UCL
Institut d'administration et de gestion
School of Management
ESPO/IAG

UNIVERSITE CATHOLIQUE DE LOUVAIN

DEPARTEMENT D'ADMINISTRATION ET DE GESTION

Transforming Phone-based Interfaces for Web Access: from WML to UsiXML

Promoteur: VANDERDONCKT Jean

Mémoire-projet
présenté par
CUI Xuefeng

en vue de l'obtention du titre de
Master in Business Administration

Année académique 2004-2005

PREFACE

Firstly, I would like to give thanks to Professor Jean Vanderdonckt, my promoter. Without his valuable direction, useful advice, and all-the-time support, I could not have finished this thesis.

Secondly, I would like to give thanks to Laurent Bouillon and Benjamin Michotte who helped me through the whole project.

Thirdly, I would like to give thanks to all the professors and assistants of IAG who once taught me and helped me.

Finally, I would like to thank my dear family. Their love and support give me the courage and strength.

TABLE OF CONTENTS

Chapter 1. INTRODUCTION

In this chapter, we present the problem, challenges, related works, the solution we have chosen to address these issues, and thesis scope & motivation. We conclude this chapter with the structure of this thesis.

1.1 Problem

An inevitable fact is that systems change over time. Original requirements will have to be modified to account for changing user needs and advances in technology. User interfaces are no exception to this rule, and current trends in user interface (UI) design are rapidly evolving. The web, as a vital medium for information transfer, has had an impact on UI design. Furthermore, the introduction of new platforms and devices, in particular mobile phones and PDAs, has added an extra layer of complication to required UI system changes, from where the concept of multiple user interfaces emerges.

In the migration of interactive systems to new platforms and architectures, many modifications have to be made to the user interface. According to the study of Myers [1] the development of the User Interface is a difficult process and very costly in time. In fact, the study shows that, in generally, it spends the 45% of time for the conception of an application and 48% of code on the creation of User Interface. Further more, the diversity of the IT device available in the market has greatly enforced this issue. Such diversity includes different categories as following: Smartphones, Pocket PC, Tablet PC, classic PC, Laptop PC, and so on.

The rapid growth of Web services has led to a situation where companies and individuals rely more and more on material that is available on the Web. For the users of web service, they want to use the same services with different devices depending on their current context of use. The services have to adapt their contents and presentation according to the constraints imposed by the target platform such as: operating system, programming language, screen resolution, interaction capabilities. It is beneficial to maintain the look and feel of the service for all the possible platforms.

Until now the wireless Internet access service is not yet widely distributed comparing with wired Internet services in common applications. However, in the near future we may see that small portable personal devices including smart phones and other sub-palm-sized devices outnumber traditional Internet terminals in generating Internet

traffic. To facilitate Internet access via wireless mobile devices, the Wireless Application Protocol (WAP) and Wireless Markup Language (WML) were released.

As each device has its own characters, the restrictions imposed by different device for the User Interface are significantly different. For instance, the main user interface restrictions of WAP devices are: small screen, limited input techniques, limited amount of memory, and slow network connections, while the personal computer with broadband cable network connection have much more capability in all the above categories. As a result, the users of mobile device try to overcome these problems by avoiding complex content. A number of service providers maintain separate and lighter versions of their web service for mobile users. This issue explains that why the User Interfaces are so various depending on the different target platforms.

The following figure shows three different User Interfaces presented in three different platforms for the same information content.

Fig.1 presentations for the same resource in different platforms

The upper two User Interfaces are respectively presented by Trium mobile phone and Ericsson PDA from left to right. Compare these two User Interfaces, we can conclude that even two different WAP devices have obviously difference appearance, for instance the content layout, for the User Interface of the same WAP information content according to their different capacities. The third one is presented by the Firefox Web browser on a Gericom laptop PC in windows 2000 OS. This User Interface shows more graphical components and more complicated structure than the previous two examples.

1.2 Challenges

As the variety of platforms has increased dramatically, to ensure the usability for all the platforms which support Web by reserving the original WAP UI design is a big challenge today.

Usability is a measure of the quality of a system from the user's point of view. Usability defines whether the system solves the right problems from the user's point of view, and whether the system solves the problems in the right way. Usability design implies learning to know the users and understanding their needs so that the user's point of view is properly taken into account in the design.

Usability has multiple components. It is defined by International Organisation for Standardization (ISO) as "The effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in particular environments". [2]

- *Effectiveness* defines whether the system includes the right features from the user's point of view.
- *Efficiency* defines how quick and easy the system is from the users point of view.
- *Satisfaction* defines that the system should be pleasant to use, so that the user are subjectively satisfied when using it.

The effectiveness problems in WAP services are often related to the limited selection of available services, and the limited contents of these services. There are no general rules for what kind of content mobile users will need. Mobiles services are often built as subsets of fixed network services, picking up a small selection of contents based on assumptions regarding what kind of services mobile users will need. The process often leads to a general content that can also probably be accessed elsewhere, and more easily than with the WAP service.

The main efficiency problems in WAP Services are related to the restrictions of mobile

devices, browsers and networks. In current WAP networks, the main efficiency problem is the time required to establish a connection to the service. On a small screen, only a limited amount of information can be displayed at a time. Because of memory restrictions, the device can only receive a limited amount of data at a time. When accessing the service, the user has to scroll the screen and wait for new downloads every now and then. Since the information has to be served to the user in "small portions" the services have to be constructed based on limitations rather than natural classification. As a result, the services often include artificial classifications, thus causing problems to the user. For instance, the services are often divided into categories such as "Utilities", "Tools" and "Entertainment", giving the user little idea which category the service he or she is seeking belongs to. The navigation efficiency in the services can be improved by trying to make the items that the user is most likely to request the easiest to access. The user should get good navigation support, including feedback in the page header on where he/she currently is in the service. There should also be easy access to back and forward.

User satisfaction is based on the total usage experience, ease of use and utility. Mobile users do not usually browse around but need the information or service quickly; Compared to fixed network users, mobile users need services that are faster and easier to use on their much more modest devices. These user requirements are not easy to fulfil.

To adapt WAP services to platforms other than mobile phone, it should concern all these three mentioned categories according to the constraints posed by different platforms. As a result, the content authors can no longer afford to develop a content that is targeted for use via a single access mechanism only. There is a need for common guidelines on how to provide multiplatform web services.

While there are WAP gateways (for example, the Nokia WAP Server) that are capable of converting HTML pages into WML decks, this technology is still pretty much in its infancy stage. A direct HTML to WML conversion often results in a clumsy user interface on the WAP device and vice versa. In any case, usability issues on the Web and WAP environments are totally distinct. It is thus difficult to transform an HTML page that is designed to be displayed on a web browser to be displayed on a WAP device without some loss of usability and vice versa.

Toward to a platform independent Web for WML User Interface, the following section will present you several existing solutions to adapt a WML User Interface to different

platforms for the Web.

1.2 Related works

There are four general approaches to displaying Web UIs on various devices: device-specific authoring; multiple-device authoring; client-side navigation; and automatic re-authoring.

1.2.1 Device-specific authoring

Device-specific authoring involves authoring a set of Web UIs for a particular display device, for example a cellular phone outfitted with a display and communications software such as the Nokia 9000. The basic philosophy in this approach is that users of such specialty devices will only have access to a select set of services, and the UIs for these services can all be designed up-front for the device's particular display. The desired UIs must be pre-defined and custom information extraction and UIs formatting software must be written to deliver the information to the target device. This is the approach taken in Unwired Planet's UP.Link service [3] which uses a proprietary mark-up language (HDML).

Device-specific authoring will typically yield the best-looking results, but limits the user's access to a small select set of web pages.

1.2.2 Multiple-device authoring

In multiple-device authoring, a range of target devices is identified, and mappings from a single source document to a set of rendered documents are defined to cover the devices within the range. One example of this is the StretchText approach [4], in which portions of the document (potentially down to the word level) can be tagged with a 'level of abstraction' measure. Upon receiving the UI document, users can specify the level of abstraction they wish to view and are presented with the corresponding detail or lack thereof. Another example of multiple-device authoring is HTML cascading style sheets (CSS) [5]. In CSS, a single style sheet defines a set of display attributes for different structural portions of a document (e.g., all top-level section headings are to be displayed in red 18-point Times font). A series of style sheets may be attached to a document, each with a weight describing its desirability to the document's author. The user can also

specify a style sheet, as can the WWW browser (the 'default' style sheet). Although the author's style sheets normally override the user's, the user can selectively enable or disable the author's, providing the ability to tailor the rendering of the document to their particular display.

Multiple-device authoring, while it spends less total effort per document than device-specific authoring, still requires significantly more manual design work than simply authoring for a single desktop platform.

1.2.3 Client-side navigation

In client-side navigation, the user is given the ability to interactively navigate a single web page by altering the portion of it that is displayed at any given time. A very trivial example of this is the use of scroll bars on the document display area. A much more sophisticated approach is that taken in the PAD++ system [6], in which the user is free to zoom and pan the device display over the document with infinite resolution. Active Outlining

Client-side navigation holds promise if a good set of techniques can be developed, but the 'peephole' approach taken in PAD++ seems very awkward to use for large documents, and the active outlining technique has limited applicability since most web pages do not use a strict section/sub-section organization.

1.2.4 Automatic re-authoring

Automatic re-authoring involves developing software which can take an arbitrary web UI designed for the original platform, along with characteristics of the target display device, and re-author the web UIs through a series of transformations so that it can be appropriately displayed on the device. This process can be performed either on the client, on the server, or on an intermediary HTTP proxy server [7] which exists solely for the purpose of providing these transformation services.

As we've discussed above, according to the shortcomings of other solutions, automatic re-authoring is thus the ideal approach to providing broad access to the web from a wide range of devices, if it can be made to produce legible, navigable and aesthetically pleasing re-authored documents without loss of information.

Within this solution, there is a popular technology which is called Transcoding Technology. For the transcoding technology, the transcoding applications automatically transform a UI code from the original platform to a new UI code for the target platform. This transformation can occur at design-time (i.e., the transformation is made only once and re-inserted in the formation is made only once and re-inserted in the new platform) or at run-time (i.e., the transformation is performed on demand when the UI is requested).

Fig.2 Example for Transcoding

The Fig.2 presents an application [8] as an example for transcoding approach. It uses an HTML/WML conversion proxy server, which converts HTML-based Web contents automatically and on-line to WML. This application gives the mobile user transparent access to their familiar Web pages from their mobile phones and other mobile devices. If HTML-based Web services follow certain guidelines, they can be converted automatically to WML and adapted to the client device.

The HTML to WML conversion consists of two main tasks: dividing the document into parts that may reasonably be viewed on a display of the target device, and converting the document type to the markup language supported by the target device. The conversion should have knowledge about the target device, such as display size, supported image formats, and the markup language support.

The layout of an HTML page must be modified, and images are either discarded or simplified. Most HTML pages are useful even without images, and the layout may be mimicked by the suitable organisation of WML cards. The danger is that the user may not find the desired information even though it is somewhere in the card set. The division into cards makes the presentation simpler, but the cards do not generally render the information unavailable. The practice has confirmed the usefulness of converted HTML pages.

The conversion first checks and validates the HTML document. Then, the server parses the document, converts the contents and rearranges the contents as WML decks and cards (Figure 3).

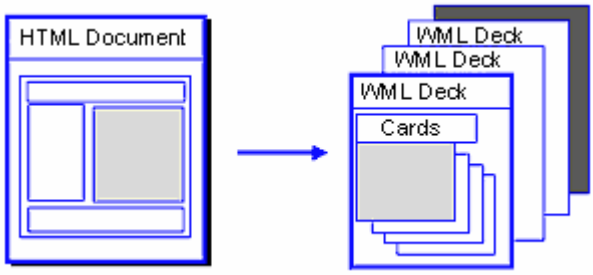


Fig. 3 HTML/WML transcoding

HTML	WML	Description
 	 	line break
<i>	<i>	italic
<u>	<u>	underline
<p>	<p>	paragraph
<b>	<b>	bold
<small>	<small>	small text
		strong text
<a>	<a>	hyperlinks
<select>, <option>	<select>, <option>	choices
<table>, <td>, <tr>	<table>, <td>, <tr>	tables
<textarea>, <input>	<input>	text area extboxes, submit buttons
META HTTP-EQUIV="refresh"	<onevent>	timed site redirection

Table 1. Mapping table from HTML to WML

The above table is a mapping table from HTML tags to WML tags. The parsing breaks the HTML data into its logical elements, such as start-tags and end-tags, attributes and text. The parser also checks the document against the given Document Type Definition

(DTD) and it corrects errors. When converting from HTML to WML, some data is inevitably lost. The terminal adaptation phase may work optimally only if it has access to all information in the original HTML.

To help the transcoding process to get extra information which is not contained in the original HTML, the application using annotations can resolve this problem. For example, the conversion tool [9] which transcoding HTML to VoiceXML firstly codify external information structures in an XML annotation language called VXPL that can be used to annotate existing HTML code to indicate where these information structures exist. VXPL is designed so that it can be used for manual of automatic annotation of HTML code. Secondly, they use automated transcoder to converts VXPL into VoiceXML with increased support for navigation and usability of the resulting interface.

However, we assumed that these transcoding tools would not produce results as good as services designed specially for the mobile clients. Because these two transcoding tools above are about to transform a final UI to another final UI, both of them lack flexibility with no design alternatives.

Another shortcoming for transcoding technology is that it is too specific. It needs several one-to-one mappings, such as HTML/WML, XHTML/WML and XML/cHTML. Each mapping is very specific and can not be generalized to other platforms. Therefore, this approach is limited in that it is very specific to the problem raised and should be multiplied by the amount of platforms.

1.3 Our Solution

In considering the shortcomings of above solutions, we choose the User Interface Re-engineering technology to be the solution of the platform independent access of WEB User Interfaces. The reason will be explained as following.

User Interface Re-engineering transforms a final user interface into a logical representation that has enough information to allow forward engineering to port a UI from one computing platform to another with maximum flexibility and minimal effort. Re-engineering is used to adapt a UI to another context. This adaptation is governed by two main tasks: the adaptation of the code itself to the new computing plating platform and the redesign of the UI to better suit the new constraints of the target platform (interaction capabilities, screen size.). [10]

The user interface reengineering process can be divided into two phases: Reverse engineering and Forward Engineering.

The general definition of Reverse engineering is the process of analyzing a subject system to: "identify the system's components and their interrelationships, and create representations of the system in another form or at a higher level of abstraction."[21] Forward Engineering is the process of generating and implementing the user interface from high-level models.

In reverse engineering, the challenge is to understand the interface code for building a high-level UI model. This UI model represents all the relevant aspects of a UI. Various high-level models have been suggested in the literature including object-oriented models and abstract UI specification, as well as general task models for the problem domain and environment dependent task models. All these models can drive the interface development process [11].

Comparing to the transcoding technology, the re-engineering technology has three advantages. Firstly, at the reverse-engineering stage, it transforms the original final UI to a logical presentation which is platform independent and modality independent. So it can be more flexible for more alternative design possibilities corresponding to different target platforms and modalities such as graphical UIs, virtual UIs etc. Secondly, not like the transcoding method which inevitable loss some design information in transforming to another target language, with the logical presentation model can store all the design information before performing the forward engineering. Thirdly, with this logical presentation model, the original UI design could be reused for all the target platforms of the transformation, as a result, it realize the reusability of resulted UIs.

1.4 Scope and Motivation

Nowadays, the web service providers are doing great efforts to satisfy the growing variety of mobile clients. As a result, the diversity of mobile Internet services and devices is rapidly increasing.

However, most of the currently available WAP services are not generic but they have been tailored to specific WAP devices. The selection of WAP devices is expected to range from mobile phones to palmtop computers. Even if the service is generic, it has

been designed according to the minimum client device. Then the service cannot utilize the more advanced features of other devices. As more devices will be available on the market, it will require more and more efforts to maintain device-specific services. Adapting the contents to different devices, networks and user preferences will be a big challenge for service designers.

In a WAP study [12], Marc Ramsay and Jakob Nielsen describe a situation where two users access the same site with different phones. One of the users describes the site as "fantastic", and the other says that the site makes him feel "aggrieved". The latter user could see the index of a restaurant guide but every time he tried to access the information itself, he got the message "wrong address". These kinds of problems arise when designers have omitted to take into account the different capabilities of the WAP devices and the different interpretations of WML code on the browsers.

Different from WAP, the Japanese i-mode is a closed specification. The operator (NTT DoCoMo) [13] is both in charge of delivering the devices and offering the services, and the operator also selects the services to be provided. In this approach, service providers do not need to worry about the adaptation of services

Focus on this issue, within the scope of this thesis, we only concern about the adaptation of WAP User Interface for Web access. As the contents of the WAP services are implemented in Wireless Markup Language (WML) and WMLScript, to be more concrete, we take the User Interfaces which is written in WML 1.1 as the start point for the User interface adaptation for the case of this thesis. A description of WML is presented in the chapter 4.

As we taking the User Interface reengineering techniques to facilitate the adaptation of WML User Interface, within the scope of this thesis, we only concentrates in the first stage, in other words, in the reverse engineering process in order to transform the existing WML UIs developed for a mobile phone into a logical representation for other platform that was not initially planned without loosing the development effort. So we don't consider constraints imposed by the target platform such as: operating system, programming language, screen resolution, interaction capabilities. The goal is to reuse the existing design if possible.

To support the reverse engineering process, we have developed a reverse engineering

tool that allows a flexible recovery of the presentation model from the WML UIs. For the target language of the presentation model, we choose the Concret User Interface (CUI) model of USer Interface eXtensible Markup Language (USIXML) among the different User Interface Description Languages (UIDLs). The reason for the choices of technology is presented in the chapter 2.

1.5 Thesis Structure

The chapter 2 present the User Interface Description Language (UIDL) that we choose to support the User Interface Reverse Engineering among other UIDLs.

The chapter 3 introduce the XML transformation tool developed for the User Interface Reverse Engineering and its related technologies.

The chapter 4 provides introduction about WML as well as its meta-model and the detailed specification. At the same time, we show the UIs by different emulators for some major widgets from "real world" examples.

The chapter 5 introduce CUI model of USIXML with its specification especially for the USIXML element which can be mapped from WML. We also show some concrete CUI Model examples for the major elements.

The chapter 6 defines the mapping rule from WML 1.1 to CUI model of USIXML with the help of XPATH notation. It also presents the corresponding XSLT template to each mapping rule which can be combined to be a complete XSLT style sheet which serves as the input file for the XSLT transformation application.

The chapter 7 presents some concrete transformation examples by using the implemented transformation application and the XSLT style sheet. Each example also serves to test the result of transformation.

The chapter 8 is dedicated to conclude this thesis.

Chapter 2. USER INTERFACE DESCRIPTION LANGUAGE FOR REVERSE ENGINEERING

A UI Description Language (UIDL) consists of what a high-level computer language for describing characteristics of interest of a UI with respect to the rest of an interactive application. Such a language involves defining a syntax (i.e., how these characteristics can be expressed in terms of the language) and semantics (i.e., what do these characteristics mean in the real world). It can be considered as a common way to specify a UI independently of any target language (e.g. programming or markup) that would serve to implement this UI. [15]

Today, the challenge of the UI development is brought by Diversity of users, Richness of cultures, Complexity of interaction devices and styles, Heterogeneousness of computing platforms, Multiplicity of working environments, and Multiplicity of contexts of use. [23] With the help of UIDLs, we can achieve various goals concerning the above challenges, for instance: Ensuring portability of UIs from one computing platform to another while preserving some consistency between or with the target computing platform; Making one UI design for multiple device, platforms, or appliances; Using a UI description to enable automated generation of UI code; Improving the reusability of UI design, etc. [16]

As we've presented in the previous section, we use the Reverse Engineering approach to attain the goal of this thesis. The Reverse Engineering is the process of analyzing software code with the objective of recovering its design and specification. In the reverse engineering of interactive systems, the ideal behavioural specification of the system is an abstract UI model with enough detail to allow appropriate user interface techniques, in particular model-based approaches, to be chosen in the new interface domain [14].

With the guide of the discipline of Human-Computer Interaction (HCI), in addition to the above goals, User Interface Description Languages (UIDLs) can also support the reverse engineering. To present our choice of the UIDL, we firstly introduce some UIDLs such as UIML, XIML, TERESAXML. Secondly, we will present the UsiXML in depth to show its advantages over the other UIDLs.

2.1 UIML

User Interface Markup Language (UIML) is a meta-language that allows designers to describe the user interface (UI) in generic terms, and to use a style description to map the UI to various operating systems, languages and devices. [15] UIML is used to define interface elements such as buttons, menus, lists, and other controls. It also defines the layout and design of the controls, and actions to take place when certain events occur. Events may be created by the user interacting with the interface. Developing user interface using UIML involves writing UIML code which is a low level specification of the user interface. The advantages of using UIML include using UIML to describe the UI's behavior in a device independent manner, its ability to give as much power to a UI implementer as a native toolkit, its ability to describe content, structure, behavior and style of UI separately.

UIML supports all the features pertaining to a platform. This complete support is made possible by defining vocabularies. A vocabulary is a set of names, properties and associated behaviors for UI elements. Just like a programmer would use pre-defined libraries, UI designer can use pre-defined vocabularies to create user interfaces using UIML. UIML is a language to describe user interfaces for multiple devices; However, UIML does not provide any facility to write one description for multiple platforms. A UI designer has to create separate UIs for each platform using its own vocabulary. [24]

2.2 XIML

eXtensible Interface Markup Language, maintained by XIML Forum(http://www.ximl.org) – an independent Consortium, is an XML-based language for developing multiple user interfaces by transforming and refining user tasks and UI models. It provides representation framework for industry and universal support of functionality across interface lifecycle which includes phases of "design, development, operation, management, organization and evaluation" [17].

XIML is very abstract interface definition language which divides definition of interface into "components", high level building blocks of an interface. Examples of components include task(business process), domain(defines a hierarchy of components), user(defines hierarchy of end-users), presentation, and dialog(defines actions within the interface). XIML supports task modeling, that is, it has the ability to represent abstract concepts such as user tasks, domain objects, and user profiles. Components are mapped to "elements" which are concrete representations such as "widgets". XIML representation framework provides support for relational modeling between components.

XIML can specify any type of model, of model element, and relationships between. Although some predefined models and relationships exist, one can expand the existing set to fit a particular context of use. XIML has been used in MANNA for platform adaptation, in VAQUITA to support reverse engineering, and in Envir3D to transform graphical UI into a virtual one [18]

2.3 TERESA

The TERESA (Transformation Environment for interactive Systems representations) exploits a UIDL called TERESAXML produces different UIs for multiple computing platform from a general task model which is progressively refined for the different platforms. Then, various presentation and dialogues techniques are used to map the general specifications expressed into XHTML code for each platform such as web, PocketPC, and mobile phones. [19]

The TERESAXML is the XML-compliant language that was developed inside the Teresa project, which is intended to be a transformation-based environment designed and developed at the HCI Group of ISTI-C.N.R. It composed of two parts: (i) a XML-description of CTT notation which was the first XML language for task models; (ii) a language for describing user interfaces. This last part will be more deeply investigated. Teresa XML for describing UIs specifies how the various AIO composing the UI are organized, along with the specification of the UI dialog. [15]

2.4 UsiXML

USer Interface eXtensible Markup Language (UsiXML) defined by the ISYS unity of IAG consists of a User Interface Description Language (UIDL) allowing designers to specify a user interface at multiple levels of abstraction depending on the development path they are following: task and concepts, abstract user interface (AUI), concrete user interface (CUI), and final user interface (FUI) [16]. USIXML can be used to specify a platform-independent, a context-independent, and a modality-independent UI. For instance, a UI that is defined at the AUI level is assumed to be independent of any modality and platform. Therefore, it can be reified into different situations. Conversely, a UI that is defined at the CUI level can be abstracted into the AUI level so as to be transformed for another context of use.

UsiXML is structured according to the four basic levels of abstractions defined in the Cameleon reference framework [20] that is intended to express the UI development life cycle. The following picture shows this framework.

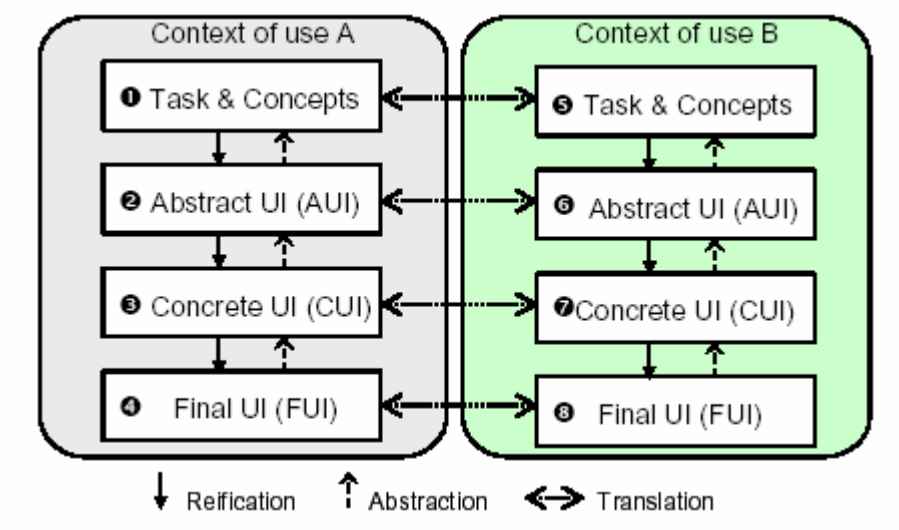


Fig.4 Cameleon framework

2.4.1 The Task & Concept level

At the top of the framework is the Task & Concepts level where the interactive task to be carried out by the end user is defined according to her viewpoint, along with the various objects that are manipulated by these tasks. These objects are considered as instances of classes representing the concepts manipulated. Within this level, there are three different models which are task model, domain model and context model.

