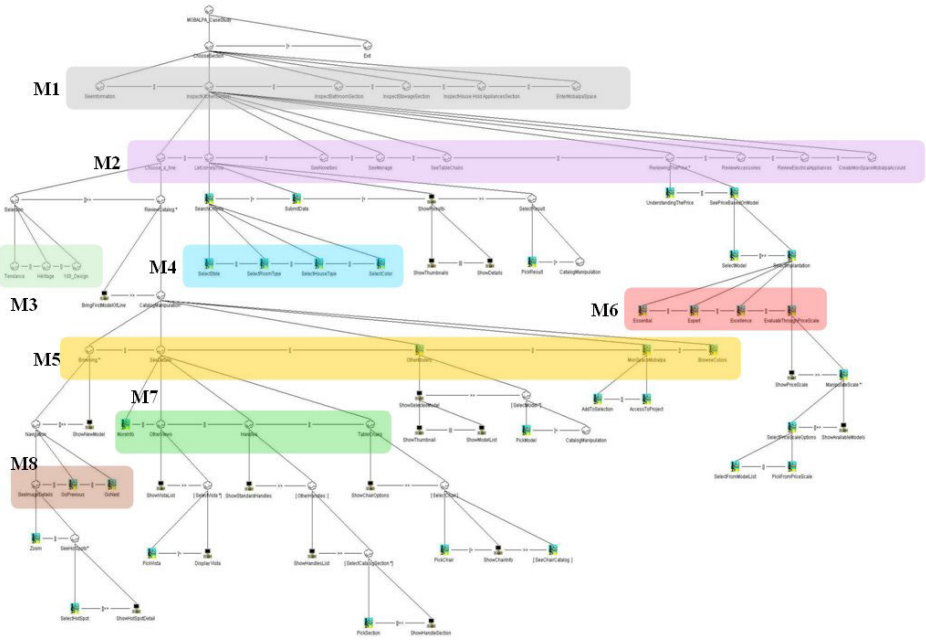


Figure 4-15: Menu in the task tree of MRE

Menu list	Menu item or <i>sub menu</i>	Member of main menu, or isolated
choose section M1	seeInformation, InspectKitchenSection , InspectBathroomSection, InspectStowageSection, InspectHouseHoldApplianceSection, EnterMobalpaSpace	Main menu member
Inspect Kitchen Section M2	Choose a Line, LetUsHelpYou, SeeNovelties, SeeMenage, SeeTableChairs, ReviewingThePrice, ReviewAccesories, ReviewElectricalAppliances, CreateMonSpaceMobalpaAccount	Main menu member
Selection M3	Tendance, Heritage, 100_Design	isolated
Search Criteria M4	SelectStyle, SelectRoomType, SelectHouseType, SelectColor	Isolated
Catalogue Manipulation M5	Browsing, SeeDetails , OtherModels, MonSpaceMobalpa, BrowseColors	isolated
Select implantation M6	Essential, Expert, Excellence, evaluateThroughPriceScale	isolated
SeeDetails M7	MoreInfo, OtherViews, Handles, TableChairs	isolated
Navigation M8	SeeImageDetails, GoPrevious, GoNext	isolated

Table 4-3 Resultant menu structure for MRE

4.3.3 Proposing a taxonomy of menu objects

The general features that menu in UI are shown in table 4-2 this is a compilation of all the features that a menu could have.

	Features	Possible values
Menu	Use of space	Full screen, localized
	Area	Fixed, variable
	Dimensions	1D, 2D, 3D
	Representation	Implicit, explicit
	Selection method	Key combination, Mouse, digital device, voice recognition and haptic devices
	Influence area	Full screen, localized
	Persistence	Application life, event life, mixed
	Presentation	Locked, flying, user preference
	Exploration	Selection, continuous, accelerators
	Status	Compacted, full or partially opened (for instance more used functions are available)
	Reinforcer	Menu path, text, tool tip
	Roll over method	Colour, animation, text accentuation, none
	Disposition	Horizontal, vertical, oblique, circular, polygonal, mixed
	Modality types	Textual, iconic, vocal, bimodal
	Level inflexion by lower levels	Full, summarized, title, removed
	Orientation	Left to right, centre, top-down
Submenu	Level inflexion by upper levels	Expandable, drop-down, cascade, submenu, emergent
Menu item	Type	Command execution, dialogue window, submenu, toggle item, radio item
	Roll over method	Colour, animation, text accentuation, none

Table 4-4 Taxonomy of Menu elements

4.3.3.a Use of space

This feature means the space that is used by the menu. The options are: full screen or a region of the available space (a special case is full Web browser space that we are going to discuss later).

4.3.3.b Area

The menu can have an area (of course with 1D we would have a distance and a volume in 3D) and this feature can be fixed or variable (this capability can be modified by the user or for application adaptability).

4.3.3.c Dimensions

The dimension of a menu is related with the type of UI for instance in a GUI we can have 2D or 3D menus meanwhile in a Character User Interface we have only the command interpreter and one dimension.

4.3.3.d Representation

We have two possible options for the representation: the menu could be explicit that is always available or implicit that is after some conditions is presented to the user. Then we could see this feature as a subcategory of the persistence.

4.3.3.e Selection method

Here we count the possible ways to interact with the elements of the menu: Key combination, Mouse, digital device, voice recognition and haptic devices.

4.3.3.f Influence area

The menu could exist in a specific area of the available space of the application but the result could be reflected in other section that we call influence area. For instance, typical webapps include a top menu which is living in a frame while the result of the action of interacting with the menu is executed in other frame.

4.3.3.g Persistence

This feature is about the life span of the menu. Some menus have the same life cycle than the application and others are available under demand as result of an event trigger and some menus could me a combination. For instance, we could configure our system to always bring at start some menus that normally are only active by demand.

4.3.3.h Presentation

This feature contemplates the position of the menu and their possible combinations: some menu are in a fixed position all the time (Locked), others are

presented near an event, e.g., a selection that trigger a flying menu (a non specific position), or in other cases the user can modify their position according to her/his preferences.

4.3.3.i Exploration

The way of navigate through the levels of the menu hierarchy. The first option is by Selection, that is indicate that you want to explore a section and you have to click (in the case of using a mouse) and after that the next section –if any- is shown. Meanwhile the continuous option implies that moving over the elements expand the inner levels, and finally the use of accelerators implies the use of key combinations that expand the menu to the desired level.

4.3.3.j Status

The expression of menu depending on the design settings could be presented to the user as compacted; full or partially opened (for instance more used functions are available).

4.3.3.k Reinforcer

The interaction with the Menu could be frightening for the user so the menu has to include information to reinforce the selection as the right path. First we could have a path to show the user what was the exploration path over the menu, a brief explanatory text, and finally a minimal text over the menu item in a flying box: a tool tip.

4.3.3.l Rollover method

This is the feedback of the menu in respond to the user interaction: Colour, animation, text accentuation, none.

4.3.3.m Disposition

The position of the menu in the overall structure of the application is very important and could be disposed in one of these general categories: Horizontal, vertical, oblique, circular, polygonal, mixed.

4.3.3.n Modality types

This feature deals with the interaction modality of the menu. The possible options are: Textual, iconic, vocal, bimodal.

4.3.3.o Level Inflexion by lower levels

While the menu is explored and for the sake of the application space the higher levels could be affected or not and change their status to: Full, summarized, title, removed. The idea is to leave the user with the better presentation of the menu

without a saturation of levels. Because according to [Zaph01] the better combinations are the ones with fewer levels in length than in width.

4.3.3.p Level inflexion by upper levels

The presentation of the submenus as the result of the selection of their parent menus are: Expandable, drop-down, cascade, submenu, emergent.

4.3.3.q Type

The menu item could be of one of these types: Command execution, dialogue window, submenu, toggle item, radio item. The sub menu in figure 4-8 is not presented because the treatment of a menu or a submenu in terms of tasks to fulfil is the same.

4.3.4 Menus on RIA applications

The menus have a capital importance in the RIAs because the menu could be seen as a list of shortcuts to commands and operations within an application but in RIA They are also the links between Web pages. There is an intrinsic sense of navigation in the menu object. The menu objects in webapps are a reflex of the navigation structure of the site and in the next table (4-3) we present the extensions to the taxonomy that has been presented in the previous section.