Task model

This model describes the various tasks to be carried out by a user in interaction with an interactive system. An extended version of ConcurTaskTree (CTT) [27] has been chosen as a task modeling technique to represent user's tasks and their logical and temporal ordering. A task model is therefore composed of tasks and task relationships. Each task is described by a name, a type, a frequency value, and an importance value. The task could be decomposed to sub-task until the possible lowest level.

Domain model

A domain model describes the real-world concepts and their inter-actions as understood

by users. USIXML domain model has the form of a UML class diagram. Domain model concepts are classes, attributes, methods and domain relationships. Concepts contained in USIXML domain model are at a certain point manipulated by users. By manipulated, it is meant that either attribute values are rendered through the UI or that methods attached to classes of objects are used by a user.

Context model

A context model describes all the entities that may influence carrying out the interactive task of user with the intended UI. It is assumed to capture any relevant attribute of the context of use, in which the user is. A context model consists of:

• A user model that recursively decomposes the user population into stereotypes (or profiles) and sub-stereotypes, each stereo-type sharing a same series of attributes and associated values.

• A platform model captures relevant attributes for each couple software-hardware platform that may significantly influence the context-sensitivity.

• An environment model describes any property of interest of the physical environment where the user is using the UI on the computing platform to accomplish her interactive tasks. Such attributes may be lighting conditions, level of stress, etc.

2.4.2 The Abstract User Interface Level

In this level, the AUI model represents a canonical expression of the renderings and manipulation of the domain concepts and functions independently from any modality and computing platform. An AUI is populated by abstract interaction objects and abstract user interface relationship. An AIO describes an abstraction of widgets found in most toolkits like windows, buttons but, also, vocal output widget in auditory interface. As a result, the AIO is independent of any modality of interaction and any platform. For the Abstract User Interface Relationship, it is an abstract relationship among AUI objects that indicate the existence of some spatio-temporal setting among them. The Spatio-temporal relationships characterise the physical constraints between AIOs as they are presented in time and space.[19]

2.4.3 The Concrete User Interface Level

In this level, the CUI model concretises an abstract UI for a given context of use into Concrete Interaction Objects (CIOs) so as to define widgets layout and interface navigation. Like the AUI level, the CUI level is also independent of any computing platform.

A CUI is also an abstraction of the FUI. We can realize the FUI by concretizing the different widgets defined in the CUI level. For the reverse engineering, we can transform the source final UI to the CUI level. And as the resulted CUI is platform independent, Preparing for the further forward engineering, we can modified the resulted CUI to another CUI which can better abstract a new FUI according to the constraints brought by the target platform.

2.4.4 The Final User Interface Level

At the Final UI level, The UI is expressed as source code. It is produced at the very last step of the reification process which is supported by a multi-target development environment. The Final UI can be seen with the help of interpreter according to a particular platform.

2.5 Conclusion

One theorytical resson to choose USIXML is that not like UIML and XIML, the USIXML is platform independent. For instance, one big shortcoming of UIML is that, as it just offers a single language to define the different types of user interfaces, it does not allow the creation of user interfaces for the different languages or for different devices from a single description: there is still a need to design separate UIs for each device.[15]
On the other side, as USIXML is platform indepenedent, it ensures portability of UIs from one computing platform to another while preserving some consistency between or with the target computing platform. [19]

The other theorytical reson is that USIXML has a context model which describes all the entities that may influence carrying out the interactive task of user with the intended UI. This context model consists of user model, platform model and environment model, while other UIDLs, such as UIML and XIML, don't have.

The practical reason to choose USIXML is that for the USIXML has already a complet list of supporting tools such as editors, generators, iterpretors and also transformation tools, such as ReversiXML which transforms html to USIXML. The work of this thesis can enrich such tools for USIXML.

The methodological reason to choose USIXML is that above UIDLs except USIXMl only represent an instance with some degree of coverage and restrictions of the multi-path UI development, while USIXML supports the multi-path UI development in order to better support the change of the IT environment. The multi-path UI development is characterised by the following principles [16]:

• Formal definition of UI models: any UI is expressed through to a suite of models that are analysable, editable, and exploitable by software.
• Transformational approach: each model stored according to the ontological format can be subject to transformations realizing various development steps.
• Multiple development paths: development steps can be combined together to form development paths that are compatible with the organisation's development scenario. For example, a series of transformations can be applied to progressively move from a task model to a dialog model, to recover a domain model from a presentation model, or to derive a presentation model from both the task and domain models.
• Flexible development approaches: development scenarios (e.g., forward engineering, reverse engineering, wide spreading, or middle-out) are supported by flexibly following alternate development paths. The wide spreading approach tends to apply in parallel all the required adaptations where they occur. And the middle-out approach relies on an intermediate model and propagates changes to all artefacts exploited in the development process.

Regarding the UI expressiveness for multiple contexts of use, UIML, XIML, and TERESA are UIDLs that address the basic requirements of UI modelling and expressivity. XIML is probably the most expressive one as a new model; element or relationship can be defined internally. However, there is no systematic support of these relationships until they are covered by specific software. Regarding the transformational approach, TERESA include some transformation mechanism to map a model onto another one, but the logics and the definition of transformation rules are completely hard coded with little or no control by designers. In addition, the definition of these representations is not independent of the transformation engine.

The principle advantage of USIXML is that as a UIDL it permits a multi-path UI development. [19] According to the multi-path development of UI, USIXML supports a flexible development process based on transformations. USIXML is a collection of integrated models expressed in a formal and uniform format, such as task model, CUI model, AUI model, FUI model, etc. Each of these models can be derived by each other thanks for the mapping model. In addition, USIXML can be used to specify a platform-independent, a context-independent, and a modality-independent UI. As USIXML has such advantages over other UIDLs, we choose it to support our User Interface Reverse Engineering.

Further, as it is not necessary to define the whole collection of USIXML models to get a User Interface, we can choose one USIXML model to be the target abstraction model for the User Interface Reverse Engineering. From the definition of the Reverse Engineering, we can know that the target model is a high-level UI model which represents all the relevant aspects of a UI and can be further forward engineered to a Final UI. According to this issue, we have two candidates (CUI and AUI) within the collection of USIXML models. Today, as the most popular UIs are graphical UIs, to be more concrete and direct within the scope of this thesis, we choose the CUI model which could be directly forward engineered to a graphical User Interface.

For the further forward engineering, if the target UIs are not graphical UIs such as vocal UIs, multimodal UIs, and virtual reality UIs, instead of directly perform the forward engineering on the transformed CUI model, we can firstly build a AUI model specifying constraints in time and space by the mapping of the transformed CUI model. Then we can build a new CUI in turn mapped onto more precise relationships from the AUI level which is applicable to the modalities other than graphical UIs.

For futher details of USIXML, there is a list of publications located in the following address: (http://www.usixml.org/index.php?view=page&idpage=22)

Chapter 3. XML TRANSFORMATION

The former chapters have already introduced that both of WML and USIXML are XML. This chapter introduce the related technologies that we use to perform the XML transformation. To perform the transformation, we developed a transformation tool which uses a XSL style sheet as input to define the transformation rules. The programming language is Java.

In this chapter, we firstly introduce XML. Secondly, we present the XML processing APIs with Java that we used in our transformation tool, such as SAX, DOM and JAXP. Thirdly, we present the technologies related to XSLT style sheet which serves as input of our transformation tool. Lastly, we present the transformation application and its usage.

3.1 Introduction to XML

XML is a standard textual markup language suitable for encoding almost any sort of data. It works very well for both unstructured narrative data written by people and for the record-oriented data common in computer applications. About the only thing it's not really suitable for are bitmapped things such as photographs and recorded sound.

Logically an XML document is made up of nested elements. Each element has a name, a set of attributes and some content. The content can include plain text and/or other elements. The attributes are name value pairs associated with the element. Each document has a single topmost element called the root or document element. Since all non-root elements nest completely inside other elements, an XML document has a natural tree structure. Besides elements and text nodes, XML documents can also contain comments, processing instructions, an XML declaration, and a document type declaration.

Physically, an XML document is divided into storage units called entities. These entities can be files, database records, data structures in memory, or something else. The document entity contains the root element of the document. Parsed entities contain XML markup and that will be merged to form the entire document. Parsed entities are located via general entity references such as &anaconda; in the document entity or another parsed entity. Unparsed entities contain non-XML, possibly binary data that will be identified by ENTITY type attributes in the document.

Every XML document must be well-formed. Among other things this means, every start-tag must have a matching end-tag, every attribute value must be quoted, and only certain characters can be used in element names. If a document is not well-formed, it is not an XML document; and XML parsers will not accept it. Beyond well-formedness, documents that have a schema may be (but do not have to be) valid. A valid document adheres to all the constraints listed in the schema. Schema languages include Document Type Definitions (DTDs), the W3C XML Schema Language, and the XPath-based Schema.

Since XML markup normally focuses on the structure and semantics of the contained information, before a document can be shown to a human reader, it must first be associated with a style sheet that tells the browser or other tool how to format the document for display to a person. The two most popular style languages are Cascading Style Sheets (CSS) and the Extensible Style sheet Language (XSL). CSS is a non-XML declarative language for applying simple styles such as font-weight to elements of certain types. XSL is actually two separate XML applications, the XSL-FO page description language and the XSLT Turing-complete functional language. An XSLT style sheet is used to transform a source XML document into other XML vocabularies such as USIXML.

3.2 Transforming XML documents with Java

Java is the ideal language for transforming XML documents. Its strong Unicode support in particular made it the preferred language for many early implementers. Consequently, more XML tools have been written in Java than in any other language. More open source XML tools are written in Java than in any other language. More programmers process XML in Java than in any other language.

As following, we present the three major standard APIs for processing XML documents with Java, the Simple API for XML (SAX), the Document Object Model (DOM), Java API for XML Processing (JAXP).

3.2.1 SAX

SAX, the Simple API for XML, was the first standard API shared across different XML parsers. SAX is unique among XML APIs in that it models the parser rather than the document. In particular the parser is represented as an instance of the XMLReader interface. The specific class that implements this interface varies from parser to parser.

Most of the time you only access it through the common methods of the XMLReader interface.

A parser reads a document from beginning to end. As it does so it encounters start-tags, end-tags, text, comments, processing instructions, and more. Parsing is the process of reading an XML document and reporting its content to a client application while checking the document for well-formedness.

SAX represents parsers as instances of the XMLReader interface.The parser tells the client application what it sees as it sees it by invoking methods in a ContentHandler object. ContentHandler is an interface the client application implements to receive notification of document content. The client application will instantiate a client-specific instance of the ContentHandler interface and register it with the XMLReader thats going to parse the document. As the reader reads the document, it calls back to the methods in the registered ContentHandler object.

3.2.2 DOM

The Document Object Model, DOM, provides a standard set of objects for representing HTML and XML documents, and a standard interface for accessing and manipulating them. It is the second major standard API for XML parsers. Most major parsers implement both SAX and DOM. DOM programs start off similarly to SAX programs, by having a parser object read an XML document from an input stream or other source. However, where the SAX parser returns the document broken up into a series of small pieces, the equivalent DOM method returns an entire Document object that contains everything in the original XML document. One can read information from the document by invoking methods on this Document object or on the other objects it contains. This makes DOM much more convenient when random access to widely separated parts of the original document is required.

The DOM is separated into different parts (Core, XML, and HTML)

- Core DOM - defines a standard set of objects for any structured document
- XML DOM - defines a standard set of objects for XML documents
- HTML DOM - defines a standard set of objects for HTML documents

The XML DOM views XML documents as a tree structure of elements embedded within other elements. All elements, their containing text and their attributes, can be accessed

through the DOM tree. Their contents can be modified or deleted, and new elements can be created by the DOM. The elements, their text, and their attributes are all known as nodes.

3.2.3 JAXP

Starting in Java 1.4, Sun bundled the Crimson XML parser and the SAX2, DOM2, and TrAX APIs into the standard Java class library. (TrAX is an XSLT API that sits on top of XML APIs like SAX and DOM.) They also threw in a couple of factory classes, and called the whole thing the "Java API for XML Processing" (JAXP).

The reason about include the JAXP to the XML processing API is explained as following:
DOM represents a document tree fully held in memory. It is a large API designed to perform almost every conceivable XML task. It also must have the same API across multiple languages. Because of those constraints, DOM does not always come naturally to Java developers who expect typical Java capabilities such as method overloading, the use of standard Java object types, and simple set and get methods. DOM also requires lots of processing power and memory, making it untractable for many lightweight Web applications and programs.

SAX does not hold a document tree in memory. Instead, it presents a view of the document as a sequence of events. For example, it reports every time it encounters a begin tag and an end tag. That approach makes it a lightweight API that is good for fast reading. However, the event-view of a document is not intuitive to many of today's server-side, object oriented Java developers. SAX also does not support modifying the document, nor does it allow random access to the document.

JAXP attempts to incorporate the best of DOM and SAX. It's a lightweight API designed to perform quickly in a small-memory footprint. JAXP also provides a full document view with random access but, surprisingly, it does not require the entire document to be in memory. The API allows for future flyweight implementations that load information only when needed. Additionally, JAXP supports easy document modification through standard constructors and normal set methods.

3.3 XSLT style sheet

XSLT, the Extensible Style sheet Language for Transformations is an official recommendation of the World Wide Web Consortium (W3C). An XSLT style sheet is

used to transform a source XML document into other XML vocabularies such as USIXML. It describes how documents in one format are converted to documents in another format. Both input and output documents are represented by the XPath data model. XPath expressions select nodes from the input document for further processing. Templates containing XSLT instructions are applied to the selected nodes to generate new nodes that are added to the output document.

XSLT is based on the notion of templates. An XSLT style sheet contains semi-independent templates for each element or other node that will be processed. An XSLT processor parses the style sheet and an input document. Then it compares the nodes in the input document to the templates in the style sheet. When it finds a match, it instantiates the template and adds the result to the output tree.

The biggest difference between XSLT and traditional programming languages is that the input document drives the flow of the program rather than the style sheet controlling it explicitly. When designing an XSLT style sheet, you concentrate on which input constructs map to which output constructs rather than on how or when the processor reads the input and generates the output.

As XSLT is not a procedural language, it does have the advantage of being much more robust against unexpected changes in the structure of the input data. An XSLT transform rarely fails completely just because an expected element is missing or misplaced or because an unexpected, invalid element is encountered.

The W3C has defined another standard for stylesheet which is Cascading Style Sheets (CSS), a mechanism used to define various properties of markup elements. Although CSS worked for XHTML at present, it has some drawbacks:

- CSS can't change the order in which elements appear in a document.
- CSS can't do computations.
- CSS can't combine multiple documents

3.4 Java application for XML Transformation

In this section, at first, we show the transformation procedure which has 4 steps to realise the transformation. Then, we present the transformation application's source code

which includes some short comments corresponding to the 4 steps of the transformation procedure. At last, we explain how to use the java application and its requirements.

3.4.1 Procedure of Transformation

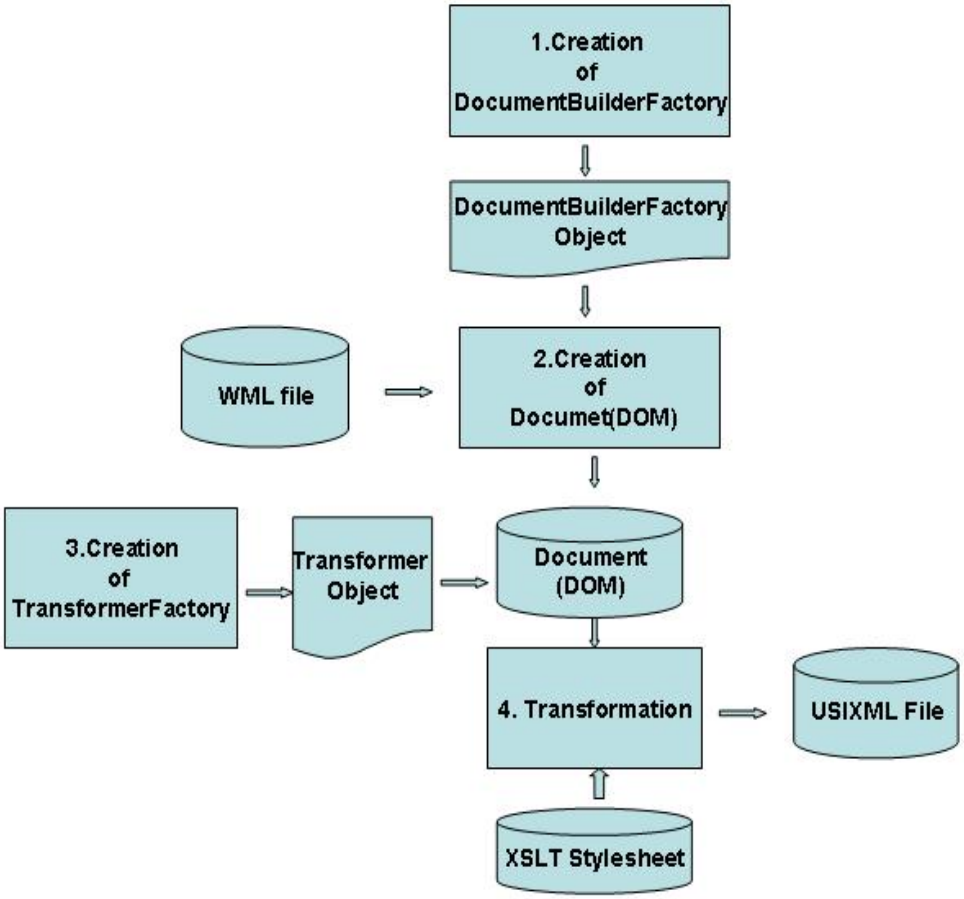


Fig. 5 Transforming Procedure

To transform the original WML file to USIXML file, we have implemented a transformation application using Java programming language. The development environment is Java 2 SDK V1.4. The Fig. 5 presents the transformation procedure which involves four steps:

1. Using the factory class "DocumentBuilderFactory" defined in the javax.xml.parsers package to instantiate a DocumentBuilder object.
2. Parsing the WML source document into a document object model (DOM), where it is represented as a tree where each node in the tree is a tag in the original WML.
3. Using the factory class "TransformerFactory" defined in javax.xml.transform package to instantiate a transformer object. The transformer is created from a set of transformation instructions. The set of transformation instructions is defined by the

XSLT Style sheet which indicates in which case the specified transformations are carried.

4. Transforming the Dom Document by the transformer. These can either insert new tags into the DOM tree or remove some nodes altogether. Positions of elements to be processed from the DOM tree are specified in terms of their XPATH. The XPATH of a node is a unique path to that node in the DOM tree.

3.4.2 Java application source code

```
// import the JAXP APIs:
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamSource;
import javax.xml.transform.stream.StreamResult;

// for the exceptions that can be thrown when the XML document is parsed:
import org.xml.sax.SAXException;

// import the W3C definition for a DOM:
import org.w3c.dom.Document;
// for reading and writing:
import java.io.*;
public class transformer {
      // Global value so it can be used by the tree-adapter
       static Document document;
       public static void main (String argv [])
       {
            if (argv.length != 3) {
                 System.err.println ("Usage: java transformer stylesheet sourceFileName
targetFileName");
                 System.exit (1);
            }
            // to obtain an instance of a factory that can give us a document builder:
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            //factory.setNamespaceAware(true);
            //factory.setValidating(true);

            try {
              File stylesheet = new File(argv[0]);
                 File datafile = new File(argv[1]);
                 File transformedfile= new File(argv[2]);
                 DocumentBuilder builder = factory.newDocumentBuilder();
                 // Parsing the WML source document into a DOM
                 document = builder.parse(datafile);
                 // Using a Transformer for output
```

```
                TransformerFactory tFactory = TransformerFactory.newInstance();
                StreamSource stylesource = new StreamSource(stylesheet);
                // to instantiate a transformer object
                Transformer transformer = tFactory.newTransformer(stylesource);
                DOMSource source = new DOMSource(document);
                StreamResult result = new StreamResult(argv[2]);
                //Transforming the Dom Document by the transformer
                transformer.transform(source, result);

            } catch (TransformerConfigurationException tce) {
                // Error generated by the parser
                System.out.println ("\n** Transformer Factory error");
                System.out.println("      " + tce.getMessage() );
                Throwable x = tce;
                if (tce.getException() != null)
                        x = tce.getException();
                x.printStackTrace();
            } catch (TransformerException te) {
                // Error generated by the parser
                System.out.println ("\n** Transformation error");
                System.out.println("      " + te.getMessage() );
                Throwable x = te;
                if (te.getException() != null)
                        x = te.getException();
                x.printStackTrace();

             } catch (SAXException sxe) {
                // Error generated by this application
                Exception    x = sxe;
                if (sxe.getException() != null)
                        x = sxe.getException();
                x.printStackTrace();
            } catch (ParserConfigurationException pce) {
                    // Parser with specified options can't be built
                    pce.printStackTrace();

            } catch (IOException ioe) {
                // I/O error
                ioe.printStackTrace();
            }
        }
    }
```

3.4.3 Usage of Java application

To execute the java Transformation Application which is named as transformer.java, suppose you've installed the Java 2 Software Development Kit which can be free downloaded from the www.java.sun.com Web site and connected to internet, you can just input the command line, shown in Fig. 6, under the directory which includes transformer.java, the XSLT style sheet file and the source WML file. The result target USIXML file will be created under the same directory. For the XSLT style sheet, we will

present it in the chapter 6, and you can find the complete XSLT style sheet source code in the appendix 4.

Fig. 6 The Execution of Transformation App. (1)

Another option instead of DOS to run the Java program is to use a Java develop tool, for more platforms other than windows, such as Eclipse which can be free downloaded from www.eclipse.com. The following picture shows you where to input the arguments.

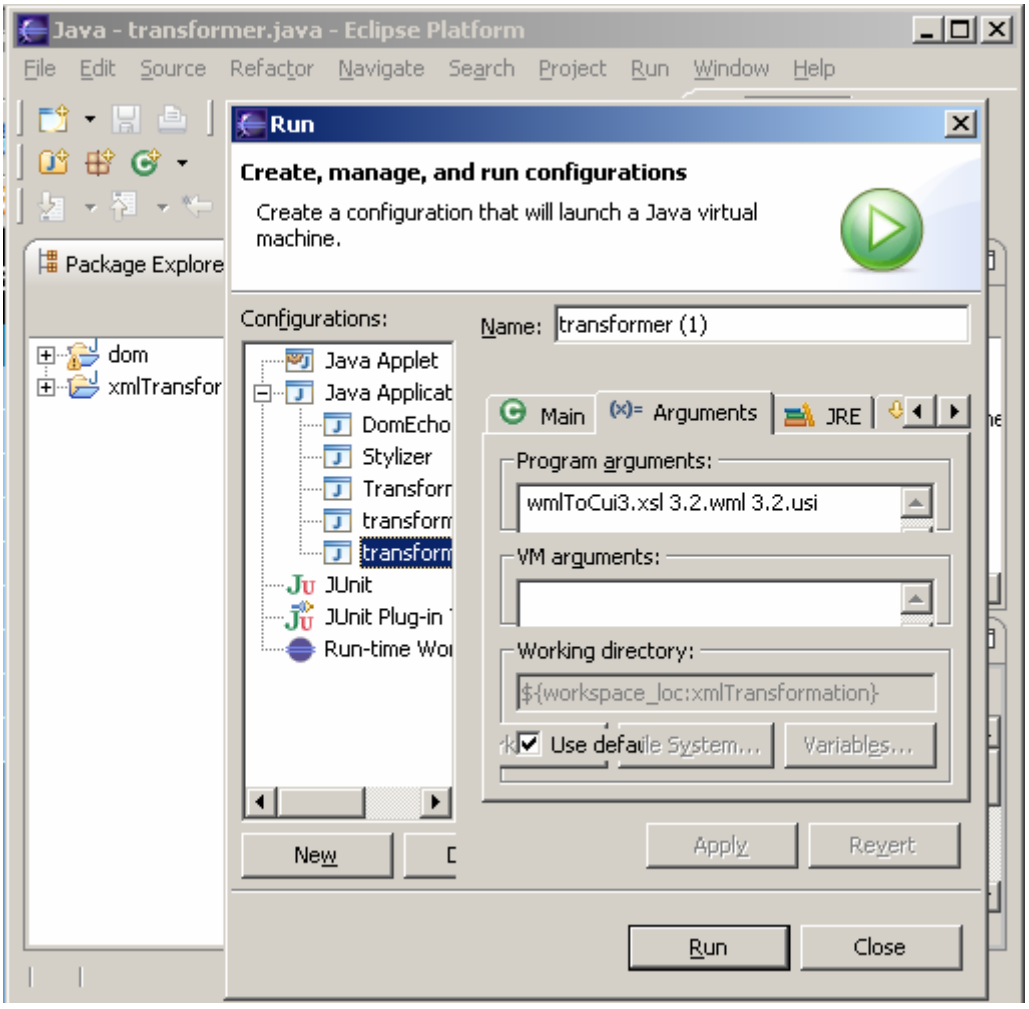


Fig. 7 The Execution of Transformation App. (2)

Chapter 4. INTRODUCTION TO WML 1.1

As our goal is to reverse engineering the existing design for WML UIs, we dedicate this chapter to introduce WML. We firstly choose the ideal WML version as the source for the reverse engineering. Then, we will explore the structure of WML for the chosen version by showing the meta-model of WML to explain the inter-relationship between WML elements. And further more, we will present each element and its attributes related for the transformation by showing the detailed specification for each element on which we perform the transformation.

4.1 Introduction to WML

The contents of the WAP services are implemented in Wireless Markup Language (WML) and WMLScript. WML is an XML-based markup language designed for low-end devices and slow, unreliable networks. WML provides basic means for document formatting and user interaction but presupposes little of how they are actually implemented. Developers of WAP services only design the interaction logic in the application. Each client device then implements these interactions in its own way.

Fig.8 Three popular WAP sites with WML 1.1

The Fig.8 shows the three most demanding WAP sites are all developed by WML 1.1. Although the current WML standard is WML 1.3, the most WML resource is written in

WML 1.1. However, today there are very few mobile sites adapting WML 1.3-specific features.