	Features	Possible values
RIA Menu	Auto update	Through Push technology
	Broadband saving version	Text, low resolution
	Loading previous	None, partial, loading icon,
	Leap	None, animated, multimedia

Table 4-5 Taxonomy of Menu elements extended to RIA

4.3.4.a Auto update

The Elements of a RIA menu can be updated using push technology. For instance in the case of an offer (in an e-commerce application) the menu could be updated to present some new category of products.

4.3.4.b Broadband saving version

The amount of information sends it in the Web world through the wire is always an issue, so the menus could have simpler versions to present information to the user.

4.3.4.c Loading previous

The menus have to inform the user that some of the information is still unavailable and present a progression icon or part of the information while the rest is retrieved.

4.3.4.d Leap

The interaction with webapps is very scarce; the user is used to jump from web place to another in seconds so in order to provide a one button option to reach some section or fulfill some task the RIA menu include hot spots leaps to specific sections. For instance, in some menus the site creators show you suggestions and they send you to some specific section⁸ that does not require explore the menu hierarchy (in a sense is an accelerator but more concise and punctual because the accelerators also could imply more than one letter besides the control/alt key to dig into the lower levels of the menu).

4.3.5 Implantation of the menu in the AUI

The heuristic for implant the menu derived of the analysis of the CTT tree is again depending of the designer preferences. Because from tables 4-4 we have the available combinations but here we propose a simple method: The inclusion of a umbrella AC that would cover all the ACs that has already created and the inclusion of an extra AC (the menu) that will include AIC with navigation facets to each element of table 4-3. Then, all the ACs in an XSLT iterative process we have to include a pair of AIC one with a navigation facet to return to the menu and other with an input facet to prepare the AIC to the inclusion of an accelerator in the CUI step.

⁸ An example of this menu option is the gold box of Amazon.com

Chapter 5 Concrete User Interface Representation

In this chapter we are going to describe the step three (see figure 5-1) first we discuss about the selection of the platform, then process to transform our AUI into its CUI representation using XSLT transformations and finally the definition of the behavior and presentation of our UI.

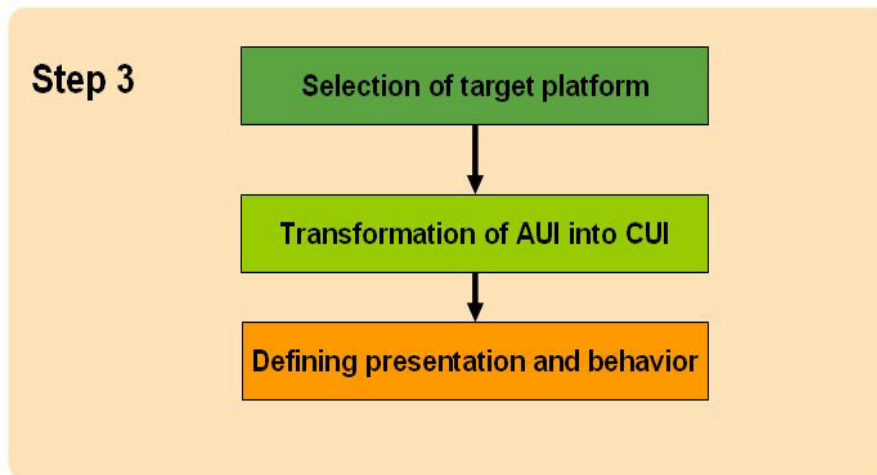


Figure 5-1: Sub steps of the transformation from AUI to CUI

5.1 Selection of target platform

The goal is still behind the hill, our UI need to pass to the next CAMELEON level to look more like a typical UI with widgets more close to the ones of the final platform. Here, we have to define the look and feel, the appearance and the behavior. The CUI representation is dependent of the modality [Limb04] and in most of the cases RIA applications have a graphic modality i.e., a combination of graphic input and output (see figure 5-1). Nevertheless, the description of the elements is independent from any existent toolkit or API the reason of this is keeping the UI as general as possible to allow a simpler translation to the last step (next chapter). In other words, the proximity to any real API could compromise the translation for other platforms and languages. Our work is based in the CUI

model from UsiXML⁹ that include a collection of concrete Interaction objects. That is a general toolkit not attached to a single platform or language besides a relationship model.

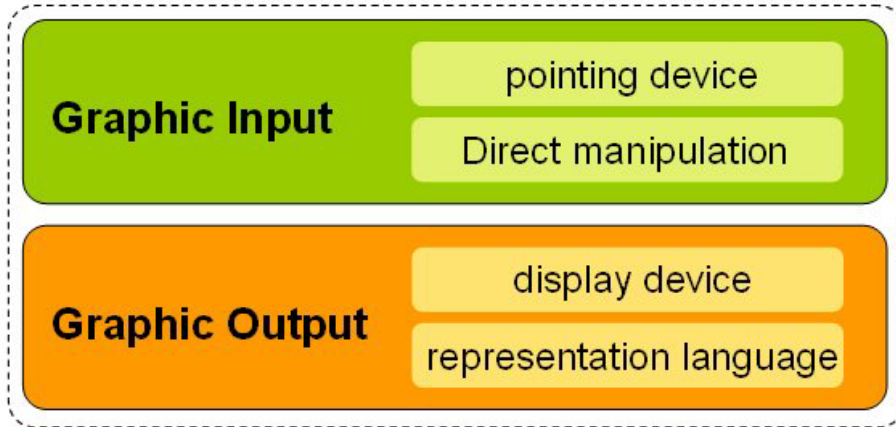


Figure 5-2: Elements of graphic modality

5.2 Transformation of AUI in CUI

5.2.1 Selection of Concrete Interface Components

The selection of Concrete Interaction Objects (CIOs) is a process that includes some XSLT templates. In the table 5-1, we define a possible set of CIOs which can be obtained by the transformation of the AIC taking into account: facet types, data types and cardinalities among others (from the AUI depicted in fig. 4-12).

AIC	Facet Specification	Relevant Information	Possible CIC
“ShowStandardHandles”	▪ Output	▪ Feedback	Images
“ShowHandlesList”	▪ Output	▪ Data type ▪ Domain ▪ Characteristics	Images
“Pick Section”	▪ Input	▪ Data type ▪ Domain ▪ Characteristics ▪ Selection Value	An ImageZone (with a link)
“ShowHandleSection”	▪ Output	▪ Data type ▪ Domain ▪ Characteristics	A window

Table 5-1 Fragment of AUI (some AIOs) from MRE and their equivalent CIOs

⁹ Available from [Http://www.usixml.org](http://www.usixml.org)

5.2.2 Defining CICs spatial position

The process of selecting the final position of CICs depends on the physical constraints of the final size of the main container (Here, a web browser). The Heuristic defined could be as simple as: put all the objects, one by one, following tasks temporal relationships and taking the centre of the window as the axis of all the CICs or it could involve more complex heuristics based on ergonomical criteria.

5.2.3 Defining Navigation

Navigation leads the user while she or he is using the application. In short, defines the visibility of components depending on tasks temporal relationships, the navigation isn't defined in an explicit form in our study case, it's hidden in all the triggers used. In some other application a back and forward buttons/links could produce a more obvious way of navigation schema.

5.2.4 Resulting CUI UsiXML specification

The UsiXML representation of a CUI is shown in Figure 5-3 it's from [Mart06]. The UsiXML document is much more extended and complex than the fragment that is presented which included the resources and events associated to the widgets besides information from the other levels. Some of the saved information in this file would be very useful for retargeting tasks. This file was created using the GrafiXML [Limb04].

```
<?xml version="1.0" encoding="UTF-8"?>
<uiModel xmlns="http://www.usixml.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.usixml.org/
  http://www.usixml.org/spec/UsiXML-ui_model.xsd"
  id="FruitStore_31" name="FruitStore"
  creationDate="2006-03-31T11:03:50.109-06:00" schemaVersion="1.6.3" xsi:type="uiModel">
  <head>
    <version modifDate="2006-03-31T11:03:50.109-06:00">1</version>
    <authorName>Javier Martinez</authorName>
    <comment>Generated by GrafiXML 1.1.999 build id : 200602081036</comment>
  </head>
  <window id="window_component_0" name="window_component_0"
    width="400" height="350">
    <box id="box_1" name="box_1" type="vertical">
      <imageComponent id="image_component_2"
        name="image_component_2"
        tooltip="/uiModel/resourceModel/cioRef[@ciId='image_component_2']/resource/@tooltip"
        defaultTooltip="Apples (£5)"
        content="/uiModel/resourceModel/cioRef[@ciId='image_component_2']/resource/@content"
```

```
        defaultContent="/resources/00/" isVisible="true"
        isEnabled="true" textColor="#000000"/>
<imageComponent id="image_component_3"
    name="image_component_3"
    tooltip="/uiModel/resourceModel/cioRef[@cioid='image_component_3']/resource/@tooltip"
    defaultTooltip="Bananas (£7)"
    content="/uiModel/resourceModel/cioRef[@cioid='image_component_3']/resource/@content"
    defaultContent="/resources/00/" isVisible="true"
    isEnabled="true" textColor="#000000"/>
<imageComponent id="image_component_4"
    name="image_component_4"
    tooltip="/uiModel/resourceModel/cioRef[@cioid='image_component_4']/resource/@tooltip"
    defaultTooltip="Grapes (£9)"
    content="/uiModel/resourceModel/cioRef[@cioid='image_component_4']/resource/@content"
    defaultContent="/resources/00/" isVisible="true"
    isEnabled="true" textColor="#000000"/>
<imageComponent id="image_component_5"
    name="image_component_5"
    tooltip="/uiModel/resourceModel/cioRef[@cioid='image_component_5']/resource/@tooltip"
    defaultTooltip="Ready to buy? double click in basket"
    content="/uiModel/resourceModel/cioRef[@cioid='image_component_5']/resource/@content"
    defaultContent="/resources/00/" isVisible="true"
    isEnabled="true" textColor="#000000"/>
</box>
</window>
```

Figure 5-3: Fragment of a CUI UsiXML file

5.3 Refining presentation and behavior for CUI

5.3.1 Behavior a basic introduction

Before going any further we have to define what behaviour in the context of this work is: Behaviour refers to the actions or reactions of an object or organism, usually in relation to the environment [wiki07].

We can observe three general models of behaviour in software engineering: Control flow, Data flow and State machines [Bock99].

5.3.1.a Control flow

This model describes the sequence of steps and simply assumes that next step is going to be processed after the current step is complete. There is not a vigilance of the inputs i.e. it's not required to start a new step to observe the state of the input as it's assumed that between steps the needed information is gathered. For instance, a store is open according to a schedule without looking to the presence or not of customers. Control flow is used extensively in the imperative languages e.g. C, Basic, FORTRAN, among many others. The specific tool was the flow chart [Kern88], [Mart98] and [Kell98]. A drawback is the difficulty of modelling concurrent steps.

5.3.1.b Data flow

This model works with the following idea: each step provides the inputs for the next step (its own outputs). For instance, this approach is more suitable for Object Oriented languages as Java. Because, objects are waiting for the needed

input for execution and also we could deal with several threads of execution since we could have multiple object instances. One of the most used tools for this kind of model is Petri Nets. [Grah96], [Film84].

5.3.1.c State Machines

This model uses/sees the input as a collection of events that would trig within the environment of the application. The input of each step is processed as part of the step. For instance, the state charts of UML. [Rational Software, et al, UML Semantics, version 1.1, Rational Software Corporation, Santa Clara, CA, September 1997, chapter 11. Harel, D., and M. Politi, Modeling Reactive Systems With Statecharts: The Statemate Approach, McGraw Hill, 1998.]

We summarize the features of the three models in table 5-2,

	Control Flow	Data Flow	State Machine
Input is determined	At start	Before Start	At Start
Start conditions	Internal	Internal	External
Basic tool	Flow chart	Petri Net	State charts
Better for	Strong coupled between input data and step order	Weak coupled between input data and step order	Input data provided by step events

Table 5-2 Dimensions of Behaviour Modelling (modified from [Bock99])

5.3.2 Behavior of the CUI representation

So far, we are trying to figure out how to create a model of behaviour based in the combination of these three models since the distributed nature of the RIAs could be badly represented if we select only one medium. As we have discussed in the previous chapter a nice candidate is the SMIL language that is also used as starting point in [Lina07].

Chapter 6 Generation of Final User Interface

In this step are produced operational UIs that are executed, compiled or interpreted on a particular platform (e.g., .NET, LZX, SWF and GWT among others¹⁰). The code that is obtained is translated again with XSLT stylesheets and finally we have code in the target language that could be treated by the interpreters, compilers, generators or converters of platform.

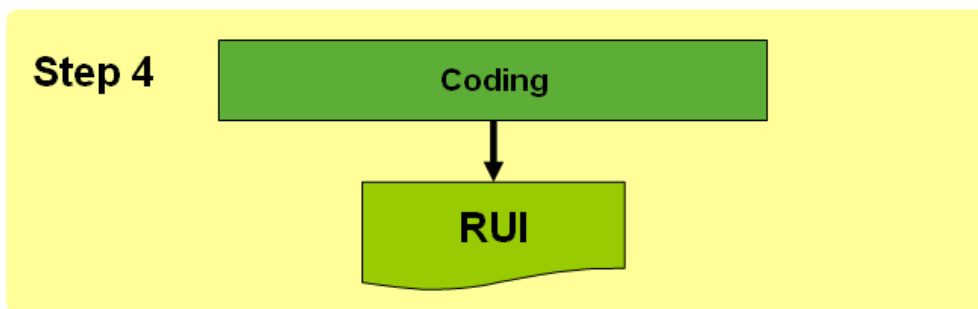


Figure 6-1: The final step: delivering a RUI

6.1 Processing the CUI to generate the Final User Interface

In this phase we transform CUI specifications to native widgets sets present in popular graphical toolkits (GWT or XAML among others) thanks to XSLT transformations the CUI objects are translated to the platform/language specific elements. An important feature of the method is its capability to redirect the target FUI e.g., in the figure 6-2, as example we present three target transformations: GWT [GWT07], Open Laszlo [open06] and XAML [XAML06] to endeavour these transformations is needed to generate adequate XSLT templates to translate CIOs described in UsiXML to the target language. This section describes the way XSL transformations are applied to generate a hypothetical (minimal) XAML output.

¹⁰ The creation of generators and converters of UsiXML to the Final code has already begun.

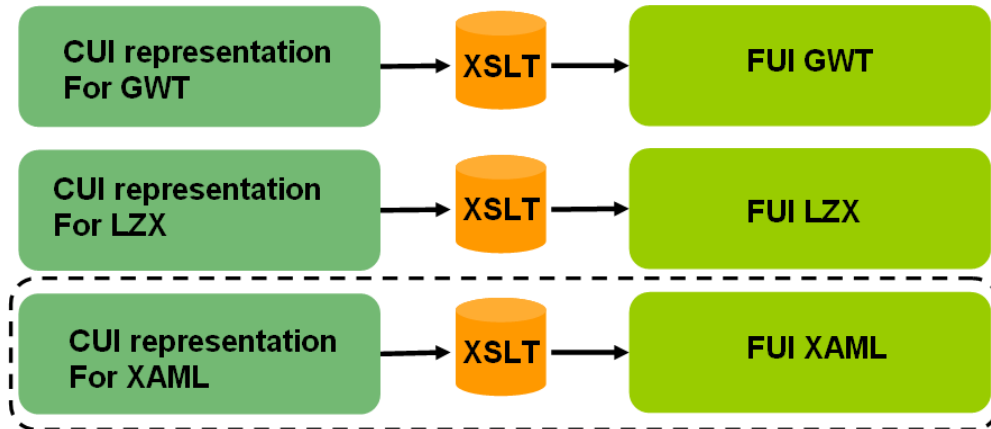


Figure 6-2: The final step: delivering FUI code for the interpreters or compilers.

This is an excerpt of the final version of the XSLT template rules (Fig. 6-3) we just add here some rules to make clear the example since the XML source document is very different to the final document, some of the code is restricted to default values.