The WML 2.0 document type extends XHTML Mobile Profile is a compact core module, which is to be supported by most of the browsers the WML documents written using earlier WML versions (WML 1.x) can be transformed into WML 2 format. The WML 2.0 is a document type with an XHTML core and WAP extensions. The core document type, known as XHTML Mobile Profile, can be used to author content convergent with W3C specifications. The WML2 document type (XHTML core plus WAP extensions) can be used to deliver WML1 content to WAP 2 clients, achieving backward compatibility with WML1 in a manner that is transparent to the end user. The structure and relationship of the document types allows an efficient implementation of a user agent that supports both types.[22]

The principle differences between WML1.1 and WML 2.0 are explained as following:

- While WML1.1 has noting to do with XHTML, WML 2.0 defers to XHTML in the case of duplicated semantics (elements, attributes, and attribute values). (eg.., optgroup, table)
- WML 2.0 includes some XHTML elements and attributes which WML 1.1 don't has (eg.., textarea, lable, h1).
- WML 2.0 removes WML elements, attributes, and attribute values when they can be expressed in XHTML and CSS (eg.., wml, template) .
- WML 2.0 includes WML1 elements and attributes when WML1 features cannot be expressed in XHTML and CSS. These elements are included using the WML namespace, identified by the "wml:" prefix (eg.., wml:card)

The new WML 2.0 user interface has been introduced to support the forthcoming WAP 2.0 compliant mobile phones. But the time of writing this thesis, there are few WAP 2.0 devices and services on the market. There are no guidelines either for developing usable mobile services with WML 2.0 today. As a result, many mobile devices support only the WML 1.1 standard. So we choose the UIs of WML 1.1 as the source UIs for the reverse engineering.

4.2 Meta-model of WML 1.1

In the last section, we have decided to choose the WML1.1 as the source WAP UIs development language for the further transformation. In this section, we present the

meta-model of WML 1.1 to show the structure of WML and explore the inter-relationship between WML elements. For constructing the meta-model of WML 1.1, we use UML class diagram which is implemented by Rational Rose. There are several remarks for this meta-model. Firstly, the normal classes present the elements of WML 1.1. Secondly, the classes begin with "%" are the entities defined in the WML 1.1 Document Type Definition (DTD) (see Appendix 1). Thirdly, for avoiding too much inter-relationships in order to improve the readability of the meta-model, we invent some classes which begin with "+" to aggregate some elements with similar nature. The Fig.9 below shows the meta-model of WML 1.1

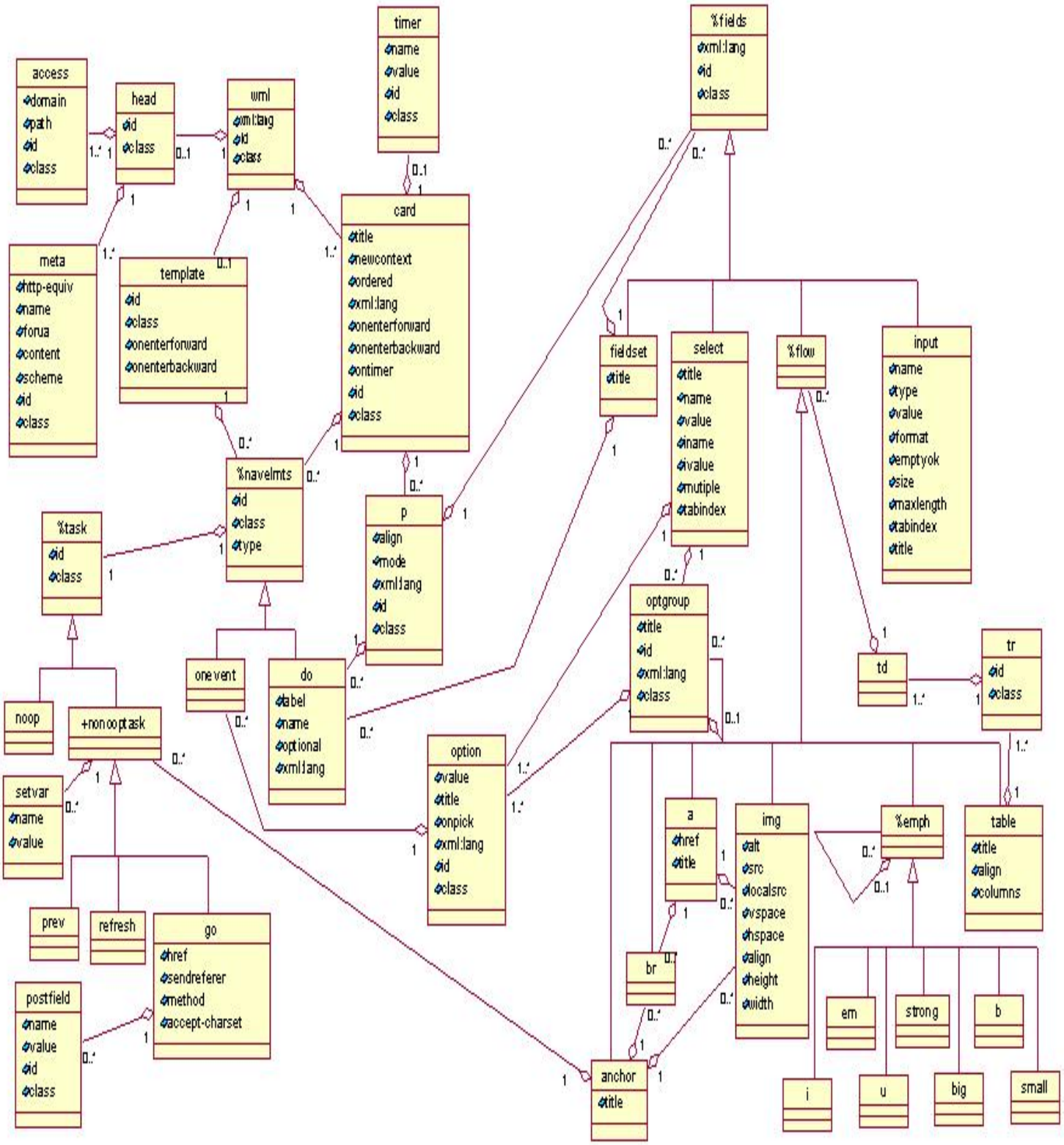


Fig. 9 Meta-model of WML1.1

Having the meta-model shown in the Fig.9, we can further conclude its four major functional areas. Firstly, it is Deck/card organisational metaphor, in other words, all information in WML is organised into a collection of cards and decks. Cards specify one or more units of user interaction (e.g., a choice menu, a screen of text or a text entry field). Logically, a user navigates through a series of WML cards, reviews the contents of each, enters requested information, makes choices and moves on to another card. Cards are grouped together into decks. A WML deck is similar to an HTML page, in that it is identified by a URL. Secondly, WML includes text and image support, including a variety of formatting and layout commands. For example, boldfaced text may be specified. Thirdly, WML includes support for explicitly managing the navigation between cards and decks. WML also includes provisions for event handling in the device, which may be used for navigation. Lastly, all WML decks can be parameterised using a state model. Variables can be used in the place of strings and are substituted at run-time. This parameterisation allows for more efficient use of network resources.

There are some WML 1.1 elements, such as prev, refresh, template, and timer etc., for which the directly mapping to the CUI Model of USIXML is not currently available, in other word, there are not such corresponding elements exist in USIXML. As a result, we can not map all the WML 1.1 elements to the USIXML. The following figure will show the potion of the meta-model which can be mapped to USIXML at the time of this thesis.

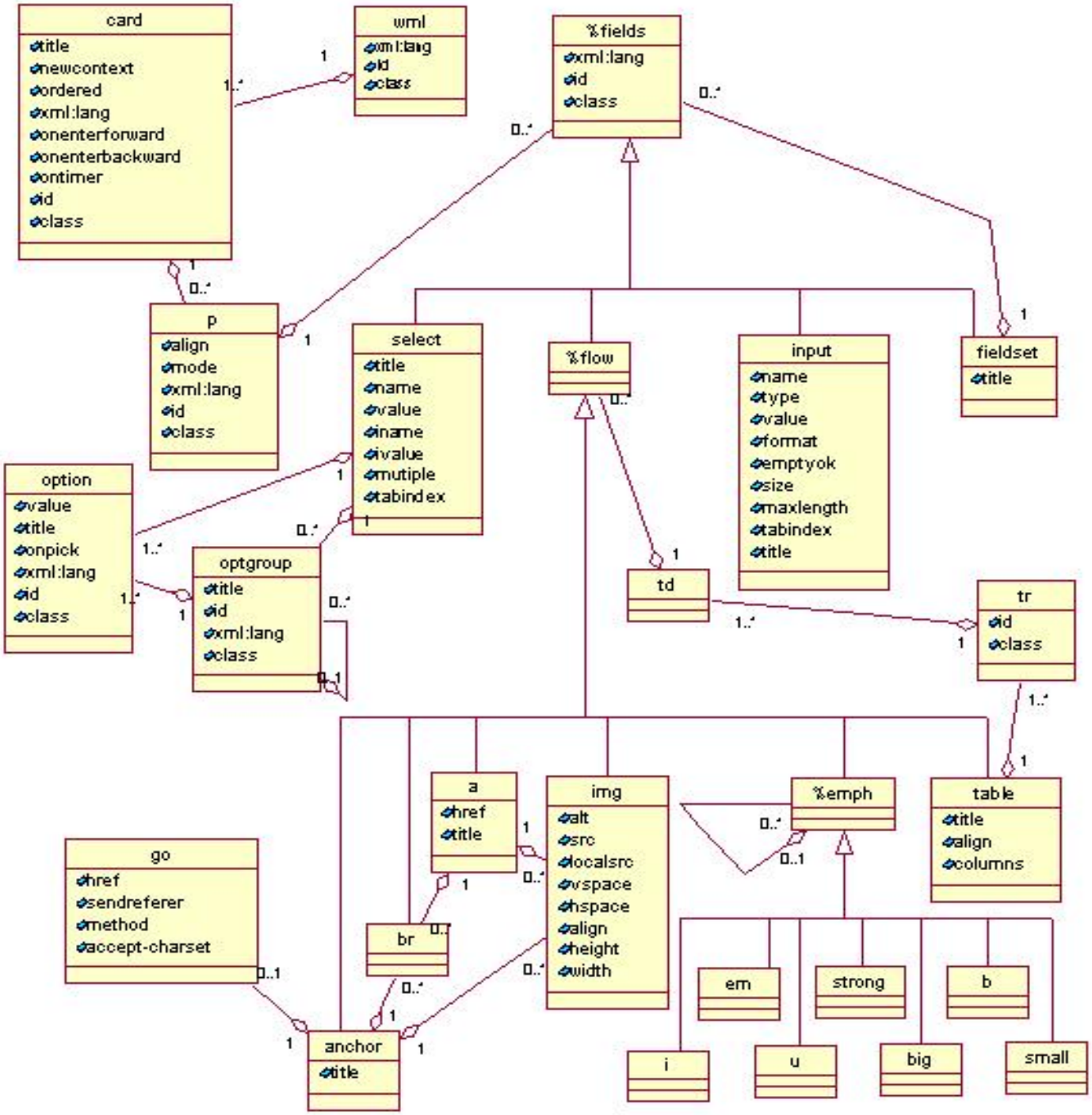


Fig. 10 Potion of Interest of WML 1.1

The Fig.10 above shows the potion of interest of the WML 1.1 meta-model which can be mapped to the CUI Model of USIXML (USer Interface eXtensible Markup Language). In the next section, we will give the specification for all the elements in this potion of the Meta-model.

4.3 Specification of elements and attributes of WML

In the former section, we have presented the meta-model of WML. In this section, we will show the detailed specification for each of the element in the portion of interest meta-model of WML 1.1 as well as its attributes. All these elements and its corresponding attributes are related for the later transformation.

This specification is referred to the Wireless Markup Language (WML) 1.1 Document Type Definition (DTD) (see Appendix 1) and the official WML specification developed and maintained by the WAP Forum.

4.3.1 Card Element

ELEMENT card

Specification: The card element is a container of text and input elements that is sufficiently flexible to allow presentation and layout in a wide variety of devices, with a wide variety of display and input characteristics. Its attribute title specifies advisory information about the card. The title may be rendered in a variety of ways by the user agent.

4.3.2 User Inputs: select, optgroup, option and input

ELEMENT select

Specification: The select element lets users pick from a list of options. The title attribute specifies a title for this element, which may be used in the presentation of this object. The ivalue attribute indicates the default-selected option element. The multiple attribute indicates that the select list should accept multiple selections. When not set, the select list

should only accept a single selected option.

ELEMENT optgroup

Specification: The optgroup element allows the author to group related option elements into a hierarchy. The title attribute specifies a title for this element, which may be used in the presentation of this object.

ELEMENT option

Specification: This element specifies a single choice option in a select element. The title attribute specifies a title for this element, which may be used in the presentation of this object. The onpick attribute occurs when the user selects or deselects this option. A multiple-selection option list generates an onpick event whenever the user selects or deselects this option. A single-selection option list generates an onpick event when the user selects this option.

```
<wml>
<card>
<p align="center">Select Boxes
<select>

<option onpick="link1.wml">Selection1</option>

<option onpick="link2.wml">Selection2</option>

<option onpick="link3.wml">Selection3</option>
</select>
```

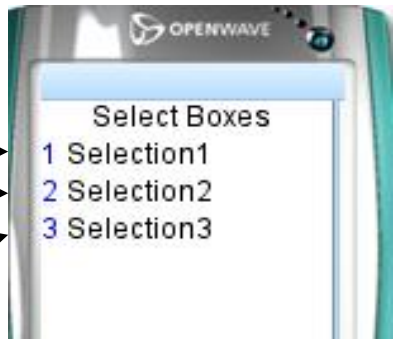


Fig. 11 Example for “selection”

Fig.12 "Real World" Example for "selection"

ELEMENT input

Specification: The input element specifies a text entry object. The type attribute specifies the type of text-input area. ("text", "password"); The value attribute indicates the default value of the variable named in the name; The size attribute specifies the width, in characters, of the text-input area. The maxlength attribute specifies the maximum number of characters that can be entered by the user in the text-entry area; The title attribute specifies a title for this element, which may be used in the presentation of this object.

```
<wml>
<card>
<p align="center">Input Box

<input name="Name"
value="Template"/>

</p>
</card>
```

Fig.13 Example for “input”

Fig.14 “Real World” Example for “input”

From Fig.14, we can see for input field the different devices have very different presentations.

ELEMENT fieldset

Specification: The fieldset element allows the grouping of related fields and text.

Remarks: The title attribute specifies a title for this element, which may be used in the presentation of this object.

4.3.3 Text presentation and layout

ELEMENT table

Specification: The table element is used together with the tr and td elements to create sets of aligned columns of text and images in a card. The title attribute specifies a title for this element, which may be used in the presentation of this object. The align attribute

specifies the layout of text and images within the columns of a row set. The columns attribute specifies the number of columns for the row set.

ELEMENT tr

Specification: The tr element is used as a container to hold a single table row. Table rows may be empty . Empty table rows are significant and must not be ignored.

ELEMENT td

Specification: The td element is used as a container to hold a single table cell data within a table row.

```
<wml>
<card>
<p align="center">Table
<table title="Title" columns="3">
    <tr>
        <td>L1C1</td>
        <td>L1C2</td>
        <td>L1C3</td>
    </tr>
    <tr>
        <td>L2C1</td>
        <td>L2C2</td>
        <td>L2C3</td>
    </tr>
</table>
```

Fig.15 Example for “table, tr, td”

Fig.16 "Real world" Example for "table, tr, td"

The Fig.16 shows different ways of visualising table. The two devices are Nokia 9210 and Ericsson R380. The original table does not have visible table borders, but the Nokia 9210 uses them. The Ericsson R380 opts to drop the borders off in every table.

ELEMENT strong

Specification: Render with emphasis.

ELEMENT b

Specification: Render with a bold font.

ELEMENT i & em

Specification: Render with an italic font.

ELEMENT u

Specification: Render with underline.

ELEMENT big

Specification: Render with a large font.

ELEMENT small

Specification: Render with a small font.

```
<wml>
<card>

<p>normal<em>em</em><strong>strong</strong></p>

<p>normal <b>b</b> <i>i</i> <u>u</u></p>

<p>normal<big>big</big><small>small</small></p>
```

Fig.17 Example for Inline Layout elements

From Fig.17 above, we can see all the presentation of the inline layout elements.

ELEMENT p

Specification: The p element establishes both the line wrap and alignment parameters for a paragraph. If the text alignment is not specified, it defaults to left. If the line-wrap mode is not specified, it is identical to the line-wrap mode of the previous paragraph in the current card. The align attribute defines horizontal alignment of content. The default value is "left".

```
<wml>
<card>

<p align="left">First paragraph</p>

<p align="center">Second paragraph</p>

<p align="right">Third paragraph</p>
</card>
</wml>
```

Fig.18 Example for "p" element

From Fig. 18 above, we can see the different horizontal alignments for the p element.

Fig.19 "Real world" Example for "p" element

From Fig.19, we can realize that the device with bigger screen can show more lines and more words per line for the element p.

4.3.4 Images, Anchors

ELEMENT img

Specification: The img element indicates that an image is to be included in the text flow. Image layout is done within the context of normal text layout. The attribute src specifies the URI for the image; The vspace attribute specify the amount of white space to be inserted to the above and below an image or object. The hspace attribute specify the amount of white space to be inserted to the left and right an image or object; The align

attribute specifies image alignment within the text flow and with respect to the current insertion point; The height attribute give user agents an idea of the height of an image or object so that they may reserve space for it and continue rendering the card while waiting for the image data. User agents may scale objects and images to match these values if appropriate; The width attribute give user agents an idea of the width of an image or object so that they may reserve space for it and continue rendering the card while waiting for the image data.

```
<wml>
<card id="splash"
title="Hollywood.com">
<p align="center">
<small><b>Welcome To</b></small>
</p>
<p align="center">

<img src="images\logo_holly.gif"
alt="Hollywood.com" align="middle"
/>

</p>
</card>
</wml>
```

Fig.20 "Real world" Example for "img" element

From Fig.20 above, we can see the M3Gate simulator can not show the image as openwave emulator doing. As instead of, it show an alternative text content.

ELEMENT anchor

Specification: The anchor element specifies the head of a link. The title attribute specifies a brief text string identifying the link.

ELEMENT a

Specification: Element “a” provides basic hypertext linking capabilities. You can make conections between individual cards or WML decks. All WML elements can contain two core attributes, id and class. Attribute id in “card” element serves as anchor for intercard links. The href attribute specifies the destination URI. The title attribute specifies a brief text string identifying the link.

```
<wml>
<card id='card_aaa'>
<p>This is card <strong>aaa</strong></p>
<p>Go to <a href='#card_bbb'>bbb</a></p>
<p>Go to <a href='/hyperLink'>target
address</a></p>
</card>

<card id='card_bbb'>
<p>This is card <strong>bbb</strong></p>
<p>Go to <a href='#card_aaa'>aaa</a></p>
</card> Forever Young
</wml>
```

Fig.21 Example for “a”

Fig.22 "Real world" Example for "a" element

From Fig.22 above, we can see for the a element these two devices have the same appearance.

4.4 A WML E-mail Application

Web mail accounts such as the ubiquitous 'Hotmail', or 'Yahoo! Mail' are fast becoming the most popular kind of e-mail. There are almost 170 million subscribers to this type of account. Web-based mail allows users convenient web access to their mail account without any messy machine configuration issues. The WML E-mail application on the yahoo WAP site - http://home.mobile.yahoo.com implements a WAP mail system, allowing access to an SMTP/POP3-based e-mail account.

This WAP e-mail application conclude services such as:

- Compose, send and reply to mail via an SMTP server
- View an inbox and read mail using a POP3 service
- Determine the number of messages waiting in the inbox

- Delete mail, etc.

There are some screen shots to show the above services by this application as following:

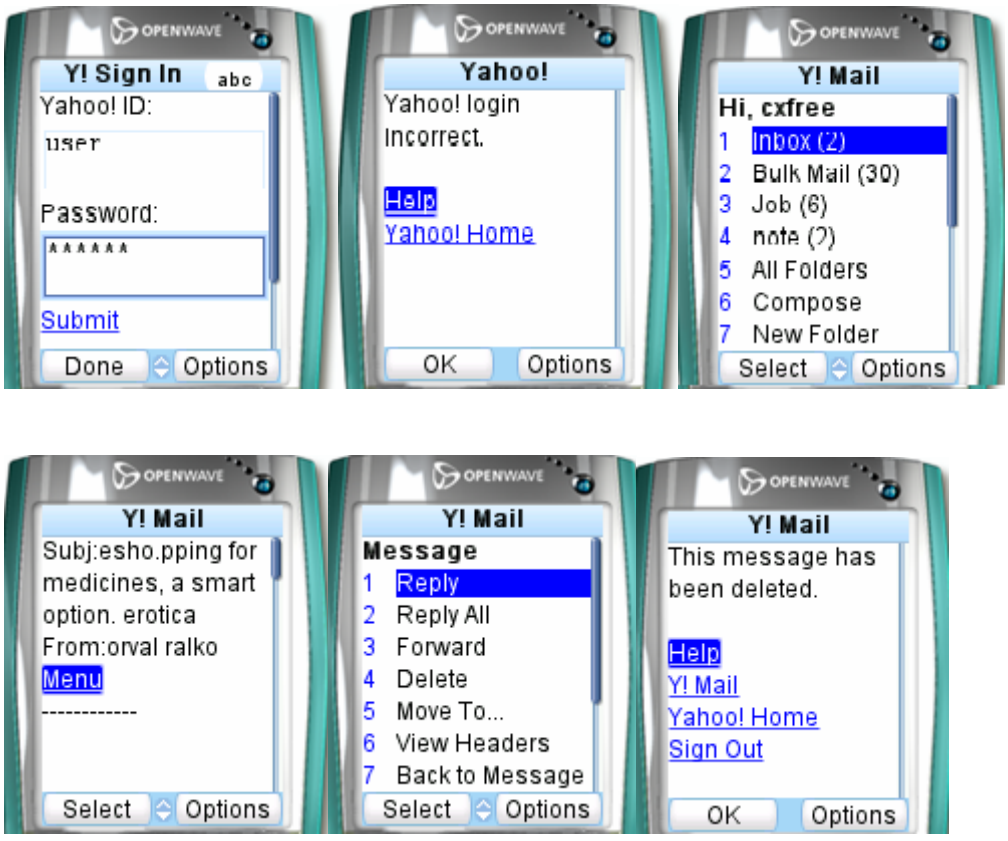


Fig.23 WML1.1 WAP E-mail Application

4.5 Conclusion

In this chapter, we have presented the detailed specifications about the elements of interest of the WML 1.1 for the mapping to CUI elements. To illustrate WML's element, we have presented some examples to show the different appearances which are not only caused by different attribute but also by different devices. We also present a "real world" WAP application using WML 1.1 to show the feel and look of WML UIs.

Chapter 5. THE TARGET CUI MODEL OF USIXML

In the chapter 2, we we've chosen CUI model as the target model of the UI Reverse Engineering for the WML UIs. And in the last chapter, we've introduce the WML elements which could be mapped to that of CUI model. Before we define the mapping rules in the next chapter, in this chapter, we will present the components and the detailed specification for the mapped elements of CUI Model.

5.1 Introduction

A CUI is a UI model allowing a specification of an appearance and behaviour of a UI with elements that can be perceived by users. A CUI consists of:

1. Modality dependent i.e., an instance of a CUI addresses a single modality at a time. Two modalities lie in the intended scope of USIXML: graphical and auditory.

2. Platform independent i.e., elements populating a CUI realize an abstraction of common languages used to program UIs.

```
<cuiModel id="f-cui_19" name="f-cui">
        <window id="window_1" name="Main window"
            isVisible="true" width="400"
            height="350" windowLeftMargin="0" windowTopMargin="0">
            <box id="box_1" name="box_1" isVisible="true">
                <button id="button_component_0"
                    name="button_component_0" isVisible="true"
                    isEnabled="true" textColor="#000000"/>
                <textComponent id="text_component_3"
                    name="text_component_3" isVisible="true"
                    isEnabled="true" textColor="#000000" maxLength="50"
                    numberOfColumns="15" isEditable="true"
                    defaultContent="HELLO"/>
            </box>
        </window>
</cuiModel>
```

Fig. 24 Example for CUI Model

Fig. 24 shows a simple declaration of a window containing a top-centred label and a

button.

Before we define the mapping rules from WML to CUI Model level which define widgets layout and interface navigation, we present the components of CUI model and the specification of elements which is related to the mapping.

5.2 Specifications for CUI elements

All the following specification for CUI elements refers to the USIXML V1.4.5 documentation downloaded from the official USIXML site www.usixml.org. For some elements we present some USIXML code to illustrate them.

5.2.1 Upper- level elements

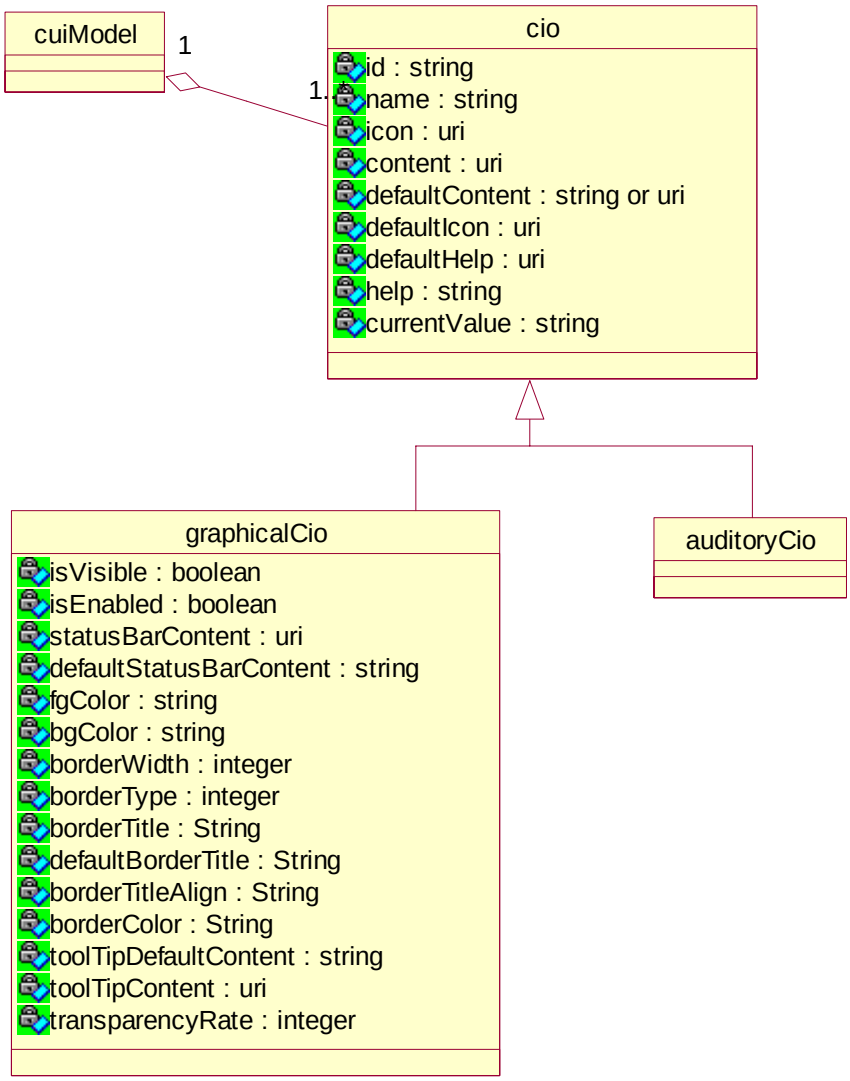


Fig.25 upper-level elements

Element cio

Specification: Is an entity of the UI that users can perceive (e.g., text, image, animation) and/or manipulate (e.g., a push button, a list box, a check box). A widget provided by a toolkit, physical interactor and physical interaction object.

Attributes:

- Id: Identifies a cio. An id is internally attributed to a cio and is, consequently, not supposed to reflect its cio type, or content,...

- Name: Is a name given to a cio. A name gives a first insight on a cio's type, function or content.
- Icon: Is a context dependent icon associated with a cio.
- Content: Is text content associated with any cio. All cios content are defined in a remote file to allow a run-time language content selection to work easily.
- defaultContent: Is a default caption for any cio.
- defaultIcon: Is a default icon for any cio.
- defaultHelp: Is the default help for any cio.
- Help: Is the help for any cio.
- currentValue

Element graphicalCio

Inherits from: cio

Specification: Is an element composing a graphical user interface. It may be either a container or an individual component.

Attributes:

- isVisable: Is set to true if a graphicalCio is visible.
- isEnable: Is set to true if a graphicalCIO is enabled.
- statusBarContent: Is the status bar content of a graphical CIO.
- isDefaultStatusBarContent: Is the default status bar content of a graphical CIO.
- fgColor: Is a graphicalCio foreground color expressed with html color codes.
- BgColor: Is a graphicalCio backgroung color expressed with html color codes.
- borderWidth: Is the width of the border.
- borderType: Is the type of the border.
- borderTitle: Is the title of the border(i.e., a label appearing in the border of this component).
- defaultBorderTitle: Is the context dependent border title.
- borderTitleAlign: Indicates the alignment of the title of the border. Allowed values: left, middle, right.

- BorderColor: Is the color of the border. Expressed with html color codes.
- defaultToolTipContent: Is the default tooltip associated with a CIO.
- toolTipContent: Is the context dependent tooltip associated with a CIO.
- transparencyRate: Is the transparency rate of a CIO. Is expressed in percent.

5.2.2 GraphicalContainer elements

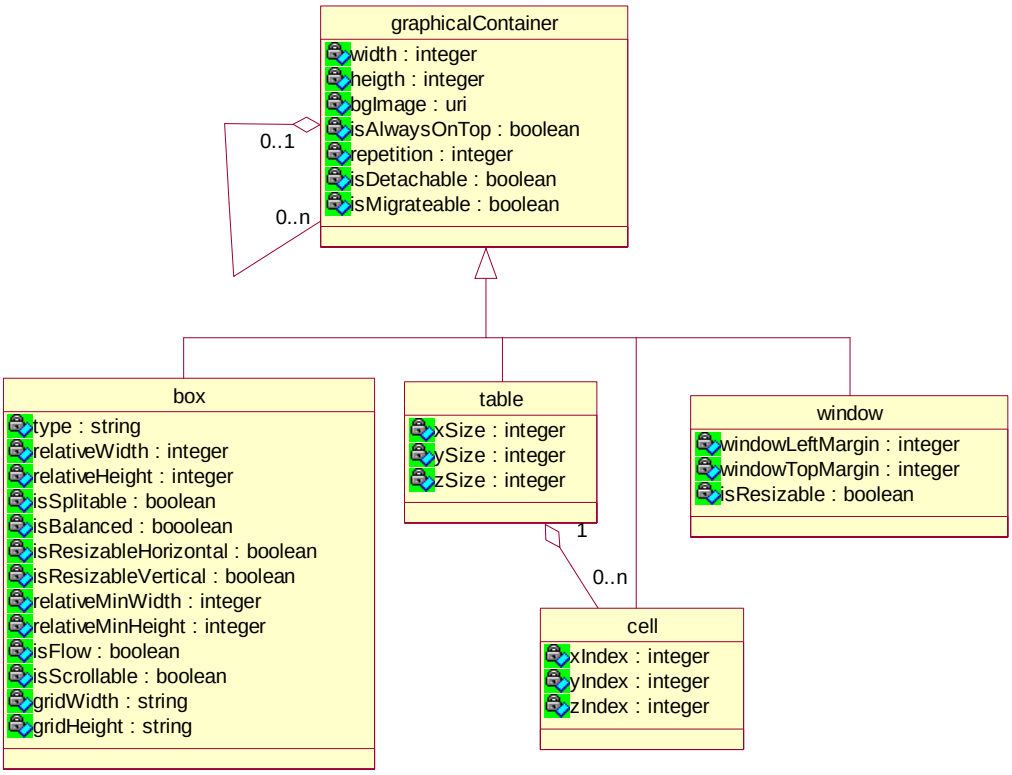


Fig.26 Portion of interest graphicalContainer elements

Element graphicalContainer

Fig.26 shows the graphicalContainer element and its portion of interest children elements which can be mapped from WML 1.1.

Inherits from: graphicalCio

Specification: Contains a collection of cio's (either graphicalIndividualComponents or graphicalContainers) that support the execution of a set of logically/semantically

connected tasks.

Attributes:

- width: Is the width of the graphicalContainer.
- Height: Is the height of the graphicalContainer.
- bgImage: Is the background image of the graphicalContainer.
- isAlwaysOnTop: Is true if is always on the top.
- Repetition: Indicates how many times a graphicalContainer is repeated in the specification.
- isDetachable: Indicates if a container may be detached or not from its graphicalContainer. This attribute may help for multi-surface distributed interfaces.
- IsMigrateable: Indicates is the container can pass from one platform to another.

Element window

Inherits from: graphicalContainer

Specification: Is a window

Attributes:

- windowLeftMargin: Indicates a left margin size in pixel.
- windowRightMargin: Indicates a right margin size in pixel.
- isResizable: Specifies if a window is resizable or not. Default : true.

Element box

Inherits from: graphicalContainer

Specification: Is a containers that enables an unambiguous structuring of graphicalIndividualComponents within a window, a tabbedItem, a dialogBox. Boxes are embedded one into each other. They may be of type main (the topmost box in a container), horizontal, or vertical.

Attributes:

- type: Equals horizontal, vertical, horizontalGrid or verticalGrid.
- relativeWidth: Expresses in percent the relative width of a box container.
- relativeHeight: Expresses in percent the relative height of a box container.

- isSplitable: Indicates if a box is splitable or not. This information is notably used during adaptation heuristics to reshuffle containers and potentially redistribute boxes between several abstract containers.
- isBalanced: Indicates that all graphicalIndividualComponents are topologically balanced within a box.
- isResizableHorizontal: indicates if a box is horizontally resizable.
- isResizableVertical: indicates if a box is vertically resizable.
- relativeMinWidth: Indicates a minimal width in percentage of the initial size of a box.
- relativeMinHeight: Indicates a minimal height in percentage of the initial size of a box.
- isFlow: Indicates if the layout algorithm should create a new line in the box if all components belonging to one box can not be displayed in one line.
- isScrollable: Indicates if a box may be equipped with a scrollbar if its content cannot be displayed on its surface.
- gridWidth: Indicates the width (in absolute number) of the box in case it is of type grid.
- gridHeight: Indicates the width (in absolute number) of the box in case it is of type grid.

Element table

Inherits from: graphicalContainer

Specification: Is composed of cells. A table is characterized by its size. It may be uni,bi,tri-dimensionnal.

Attributes:

- xSize: Designates a number of lines of a table.
- ySize: Designates a number of column of a table.
- zSize: Designates a number of layer of a table.

Element cell

Inherits from: graphicalContainer

Specification: Composes a table (and may contain a table). It is a container in itself but it must always be part of a table.

Attributes:

- xIndex: Designates a line number.
- yIndex: Designates a column number.
- zIndex: Designates a layer number.

```
<CuiModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<window id="window_1" isVisible="true" isEnable="true" windowLeftMargin="0"
windowTopMargin="0" name="window">
<box id="box_1" borderTitle="Yahoo!" isVisible="true" isEnable="true">
<table id="table_1" ySize="2" xSize="2" isVisible="true" isEnable="true" name="table_example"
glueHorizontal="center">
<cell id="cell_1" xIndex="1" yIndex="1" isVisible="true" isEnable="true">
<textComponent id="textComponent_2" glueHorizental="left" isVisible="true" isEnable="true"
defaultContent="L1C1"/>
</cell>
<cell id="cell_2" xIndex="1" yIndex="2" isVisible="true" isEnable="true">
<textComponent id="textComponent_3" glueHorizental="left" isVisible="true" isEnable="true"
defaultContent="L1C2"/>
</cell>
<cell id="cell_3" xIndex="2" yIndex="1" isVisible="true" isEnable="true">
<textComponent id="textComponent_4" glueHorizental="left" isVisible="true" isEnable="true"
defaultContent="L2C1"/>
</cell>
<cell id="cell_4" xIndex="2" yIndex="2" isVisible="true" isEnable="true">
<textComponent id="textComponent_5" glueHorizental="left" isVisible="true" isEnable="true"
defaultContent="L2C2"/>
</cell>
</table>
</box>
</window>
</CuiModel>
```

Fig. 27 Example for element “table & cell” of CUI Model

The Fig.27 shows the code of CUI Model for table and cell elements. The table has two lines and two columns. The position of each cell is marked by the content of the cell, for instance, L1C1 present the data in the first line and first column.

5.2.3 GraphicalIndividualComponent elements

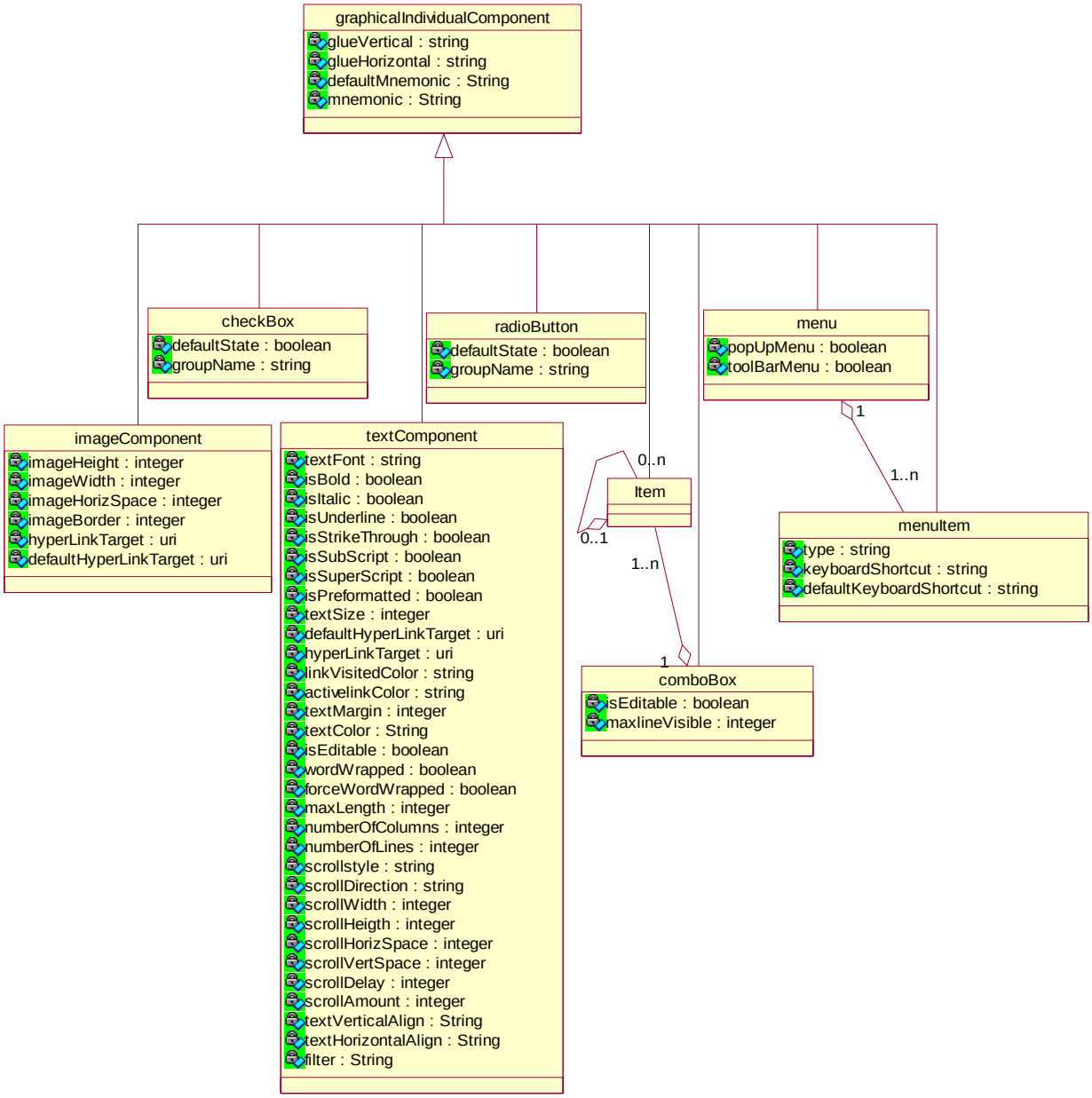


Fig.28 Portion of interest of graphicalIndividualComponent elements

Element graphicalIndividualComponent

Fig.28 shows the graphicalIndividualComponent element and its portion of interest children elements which can be mapped from WML 1.1.

Inherits from: graphicalCio

Specification: Is a GraphicalCio contained in a graphicalContainer. Its specific attributes

(offsetVertical and offsetHorizintal) enable to specify its layout relationship with its graphicalContainer.

Attributes:

- glueVertical: Is the specification of a layout constraint (on a vertical axis) between a graphicalIndivudualComponent and its graphicalContainer. Allowed values : top, middle, bottom.
- glueHorizontal: Is the specification of a layout constraint (on a horizontal axis) between a graphicalIndivudualComponent and its graphicalContainer. Allowed values : left, middle, right.
- defaultMnemonic: Is the default mnemonic for a control graphical individual component.
- Mnemonic: Is the mnemonic depending on the context (especially the language).

Element textComponent

Inherits from: graphicalIndividualComponent

Specification: Is a graphicalIndividualComponent specialized for handling textual content.

Attributes:

- textFont: Specifies a font style for the textComponent.
- isBold: Specifies if a textComponent is bold or not.
- isItalic: Specifies if a textComponent is italic or not.
- isUnderline: Specifies if a textComponent is underlined or not.
- isStrikeThrough: Specifies if a textComponent is striked through or not.
- isSubScript: Specifies if a textComponent is subscripted or not.
- isSuperScript: Specifies if a textComponent is superscripted or not.
- isPreformatted: If is set to true no style attribute can be modified from outside a specification.
- textSize: Specifies a size in points for a textComponent.
- defaultHyperLink: Is the default hyperLink.
- defaultHyperLinkTarget: Designates a hyperlink target file.

- linkVisitedColor: Is a hyperlink color after being visited one time.
- activeLinkColor: Is a color of an hyperLink textComponent when clicking on it.
- textMargin: Specifies a textmargin size in pixels.
- textColor: Specifies a text color.
- isEditable: Specifies if a textComponent is editable or not i.e., subject to user input or not.
- wordWrapped: Indicates if a text is wrapped or not.
- forceWordWrapped: Is a graphicalIndividualComponent specialized for handling textual content.
- maxLength: Is a maximum length for a content of a textComponent. Expressed in number of characters.
- numberOfColumns: Is a number of columns of a textComponent.
- numberOfLines: Is a number of lines of a textComponent.
- scrollStyle: Is the style of the scroll. Allowed values : scroll, slide, alternate.
- scrollDirection: Is the scroll direction. Alllowed values : left, right.
- scrollWidth: Indicates a width in pixel of a scrolling textComponent.
- scrollHeight: Indicates a height in pixel of a scrolling textComponent.
- scrollHorizSpace: Indicates a blank space in pixel at the right and the left of a scrolling textComponent.
- scrollVertSpace: Indicates a blank space in pixel left, bellow and above a scrolling textComponent.
- scrollDelay: Expresses a scrolling delay in millisecond.
- scrollAmount: Expresses in pixel scrolled distance after each delay.
- textVerticalAlign: Indicates a vertical alignment constraint of the text contained in a textComponent. Allowed values : top, middle, bottom.
- textHorizontalAlign: Indicates an horizontal alignment constraint of the text contained in a textComponent. Allowed values: left, middle, right.
- Filter: Is a regular expression constraining the content of a textComponent.

Elelment imagComponent

Inherits from: graphicalIndividualComponent

Specification: Is a graphicalIndividualComponent specialized for handling image content.

Attributes:

- imageHeight: Is the heigth of an ImageComponent expressed in pixels.
- imageWidth: Is the width of an ImageComponent expressed in pixels.
- imageHorizSpace: Expresses an horizontal offset (in pixels) with respect to an imageComponent's container.
- imageBorder: Expresses a border width in pixels.
- defaultHyperLinkTarget: Specifies a target file reachable from an imageComponent (depends of the context).
- defaultDefaultHyperLinkTarget: Is the default hyperLink.

Element radioButton

Inherits from: graficalIndividualContainer

Specification: Enables a Boolean choice by checking a circle aside of a label. A radioButton may be differentiated from a checkBox by the fact that when grouped optionButton selection is mutually exclusive while checkBox allows multiple choices.

Attributes:

- defaultState: Is true if a radioButton is selected.
- groupName: Is the name of the group.

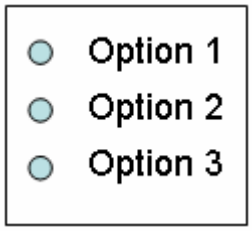


Fig.29 Appearance of radioButton

```
<CuiModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<window id="window_1" isVisible="true" isEnable="true" windowLeftMargin="0"
windowTopMargin="0" name="Wordaholic">
<box id="box_1" borderTitle="Wordaholic" isVisible="true" isEnable="true">
<textComponent id="textComponent_1" glueHorizental="center" isVisible="true"
isEnable="true" isBold="true" defaultContent="Wordaholic(tm)"/>
<radioButton id="radioButton_1" glueHorizental="left" defautContent="First
Selection" isVisible="true" isEnable="true" defaultState="false"/>
<radioButton id="radioButton_2" glueHorizental="left" defautContent="Second
Selection" isVisible="true" isEnable="true" defaultState="false"/>
<radioButton id="radioButton_3" glueHorizental="left" defautContent="Third
Selection" isVisible="true" isEnable="true" defaultState="false"/>
</box>
</window>
</CuiModel>
```

Fig. 30 Example for element "radioButton" of CUI Model

Element checkBox

Inherits from: graficalIndividualContainer

Specification: Enables a boolean choice by checking a square box aside of a label. A checkBox may be differentiated from an radioButton by the fact that when grouped checkBoxes allow multiple choices while radioButton selection is mutually exclusive.

Attributes:

- defaultState: Indicates a default state for a checkbox..
- groupName: Is the name of the group.

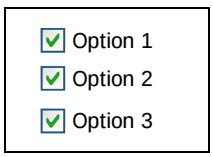


Fig.31 Appearance of checkBox

Element comboBox

Inherits from: graphicalIndividualComponent

Specification: Enables a direct selection over a collection of sequentially ordered items.

Attributes:

- isEditable: Specifies if the content of the textbox (composing a comboBox) is editable or not.
- maxLineVisible: Indicates the number of visible lines.

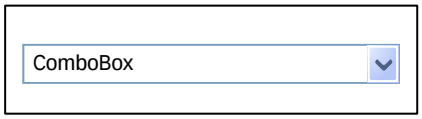


Fig.32 Appearance of comboBox

Element item

Inherits from: graphicalIndividualComponent

Specification: Specifies an item populating either a comboBox or a spin.

```
<CuiModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<window id="window_1" isVisible="true" isEnable="true"
windowLeftMargin="0" windowTopMargin="0" name="Wordaholic">
<box id="box_1" borderTitle="Wordaholic" isVisible="true" isEnable="true">
<comboBox id="comboBox_1">
<item id="item_1" defautContent="Rule1"/>
<item id="item_2" defautContent="Rue2"/>
<item id="item_3" defautContent="Rule3"/>
<item id="item_4" defautContent="Rule4"/>
<item id="item_5" defautContent="Rule5"/>
<item id="item_6" defautContent="Rule6"/>
<item id="item_7" defautContent="Rule7"/>
</comboBox>
</box>
</window>
</CuiModel>
```

Fig. 33 Example for elements "comboBox & item" of CUI Model

5.3 Conclusion

Comparing to the structure and elements of source WML 1.1 explored in the chapter 4, in this chapter, we present the components and elements of the target CUI Model of USIXML preparing for the mapping to WML 1.1 elements.

Concrete Interaction Objects realize an abstraction of widget sets found in popular graphical toolkits (Java AWT/Swing, HTML 4.O, Flash DRK6). A CIO is defined as an entity that users can perceive and/or manipulate (e.g., a push button, a list box, a check box). Orthogonally to AIOs, CIOs are divided into two types graphicalContainers (e.g., window, panel, table, cell, dialog box, etc.) and graphicalIndividual-Components (e.g., a button, a text component, a video component, a menu, a spin button, etc.).

The layout of the CUI is defined without any absolute coordinates. A box embedding mechanisms is used to specify a lay-out. Alignments between CIOs are defined with a special relationship called alignment

The elements chosen to be mapped from those of WML1.1 are the graphicalContainer and graphicalIndividualComponent elements and its children elements. In the next chapter, we present the mapping rules from WML 1.1 to CUI model.

Chapter 6. MAPPING RULES AND ITS CORRESPONDING XSLT TEMPLATES

As we've introduced the Java transformation application in the chapter 3, the rest work for the transformation is to create a XSLT style sheet which combines all the XSLT templates. For building the XSLT style sheet, we deduct the XSLT templates concerning to the mapping rules from WML to CUI Model. In this chapter, we firstly present the mapping types and the way we organise the mapping rules. Secondly, we introduce the XPath notation with which we can express the mapping rules in a concise and logic way. Lastly, we will show the mapping rules of each chosen element of WML and its corresponding XSLT template.

6.1 Types of mapping rules and its presentation

For the mapping between WML element and CUI element, there are four types of mapping. The first two types are element to element mapping which are respectively one-to-one mapping, one-to-several mapping. In the case of one-to-several mapping, for instance, the select element of WML 1.1 can be mapped to comboBox or a drop down menu according to different conditions. Another example is the option element of WML 1.1 can be optionally mapped to check box, radio button, linked textComponent, combBox item, and menu item in CUI by checking different conditions. These will be shown in the section 6.3.2.2 for the detailed mapping rules. The third type is element-to-attribute mapping specially for the element such as i, em, b, strong, u, big and small. The detailed mapping will be presented in the section 6.3.3.4. The last type is text-content- to-element mapping which will be presented in detailed in the section 6.3.3.4.

To determine the value of attribute of the mapped CUI element, there are three possibilities. The first one is to directly determine the value by the corresponding attribute of the mapping WML element. The second one is to determine the value by

calculating using XPath functions when there is not a correspondent attribute of the mapping WML element. The functions of XPath are included in the function library (see Appendix 2). Unlike the first two possibilities which are not conditional, the third one is to determine the value depending on the condition of the source WML element's context.

To present the mapping rules in a comprehensive way in considering the different possibilities of determination of attribute value, we divide the mapping table into three sub-mapping tables.

The WMLelementName_table 1 presents the mapping elements from WML to CUI for the one-to-one mapping and several-to-one mapping without "condition" field while for one-to-several mapping with "condition" field. It also presents the attributes that could be determined its value by directly mapping from WML attributes.

The WMLelementName_table 2 presents the attributes of the mapped CUI element which can not be directly mapped from an attribute of WML element. These attributes have unconditional values which are determined only by calculation.

Comparing to The WMLelementName_table 2, the WMLelementName_table 3 has an additional "condition" field to present the attributes with conditional values.

6.2 The XPath notation for mapping rules

As we've introduced in the section 3.3, XPath expressions select nodes from the input XML document for further transforming into other XML vocabularies. Both input and output documents are represented by the XPath data model. For the mapping rules, some attribute value and mapping conditions are complicated and hard to be expressed by normal language. As XPath expression can express the attribute value and mapping

conditions in a robust and understandable way, we use the XPath notation in our mapping rules.

6.2.1 Introduction

XPath is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer. In general, an XPath expression specifies a pattern that selects a set of XML nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes. XSLT templates then use those patterns when applying transformations.