The resulting XAML UI definition is shown below (Figure 6-4). The UI definition in UsiXML documents describe the event response (code included in a separated section). This is also the case of XAML that in a separated document, denoted by the “CodeBehind” tag includes this information. Also, the size of the widgets is omitted for the sake of simplicity.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:wf="http://schemas.microsoft.com/2003/xaml/" version="1.0">
  <xsl:output method="xml" omit-xml-declaration="yes" />
  <xsl:template match="*/" />
  <wf:UserControl Name="WebForm1" ClientSize="200, 200" xmlns="http://schemas.microsoft.com/2003/xaml/"
  xmlns:def="Definition" xmlns:wf="wf" def:Class="XamlonApplication8.WebForm1" def:CodeBehind="WebForm1.xaml.cs">
    <xsl:apply-templates select="/cuiModel/window"/>
  </wf:UserControl>
</xsl:template>
<xsl:template match="window">
  <wf:UserControl.Controls>
    <xsl:apply-templates select="/cuiModel/window/box/inputText"/>
    <xsl:apply-templates select="/cuiModel/window/box/button"/>
    <xsl:apply-templates select="/cuiModel/window/box/outputText"/>
    <xsl:apply-templates select="/cuiModel/window/box/slider"/>
  </wf:UserControl.Controls>
</xsl:template>
<xsl:template match="inputText">
```

```
<wf:TextBox Text="{@defaultContent}" TabIndex="1" Name="{@name}"/>
</xsl:template>
<xsl:template match="button">
  <wf:Button Text="{@defaultContent}" TabIndex="1" Name="{@name}"/>
</xsl:template>
<xsl:template match="outputText">
  <wf:Label Text="{@defaultContent}" TabIndex="1" Name="{@name}"/>
</xsl:template>
<xsl:template match="slider">
  <wf:TrackBar Text="{@defaultContent}" TabIndex="1" Name="{@name}"/>
</xsl:template>
</xsl:stylesheet>
```

Figure 6-3: XSL transformation document

Then the code that was shown in figure 5-3 would deliver something like the code presented in figure 6-4. Now we have the equivalent to the CUI windows previously defined but settle down to Microsoft technology.