Much of the notation of directory paths is carried over intact:

- The forward slash (/) is used as a path separator.
- An absolute path from the root of the document starts with a /.
- A relative path from a given location starts with anything else.
- A double period (..) indicates the parent of the current node.
- A single period . indicates the current node.

In an xHTML document, for example, the path /h1/h2/ would indicate an h2 element under an h1. In a pattern-matching specification like XSLT, the specification /h1/h2 selects all h2 elements that lie under an h1 element. To select a specific h2 element, square brackets ([]) are used for indexing (like those used for arrays). The path /h1[4]/h2[5] would therefore select the fifth h2 element under the fourth h1 element.

A name in XPath specification refers to an element. To refer to attribute, you prefix its name with an "@" sign. For example, @type refers to the type attribute of an element. Assuming you have an XML document with list elements, for example, the expression list/@type selects the type attribute of the list element. Since the expression does not begin with /, the reference specifies a list node relative to the current context

6.2.2 Location steps

The location step for a specific node has three parts:

- an axis, which specifies the tree relationship between the nodes selected by the location step and the context node,
- a node test, which specifies the node type and expanded-name of the nodes selected by the location step, and
- zero or more predicates, which use arbitrary expressions to further refine the set of nodes selected by the location step.

6.2.2.1 Axes

The following axes are available:

- the child axis contains the children of the context node.
- the descendant axis contains the descendants of the context node; a descendant is a child or a child of a child and so on; thus the descendant axis never contains attribute or namespace nodes
- the parent axis contains the parent of the context node, if there is one.
- the ancestor axis contains the ancestors of the context node; the ancestors of the context node consist of the parent of context node and the parent's parent and so on; thus, the ancestor axis will always include the root node, unless the context node is the root node
- the following-sibling axis contains all the following siblings of the context node; if the context node is an attribute node or namespace node, the following-sibling axis is empty
- the preceding-sibling axis contains all the preceding siblings of the context node; if the context node is an attribute node or namespace node, the preceding-sibling axis is empty
- the following axis contains all nodes in the same document as the context node that are after the context node in document order, excluding any descendants and excluding attribute nodes and namespace nodes
- the preceding axis contains all nodes in the same document as the context node that are before the context node in document order, excluding any ancestors and excluding attribute nodes and namespace nodes

- the attribute axis contains the attributes of the context node; the axis will be empty unless the context node is an element
- the namespace axis contains the namespace nodes of the context node; the axis will be empty unless the context node is an element
- the self axis contains just the context node itself
- the descendant-or-self axis contains the context node and the descendants of the context node
- the ancestor-or-self axis contains the context node and the ancestors of the context node; thus, the ancestor axis will always include the root node

6.2.2.2 Node Tests

Every axis has a principal node type. If an axis can contain elements, then the principal node type is element; otherwise, it is the type of the nodes that the axis can contain. Thus,

- For the attribute axis, the principal node type is attribute.
- For the namespace axis, the principal node type is namespace.
- For other axes, the principal node type is element.

The attribute node text can be expressed as @nameOfAttribute. For example, @href selects the href attribute of the context node; if the context node has no href attribute, it will select an empty set of nodes.

6.2.2.3 Predicates

A predicate filters a node-set with respect to an axis to produce a new node-set. For each node in the node-set to be filtered, the PredicateExpr is evaluated with that node as the context node, with the number of nodes in the node-set as the context size, and with the proximity position of the node in the node-set with respect to the axis as the context position; if PredicateExpr evaluates to true for that node, the node is included in the new node-set; otherwise, it is not included.

6.3 Mapping Rules and XSLT templates

For the scale of mapping which we've presented in the section 4.3, we present the mapping rules for elements of WML 1.1 which could be mapped to that of CUI model of USIXML.

According to the structure of WML 1.1, we organise the WML 1.1 elements into four categories which are respectively card element, user input elements, text presentation and layout elements and image & anchor elements. Following each mapping rule, we deduct a XSLT template which we will use to construct the XSLT stylesheet for the transformation.

6.3.1 card element :

Language	**WML**	**USIXML**
Element	Card	Window
Attribute	Title	Name

card_table 1

As the card element of WML is a top level container of text and input elements. The most proper mapping element for USIXML is window element. The window's attribute name's value equals to the title attribute of card element.

Element	Attribute	Value
window	Id	window_"number of preceding card +1"
	isVisible	TRUE
	isEnable	TRUE
	windowLeftMargin	0
	windowTopMargin	0

card_table 2

We identify the window element by its presenting position of window element in the target USIXML file. To get the value of id, we first find out the position of the corresponding window element in the source WML file by calculating the number of

preceding card elements plus 1. As there are not corresponding attributes for isVisible, isEnable, widowLeftMargin and widowTopMargin within card element in WML, we give the value of these attributes as default in considering the properties of the mapping. Here, the window are both visible and enable. And the window is full screen, so the left and top margin equal zero.

Template deal with the card element

```
<xsl:template match="card">
    <!--set condition for the card element which has the p element inside-->
    <xsl:if test="descendant::p">
        <xsl:element name="window">
            <xsl:attribute name="id"><xsl:value-of
select="concat('window_',string(count(preceding::card)+1))"/></xsl:attribute>
            <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
            <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
            <xsl:attribute name="windowLeftMargin"><xsl:value-of
select="0"/></xsl:attribute>
            <xsl:attribute name="windowTopMargin"><xsl:value-of
select="0"/></xsl:attribute>
            <xsl:if test="@title">
                <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
            </xsl:if>
            <!--add a box element inside the window element-->
            <xsl:element name="box">
                <xsl:attribute name="id"><xsl:value-of
select="concat('box_','1')"/></xsl:attribute>
                <xsl:if test="@title">
                    <xsl:attribute name="borderTitle"><xsl:value-of
select="@title"/></xsl:attribute>
                </xsl:if>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:apply-templates/>
            </xsl:element>
```

```
            </xsl:element>
        </xsl:if>
    </xsl:template>
```

The concrete example in the section 7.2 shows the resulted CUI Model from WML by this template.

6.3.2 User input elements:

All the following elements of WML we will present in this section, such as optgroup, select, option and input, are all serve to collect the user input.

6.3.2.1 optgroup element

Language	**WML**	**USIXML**
Element	optgroup	Box
Attribute	title	Name
		borderTitle

optgroup_table 1

The optgroup element which groups related option elements into a hierarchy which is very similar to box element in USIXML which is a container that enables an unambiguous structuring of all graphicalIndividualComponents within a window. Therefor, we map the optgroup element of WML to the box element of USIXML. The reason to map title attribute of WML to both name and borderTitle attributes is for box element description and the presentation of the title of the box in the final UI of USIXML.

Element	Attribute	Value
box	id	box_"number of preceding fieldset+ancestor fieldset+preceding optgroup+2"

Optgroup_table 2

The explication of the value of id attribute for the box element of USIXML is shown at fieldset_table 2.

Template deal with the optgroup element

```
<xsl:template match="optgroup">
    <xsl:element name="box">
        <xsl:attribute name="id"><xsl:value-of
select="concat('box_',string(1+count(preceding::optgroup)+count(preceding::fieldset)
+count(ancestor::fieldset)+1))"/></xsl:attribute>
        <xsl:if test="@title">
            <xsl:attribute name="borderTitle"><xsl:value-of
select="@title"/></xsl:attribute>
            <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
        </xsl:if>
        <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
        <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

6.3.2.2 select & option element

The select element lets users pick from a list of options whereas the option element is a single choice option in a select element. As the mappings for selection and option elements are one-to-several mappings which engage 3 checking points to decide which element it should map to. According to the complexity of this issue, before we introduce the table of mapping rules, we present you the decision procedure for the mapping.

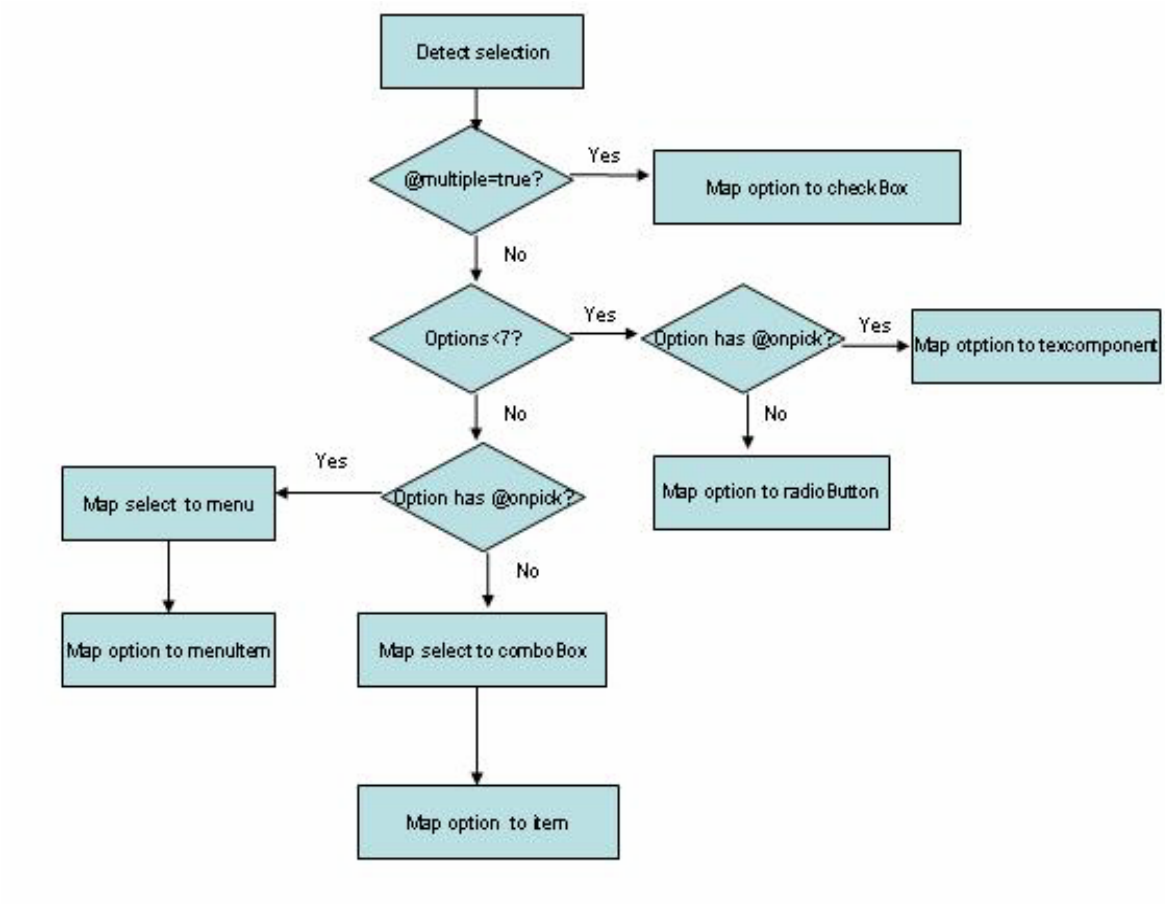


Fig. 34 Decision procedure for mapping of select and option

As we've shown in the fig.34, the first condition to check is the value of multiple attribute of select element. If the value is true, it means that the selection could be one or more than one. In this case, on the other side, only the checkbox element of USIXML allows multiple selection, for this reason, we map all the enclosed option elements to checkbox elements of USIXML.

Then, we reach to the second check point which is whether the total option number is less than 7. If the answer is yes, there will be two possibilities for the mapping of the option element. The two possibilities are radioButton and linked textComponent of USIXML. If the answer is not, there are also two possibilities. The first possibility is select maps to menu and option maps to menuItem. The other one is select maps to comboBox and option maps to item. The reason is that each of radioButton and textComponent takes one line place, so if there are too many this kind of widgets in one

UI, it will be take a lot of place which may make the resulted UI very crowded. Instead of radioButton and textComponent, the comboBox and drop-down menu can take only one line place by hiding comboBox items and menu items.

The reasons for choosing between radioButton and linked textComponent and between menuItem and comboBox item are the same. If the option element has onpick arttribute, to select this option means to navigate to the address corresponding to the value of onpick attribute. As the menuItem with drop-down menu and the linked textComponent has the function to direct navigate to another address when they are chosen, it is better to map the option element to them than to the comboBox item and radioButton.

We can draw some conclusion as following:

1. If the value of multiple attribute is true, option elements map to checkBox.

2. Else, Mapping elements of USIXML for option and select are shown in the following table.

	@onpick	Not(@onpick)
options>6	Menu	ComboBox

Mapping table for select

	Not(@onpick)	@onpick
options<7	radioButton	textComponent
options>6	menuItem	ComboBox item

Mapping table for option

The complete mapping rules with XPath notation are shown as following:

Language	WML	USIXML	Condition
Element	select	comboBox	not(@multiple="true") & count(option)>6 & not(./option[@onpick])
		menu	not(@multiple="true") & count(option)>6 & ./option[@onpick]
Attribute	@title	@defaultContent	

select_table 1

The value of the defaultContent attribute of comboBox and menu element of USIXML equals to the string value of the title attribute of select element.

Language	WML	USIXML	Condition
Element	option	checkBox	../@multiple="true"
		radioButton	not(../@multiple="true") & count(option)<7 & not(@onpick)
		textComponent	not(../@multiple="true") & count(option)<7 & @onpick
		menuItem	not(../@multiple="true") & count(option)>6 & @onpick
		item	not(../@multiple="true") & count(option)>6 & not(@onpick)

option_table 1

Element	Attribute	Value
checkBox	id	checkBox_" number of preceding checkBox's option +1"
radioButton	id	radioButton_"number of preceding radioButton option+1"
item	id	item_"number of preceding item option +1"
	defaultContent	text content enclosed in option
menuItem	id	menuItem_"number of preceding menu item option +1"
	defaultContent	text content enclosed in option
textComponent	id	textComponent_"number of preceding textComponent+1"
	defaultContent	text content enclosed in option
	defaultHyperLinkTarget	Value of @onpick

option_table 2

We identify elements of USIXML by its presenting position of such element in the target USIXML file. To get the value of id, we find out the position of the corresponding option element in the source WML file by calculating the number of preceding corresponding option elements plus 1.

Element	Attribute	Value	Condition
checkBox & radioButton	defaultState	TRUE	select/@ivalue contains the checkBox's position number.
	defaultState	FALSE	select/@ivalue doesn't contain the checkBox's position number.
	glueHorizontal	p/@align	text enclosed in p element & p has align attribute
	glueHorizontal	Left	text enclosed in p element & p doesn't have align attribute

option_table 3

The value of the defaultState attribute depends on whether the ivalue attribute of the outer select element contains the checkBox or radioButton's positon number. If yes, the value is "true", and "false" vice versa. For the value of glueHorizontal attribute, it equals to the value of the align attribute of the outer p element. If the outer p element has not align attribute. The value is "left" as default.

Template deal with the select & option element

```
<xsl:template match="select">
        <xsl:choose>
            <xsl:when test="@multiple='true'">
<!--Mapping option to checkBox -->
                <xsl:for-each select="./option">
                    <xsl:element name="checkBox">
                        <xsl:attribute name="id"><xsl:value-of
select="concat('checkBox_',string(count(preceding::option[../@multiple='true'])+1))"/
></xsl:attribute>
                        <xsl:choose>
                            <xsl:when test="ancestor::p[@align]">
                                <xsl:attribute
name="glueHorizontal"><xsl:value-of
select="ancestor::p[@align]/@align"/></xsl:attribute>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:attribute
name="glueHorizontal"><xsl:value-of select="'left'"/></xsl:attribute>
                            </xsl:otherwise>
                        </xsl:choose>
                        <xsl:if test="../@title">
                            <xsl:attribute name="groupName"><xsl:value-of
select="../@title"/></xsl:attribute>
                        </xsl:if>
```

```
                                <xsl:attribute name="defautContent"><xsl:value-of
select="./text()"/></xsl:attribute>
                                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                                <xsl:choose>
                                    <xsl:when
test="contains(string(../@ivalue),string(position()))">
                                        <xsl:attribute name="defaultState"><xsl:value-of
select="true()"/></xsl:attribute>
                                    </xsl:when>
                                    <xsl:otherwise>
                                        <xsl:attribute name="defaultState"><xsl:value-of
select="false()"/></xsl:attribute>
                                    </xsl:otherwise>
                                </xsl:choose>
                            </xsl:element>
                        </xsl:for-each>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:if test="count(./option) &gt; 6">
                           <xsl:choose>
                               <xsl:when test="./option[@onpick]">
<!--Mapping select to menu -->
                                <xsl:element name="menu">
                                   <xsl:attribute name="id"><xsl:value-of
select="concat('menu_',string(count(preceding::select[count(./option) &gt;
6])+1))"/></xsl:attribute>
                                <xsl:attribute name="defaultContent"><xsl:value-of
select="@title"/></xsl:attribute>
<!--Mapping option to menuItem -->

                                <xsl:for-each select="./option">
                                   <xsl:element name="menuItem">
                                       <xsl:attribute name="id"><xsl:value-of
select="concat('menuItem_',string(count(preceding::option)+1))"/></xsl:attribute>
                                       <xsl:attribute
name="defautContent"><xsl:value-of select="./text()"/></xsl:attribute>
                                   </xsl:element>
                                </xsl:for-each>
                             </xsl:element>
                           </xsl:when>
                            <xsl:otherwise>
```

```
<!--Mapping select to comboBox -->
                        <xsl:element name="comboBox">
                          <xsl:attribute name="id"><xsl:value-of
select="concat('comboBox_',string(count(preceding::select[count(./option) &gt;
6])+1))"/></xsl:attribute>
                          <xsl:attribute name="defaultContent"><xsl:value-of
select="@title"/></xsl:attribute>
                          <xsl:for-each select="./option">
<!--Mapping option to item -->
                              <xsl:element name="item">
                                  <xsl:attribute name="id"><xsl:value-of
select="concat('item_',string(count(preceding::option)+1))"/></xsl:attribute>
                                  <xsl:attribute
name="defautContent"><xsl:value-of select="./text()"/></xsl:attribute>
                              </xsl:element>
                          </xsl:for-each>
                        </xsl:element>
                      </xsl:otherwise>
                    </xsl:choose>
                  </xsl:if>
                      <xsl:if test="count(./option) &lt; 7">
                        <xsl:choose>
                          <xsl:when test="./option[@onpick]">
                            <xsl:for-each select="child::option">
<!--Mapping option to textComponent -->
                              <xsl:apply-templates/>
                            </xsl:for-each>
                          </xsl:when>
                          <xsl:otherwise>
                            <xsl:for-each select="child::option">
<!--Mapping select to radioButton -->
                              <xsl:element name="radioButton">
                                  <xsl:attribute name="id"><xsl:value-of
select="concat('radioButton_',string(count(preceding::option)-count(preceding::option
[../@multiple='true'])+1))"/></xsl:attribute>
                                  <xsl:choose>
                                      <xsl:when test="ancestor::p[@align]">
                                          <xsl:attribute
name="glueHorizontal"><xsl:value-of
select="ancestor::p[@align]/@align"/></xsl:attribute>
                                      </xsl:when>
                                      <xsl:otherwise>
                                          <xsl:attribute
name="glueHorizontal"><xsl:value-of select="'left'"/></xsl:attribute>
```

```
                                        </xsl:otherwise>
                                    </xsl:choose>
                                    <xsl:if test="../@title">
                                        <xsl:attribute
name="groupName"><xsl:value-of select="../@title"/></xsl:attribute>
                                    </xsl:if>
                                    <xsl:attribute
name="defautContent"><xsl:value-of select="./text()"/></xsl:attribute>
                                    <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                                    <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                                    <xsl:choose>
                                        <xsl:when
test="position()=number(../@ivalue)">
                                            <xsl:attribute
name="defaultState"><xsl:value-of select="true()"/></xsl:attribute>
                                        </xsl:when>
                                        <xsl:otherwise>
                                            <xsl:attribute
name="defaultState"><xsl:value-of select="false()"/></xsl:attribute>
                                        </xsl:otherwise>
                                    </xsl:choose>
                                </xsl:element>
                            </xsl:for-each>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:if>
            </xsl:otherwise>
        </xsl:choose>
        </xsl:template>
```

The concrete example in the 7.4 shows the resulted CUI Model from WML by this template.

6.3.2.3 input element

Language	**WML**	**USIXML**
Element	Input	textComponent
Attribute	Title	name
	Size	size
	Maxlength	maxlength
	Value	defaultContent
	type="password"	isPassword="true"

input_table 1

As the input element specifies a text entry object of WML, we map it to an empty textComponent element which also defines a textual content of USIXML. There is not a widget which specifically presents a password field. Instead, if the type attribute of input element equals to “password”, the value of ispassword attribute of textComponent element equals to “true” which makes the empty textComponent to present the password field.

Element	Attribute	Value
Text Component	Id	textComponent_"number of preceding input preceding text() + ancestor text()+1"
	isEditable	TRUE
	isVisible	TRUE

input_table 2

We identify the textComponent element of USIXML by its presenting position of textComponent element in the target USIXML file. To get the value of id, we find out the position of the corresponding img element in the source WML file by calculating the number of preceding img elements plus 1.

Template deal with the input element

```
<xsl:template match="input">
    <xsl:element name="textComponent">
        <xsl:attribute name="id"><xsl:value-of
select="concat('textComponent_',string(count(preceding::input)+count(preceding::tex
t())+count(ancestor::text())+1))"/></xsl:attribute>
        <xsl:choose>
            <xsl:when test="ancestor::p[@align]">
                <xsl:attribute name="glueHorizental"><xsl:value-of
select="ancestor::p[@align]/@align"/></xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="glueHorizental"><xsl:value-of
select="'left'"/></xsl:attribute>
            </xsl:otherwise>
```

```
            </xsl:choose>
            <xsl:if test="@title">
                <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
            </xsl:if>
            <xsl:if test="@size">
                <xsl:attribute name="size"><xsl:value-of
select="@size"/></xsl:attribute>
            </xsl:if>
            <xsl:if test="@maxlength">
                <xsl:attribute name="maxlength"><xsl:value-of
select="@maxlength"/></xsl:attribute>
            </xsl:if>
            <xsl:if test="@type='password'">
                <xsl:attribute name="isPassword"><xsl:value-of
select="true()"/></xsl:attribute>
            </xsl:if>
            <xsl:attribute name="defaultContent"><xsl:value-of
select="@value"/></xsl:attribute>
            <xsl:attribute name="isEditable"><xsl:value-of
select="true()"/></xsl:attribute>
            <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
        </xsl:element>
    </xsl:template>
```

The concrete example in the section 7.3 shows the resulted CUI Model from WML by this template

6.3.3 Text presentation and layout

All the following elements of WML we will present in this section, such as table, td, fieldset and text content, etc., are all related to text presentation and layout.

6.3.3.1 table element

Language	WML	USIXML
Element	Table	table
Attribute	Title	name
	columns	ySize
	Align	glueHorizontal

table_table 1

Both the WML and USIXML have table element which has similar definition. But the names of its attributes are different. The table 1 shows the attributes that can be directly mapped from WML to USIXML.

Element	Attribute	Value
table	Id	table_"number of preceding table +1"
	xSize	number of tr inside the table

table_table 2

We identify the table element of USIXML by its presenting position of the table element in the target USIXML file. To get the value of id, we first find out the position of the corresponding table element in the source WML file by calculating the number of preceding table elements plus 1. For the value of xSize, we calculate the number of tr inside the corresponding table of WML file.

Template deal with the table element

```
    <xsl:template match="table">
        <xsl:element name="table">
            <xsl:attribute name="id"><xsl:value-of
select="concat('table_',string(count(preceding::table)+1))"/></xsl:attribute>
            <xsl:attribute name="ySize"><xsl:value-of
select="@columns"/></xsl:attribute>
            <xsl:attribute name="xSize"><xsl:value-of
select="count(child::tr)"/></xsl:attribute>
            <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
            <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
            <xsl:if test="@title">
                <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
            </xsl:if>
            <xsl:if test="@align">
                <xsl:attribute name="glueHorizontal"><xsl:value-of
select="@align"/></xsl:attribute>
            </xsl:if>
            <xsl:apply-templates/>
```

```
        </xsl:element>
    </xsl:template>
```

The concrete example in the section 7.6 shows the resulted CUI Model from WML by this template.

6.3.3.2 td element

Language	WML	USIXML
Element	td	Cell

td_table 1

As the td element is where to put the data in the table of WML, we map it to the cell element of USIXML which can contain not only graphicalIndividualComponent, but also graphicalContainer for the table element.

Element	Attribute	Value
cell	Id	cell_"number of preceding td + 1"
	xIndex	number of preceding-cibling tr of parent node + 1
	yIndex	number of preceding-cibling td + 1

td_table 2

We identify the cell element of USIXML by its presenting position of cell element in the target USIXML file. To get the value of id, we first find out the position of the corresponding td element in the source WML file by calculating the number of preceding td elements plus 1. For the value of xIndex, we calculate the line position of the corresponding td in the table of WML file. We can get the value by calculate the number of preceding-cibling tr of parent node plus 1 for the parsed DOM tree of source WML file. For the value of yIndex, we calculate the column position of the corresponding td in the table of WML file. We can get the value by calculate the number of preceding-cibling td of current node plus 1 for the parsed DOM tree of source WML file.

Template deal with the td element

```
    <xsl:template match="td">
        <xsl:element name="cell">
            <xsl:attribute name="id"><xsl:value-of
```

```
select="concat('cell_',string(count(preceding::td)+1))"/></xsl:attribute>
                <xsl:attribute name="xIndex"><xsl:value-of
select="count(../preceding-sibling::tr)+1"/></xsl:attribute>
                <xsl:attribute name="yIndex"><xsl:value-of
select="count(preceding-sibling::td)+1"/></xsl:attribute>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:template>
```

The concrete example in the section 7.6 shows the resulted CUI Model from WML by this template.

6.3.3.3 **fieldset element**

Language	**WML**	**USIXML**
Element	Fieldset	box
Attribute	Title	name
		borderTitle

fieldset_table 1

The fieldset element of WML, which allows the grouping of related fields and text, maps to the box element of USIXML. The reason to map title attribute of WML to both name and borderTitle attributes is for box element description and the presentation of the title of the box in the final UI of USIXML.

Element	Attribute	Value
box	id	box_"number of preceding fieldset+ancestor fieldset+ preceding optgroup+2"

fieldset_table 2

We identify the box element of USIXML by its presenting position of box element in the target USIXML file. As all the fieldset and optgroup element of WML can map to the box element of USIXML, the value of id equals to the number of preceding fieldset and ancestor fieldset and preceding optgroup elements of current node plus 2 for the parsed DOM tree of source WML file. The reason to add the number of ancestor

fieldset is that the fieldset element can be self nested. And the reason to add 2 is that the first box element is always automatically added under the window element.

Template deal with the fieldset element

```
<xsl:template match="fieldset">
    <xsl:element name="box">
        <xsl:attribute name="id"><xsl:value-of
select="concat('box_',string(1+count(preceding::fieldset)+count(ancestor::fieldset)+co
unt(preceding::optgroup)+1))"/></xsl:attribute>
        <xsl:if test="@title">
            <xsl:attribute name="borderTitle"><xsl:value-of
select="@title"/></xsl:attribute>
            <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
        </xsl:if>
        <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
        <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

6.3.3.4 text content

Language	WML	USIXML
Content & Element	Text Content	textComponent

text_content_table 1

The text content of WML maps to the textComponent element of USIXML. The reason is that in USIXML the textual content is present by textComponent element.

Element	Attribute	Value
textComponent	id	textComponent_"number of preceding input + preceding text()+ ancestor text() + 1"

text_content_table 2

The explication of the value of id attribute for the textComponent element of USIXML is shown at input_table 2.

Element	Attribute	Value	Condition
textComponent	isItalic	TRUE	text enclosed in the i or em element
	isBold	TRUE	text enclosed in the b or strong element
	isUnderline	TRUE	text enclosed in the u element
	textSize	12	text enclosed in the big element
	textSize	8	text enclosed in the small element
	defaultDefaultHyperLinkTarget	go/@href	text enclosed in the anchor element
	defaultDefaultHyperLinkTarget	a/@href	text enclosed in the a element
	glueHorizontal	p/@align	text enclosed in the p element & the p has align attribute
	glueHorizontal	left	text enclosed in the p element & the p doesn't have align attribute

text_content_table 3

The text content can be nested in the inline layout elements of WML such as i, em, b, strong, big and small element. When it happens the corresponding additional attributes will be added to the textComponent. For the value of defaultDefaultHyperLinkTarget attribute, it equals to that of the href attribute of outer a element or go element which enclosed in the anchor element. For the value of glueHorizontal attribute, it equals to that of the align attribute of the outer p element. If the outer p element has not align attribute. The value is "left" as default.

Template deal with the text content

```
    <xsl:template match="text()">
            <xsl:element name="textComponent">
                <xsl:attribute name="id"><xsl:value-of
select="concat('textComponent_',string(count(preceding::text())+count(ancestor::text(
))+count(preceding::input)+1))"/></xsl:attribute>
                <xsl:choose>
                    <xsl:when test="ancestor::p[@align]">
                        <xsl:attribute name="glueHorizental"><xsl:value-of
select="ancestor::p[@align]/@align"/></xsl:attribute>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:attribute name="glueHorizental"><xsl:value-of
select="'left'"/></xsl:attribute>
```

```
                    </xsl:otherwise>
                </xsl:choose>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:for-each select="ancestor::*">
                    <xsl:if test="local-name(.)='anchor'">
                        <xsl:attribute
name="defaultHyperLinkTarget"><xsl:value-of select="*/@href"/></xsl:attribute>
                        <xsl:if test="@title">
                        <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
                        </xsl:if>
                    </xsl:if>
                    <xsl:if test="local-name(.)='a'">
                        <xsl:attribute
name="defaultHyperLinkTarget"><xsl:value-of select="@href"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='b'">
                        <xsl:attribute name="isBold"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='em'">
                        <xsl:attribute name="isItalic"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='strong'">
                        <xsl:attribute name="isBold"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='i'">
                        <xsl:attribute name="isItalic"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='u'">
                        <xsl:attribute name="isUnderline"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='small'">
                        <xsl:attribute name="textSize"><xsl:value-of
select="8"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='big'">
```

```
                    <xsl:attribute name="textSize"><xsl:value-of
select="12"/></xsl:attribute>
                </xsl:if>
            </xsl:for-each>
            <xsl:attribute name="defaultContent"><xsl:value-of
select="normalize-space()"/></xsl:attribute>
        </xsl:element>
    </xsl:template>
```

The concrete example in the section 7.2 shows the resulted CUI Model from WML by this template.

6.3.4 Images and Navigations

In this category, we have four elements which are img, anchor, a, and navigational option.

6.3.4.1 img element

Language	WML	USIXML
Element	Img	imageComponent
Attribute	Height	imageHeight
	Width	imageWidth
	Src	defaultHyperLinkTarget
	vspace(default:0)	imageVertSpace
	hspace(default:0)	imageHorizSpace
	align(default:bottom)	glueVertical

img_table 1

The img element of WML maps to the imageComponent element of USIXML both of them serve to present imag. As the default value for vspace and hspace in WML are both zero, if the img element doesn't has these attribute, the value of imageVertSpace and imageHorizontal are both zero.

Element	Attribute	Value
imageComponent	id	imageComponent_"number of preceding img + 1"

img_table 2

We identify the imageComponent element of USIXML by its presenting position of imageComponent element in the target USIXML file. To get the value of id, we find

out the position of the corresponding img element in the source WML file by calculating the number of preceding img elements plus 1.

Element	Attribute	Value	Condition
imageComponent	glueHorizontal	p/@align	text enclosed in p element & p has align attribute
	glueHorizontal	left	text enclosed in p element & p doesn't have align attribute

img_table 3

For the value of glueHorizontal attribute, it equals to the value of the align attribute of the outer p element. If the outer p element has not align attribute. The value is "left" as default.

Template deal with the img element

```
    <xsl:template match="img">
        <xsl:element name="imageComponent">
            <xsl:attribute name="id"><xsl:value-of
select="concat('imageComponent_',string(count(preceding::img)+1))"/></xsl:attribute
>
            <xsl:choose>
                <xsl:when test="ancestor::p[@align]">
                    <xsl:attribute name="glueHorizental"><xsl:value-of
select="@align"/></xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:attribute name="glueHorizental"><xsl:value-of
select="'left'"/></xsl:attribute>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:attribute name="defaultHyperLinkTarget"><xsl:value-of
select="@src"/></xsl:attribute>
            <xsl:choose>
                <xsl:when test="@vspace">
                    <xsl:attribute name="imageVertSpace"><xsl:value-of
select="@vspace"/></xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:attribute name="imageVertSpace"><xsl:value-of
select="0"/></xsl:attribute>
                </xsl:otherwise>
            </xsl:choose>
```

```
            <xsl:choose>
                <xsl:when test="@hspace">
                    <xsl:attribute name="imageHorizSpace"><xsl:value-of
select="@hspace"/></xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:attribute name="imageHorizSpace"><xsl:value-of
select="0"/></xsl:attribute>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:choose>
                <xsl:when test="@align">
                    <xsl:attribute name="glueVertical"><xsl:value-of
select="@align"/></xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:attribute name="glueVertical"><xsl:value-of
select="'bottom'"/></xsl:attribute>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:if test="@height">
                <xsl:attribute name="imageHeight"><xsl:value-of
select="@height"/></xsl:attribute>
            </xsl:if>
            <xsl:if test="@width">
                <xsl:attribute name="imageWidth"><xsl:value-of
select="@width"/></xsl:attribute>
            </xsl:if>
        </xsl:element>
    </xsl:template>
```

The concrete example in the section 7.5 shows the resulted CUI Model from WML by this template.

6.3.4.2 Navigation elements

There are three navigation element in WML such as anchor, a, and navigational option. The navigation element specify a hyper text link for a UI transition.

Language	WML	USIXML
element	a or anchor or navigational option	graphicalTransition
		Source
		Target

navigationElements_table 1

The navigation elements, such as a, anchor, navigational option, map to graphicalTransition, source, and target which define the UI transition in USIXML.

Element	Attribute	Value
graphicalTransition	id	textLink_"number of (preceding a + preceding anchor+ preceding option)+1"
	type	Open
source	sourceId	textComponet_"number of(preceding & ancestor text() + preceding input)+1"
target	targeted	a/@href or option/@onpick
		anchor/go/@href

navigationElements_table 2

The sourceId attribute of source element equals to the id attribute of the corresponding textComponent element in the USIXML file.

Template deal with the a element

```
<xsl:template name="transition_a">
    <xsl:for-each select="//a">
        <xsl:element name="graphicalTransition">
            <xsl:attribute name="id"><xsl:value-of
select="concat('textLink_',string(count(preceding::a)+count(preceding::anchor)+1))"/
></xsl:attribute>
            <xsl:attribute name="type"><xsl:value-of
select="'open"'/></xsl:attribute>
            <xsl:element name="source">
                <xsl:attribute name="sourceId"><xsl:value-of
select="concat('textComponent_',string(count(preceding::text())+count(ancestor::text(
))+count(preceding::input)+1))"/></xsl:attribute>
            </xsl:element>
            <xsl:element name="target">
                <xsl:attribute name="targetId"><xsl:value-of
select="@href"/></xsl:attribute>
            </xsl:element>
        </xsl:element>
    </xsl:for-each>
</xsl:template>
```

Template deal with the anchor element

```
<xsl:template name="transition_anchor">
    <xsl:for-each select="//anchor">
        <xsl:element name="graphicalTransition">
            <xsl:attribute name="id"><xsl:value-of
select="concat('textLink_',string(count(preceding::a)+count(preceding::anchor)+1))"/
></xsl:attribute>
            <xsl:attribute name="type"><xsl:value-of
select="'open"'/></xsl:attribute>
            <xsl:element name="source">
                <xsl:attribute name="sourceId"><xsl:value-of
select="concat('textComponent_',string(count(preceding::text())+count(ancestor::text(
))+count(preceding::input)+1))"/></xsl:attribute>
            </xsl:element>
            <xsl:element name="target">
                <xsl:attribute name="targetId"><xsl:value-of
select="child::go/@href"/></xsl:attribute>
            </xsl:element>
        </xsl:element>
    </xsl:for-each>
</xsl:template>
```

The concrete example in the section 7.2 shows the resulted CUI Model from WML by this template.

To present the mapping rules in a whole picture, we put the complete mapping table in Appendix 3. For the complete code of XSLT style sheet, see Appendix 4.

6.4 Conclusion

Based on the earlier chapter, such as chapter 4 for the source WML, and chapter 5 for the target CUI we implement the mapping rules with XPATH notation. We implement the complete XSLT style sheet by combining all the XSLT templates deducted by the mapping rules.

WML	USIXML	Mapping Condition
card	window	
table	table	
td	cell	
fieldset, optgroup	box	
input	textComponent	
text content		
select	comboBox	<option> don't has @onpick & has more than 6 options
	menu	<option> has @onpick & has more than 6 options
option	radioButton	<option> don't has @onpick & has less than 7 options
	item	<option> don't has @onpick & has more than 6 options
	menuItem	<option> has @onpick & has more than 6 options
	textComponent	<option> has @onpick & has less than 7 options
	checkBox	<select> has @multiple="true"
img	imageComponent	
a, anchor	source, target, graphicalTransition	
option		<option> has @onpick

Table 2. Element-to-element Mapping table

The above table shows the mapping from WML elements to USIXML elements which include one-to-one mapping and one-to-several mapping. For the one-to-several mapping, we use the explicite mapping condition to decide which USIXML element will be mapped from certain WML element.

WML	USIXML textComponent's attribute
i, em	isItalic="true"
b, strong	isBold="true"
u	isUderline="true"
big	textSize="12"
small	textSize="8"

Table 3. Element-to-attribute Mapping table

The element-to-attribute mapping is specially for the element such as i, em, b, strong, u, big and small. All these element map to a textComponent attribute.

source:WML1.1		target:USIXML	
Element	Attribute	Attribute	Element
card	title	name	window
table	title	name	table
	columns	ySize	
	align	glueHorizontal	
fieldset	title	name	box
	title	borderTitle	
optgroup	title	name	
	title	borderTitle	
input	title	name	textComponent
	size	size	
	maxlength	maxlength	
	value	defaultContent	
	type="password"	isPassword="true"	
anchor	title	name	
anchor/go	herf	hyperLinkTarget	
a	herf	hyperLinkTarget	
p	align	glueHorizontal	
select	title	defaultContent	comboBox
			menu
option	../@title	groupName	checkBox
	p/@align	glueHorizontal	
	../@title	groupName	radioButton
	ivalue	defaultState	
	p/@align	glueHorizontal	
	onpick	defaultHyperLinkTarget	textComponent
img	vspace	imageVertSpace	imageComponent
	hspace	imageHorizSpace	
	align	glueVertical	
	height	imageHeight	
	width	imageWidth	
	src	hyperLinkTarget	
p	align	glueHorizontal	
a	href	targetId	target
anchor/go	href	targetId	
navigational option	onpick	targetId	

Table 4. Attribute-to-attribute Mapping table

The above mapping table shows the attribute-to-attribute mapping, in which the value of

USIXML attribute is directly determined by the value of the corresponding attribute of the WML element. For more detailed mapping rules which also include the attribute which has not direct mapping, you can consult the appendix 3.

In the next chapter, with the transformation application and the XSLT style sheet, we will perform the transformation on the "real world" WML UIs to get the corresponding CUI model of USIXML.

Chapter 7. TRANSFORMATION EXAMPLES

From previous chapters, we get the transformation application and the XSLT style sheet for the transformation from WML 1.1 to USIXML. In this chapter, we will present some resulted CUI Model from the transformation with the source "real world" WAP UIs. Then, we verify the result of the transformations preformed by our transformation application in considering the mapping rules.

7.1 Introduction

In this chapter, we have realised 5 examples to compare the original WML code and the target USIXML code. Each example shows the result of transformation for certain WML elements. For showing the final UIs, we use The GraphiXML application. As the GraphiXML tool is an on going project with limited capacity, for instance, for the radioButton, combobox, item and cell elements it can't display properly yet, we use it to show the final UI of USIXML comparing the UI of WML only for the first 2 examples. We use both Openwave V7 Simulator and M3Gate Wap Simulator or Wapsilon V2.4 to show all the WML UIs from which we can also see the different look and feel by different WAP devices.

For the source of the first 4 examples, we present the results of the transformation for the real mobile site http://home.mobile.yahoo.com which is written in WML 1.1. For the last one, we write a WML file to show the transformation for table and td elements. The transformation is performed by the transformation application in using the XSLT style sheet (see section 3.4).

These examples cover all the major mapping from WML1.1 to USIXML. Further more, we show all the possible choices of mapping for WML elements through these

examples.

7.2 Example for element wml, card, p, b, anchor, a

This example shows a WML card which includes three kinds of basic elements: (1) The structure elements are wml element and card element. (2) The text layout elements are p element and b element. (3) The navigation elements are anchor element and a element. Through this example, we can test the transformation program and the XSLT style sheet for the above elements.

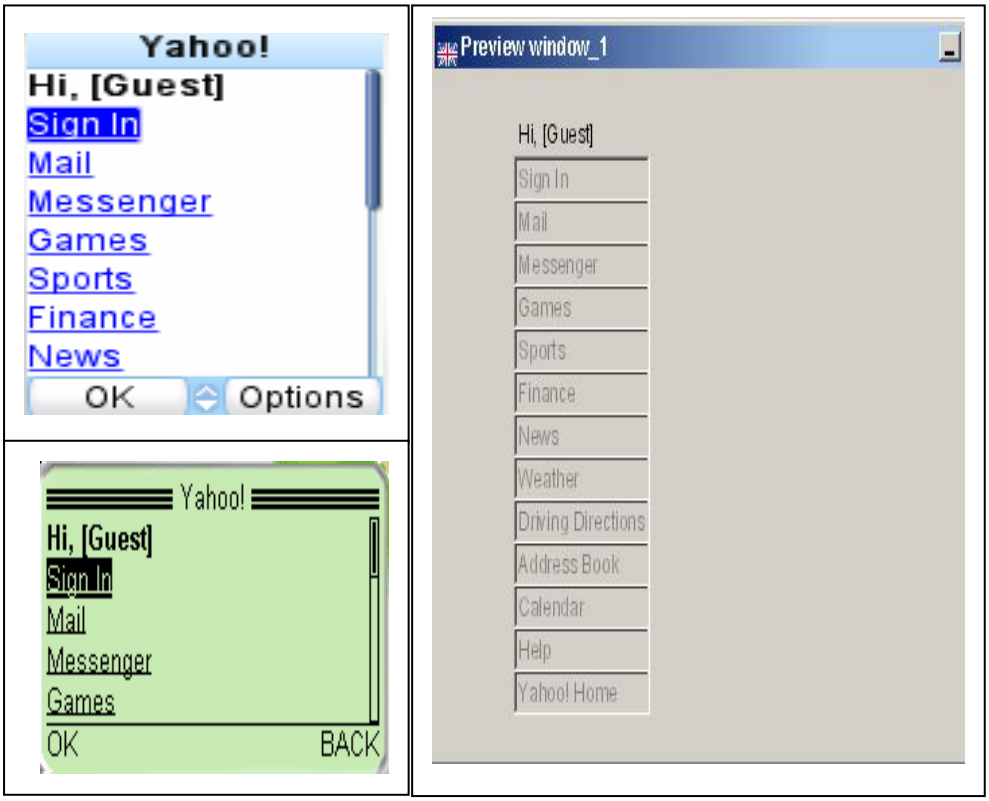


Fig. 35 UI of WML Vs. UI of USIXML

The Fig. 35 shows the comparable UIs displayed in the mobile simulator for the WML file and in the GraphiXML application for the transformed USIXML file. The mobile UI is the main card which has several web services provide to the user. From the resulted UI in the GraphiXML, we can see that it has just the same layout comparing that of GSM. For example, all the text links are left aligned, and the style of "Hi, [Guest] is bold.

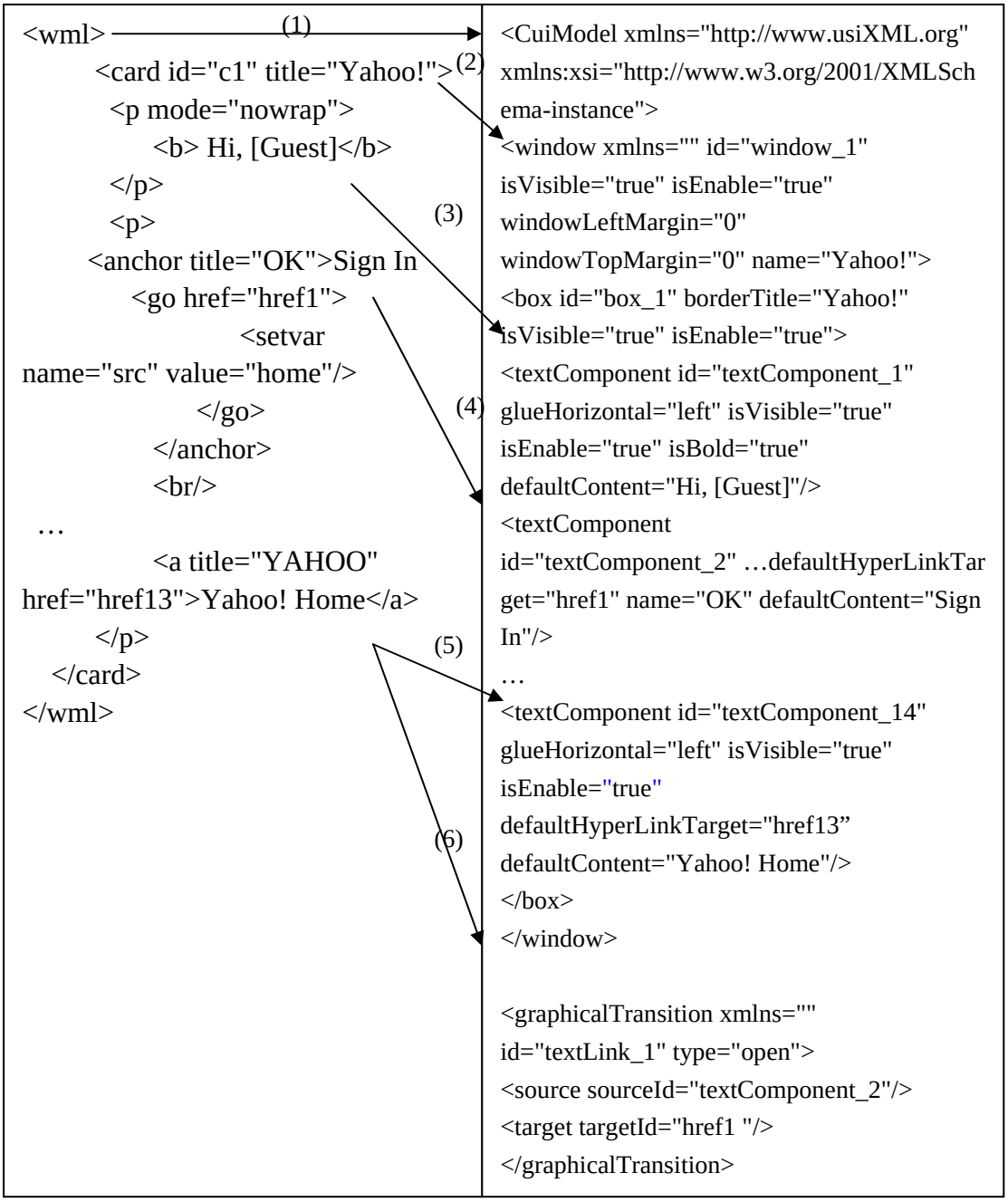


Fig. 36 WML Code Vs. CUI Model Code

The Fig.36 shows the resulted CUI Model transformed from the original WML code is perfectly corresponding the mapping rules presented in the chapter 6. In the resulted USIXML file: (1) the "CuiModel" element is created with the namespace attribute mapping to the "wml" element; (2) The only "window" element is created with three default attributes, and two mapped attributes: id = "window_1" , name = "Yahoo" respectively corresponding the "card" element's position and the value of title attribute in the source WML file; (3) For the "textComponent" element, taking the text content

"Hi, [guest]" for example, it create a "textComponent" element with defaultContent = "Hi, [guest]", also with isBold = "true" as the text is enclosed in a <b> tag, and with the value of glueHorizontal is equal to default value "left" as the <p> tag which enclose the text has not alignment attribute. (4) The "textComponent" with id = "textComponent_2", defaultContent= "Sign in", name = "ok" and defaultHyperLinkTarget = "href1" corresponding the "Sign in" text link enclosed in the "anchor" element. (5) The "textComponent" with id = "textComponent_14", defaultContent= "Yahoo! Home", and defaultHyperLinkTarget = "href13" corresponding the "Yahoo! Home" text link enclosed in the "a" element. (6) The "graphicalTransition" element with id = " textLink_13", the enclosed "source" element with sourceId = "textComponent_14" and "target" element with targetId = "href13" are created corresponding the "Yahoo! Home" text link.

7.3 Example for element input

This example shows a WML card with two types of input element. One is textual input element which maps to an empty textComponent element of USIXML. The other one is the password input element which maps to an empty textComponent element with isPassword attribute.

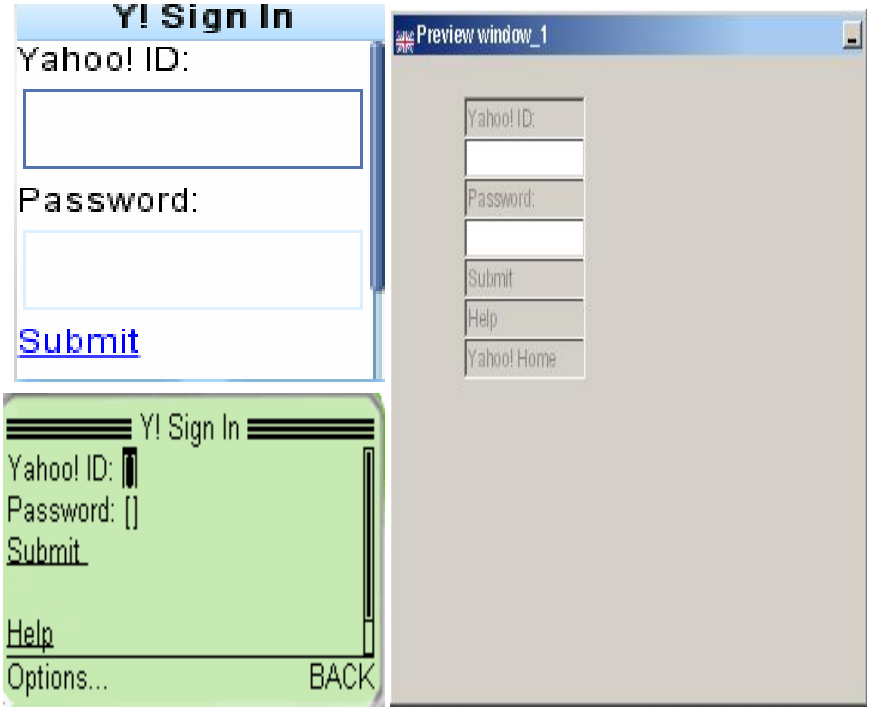


Fig.37 UI of WML Vs. UI of USIXML

The Fig. 37 shows the comparable UIs for the input example displayed in the mobile simulator for the WML file and in the GraphiXML application for the transformed USIXML file. The mobile UI ask for the user to input his Yahoo ID and corresponding password in the input fields. After input these information. The user can push the Submit text to send these information to the web service provider.

```
<wml>
      <card id="c2" title="Y! Sign
In">
          …
          <p>
                Yahoo! ID:
                <input name="login"
value="" format="*m" />
                Password:
                <input type="password"
name="passwd" value=""
format="*m" />
                <anchor
title="OK">Submit
                     <go method="post"
href="href1">
                       <postfield
name="dp" value="auth"/>
                       …
                     </go>
                </anchor>
                <br/>
                <a title="OK"
href="href2">Help</a>
                <br/>
…
                     </p>
     </card>
</wml>
```

(1)

(2)

```
<CuiModel xmlns="http://www.usiXML.org"
xmlns:xsi="http://www.w3.org/2001/XMLSch
ema-instance">
<window xmlns="" id="window_1"
isVisible="true" isEnable="true"
windowLeftMargin="0"
windowTopMargin="0" name="Y! Sign In">
<box id="box_1" borderTitle="Y! Sign In"
isVisible="true" isEnable="true">
<textComponent id="textComponent_1"
glueHorizontal="left" isVisible="true"
isEnable="true" defaultContent="Yahoo!
ID:"/>
<textComponent id="textComponent_2"
defaultContent=""… >
<textComponent id="textComponent_3"
defaultContent="Password:" … />
<textComponent id="textComponent_4"
isPassword="true" defaultContent="" … />
<textComponent id="textComponent_5"
defaultHyperLinkTarget="href1" name="OK"
defaultContent="Submit" … />
<textComponent id="textComponent_6"
defaultHyperLinkTarget="href2"
defaultContent="Help"…/>
…
</box>
</window>
<graphicalTransition xmlns=""
id="textLink_1" type="open">
<source sourceId="textComponent_5"/>
<target targetId="href1"/>
</graphicalTransition>
```

Fig. 38 WML Code Vs. CUI Model Code

The Fig.38 shows the resulted CUI Model transformed from the original WML code is

perfectly corresponding the mapping rules presented in the chapter 6. In the resulted USIXML file: (1) The "textComponent" element with id = "textComponent_2", defaultContent= " " is created corresponding the first "input" element in the source WML file. (2) The "textComponent" element with id = "textComponent_4", defaultContent= " ", and isPassword = "true" is created corresponding the second "input" element whose type attribute equals to password in the source WML file.