```
<wf:UserControl xmlns:wf="wf" xmlns="http://schemas.microsoft.com/2003/xaml/" xmlns:def="Definition" Name="WebForm1"
ClientSize="200, 200" def:Class="XamlonApplication8.WebForm1" def:CodeBehind="WebForm1.xaml.cs">
  <wf:UserControl.Controls xmlns:wf="http://schemas.microsoft.com/2003/xaml/"><wf:TextBox Text="" TabIndex="1"
Name="input_text_component_9"/><wf:TextBox Text="" TabIndex="1" Name="input_text_component_11"/><wf:TextBox
Text="0.00" TabIndex="1" Name="input_text_component_13"/><wf:Button Text="Submit order" TabIndex="1"
Name="button_component_14"/><wf:Label Text="Name" TabIndex="1" Name="output_text_component_8"/><wf:Label
Text="Address" TabIndex="1" Name="output_text_component_10"/><wf:Label Text="Total to Pay:" TabIndex="1"
Name="output_text_component_12"/>
  </wf:UserControl.Controls>
</wf:UserControl>
```

Figure 6-4: XAML resultant file

Chapter 7 Conclusion

The purpose of this work was to establish the master plan of our method to develop RUIs (see complete method, figure 7-1). All the steps have been presented in the current document. These steps are the beginning of a variation/expansion of the UsiXML family of tools and models in order to target RIAs is an ongoing work.

In this dissertation we proposed a novel approach to model RUIs which includes the complete software development life cycle. The proposed method organizes the development life cycle for RUIs from the conceptual to the final implementation stages using as guide the user requirements instead of being focus in the content, furthermore, our method is Model Driven Engineering compliant since we are concern with the separation of different aspects of the problem within abstract models that could be, progressively expanded to concrete models. That is RIAUI development cycle is progressively refined from the Computing Independent Models (CIM) as defined by OMG [OMG07] to the concrete models: Platform Specific models.

7.1 Summary of contributions

The list of contributions is listed below:

- The contributions are expanded in a series of exploratory papers in which we began to understand more the model of the RIAs:
 - [Mart06], where present the first attempt to tackle the problem
 - [Muño06] where we propose a taxonomy of RIAs
 - [Mart06a] a study case based on XAML UIs.
- We have integrated here a proposal of extensions to the task types and menu features relevant to RIAs (see chapter 3 and 4).
- A proposal of a method for the generation of the container structure
- And Finally, The generation of menus based on task trees

The goal of having a robust method to deliver RUIs is still far away but we have a good starting point.

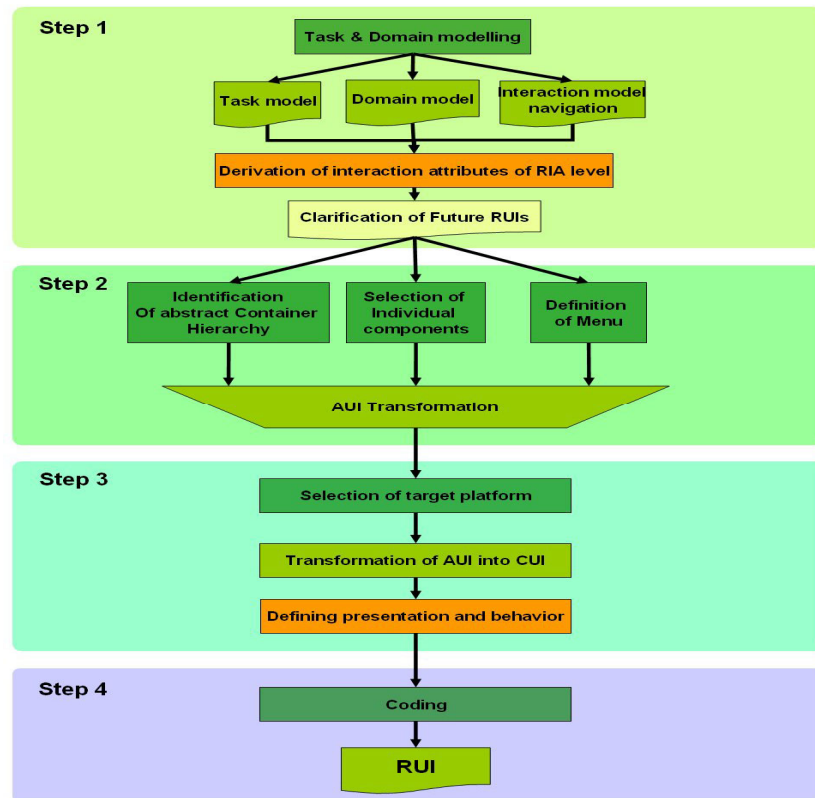


Figure 7-1: a development method for User Interfaces of RIAs

7.2 Brief discussion of future work

There are some activities that we are doing right now because are pending tasks for instance the process of making grow our repository of the UI widgets used in RIAs (for complete the XSLT templates). Also we want to pursuit the integration of the RIA frameworks GWT into the GrafXML tool as target language also the integration of some elements to make simpler the translation between models, specifically the collecting of patterns to reduce the process of conversion.

And finally, the web is a distributed environment, we have to profit of that and integrate to the solution the power of the cooperative systems i.e., web agents.

References

A

[Abra99]

Abrams, M., Phanouriou, C., Batongbacal, A. L., M. Williams, S. and Shuster, J.E.. Uiml: an appliance-independent xml user interface language. In WWW '99: Proceeding of the eighth international conference on World Wide Web, pages 1695–1708, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[Adler 95]

Adler, R. M. *Distributed Coordination Models for Client/Sever Computing*. Computer 28, 4 (April 1995): 14-22.

B

[Bles90]

Bleser, T. W. & Sibert, J., "Toto: a tool for selecting interaction techniques". In: Proceedings of user interface software and technology (Snowbird, Utah, Oct.3-5,1990) . New York: ACM, 1990, pp. 135-142.

[Boda94a]

Bodart, F., Hennebert, A., Leheureux, J., and Vanderdonckt, J. 1994. *Towards a dynamic strategy for computer-aided visual placement*. In Proceedings of the Workshop on Advanced Visual interfaces (Bari, Italy, June 01 - 04, 1994). M. F. Costabile, T. Catarci, S. Levialdi, and G. Santucci, Eds. AVI '94. ACM Press, New York, NY, 78-87.

[Boda94b]

Bodart, F. and Vanderdonckt, J. 1994. *Guide ergonomique de la présentation des applications hautement interactives*. Namur : Presses Universitaires de Namur.

[Bock99]

Bock, C., "Three Kinds of Behaviour Model," Journal of Object-Oriented Programming, 12:4, July/August 1999.

[Booc05]

Booch, G., Rumbaugh, J., and Jacobson, I., 2005. *Unified Modeling Language User Guide*, 2nd Edition. Addison-Wesley. USA.

[Boui05]

References

Bouillon, L., Limbourg, Q., Vanderdonckt, J., Michotte, B., 2005. *Reverse Engineering of Web Pages based on Derivations and Transformations*, Proc. of 3rd Latin American Web Congress LA-Web'2005 (Buenos Aires, October 31-November 2, 2005), IEEE Computer Society Press, Los Alamitos, pp. 3-13

[Boye06]

Boyer, J.M., Landwehr, D., Merrick, R., Raman, T. V., Dubinko, M. and Klotz, L. L., *XForms 1.0 (Second Edition)*, W3C Recommendation. World Wide Web Consortium, March 2006.

[Bozz06]

Bozzon, A., Comai, S., Fraternali, P., Toffetti Carughi, G. (2006), *Capturing RLA concepts in a web modeling language*, Proc. of the 15th International Conference on World Wide Web WWW'2006 (Edinburgh, May 23-26, 2006), pp. 907-908.

[Bres97]

Breslow, L. A. and Aha, D. W., 1997. *Simplifying decision trees: a survey*. Knowledge Engineering Review, 12(1):1-40.

C

[Cach02]

Cachero, C. and Gómez, J., 2002. *Advanced conceptual modeling of Web applications: Embedding operation interfaces in navigation design*. In the 21st International Conference on Conceptual Modelling (JISBD). El Escorial, Madrid, Spain.

[Calh84]

Calhoun, G. C.; Arbak, C. L. & Boff, K. R. "Eye-controlled switching for crew station design". In: Proceedings of the Human Factors Society 28th annual meeting, Santa Monica (CA): Human Factors Society, 1984, pp. 258-262.

[Calv03]

G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. "A Unifying Reference Framework for Multi-Target User Interfaces". *Interacting with Computers*, 15(3): 289-308, 2003.

[Canf90]

Canfield Smith, D., Irby, C., Kimball, R., Berplank, B. and Harslem, E. *Designing the star user interface*. Pages 237-259, 1990.

[Casn91]

Casner, S. M. 1991. *Task-analytic approach to the automated design of graphic presentations*. ACM Trans. Graph. 10, 2 (Apr. 1991), 111-151.

References

[Ceri01]

Ceri, S., Fraternali, P., Matera, M., and Maurino, A.(2001). *Designing Multi-Role, Collaborative Web Sites with WebML: a Conference Management System Case Study*. IWWOST'01, Valencia, Spain, June 2001.

[Chen05]

Chen, Y., Xie, X., Ma, W.-Y., and Zhang, H.-J. Adapting Web Pages for Small-Screen Devices. *IEEE Internet Computing*, 09(1) (2005), 50-56.

[Chu04]

Chu, H., Song, H., Wong, C., Kurakake, S., and Katagiri, M. Roam, a seamless application framework. *Journal of System and Software* 69(3) (2004), 209-226.

[Cona02]

CONALLEN, J. 2002. *Building Web Applications with UML*. Addison-Wesley

[Cran05]

Crane, D., Pascarello, E., James, D. (2005), *Ajax in Action*, Manning Publications, USA.

[Cout87]

Coutaz, J., (1987). "PAC: an Implementation Model for Dialog Design". H.-J. Bullinger, B. Shackel (ed.) *Proceedings of the Interact'87 conference, September 1-4, 1987, Stuttgart, Germany*: pp. 431-436, North-Holland.

D

[Detr03]