7.4 Examples for element select, option

As we've mentioned in the last chpter, the mappings for the select and option are one-to-several depending on different conditions. In this section, we present a series of examples to show all the possibilities for the mapping of select and option. For select there are two different mapping elements in USIXML which are comboBox and drop-down menu. For option there are 5 possibilities which are linked textComponent, comboBox item, menuItem, radioButton and checkbox.

7.4.1 Mapping option to linked textComponent

In this section, we show and test the resulted transformation for the mapping from option to linked textComponent. The mapping conditions are as following: (1) the value of multiple attribute is not equals to true; (2) there are no more than 6 options; (3) the option element has onpick attribute.

Fig.39 UI of WML

The Fig. 39 shows the UI for the select and option example displayed in the mobile simulator. It shows a game application. The user has three choices respectively for Playing game, checking the highest score, and consulting the game's rules.

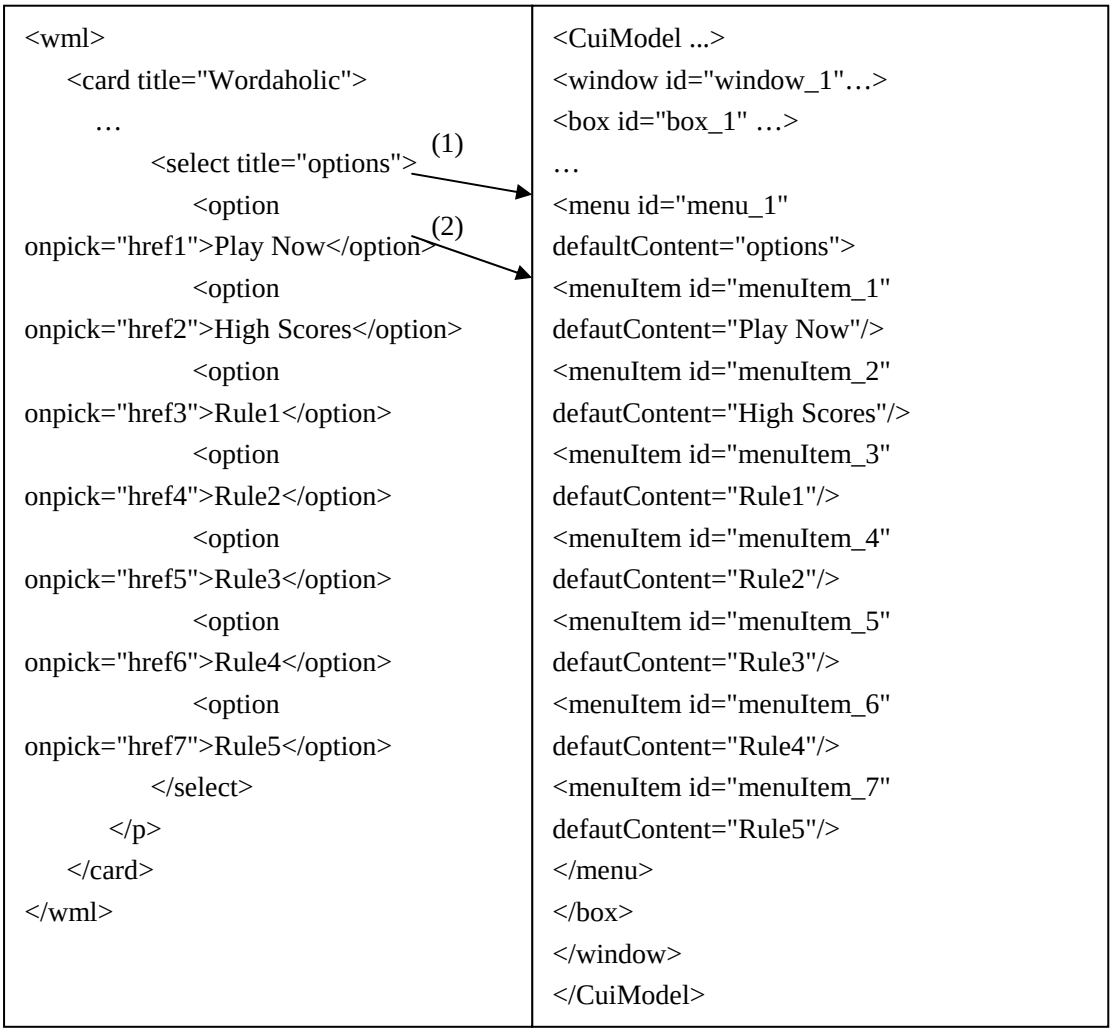
Fig. 40 WML Code Vs. CUI Model Code

The Fig.40 shows the resulted CUI Model transformed from the original WML code is perfectly corresponding the mapping rules presented in the chapter 6. In the resulted USIXML file: (1) The "textComponent" element with id = "textComponent_1", defaultContent= "Wordaholic(tm)", and glueHorizontal = "center" is created

corresponding the "Wordaholic(tm)" text in the source WML file. (2) The linked "textComponent" element with id = "textComponent_1", defaultContent= "Play Now", and glueHorizontal = "left" is created corresponding the first "option" element in the source WML file as the outer "select" element has not "multiple" attribute. (3) The "graphicalTransition" element with id = " textLink_3", the enclosed "source" element with sourceId = "textComponent_4" and "target" element with targetId = "href3" are created corresponding to the third choice option "Rules" .

7.4.2 Mapping select to drop-down Menu

The different condition between mapping option to liked textComponent and mapping it to menuItem is whether there are more than 6 options. With a little modification of last example, we add to 7 options such that it transforms the select element to the drop-down menu and the option element to menu item.




```
<wml>
    <card title="Wordaholic">
      …
          <select title="options">
              <option
onpick="href1">Play Now</option>
              <option
onpick="href2">High Scores</option>
              <option
onpick="href3">Rule1</option>
              <option
onpick="href4">Rule2</option>
              <option
onpick="href5">Rule3</option>
              <option
onpick="href6">Rule4</option>
              <option
onpick="href7">Rule5</option>
          </select>
      </p>
    </card>
</wml>
```

```
<CuiModel ...>
<window id="window_1"…>
<box id="box_1" …>
…
<menu id="menu_1"
defaultContent="options">
<menuItem id="menuItem_1"
defautContent="Play Now"/>
<menuItem id="menuItem_2"
defautContent="High Scores"/>
<menuItem id="menuItem_3"
defautContent="Rule1"/>
<menuItem id="menuItem_4"
defautContent="Rule2"/>
<menuItem id="menuItem_5"
defautContent="Rule3"/>
<menuItem id="menuItem_6"
defautContent="Rule4"/>
<menuItem id="menuItem_7"
defautContent="Rule5"/>
</menu>
</box>
</window>
</CuiModel>
```

Fig. 41 WML Code Vs. CUI Model Code

The Fig.41 shows the resulted CUI Model transformed from the original WML code is perfectly corresponding the mapping rules presented in the chapter 6. In the resulted USIXML file: (1) The "menu" element with id = "menu_1", and defaultContent="options" is created corresponding the select element in the source WML file. (2) The "menuItem" element with id = "menuItem_1" and defaultContent="Play Now" is created in the resulted USIXML code.

7.4.3 Mapping select to comboBox

Deferent from the above two example by not having onpick attribute for the option element, by the following example, we will show the transformation from select to comboBox and from option to comboBox item with the condition that there are more than 6 options.

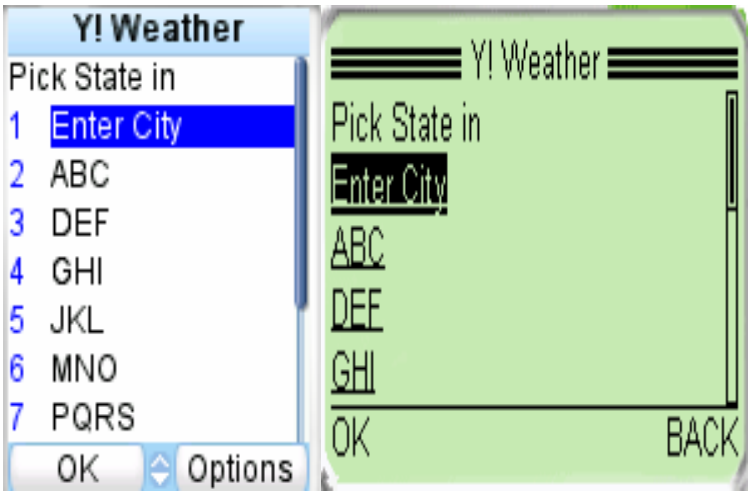


Fig. 42 UI of WML

The Fig. 42 shows the UI for the select and option example displayed in the mobile simulator. It shows a weather report application. The user has more than 6 options to choose the first character for the city in which the user is interested.

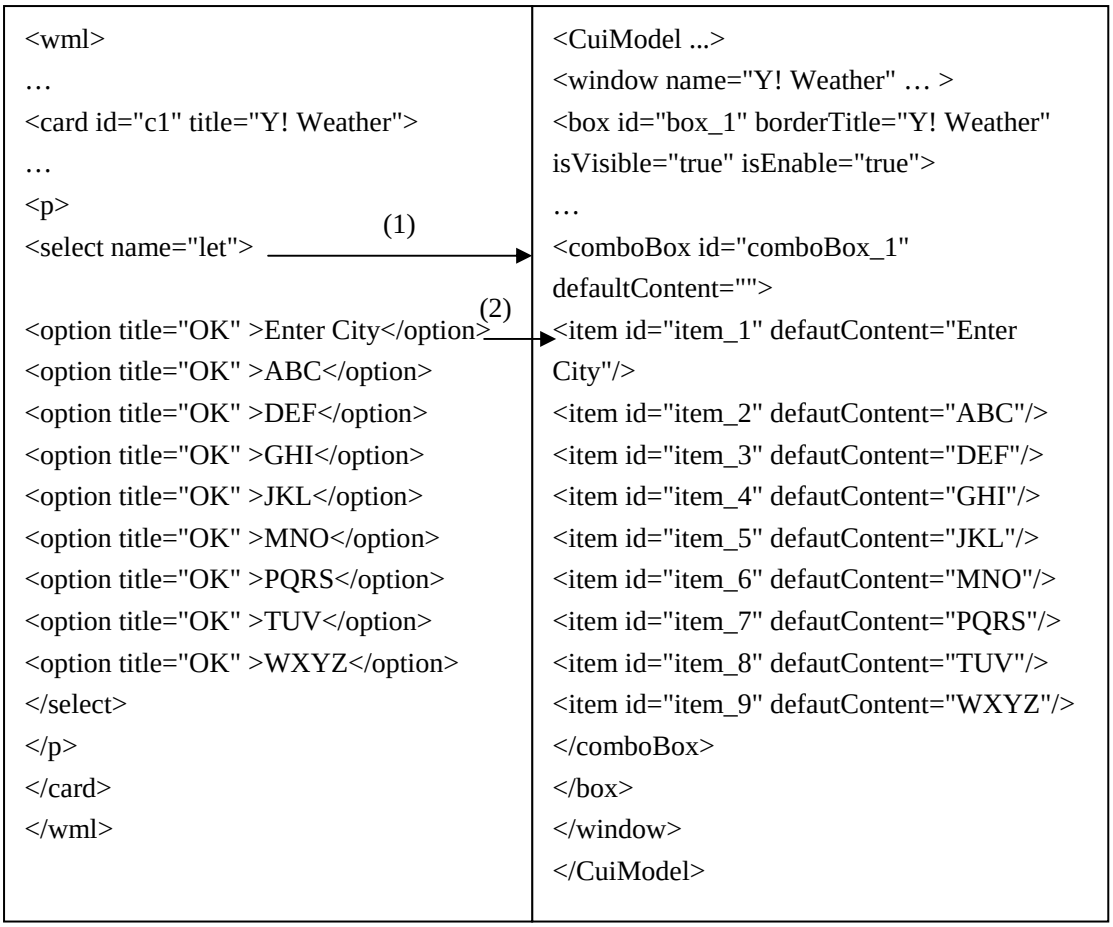


Fig. 43 WML Code Vs. CUI Model Code

From The Fig.43 above, we can see the wml source code has more than 6 options in a single "select" element. The resulted CUI Model transformed from the original WML code is perfectly corresponding the mapping rules presented in the chapter 6. In the resulted USIXML file: (1) The "comboBox" element with id = "comboBox_1" is created corresponding the "select" element in the source WML file as the enclosed options are more than 6. It is defined in our mapping rules for alternative mapping of "select" element. (2) The "item" element with id = "item_1" and defaultContent = "Play Now" is created corresponding the first option in the source WML file.

7.4.4 Mapping option to radioButton

Similar to the last example with no onpick attribute for the option element, we decrease the number of options to 6 such that the transformation program can transform option to radioButton with the condition that there are not more than 6 options.

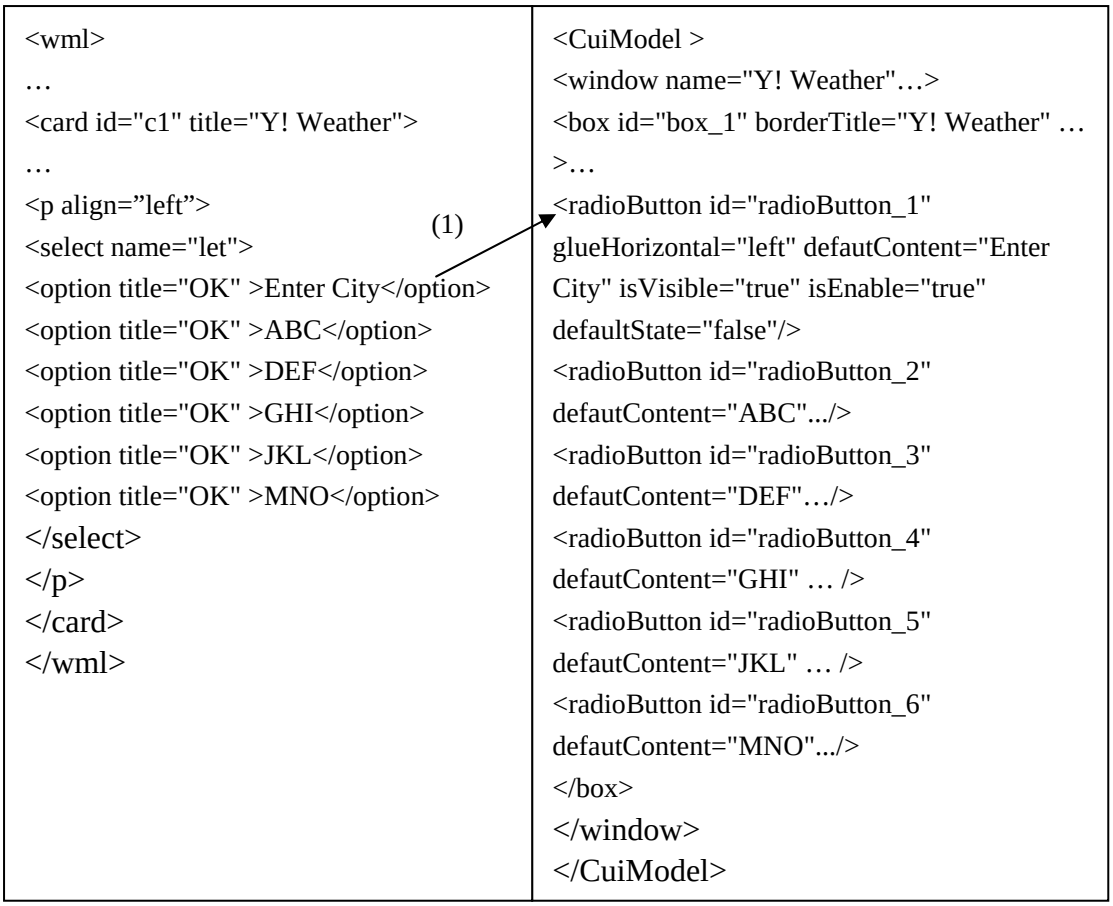


Fig. 44 WML Code Vs. CUI Model Code

From The Fig.44 above, we can see the wml source code has less than 7 options in a single “select” element. The resulted CUI Model transformed from the original WML code is perfectly corresponding the mapping rules presented in the chapter 6. In the resulted USIXML file: (1) The “radioButton” element with id = “radioButoon_1” and with defaultContent = “enter city” is created corresponding the “select” element in the source WML file as the enclosed options are less than 7.

7.4.5 Mapping option to checkbox

Deferent from all the above examples by having attribute multiple = “true” for the select element, Through the following multiple selection example, we show the transformation from option elemenet to checkbox element ignoring how many options and whether there is onpick attribute in the option element.

Fig. 45 WML Code Vs. CUI Model Code

From The Fig.45 above, we can see the select element has attribute multiple = "true" which means enable multiple choices for the following options. The resulted CUI Model transformed from the original WML code is perfectly corresponding the mapping rules presented in the chapter 6. In the resulted USIXML file: (1) The "checkBox" element corresponding to the multiple choice option. The glueHorizontal = "left" because it is enclosed in a p element with align="left" .The defaultContent = "Rule 1" which is the enclosed text content. And the defaultState = "false" is created which means this option is not pre-selected.

7.5 Example for element img

The following example shows the resulted transformation for img element to imageComponent in USIXML.

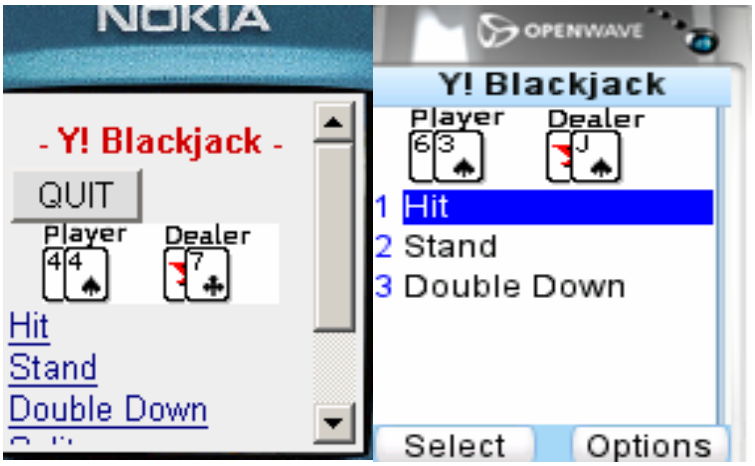


Fig.46 UI of WML

The Fig. 46 shows the UI for the "img" element example displayed in the mobile simulator. It shows a game application. The image shows the players pokers.

<wml> <card title="Y! Blackjack"> … </card> </wml>	<CuiModel ... > <window name="Y! Blackjack" … > <box borderTitle="Y! Blackjack" … > <imageComponent id="imageComponent_1" glueHorizontal="bottom" hyperLinkTarget="hrefOfImage" imageVertSpace="0" imageHorizSpace="0" glueVertical="bottom"/> … </box> </window> </CuiModel>

(1) →

Fig.47 WML Code Vs. CUI Model Code

From The Fig.47 above, we can see the resulted CUI Model transformed from the original WML code is perfectly corresponding the mapping rules presented in the chapter 6. In the resulted USIXML file: (1) The "imageComponent" element is created corresponding the "img" element in the source WML file. The values of glueHorizontal , imageVertSpace and imageHorizSpace are set by default and hyperLinkTarget equals to the src attribute of "img" element.

7.6 Example for element table, td

The following example shows the transformation result for the table and td element.

Fig. 48 UI of WML

As the example is not stored on the Web, the emulator other than Openwave can not locate to the wml file. We only show this UI by Openwave emulator. The Fig. 48 shows the UI for the "table" & "td" elements example displayed in the mobile simulator. It shows a employee information table. The first column is the employee's id, and the second is the employee's name.

Fig.49 WML Code Vs. CUI Model Code

(1) The “table” element with id = “table_1”, xSize = “3” which equals to the number of lines, ySize = “2” which equals to the number of columns, name = “employee_info” and glueHorizontal = “center” is created corresponding the “table” element in the source WML file.

(2) The “cell” element with id = “cell_1”, xIndex = “1”, yIndex = “1” and enclosed “textComponent” element with id = “textComponent_2” and defaultContent = “ID” is created corresponding the “td” element in line 1, column 1 of the source WML table.

(3) The “cell” element with id = “cell_2”, xIndex = “1”, yIndex = “2” and enclosed “textComponent” element with id = “textComponent_3” and defaultContent = “NAME” is created corresponding the “td” element in line 1, column 2 of the source WML table.

(4) The “cell” element with id = “cell_3”, xIndex = “2”, yIndex = “1” and enclosed “textComponent” element with id = “textComponent_4” and defaultContent = “7839” is created corresponding the “td” element in line 2, column 1 of the source WML table.

(5) The “cell” element with id = “cell_4”, xIndex = “2”, yIndex = “2” and enclosed “textComponent” element with id = “textComponent_5” and defaultContent = “KING” is created corresponding the “td” element in line 2, column 2 of the source WML table.

(6) The “cell” element with id = “cell_5”, xIndex = “3”, yIndex = “1” and enclosed “textComponent” element with id = “textComponent_6” and defaultContent = “7788” is created corresponding the “td” element in line 3, column 1 of the source WML table.

(7) The “cell” element with id = “cell_6”, xIndex = “3”, yIndex = “2” and enclosed “textComponent” element with id = “textComponent_7” and defaultContent = “SCOTT” is created corresponding the “td” element in line 3, column 2 of the source WML table.

Chapter 8. CONCLUSION

Today, the web services are growing at full speed with the development of information technology. At the other hand, the diversity of mobile devices is dramatically increasing. More and more people want to not only reach any information, anywhere, at anytime on the web, but also use as much different devices as possible. At the same time, the web service provider desire distributing their service for all the mobile devise. Therefore, the adaptation for the web UI to other platforms is a very hot issue.

As the WAP content authors can no longer afford to develop a content that is targeted for use for every platform. There is a need for common guidelines on how to provide multiplatform web services. In this thesis, our goal is to transform the WML, which is the dominant language for WAP service and therefore supported by all the mobile devise, to the CUI model of USIXML for adapt the WML UIs from one platform to others using according to the reverse-engineering approach. As the CUI model is more general than the Final UI, it can not only provide more flexibility for other platform adaptation, but also reserve the original design for the UIs without loosing information during the process of transformation. Therefore, by this way we can increase the WML UIs portability for remain the UIs usability by minimal efforts.

8.1 New results

For the UI reverse engineering, we firstly build the WML 1.1 meta-model with rational rose in comparing that of the USIXML. Then, we describe the mapping rules for WML 1.1 and USIXML. These mappings have four different types, which are element-to-element, element-to-attribute, text-to-element and attribute-to-attribute. In the element-to-element mapping, there are one-to-one mapping and one-to-several mapping. As for the one-to-several mapping, we use XPath and XSLT technology to perform the complex mapping procedure with respecting different mapping conditions.

For the attribute's value which can not be directly mapped from WML1.1, we use XPATH fuction to calculate it.

8.2 Advantages and disadvantages

To perform the transformation, we've developed a transformation tool in Java using JAXP. For the input style sheet, we use XSLT style sheet, which is an official recommendation of the World Wide Web Consortium (W3C) for XML transformation, to execute the transformation. It is a flexible, powerful language for transforming one XML document into another XML document. With the experience of XML transformation using XSLT, we found that the XSLT is not only an efficient but also an elegant tool which can satisfy most of the need for transformation from WML to USIXML.

One advantage of our approach is that, with the help of reverse engineering, we can reserve the original UIs design by CUI model which is platform-independent. Another advantage is that the transformation is very flexible. Especially for the transformations of select and option element, shown in the last chapter, we have a lot of choices for the mapped USIXML elements. For instance, the select element of WML may maps to comboBox or drop-down menu, while the option element of WML may maps to checkbox, textComponent, radioButton, menuItem, and comboBox item. In addition, there is no transformation that would be impossible to implement with Transformer based on XSLT as long as the corresponding transformation rule can be expressed in XPath notation.

In order to evaluate the performance of our transformation tool, average processing time of transformation from WML to USIXML, and mean transformation delay are measured. Table 1 shows the processing time of transformation, the average size of the input and output documents, and the number of WML document's nodes. As shown in Table 1, it shows similar processing time for the transformation of examples in last chapter, which shows the additional overhead due to number of source document's node is negligible. On average the average number of nodes is about 57, and the processing delay is about

0.88 seconds, which is relatively short considering the number of node. As a result, the transformation has a relative good performace with a short execution time.