De Troyer, O., Casteleyn, S.: *Modeling Complex Processes for Web Applications using WSDM*, In Proceedings of the Third International Workshop on Web-Oriented Software Technologies (held in conjunction with ICWE2003), Eds. Daniel Schwabe, Oscar Pastor, Gustavo Rossi, Luis Olsina, Oviedo, Spain (2003).

[Diaz97]

Díaz Pérez P., Catenazzi N., Aedo Cuevas, I. (1997), *De la Multimedia a la Hipermedia*, Ed. Alfaomega, Spain.

[Dyck90]

Dychhoff, H. *A typology of cutting and parking problems*. *European Journal of Operational Research*, 44:145–159, 1990.

References

E

[Ecke95]

Eckerson, W. "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications." *Open Information Systems* 10, 1 (January 1995): 3(20).

F

[Ferr03]

Ferraiolo, J., Jun, F., Jackson, D., Scalable Vector Graphics (SVG) 1.1 Specification, W3C Recommendation. World Wide Web Consortium, January 2003.

[Flas06]

FLASH (2006). Retrieved March 26, 2007, from <http://www.adobe.com/products/flash/>

[Flex06]

FLEX (2006). Retrieved March 26, 2007, from <http://www.macromedia.com/software/flex/>

[Flor06]

Florins, M., Simarro, F. M., Vanderdonckt, J., and Michotte, B. 2006. *Splitting rules for graceful degradation of user interfaces*. In Proceedings of the Working Conference on Advanced Visual interfaces (Venezia, Italy, May 23 - 26, 2006). AVI '06. ACM Press, New York, NY, 59-66.

[Fiel00]

Fielding, R. T., *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis, UC Irvine, 2000.

[Film84]

Filman, R. E. and D. P. Friedman, *Coordinated Computing: Tools and Techniques for Distributed Software*, McGraw Hill, New York, NY, 1984, chapter 9

[Fole84]

Foley, V. W., Chan, V. "The human factors of computer graphics interaction techniques", In *IEEE Computer Graphics & Applications*, (4), pp. 13-48 (1984).

[Fran98]

Franklin, M. and Zdonik, S. (1998), *Data in your Face: Push Technology in Perspective*, Proc. of ACM SIGMOD International Conference on Management of Data SIGMOD'98 (Seattle, June 2-4, 1998), ACM Press, New York, 1998, pp. 516-519.

G

References

[Garr05]

Garrett, J., *Ajax: A new approach to web applications*. Technical report, Adaptive Path, 2005.

[Gill03]

Giller, V., Melcher, R., Schrammel, J., Sefelin, R., and Tscheligi, M. *Usability Evaluations for Multi-device Application Development - Three Example Studies*. In Proceedings of Mobile HCI'03 (Udine, Italy, Sept. 8-11, 2003).

[Göbe01]

Göbel, S., Buchholz, S., Ziegert, T., and Schill, A. Device Independent Representation of Web-based Dialogs and Contents. In Proceedings of the IEEE YUFORIC '01

(Valencia, Spain, Nov. 2001).

[Gome02]

Gomez, J. and Cachero, C., *OO-H: Extending UML to Model Web Interfaces*. *Information Modeling for Internet Applications*. Idea Group Publishing, 2002. Available at <http://www.dlsi.ua.es/~ccachero/papers/igp.pdf>

[Gonz07]

Gonzalez Calleros, J.M., (2007), *Model-based development of Three-dimensional user Interfaces*, EpreuveDeConfirmation . Université Catholique de Louvain, Belgium.

[Grah96]

Graham, P., *ANSI Common Lisp*, Prentice Hall, Englewood Cliffs, NJ, 1996.

[Gree88]

Greenstein, J. S. & Arnaut, L. Y. "Input devices". In: M. Helander, (Ed.), *Handbook of Human-Computer Interaction*, Amsterdam: North-Holland, 1988, pp. 495-519.

[GWT07]

Google Web Toolkit. Retrieved August 17, 2007, from <http://code.google.com/webtoolkit/overview.html>

H

[Holl99]

Holland, S. and Oppenheim, D., *Direct combination*. In CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 262–269. ACM Press, 1999.

[Holm06]

Holmes, J., *Struts: The Complete Reference*, 2nd Edition, McGraw Hill/Osborne, 2006.

References

[Honk07]

Honkala, M. (2007). *Web User Interaction a Declarative Approach Based on XForms*. PhD Thesis, Helsinki University of Technology.

[Hyat01]

Hyatt, D., XML user interface language (XUL) 1.0. Mozilla.org, 2001.

[Hyat00]

Dave Hyatt. XBL - extensible binding language 1.0. Netscape, 2000.

J

[Jaco04]

Jacobs, I. (2004). *Architecture of the World Wide Web, Volume One*. Retrieved March 26, 2007, from <http://www.w3.org/TR/webarch/#def-representation>.

[Java07]

Java Programming language (2007). *Java Standard Edition, APIs and documentation*. Retrieved March 26, 2007, from <http://java.sun.com/>

K

[Kay03]

Kay, M. XSL Transformations (XSLT), Version 2.0. Technical report, W3C, 2003. <http://www.w3.org/TR/xslt20/>.

[Kazo07]

Kazoun, C., Lott, J., *Programming Flex 2: The comprehensive guide to creating rich media applications with Adobe Flex (Programming)*. O'Reilly, 2007.

[Kell98]

Keller, G. and T. Teufel, *SAP R/3 Process Oriented Implementation: Iterative Process Prototyping*, Addison-Wesley, MA, 1998

[Keps04]

Kepser, Stephan. *A Simple Proof for the Turing-Completeness of XSLT and XQuery*. In *Proceedings of Extreme Markup Languages 2004*.

[Kern88]

Kernighan, B. W. and D. M. Ritchie, *The C Programming Language*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[Klei04]

References

Klein, R., Six, H., Wegner, L., (2004). *Computer Science in Perspective: Essays Dedicated to Thomas Ottmann*. Lecture Notes in Computer Science, vol. 2598, Springer-Verlag, Berlin, pp. 100-101.

[Koch02]

Koch, N. and Kraus, A. 2002. *The expressive power of UML-based engineering*. In Second International Workshop on Web Oriented Software Technology (CYTED). 105–119.

L

[Leno84]

Lenorovitz, D.R.; Phillips, M.D.; Ardrey, R.S. & Kloster, G.V. “*A taxonomic approach to characterizing human-computer interaction*”. In: G. Salvendy (Ed.), *Human-Computer Interaction*. Amsterdam: Elsevier Science Publishers, 1984, pp.111-116.

[Lewi04]

Lewis Ship, H., *Tapestry in Action*. Manning Publications. 2004.

[Limb03a]

Limbourg, Q. and Vanderdonckt, J., *Comparing Task Models for User Interface Design*, in Diaper, D., Stanton, N. (Eds.), *The Handbook of Task Analysis for Human-Computer Interaction*, Lawrence Erlbaum Associates, Mahwah, pp. 135-154.

[Lim05]

Lim, A. and Zhang, X. 2005. *The container loading problem*. In Proceedings of the 2005 ACM Symposium on Applied Computing (Santa Fe, New Mexico, March 13 - 17, 2005). L. M. Liebrock, Ed. SAC '05. ACM Press, New York.

[Limb04]

Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and Lopez, V. *UsiXML: a Language Supporting Multi-Path Development of User Interfaces*, Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004 (Hamburg, July 11-13, 2004). Lecture Notes in Computer Science, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 207-228.

[Lina07]

Linaje, M., Preciado, J.C., Sánchez-Figueroa, F., 2007. *A Method for Model Based Design of Rich Internet Application Interactive User Interfaces*. In The Seventh International Conference on Web Engineering (ICWE'07).

References

M

[Mack86]

Mackinlay, J. 1986. *Automating the design of graphical presentations of relational information*. ACM Trans. Graph. 5, 2 (Apr. 1986), 110-141.

[Mand02]

Mandyam, S., Vedati, K., Kuo, C. and Wang, W., *User Interface Adaptations: Indispensable for Single Authoring*. In Workshop on Device Independent Authoring Techniques (St. Leon-Rot, 15-26 September 2002).

[Mart06]

Martínez-Ruiz, F.J., Muñoz Arteaga, J., Vanderdonckt, J., González-Calleros, J.M. (2006), A first draft of a Model-driven Method for Designing Graphical User Interfaces of Rich Internet Applications, Proc. of 4th Latin American Web Congress LA-Web'2006 (Puebla, October 25-27, 2006), IEEE Computer Society Press, 2006.

[Mart06a]

Martínez-Ruiz, J., Muñoz Arteaga, J., Vanderdonckt, J., Transformation of XAML schema for RIA using XSLT & UsiXML, Proc. of XIX Congreso Nacional y V Congreso Internacional de Informática y Computación de la ANIEI, Avances en Tecnologías de la Información CNCIC'2006 (Tuxtla Gutiérrez, 25-27 October 2006), 2006

[Mart98]

Martin, J., and J. J. Odell, *Object-Oriented Methods: A Foundation (UML edition)*, Prentice Hall, Englewood Cliffs, NJ, 1998.

[Mahe06]

Mahemoff, M., *Ajax Design Patterns*. O'Reilly & Associates, Inc., USA, 2006

[Moba07]

Mobalpa (2007). Mobalpa : cuisines, salles de bains et rangement (web site), Retrieved June 26, 2007, from [http:// http://www.mobalpa.fr/](http://www.mobalpa.fr/)

[Mont06]

Montero, F., López-Jaquero, V., *Fast HI-FI prototyping by using IdealXML*, Technical report DIAB-06-03-1, Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, Albacete, 24 March 2006.

References

[Mozi07]

Mozilla Foundation (2007). *XML User Interface Language (XUL) 1.0*, Retrieved March 26, 2007, from <http://www.mozilla.org/projects/xul/xul.html>.

[Muel04]

Mueller, W., Schaefer, R., Bluel, S., "Interactive multimodal user interfaces for mobile devices" Paderborn University, January 05 - 08, 2004. Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) IEEE 2004

[Muño06]

Muñoz Arteaga, J., Martínez-Ruiz, Francisco J., Vanderdonckt, J., Ochoa, A., *Categorization of Rich Internet Applications based on Similitude Criteria*. XI Simpósio de Informática e VI Mostra de Software Acadêmico da PUCRS (SIMS 2006) - Brasil.

O

[Olso90]

Olson, R. and Olson, G. M., "The growth of cognitive modeling in human-computer interaction since GOMS," *Human-Computer Interaction*, 5 (1990), pp. 221-265

[OMG07]

OMG (2007). *The Object Management Group*. Retrieved July 2nd 2007 from <http://www.omg.org>.

[Open06]

Open Laszlo (2006). *Open Laszlo documentation*, Retrieved July 2, 2006, from <http://www.openlaszlo.org/documentation>

P

[Paye02]

Payet, D. (2002), *LYPO: vers une perception applicative du WEB*, Proceedings of the 14th French-speaking Conference on Human-Computer Interaction IHM'2002 (Poitiers, November 26-29, 2002), Association Francophone de l'Interaction Homme-Machine, 2002, pp. 231-234.

[Pisi95]

Pisinger, D., (1995). *Algorithms for Knapsack Problems*, Ph.D. thesis, DIKU, University of Copenhagen, Report 95/1.

[Pate99]

Paternò F. *Model-Based Design and Evaluation of Interactive Applications*. Springer-Verlag, London,

References

UK, 1999.

[Pemb02]

Pemberton, S., XHTML 1.0: The extensible hypertext markup language (2nd edition).W3C Recommendation, August 2002.

[Prec05]

Preciado, J.C., Linaje, M., Sanchez, F., Comai, S. (2005), *Necessity of methodologies to model Rich Internet Applications*, Proc. of 7th IEEE International Symposium on Web Site Evolution WSE'2005, IEEE Computer Society Press, 2005, pp. 7-13.

[Prib02]

Pribeanu, C., Vanderdonckt, J., (2002). *Exploring Design Heuristics for User Interface Derivation from Task and Domain Models*, Chapter 9, in Proceedings of 4th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2002 (Valenciennes, 15-17 mai 2002), Kluwe Academics Pub., Dordrecht, 2002, pp. 103-110.

[Pont04]

Pontico, F., Farenc, C., and Winckler, M., (2004). *Une architecture de dialogue basée sur un modèle pour les applications web*. In Proceedings of the 16th Conference on Association Francophone D'interaction Homme-Machine (Namur, Belgium, August 30 - September 03, 2004). IHM 2004. ACM Press, New York, NY, 251-254.

[Puer02]

Angel Puerta and Jacob Eisenstein. *Ximl: a common representation for interaction data*. In IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces, pages 214–215, New York, NY, USA, 2002. ACM Press.

R

[Ragge99]

Ragget, D., *HTML 4.0.1 specification*. W3C Recommendation, December 1999.

[Ross03]

ROSSI, L., SCHMID, H., AND LYARDET, F. 2003. *Engineering business processes in web applications: Modeling and navigation issues*. In Third International Workshop on Web Oriented Software Technology. Oviedo, Spain, 81–89.

S

[Schm06]

Schmidt, D.C., 2006. *Guest Editor's Introduction: Model-Driven Engineering*. Computer, vol. 39, no. 2, pp. 25-31, Feb., 2006.

References

[Schn83]

Schneiderman, B. *Direct manipulation : a step beyond programming languages*. IEEE Computer, 1983

[Sesh99]

Seshadri, G., *Understanding JavaServer Pages Model 2 architecture. Exploring the MVC design pattern*. JavaWorld.com, 12/29/99.

[Souc03]

Souchon, N. and Vanderdonckt, J. *A review of XML-compliant user interface description languages*. In Proceedings of the 10th International Conference on Design, Specification, and Verification of Interactive Systems, pages 377–391, Madeira, Portugal, June 4-6 2003. Springer-Verlag.

[Spri03]

Priestersbach, A., Ziegert, T., Grassel, G., Wasmund, M., and Dermier, G. *Flexible pagination and layouting for device independent authoring*. In WWW2003 Emerging Applications for Wireless and Mobile access Workshop.

[Stal01]

Stallings, W., *Operating Systems: Internals and Design Principles*. Prentice-Hall Inc., 2001.

U

[USIX07]

UsiXML Consortium. UsiXML, a General Purpose XML Compliant User Interface

Description Language, UsiXML V1.8, 23 February 2007.

Available at <http://www.usixml.org/index.php?view=page&idpage=6>

V

[Vand98]

VANDERDONCKT J., *Une description orientée objet des objets interactifs abstraits utilisés dans les Interfaces Homme-Machine*, FNDP, Namur, 1998.

W

[Weis07]

Weisstein, Eric W. "Bell Number." From MathWorld. A Wolfram Web Resource. Retrieved March 26, 2007, from <http://mathworld.wolfram.com/BellNumber.html>

[Wiki07]

Wikipedia the free encyclopedia available on [<http://en.wikipedia.org/wiki/Behavior>] (Accessed August 17, 2007).

References

[W3C06]

W3C consortium, , *XMLHttpRequest*. 2006, <http://www.w3.org/TR/XMLHttpRequest/>

[W3C05]

W3C consortium, *Synchronized Multimedia Integration Language*, specification 2.1, W3C Recommendation, 2005. Available at <http://www.w3.org/TR/2005/REC-SMIL2-20051213/>

X

[XAML06]

"Microsoft Extensible Application Markup Language (XAML)". Available:

<http://msdn2.microsoft.com/en-us/library/ms752059.aspx>. Accessed in June2006

Y

[Ye04]

Ye, J., and Herbert, J. *User Interface Tailoring for Mobile Computing Devices*. In Proceedings of UI4All, 8th ERCIM Workshop « User Interfaces for All » (Vienna, Austria, 28-29 June 2004).

Z

[Zaph01]

Zaphiris, P. (2001). *Age Differences and the Depth-Breadth Tradeoff in Hierarchical Online Information Systems*. In Stephanidis, C. (Ed.). Proceedings of 1st International Conference on Universal Access in HCI UAHCI'2001 (New Orleans, August 5-10, 2001). Mahwah: Lawrence Erlbaum Associates, pp. 540-544.

Annex A. Task Type Taxonomy

This task type taxonomy comes from [Gonz07]

Task Type	Synonyms/sub-task types	Definition
Communicate	Convey, Transmit, call, acknowledge, respond/answer, suggest, direct, instruct, request	The action to exchange information
Create	Input/Encode/Enter Associate, name, group, introduce, insert, (new), assemble, aggregate, overlay (cover), add	Specifies the creation of an item instance
Delete	Eliminate, Remove/cut, ungroup, disassociate, ungroup	The action of deleting an item
Duplicate	Copy	Specifies the copy of an item
Filter	Segregate, set aside	The action of filtering an item
Mediate	Analyze, synthesize, compare, evaluate, decide	The action of intercede task items
Modify	Change Alter, transform, tuning, rename, segregate, resize, and collapse/expand?	An action of modifying an item
Move	Relocate, Hide, show? position? Orient? Path or travel? X	the action to change the location of an item
Navigation	Go/To	the action to find the way through containers
Perceive	Acquire/detect/search for/scan/extract, identify / discriminate / recognize, Locate, Examine, monitor, scan, detect,	The action of identifying items and/or information from the items
Reinitialize	Wipe out, Clear, Erase	The action of cleaning an item
Select/choose	Pick	selection between items
Start	Initiate/Trigger, Play, Search, active, execute, function, record, purchase	Specifies the beginning of an operation
Stop	End / finish/exit/suspend?/complete? /Terminate/Cancel	Specifies the end of an action
Toggle	activate/ deactivate, /switch	The existence of two different states of an item

Table A-1: Task types

Annex B. Comparing Standard Web Applications and RIAs

In order to create a categorization we need to define the range that every feature should cover (see table 1) and the proposed weight given to every feature. The most important characteristics have received a weight near 1 and characteristics not relevant have scored almost 0.

Features	Dynamical retrieval		Perceptive continuity			Adaptability			Multimedia			
Feature Attribute	no	yes	none	partial	Full	None	partial	Full	none	animation	sound	embedded streaming video/sound
Values	0	100	0	50	100	0	50	100	0	30	30	40
Dim. Weight	1		0.8			0.8			0.6			
Features	Collaborative faculties			User Interface language		Push Technology		use of Browser area (main or popup one)				
Feature Attribute	none	partial	full	no	Yes	no	yes	minimal	partial	Full		
Values	0	50	100	0	100	0	100	30	60	100		
Dim. Weight	0.2			0.6		0.2		0.2				

Table B-1 Features and Weights needed to categorize a RIA

A strategy that we take to contrast the sometimes not evident features of RIAs was to compare a RIA version of some Web application to get a clearer image of the differences. The results of these evaluations are grouped by pairs.

Annex B. Comparing Standard Web Applications

RIA	URL	http://www.openlaszlo.org/lps/demos/amazon/amazon.lzo?fb=1&lzt=html	
			
SWA	URL	http://www.amazon.com/Wolfgang-Amadeus-Mozart-Complete-Works/dp/B000BL13K2/sr=8-2/qid=1171641406/ref=pd_bbs_sr_2/105-5669443-2613265?ie=UTF8&s=music	
			
	Features	Dynamical retrieval	Yes
		Perceptive continuity	Yes
		UI Adaptability	Yes
		Multimedia	Yes
		Collaborative facilities	No
		User Interface description language	Open Laszlo
		UIDL	
		Push Technology	No
		use of Browser area	100%
	Features	Dynamical retrieval	yes
		Perceptive continuity	No /refresh page
		Adaptability	No
		Multimedia	No
		Collaborative facilities	No
		User Interface language	HTML
		Push Technology	No
		use of Browser area	100%

Table B-2 Comparison between e-commerce applications

Annex B. Comparing Standard Web Applications


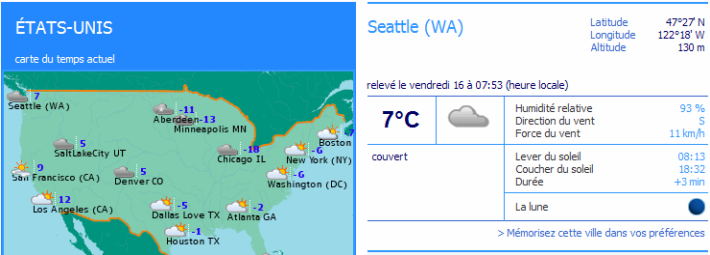
RIA	URL	http://www.openlaszlo.org/lps/demos/weather/weather.html	
	Features		
SWA	URL	http://www.tv5.org/TV5Site/meteo/detail_ville.php?langue=fr&id_ville=1705&id_pays=0&mVille=saisissez+le+nom+d%27une+ville	
	Features		

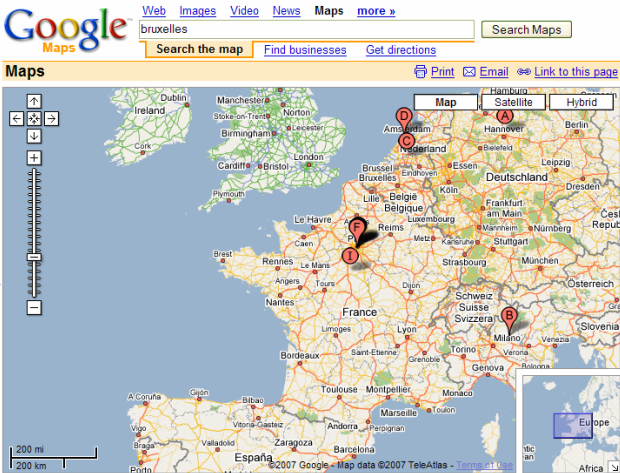

Table B-3 Comparison between weather web applications

Annex B. Comparing Standard Web Applications

RIA	URL	https://login.yahoo.com/	
	Features	Dynamical retrieval	Yes
		Perceptive continuity	Yes
		Adaptability	Yes
		Multimedia	Yes
		Collaborative faculties	No
		User Interface language	AJAX
		Push Technology	Yes
		use of Browser area	100%
SWA	URL	https://login.yahoo.com/	
	Features	Dynamical retrieval	No
		Perceptive continuity	No
		Adaptability	Partial
		Multimedia	No
		Collaborative faculties	No
		User Interface language	DHTML
		Push Technology	No
		use of Browser area	100%

Table B-4 Comparison between web mail applications

Annex B. Comparing Standard Web Applications

<p>RIA</p>	<p>URL http://maps.google.com/</p> 																				
<p>Features</p>	<table border="1"> <tr> <td>Dynamical retrieval</td> <td>Yes</td> <td rowspan="5">The marks can be available to other users</td> </tr> <tr> <td>Perceptive continuity</td> <td>Yes</td> </tr> <tr> <td>Adaptability</td> <td>Yes</td> </tr> <tr> <td>Multimedia</td> <td>Yes</td> </tr> <tr> <td>Collaborative faculties</td> <td>Yes</td> </tr> <tr> <td>User Interface language</td> <td>ajax</td> <td></td> </tr> <tr> <td>Push Technology</td> <td>Yes</td> <td></td> </tr> <tr> <td>use of Browser area</td> <td>partial</td> <td></td> </tr> </table>	Dynamical retrieval	Yes	The marks can be available to other users	Perceptive continuity	Yes	Adaptability	Yes	Multimedia	Yes	Collaborative faculties	Yes	User Interface language	ajax		Push Technology	Yes		use of Browser area	partial	
Dynamical retrieval	Yes	The marks can be available to other users																			
Perceptive continuity	Yes																				
Adaptability	Yes																				
Multimedia	Yes																				
Collaborative faculties	Yes																				
User Interface language	ajax																				
Push Technology	Yes																				
use of Browser area	partial																				
<p>SWA</p>	<p>URL http://www.maps-of-mexico.com/distrito-federal-df-mexico/mexico-df-distrito-federal-mexico-map-main.shtml</p> 																				
<p>Features</p>	<table border="1"> <tr> <td>Dynamical retrieval</td> <td>No</td> <td>The map is segmented using low resolution thumbnails to link to</td> </tr> </table>	Dynamical retrieval	No	The map is segmented using low resolution thumbnails to link to																	
Dynamical retrieval	No	The map is segmented using low resolution thumbnails to link to																			

Annex B. Comparing Standard Web Applications

			the real size maps
	Perceptive continuity	No	
	Adaptability	No	
	Multimedia	No	
	Collaborative faculties	No	
	User Interface language	HTML	
	Push Technology	No	
	use of Browser area	100%	

Table B-5 Comparison between map dispatcher applications

Annex B. Comparing Standard Web Applications

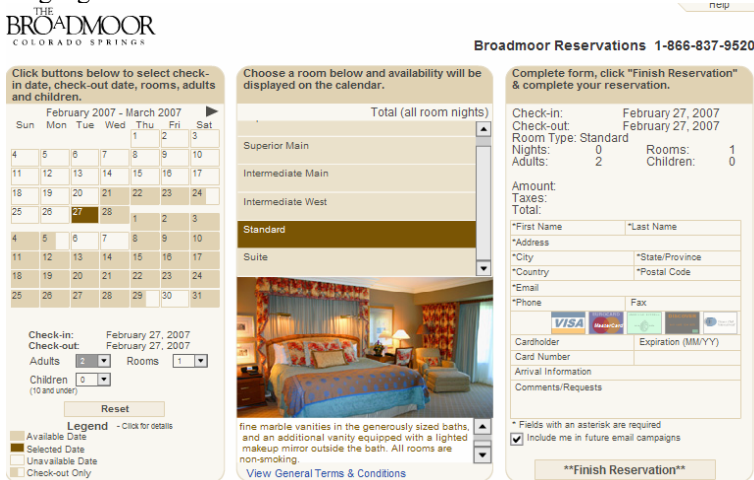

RIA	<p>URL https://reservations.ihotelier.com/onescreen.cfm?hotelid=2054&languageid=1&rezT=2054</p> 																
Features	<table border="1"> <tr><td>Dynamical retrieval</td><td>Yes</td></tr> <tr><td>Perceptive continuity</td><td>Yes</td></tr> <tr><td>Adaptability</td><td>Yes</td></tr> <tr><td>Multimedia</td><td>Yes</td></tr> <tr><td>Collaborative faculties</td><td>No</td></tr> <tr><td>User Interface language</td><td>MXML(not known) /Flash</td></tr> <tr><td>Push Technology</td><td>No</td></tr> <tr><td>use of Browser area</td><td>100%</td></tr> </table>	Dynamical retrieval	Yes	Perceptive continuity	Yes	Adaptability	Yes	Multimedia	Yes	Collaborative faculties	No	User Interface language	MXML(not known) /Flash	Push Technology	No	use of Browser area	100%
Dynamical retrieval	Yes																
Perceptive continuity	Yes																
Adaptability	Yes																
Multimedia	Yes																
Collaborative faculties	No																
User Interface language	MXML(not known) /Flash																
Push Technology	No																
use of Browser area	100%																
SWA	<p>URL http://www.mx.despegar.com/paginas/paquetes/busquedapaquetes.asp</p> 																
Features	<table border="1"> <tr><td>Dynamical retrieval</td><td>No</td></tr> <tr><td>Perceptive continuity</td><td>No</td></tr> <tr><td>Adaptability</td><td>yes</td></tr> <tr><td>Multimedia</td><td></td></tr> <tr><td>Collaborative faculties</td><td>No</td></tr> <tr><td>User Interface language</td><td>Dhtml</td></tr> <tr><td>Push Technology</td><td>No</td></tr> <tr><td>use of Browser area</td><td>partial</td></tr> </table> <p style="text-align: right;">Using javascript</p>	Dynamical retrieval	No	Perceptive continuity	No	Adaptability	yes	Multimedia		Collaborative faculties	No	User Interface language	Dhtml	Push Technology	No	use of Browser area	partial
Dynamical retrieval	No																
Perceptive continuity	No																
Adaptability	yes																
Multimedia																	
Collaborative faculties	No																
User Interface language	Dhtml																
Push Technology	No																
use of Browser area	partial																

Table B-6 Comparison between online reservation systems