Table 1 also shows that the output USIXML document is relatively larger than the input WML document. This is because the USIXML, which is multiple models language, is relatively more expressive than WML.

	Example7.2	Example7.3	Example7.4.1	Example7.5	Example7.6	Average Length
WML Document Size	1865	2017	735	969	750	1267.2(bytes)
WML Document nodes	81	95	31	33	43	57
USIXML Document Size	5429	1926	1380	1896	1829	2492(bytes)
Transformation Time	0.99	1.02	0.58	0.91	0.92	0.88(secs)

Table 1 Results of transformation

As the mapping is not very complete, due to that there is not always a USIXML element could be mapped to WML element, the disadvantage of this approach is lost of information after the transformation. For instance, as we didn't map the task elements, such as prev and refresh, to USIXML elements, if the target platform is also a small screen device which need such inter-navigation function, it should rebuild such mechanism for the final UIs.

8.3 Future works

In the future work, on the side of USIXML, it can implement a mechanism which corresponds the WML refresh and prev elements such that we can hold the original WMl UIs design for small screen devices. An additional ontimer element is also very helpful for automatic redirection for the target devices which have limited bandwidth access capability.

Because of XSLT contains semi-independent templates for each element or other node that will be processed, it is easier to modify or add transformation rules using XSLT than

other procedural programming language. On the other side of XSLT, if there has a new mapping, it should only add a semi-independent template for such elements in the XSLT style sheet without change the other part of it.

REFERENCES

1. MYER BRAD A., ROSSON M. B., Survey on user interface programming, Proceedings of CHI'92, May 3-7 1992, 195-202.

2. ISO9241 :1998. Ergonomic requirements for office work with visual display terminals. International Standard. The International Organization for Standardization. 1998.

3. REDWOOD SHORES, Link Developer's Guide Version 1.0., Unwired Planet, Inc. UP. California, July 1996.

4. COOPER and R. SHUFFLEBOTHAM, PDA Web Browsers: Implementation Issues, University of Kent at Canterbury Computing Laboratory WWW Page, November 1995.

5. H. LIE and BERT BOS, Cascading Style Sheets, WWW Consortium, September 1996.

6. B. BEDERSON and J. HOLLAN, Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. Proceedings of ACM UIST '94, ACM Press, 1994.

7. C. BROOKS, M. MAZER, S. MEEKS, J. MILLER, Application-Specific Proxy Servers as HTTP Stream Transducers, Fourth International World Wide Web Conference, Boston, December 1995.

8. JUHA KOLARI, TIMO LAAKKO, EIJA KAASINEN, Net in Pocket? Personal mobile access to web services, VTT Publications 464, ESPOO 2002.

9. ZHIYAN SHAO, ROBERT CAPRA, MANUEL A. PEREZ-QUINONES, Transcoding HTML to VoiceXML Using Annotation, 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03), Sacramento, California, USA, 2003

10. BOUILLON, L., VANDERDONCKT, J., CHOW, K.C., FLEXIBLE RE-ENGINEERING OF WEB SITES, PROC. OF 8TH ACM INT. CONF. ON IN-TELLIGENT USER INTERFACES IUI'2004 (FUNCHAL, 13-16 JANUARY 2004), ACM PRESS, NEW YORK, 2004, PP. 132-139.

11. BOMSDORF, B., SZWILLUS, G.: From Task to Dialogue: Task-Based User Interface Design. SIGCHI Bulletin 30, 4, 1998.

12. RAMSAY, M. AND NIELSEN, J. WAP Usability. Déjà Vu: 1994 All Over Again. Report

from a Field Study in London, Fall 2000.
13. NTT DoCoMo, Inc.(2001) All about i-mode.

14. MOORE, M.: Representation Issues for Reengineering Interactive Systems, ACM Computing Surveys, 28(4), 199-es, 1996.

15. SOUCHON, N., VANDERDONCKT, J., A Review of XML-Compliant User Interface Description Languages, Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS'2003 (Madeira, 4-6 June 2003), Jorge, J., Nunes, N.J., Falcao e Cunha, J. (Eds.), Lecture Notes in Computer Science, Vol. 2844, Springer-Verlag, Berlin, 2003, pp. 377-391

16. LIMBOURG, Q., VANDERDONCKT, J., UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence, in Matera, M., Comai, S. (Eds.), «Engineering Advanced Web Applications», Rinton Press, Paramus, 2004.

17. THEVENIN, D., COUTAZ, J.: Plasticity of User Interfaces: Framework and Research Agenda. In: Sasse, A., Johnson, Ch. (eds.): Proc. of IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact'99 (Edinburgh, August 1999). IOS Press, London (1999) 110–117.

18. BOUILLON, L., VANDERDONCKT, J., CHOW, K.C., Flexible Re-engineering of Web Sites, Proc. of 8th ACM Int. Conf. on In-telligent User Interfaces IUI'2004 (Funchal, 13-16 January 2004), ACM Press, New York, 2004, pp. 132-139.

19. LIMBOURG, Q., VANDERDONCKT, J., MICHOTTE, B., BOUILLON, L., FLORINS, M., TREVISAN, D., UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces, in Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages" (Gallipoli, May 25, 2004), Luyten, K., M. Abrams, Limbourg, Q., Vanderdonckt, J. (Eds.), Gallipoli, 2004, pp. 55-62.

20. CALVARY, G., COUTAZ, J., THEVENIN, D., LIMBOURG, Q., BOUIL-LON, L., VANDERDONCKT, J., A Unifying Reference Framework for Multi-Target User Interfaces, Interacting with Computers, Vol. 15, No. 3, June 2003, pp. 289-308.

21. EJ CHIKOFSKY, JH CROSS, Reverse Engineering and Design Recovery: A Taxonomy, 7 IEEE. Software 13 (1990) pp.14 -15.

22. www.wapforum.org, WML 2.0 Specification

23. VANDERDONCKT, J., A MDA-Compliant Environment for Developing User Interfaces of Information Systems, Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005), O. Pastor & J. Falcão e Cunha (eds.),

Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-31

24. www.uiml.org

25. ERIC M. BURKE, Java And XSLT, O'Reilly & Associates, Inc. USA, 2001

26. MANGANO S., XSLT Cookbook, O'Reilly & Associates, Inc. USA, 2003

27. Wireless Application Protocol Wireless Markup Language Specification Version 1.1, http://www1.wapforum.org/what/technical/SPEC-WML-19990616.pdf

28. USIXML Documentation Draft,

    http://www.usixml.org/index.php?download=usiXML-documentation-draft.pdf

29. ALI, M.F., PÉREZ-QUIÑONES M.A., ABRAMS M., BUILDING MULTI-PLATFORM USER INTERFACES WITH UIML, IN A. SEFFAH & H. JAVA-HERY (EDS.) MULTIPLE USER INTERFACES: ENGINEERING AND APPLI-CATION FRAMEWORK, JOHN WILEY AND SONS, 2003.

30. PATERNÒ, F., Model-Based Design and Evaluation of Interac-tive Applications, Springer-Verlag, Berlin, 2000.

31. Y. GAEREMYNCK, L. D. BERGMAN, T. LAU, "MORE for less: model recovery from visual interfaces for multi-device application design", Proc. of the international conference on Intelligent user interfaces, Jan 2003 (Miami, Florida, USA), ACM Press, New York, USA, 2003, pp. 69-76.

32. G. MORI, F. PATERNÒ, C. SANTORO, "Tool support for designing nomadic applications", Proc. of the 2003 international conference on Intelligent user interfaces, Jan 2003, (Miami ,USA), ACM Press, New York, USA, pp141-148

33. TIMOTHY W. BICKMORE, Digestor: Device-independent Access to the World Wide Web, Santa Clara, CA, 1997, http://citeseer.ist.psu.edu/bickmore97digestor.html

34. XML Tutorial, http://citeseer.ist.psu.edu/bickmore97digestor.html

35. XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath

36. XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt

APPENDIX 1. DTD OF WML 1.1

```
<!--Wireless Markup Language (WML) Document Type Definition. Copyright Wireless
Application Protocol Forum Ltd., 1998,1999. All rights reserved.
WML is an XML language. Typical usage:
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
...
</wml>
Terms and conditions of use are available from the Wireless Application Protocol
Forum Ltd. web site at http://www.wapforum.org/docs/copyright.htm.
-->
<!ENTITY % length "CDATA">
<!-- [0-9]+ for pixels or [0-9]+"%"
for percentage length -->
<!ENTITY % vdata "CDATA">
<!-- attribute value possibly containing
variable references -->
<!ENTITY % HREF "%vdata;">
<!-- URI, URL or URN designating a
hypertext node. May contain variable references -->
<!ENTITY % boolean "(true|false)">
<!ENTITY % number "NMTOKEN">
<!-- a number, with format [0-9]+ -->
<!ENTITY % coreattrs "id ID #IMPLIED
class CDATA #IMPLIED">
<!ENTITY % emph
"em | strong | b | i | u | big | small">
<!ENTITY % layout "br">
<!ENTITY % text "#PCDATA | %emph;">
<!-- flow covers "card-level" elements,
such as text and images -->
<!ENTITY % flow
"%text; | %layout; | img | anchor | a | table">
<!-- Task types -->
<!ENTITY % task "go | prev | noop | refresh">
<!-- Navigation and event elements -->
<!ENTITY % navelmts "do | onevent">
<!--============== Decks and Cards ==============-->
<!ELEMENT wml ( head?, template?, card+ )>
<!ATTLIST wml
```

```
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!-- card intrinsic events -->
<!ENTITY % cardev
"onenterforward %HREF; #IMPLIED
onenterbackward %HREF; #IMPLIED
ontimer %HREF; #IMPLIED"
>
<!-- card field types -->
<!ENTITY % fields
"%flow; | input | select | fieldset">
<!ELEMENT card (onevent*, timer?, (do | p)*)>
<!ATTLIST card
title %vdata; #IMPLIED
newcontext %boolean; "false"
ordered %boolean; "true"
xml:lang NMTOKEN #IMPLIED
%cardev;
%coreattrs;
>
<!--============== Event Bindings ==============-->
<!ELEMENT do (%task;)>
<!ATTLIST do
type CDATA #REQUIRED
label %vdata; #IMPLIED
name NMTOKEN #IMPLIED
optional %boolean; "false"
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT onevent (%task;)>
<!ATTLIST onevent
type CDATA #REQUIRED
%coreattrs;
>
<!--======== Deck-level declarations ========-->
<!ELEMENT head ( access | meta )+>
<!ATTLIST head
%coreattrs;
>
<!ELEMENT template (%navelmts;)*>
<!ATTLIST template
%cardev;
```

```
%coreattrs;
>
<!ELEMENT access EMPTY>
<!ATTLIST access
domain CDATA #IMPLIED
path CDATA #IMPLIED
%coreattrs;
>
<!ELEMENT meta EMPTY>
<!ATTLIST meta
http-equiv CDATA #IMPLIED
name CDATA #IMPLIED
forua %boolean; #IMPLIED
content CDATA #REQUIRED
scheme CDATA #IMPLIED
%coreattrs;
>
<!--================= Tasks ==================-->
<!ELEMENT go (postfield | setvar)*>
<!ATTLIST go
href %HREF; #REQUIRED
sendreferer %boolean; "false"
method (post|get) "get"
accept-charset CDATA #IMPLIED
%coreattrs;
>
<!ELEMENT prev (setvar)*>
<!ATTLIST prev
%coreattrs;
>
<!ELEMENT refresh (setvar)*>
<!ATTLIST refresh
%coreattrs;
>
<!ELEMENT noop EMPTY>
<!ATTLIST noop
%coreattrs;
>
<!--============== postfield ==============-->
<!ELEMENT postfield EMPTY>
<!ATTLIST postfield
name %vdata; #REQUIRED
value %vdata; #REQUIRED
%coreattrs;
```

```
>
<!--============== variables ==============-->
<!ELEMENT setvar EMPTY>
<!ATTLIST setvar
name %vdata; #REQUIRED
value %vdata; #REQUIRED
%coreattrs;
>
<!--============== Card Fields ==============-->
<!ELEMENT select (optgroup|option)+>
<!ATTLIST select
title %vdata; #IMPLIED
name NMTOKEN #IMPLIED
value %vdata; #IMPLIED
iname NMTOKEN #IMPLIED
ivalue %vdata; #IMPLIED
multiple %boolean; "false"
tabindex %number; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT optgroup (optgroup|option)+ >
<!ATTLIST optgroup
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT option (#PCDATA | onevent)*>
<!ATTLIST option
value %vdata; #IMPLIED
title %vdata; #IMPLIED
onpick %HREF; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT input EMPTY>
<!ATTLIST input
name NMTOKEN #REQUIRED
type (text|password) "text"
value %vdata; #IMPLIED
format CDATA #IMPLIED
emptyok %boolean; "false"
size %number; #IMPLIED
maxlength %number; #IMPLIED
```

```
tabindex %number; #IMPLIED
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT fieldset (%fields; | do)* >
<!ATTLIST fieldset
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT timer EMPTY>
<!ATTLIST timer
name NMTOKEN #IMPLIED
value %vdata; #REQUIRED
%coreattrs;
>
<!-- ============== Images ==============-->
<!ENTITY % IAlign "(top|middle|bottom)" >
<!ELEMENT img EMPTY>
<!ATTLIST img
alt %vdata; #REQUIRED
src %HREF; #REQUIRED
localsrc %vdata; #IMPLIED
vspace %length; "0"
hspace %length; "0"
align %IAlign; "bottom"
height %length; #IMPLIED
width %length; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!-- ============== Anchor ==============-->
<!ELEMENT anchor ( #PCDATA | br | img
| go | prev | refresh )*>
<!ATTLIST anchor
title %vdata; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT a ( #PCDATA | br | img )*>
<!ATTLIST a
href %HREF; #REQUIRED
title %vdata; #IMPLIED
```

```
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!--============== Tables ==============-->
<!ELEMENT table (tr)+>
<!ATTLIST table
title %vdata; #IMPLIED
align CDATA #IMPLIED
columns %number; #REQUIRED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT tr (td)+>
<!ATTLIST tr
%coreattrs;
>
<!ELEMENT td
( %text; | %layout; | img | anchor | a )*>
<!ATTLIST td
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!--== Text layout and line breaks ==-->
<!ELEMENT em (%flow;)*>
<!ATTLIST em
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT strong (%flow;)*>
<!ATTLIST strong
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT b (%flow;)*>
<!ATTLIST b
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT i (%flow;)*>
<!ATTLIST i
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT u (%flow;)*>
```

```
<!ATTLIST u
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT big (%flow;)*>
<!ATTLIST big
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT small (%flow;)*>
<!ATTLIST small
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ENTITY % TAlign "(left|right|center)">
<!ENTITY % WrapMode "(wrap|nowrap)" >
<!ELEMENT p (%fields; | do)*>
<!ATTLIST p
align %TAlign; "left"
mode %WrapMode; #IMPLIED
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ELEMENT br EMPTY>
<!ATTLIST br
xml:lang NMTOKEN #IMPLIED
%coreattrs;
>
<!ENTITY quot "&#34;">
<!-- quotation mark -->
<!ENTITY amp "&#38;#38;">
<!-- ampersand -->
<!ENTITY apos "&#39;">
<!-- apostrophe -->
<!ENTITY lt "&#38;#60;">
<!-- less than -->
<!ENTITY gt "&#62;">
<!-- greater than -->
<!ENTITY nbsp " ">
<!-- non-breaking space -->
<!ENTITY shy "&#173;">
<!-- soft hyphen (discretionary hyphen) -->
<!--
Copyright Wireless Application Protocol Forum Ltd., 1998,1999.All rights reserved. -->
```

APPENDIX2. XPATH FUNCTION LIBRARY

1 Node Set Functions

Function: *number* **position**()

The position function returns a number equal to the context position.

Function: *number* **count**(*node-set*)

The count function returns the number of nodes in the argument node-set.

2 String Functions

Function: *string* **concat**(*string*, *string*, *string**)

The concat function returns the concatenation of its arguments.

Function: *boolean* **contains**(*string*, *string*)

The contains function returns true if the first argument string contains the second argument string, and otherwise returns false.

Function: *string* **normalize-space**(*string*?)

The normalize-space function returns the argument string with whitespace normalized by stripping leading and trailing whitespace and replacing sequences of whitespace characters by a single space..

3 Boolean Functions

Function: *boolean* **not**(*boolean*)

The not function returns true if its argument is false, and false otherwise.

Function: *boolean* **true**()

The true function returns true.

Function: *boolean* **false**()

The false function returns false

APPENDIX 3. MAPPING TABLE OF WML AND USIXML

Mapping table of WML and UsiXML

source:WML1.1		target:USIXML	
Element	Attribute	Attribute	Element
card		id="window_'number of preceding card +1'"	window
	title	name	
		isVisible="true"	
		isEnable="true"	
		windowLeftMargin="0"	
		windowTopMargin="0"	
table		id="table_'number of preceding table +1'"	table
	title	name	
	columns	ySize	
	align	glueHorizontal	
		xSize="number of tr inside"	
td		id="cell_'number of preceding td+1'"	cell
		xIndex="number of preceding-cibling tr of parent node + 1"	
		yIndex="number of preceding-cibling td + 1"	
fieldset		id="box_'number of preceding fieldset+ancestor fieldset+preceding optgroup+2'"	box
	title	name	
	title	borderTitle	
optgroup		id="box_'number of (preceding fieldset+ancestor fieldset+preceding optgroup)+2'"	
	title	name	
	title	borderTitle	

input		id="textComponent_'number of (preceding input+preceding text()+ancestor text())+1'"	textComponent
	title	name	
	size	size	
	maxlength	maxlength	
	value	defaultContent	
	type="password"	isPassword="true"	
		isEditable="true"	
		isVisible="true"	
i		isItalic="true"	
em		isItalic="true"	
strong		isBold="true"	
u		isUnderline="true"	
b		isBold="true"	
big		textSize="12"	
small		textSize="8"	
	anchor/@title	name	
	anchor/go/@herf	hyperLinkTarget	
	a/@herf	hyperLinkTarget	
	p/@align(default:left)	glueHorizontal="left" as default	
		defaultContent="value of text()"	
select	title	defaultContent	comboBox
			menu
option		id="checkBox_'number of preceding checkBox's option +1'"	checkBox
	../@title	groupName	
	ivalue	defaultState="true" only if the ivalue contains the checkBox's position number.	

	p/@align(default:left)	glueHorizontal="left" as default	
		id="radioButton_'number of preceding radioButton option +1'"	radioButton
	select/@title	groupName	
	ivalue	defaultState	
	p/@align(default:left)	glueHorizontal="left" as default	
		id=" item_'number of preceding item+1'"	item
		defaultContent="text content"	
		id=" menuItem_'number of preceding menuItem+1'"	menuItem
		defaultContent="text content"	
		id=" textComponent_'number of preceding textComponent+1'"	textComponent
		defaultContent="text content"	
	onpick	defaultHyperLinkTarget	
img		id="imageComponent_'number of preceding img + 1'"	imageComponent
	vspace(default:0)	imageVertSpace="0" as default	
	hspace(default:0)	imageHorizSpace="0" as default	
	align(default:bottom)	glueVertical="bottom" as default	
	height	imageHeight	
	width	imageWidth	
	src	hyperLinkTarget	
p	align(default:left)	glueHorizontal="left" as default	
a		id="textLink_'number of (preceding a +preceding anchor+preceding option)+1'"	graphicalTransition
		type="open"	
		souceId="textComponet_'number of(preceding&ancestor text()+preceding input)+1'"	source
	href	targetId	target
anchor		id="textLink_'number of (preceding a +preceding anchor+preceding option)+1'"	graphicalTransition

		type="open"	
		souceId="textComponet_'number of(preceding&ancestor text()+preceding input)+1'"	source
	child:go/@href	targetId	target
navigational option		id="textLink_'number of (preceding a +preceding anchor+preceding option)+1'"	graphicalTransition
		type="open"	
		souceId="textComponet_'number of(preceding&ancestor text()+preceding input)+1'"	source
	onpick	targetId	target

APPENDIX 4. XSLT Style Sheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" indent="yes"/>
    <xsl:strip-space elements="*"/>
    <xsl:template match="/">
        <CuiModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.usiXML.org">
            <xsl:apply-templates/>
            <!--firstly, call all the templates which are not named -->
            <xsl:call-template name="transition_anchor"/>
            <xsl:call-template name="transition_a"/>
            <xsl:call-template name="transition_optionTextLink"/>
            <!--secondly, call the specific templates which deal the navigation
elements-->
        </CuiModel>
    </xsl:template>
    <xsl:template match="card">
        <!--1.template deal with the card element-->
        <xsl:if test="descendant::p">
            <!--set condition for the card element which has the p element inside-->
            <xsl:element name="window">
                <xsl:attribute name="id"><xsl:value-of
select="concat('window_',string(count(preceding::card)+1))"/></xsl:attribute>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="windowLeftMargin"><xsl:value-of
select="0"/></xsl:attribute>
                <xsl:attribute name="windowTopMargin"><xsl:value-of
select="0"/></xsl:attribute>
                <xsl:if test="@title">
                    <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
                </xsl:if>
                <xsl:element name="box">
                    <!--add a box element inside the window element-->
                    <xsl:attribute name="id"><xsl:value-of
select="concat('box_','1')"/></xsl:attribute>
                    <xsl:if test="@title">
                        <xsl:attribute name="borderTitle"><xsl:value-of
select="@title"/></xsl:attribute>
                    </xsl:if>
```

```
                        <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                        <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                        <xsl:apply-templates/>
                    </xsl:element>
                </xsl:element>
            </xsl:if>
        </xsl:template>
        <xsl:template match="optgroup">
            <!--2.1template deal with the optgroup element-->
            <xsl:element name="box">
                <xsl:attribute name="id"><xsl:value-of
select="concat('box_',string(1+count(preceding::optgroup)+count(preceding::fieldset)+co
unt(ancestor::fieldset)+1))"/></xsl:attribute>
                <xsl:if test="@title">
                    <xsl:attribute name="borderTitle"><xsl:value-of
select="@title"/></xsl:attribute>
                    <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
                </xsl:if>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:template>
    <xsl:template match="select">
            <!--2.2template deal with the option element-->
            <xsl:choose>
                <xsl:when test="@multiple='true'">
                    <xsl:for-each select="./option">
                        <xsl:element name="checkBox">
                            <xsl:attribute name="id"><xsl:value-of
select="concat('checkBox_',string(count(preceding::option[../@multiple='true'])+1))"/></
xsl:attribute>
                            <xsl:choose>
                                <xsl:when test="ancestor::p[@align]">
                                    <xsl:attribute name="glueHorizontal"><xsl:value-of
select="ancestor::p[@align]/@align"/></xsl:attribute>
                                </xsl:when>
                                <xsl:otherwise>
                                    <xsl:attribute name="glueHorizontal"><xsl:value-of
select="'left'"/></xsl:attribute>
                                </xsl:otherwise>
```

```
                                </xsl:choose>
                                <xsl:if test="../@title">
                                    <xsl:attribute name="groupName"><xsl:value-of
select="../@title"/></xsl:attribute>
                                </xsl:if>
                                <xsl:attribute name="defautContent"><xsl:value-of
select="./text()"/></xsl:attribute>
                                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                                <xsl:choose>
                                    <xsl:when
test="contains(string(../@ivalue),string(position()))">
                                        <xsl:attribute name="defaultState"><xsl:value-of
select="true()"/></xsl:attribute>
                                    </xsl:when>
                                    <xsl:otherwise>
                                        <xsl:attribute name="defaultState"><xsl:value-of
select="false()"/></xsl:attribute>
                                    </xsl:otherwise>
                                </xsl:choose>
                            </xsl:element>
                        </xsl:for-each>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:if test="count(./option) &gt; 6">
                          <xsl:choose>
                              <xsl:when test="./option[@onpick]">
                                <xsl:element name="menu">
                                  <xsl:attribute name="id"><xsl:value-of
select="concat('menu_',string(count(preceding::select[count(./option) &gt;
6])+1))"/></xsl:attribute>
                                    <xsl:attribute name="defaultContent"><xsl:value-of
select="@title"/></xsl:attribute>
                                    <xsl:for-each select="./option">
                                      <xsl:element name="menuItem">
                                          <xsl:attribute name="id"><xsl:value-of
select="concat('menuItem_',string(count(preceding::option)+1))"/></xsl:attribute>
                                          <xsl:attribute name="defautContent"><xsl:value-of
select="./text()"/></xsl:attribute>
                                      </xsl:element>
                                    </xsl:for-each>
                                </xsl:element>
                              </xsl:when>
                               <xsl:otherwise>
```

```
                        <xsl:element name="comboBox">
                          <xsl:attribute name="id"><xsl:value-of
select="concat('comboBox_',string(count(preceding::select[count(./option) &gt;
6])+1))"/></xsl:attribute>
                          <xsl:attribute name="defaultContent"><xsl:value-of
select="@title"/></xsl:attribute>
                          <xsl:for-each select="./option">
                              <xsl:element name="item">
                                  <xsl:attribute name="id"><xsl:value-of
select="concat('item_',string(count(preceding::option)+1))"/></xsl:attribute>
                                  <xsl:attribute name="defautContent"><xsl:value-of
select="./text()"/></xsl:attribute>
                              </xsl:element>
                          </xsl:for-each>
                        </xsl:element>
                      </xsl:otherwise>
                    </xsl:choose>
                  </xsl:if>
                      <xsl:if test="count(./option) &lt; 7">
                        <xsl:choose>
                          <xsl:when test="./option[@onpick]">
                            <xsl:for-each select="child::option">
                              <xsl:apply-templates/>
                            </xsl:for-each>
                          </xsl:when>
                          <xsl:otherwise>
                            <xsl:for-each select="child::option">
                              <xsl:element name="radioButton">
                                  <xsl:attribute name="id"><xsl:value-of
select="concat('radioButton_',string(count(preceding::option)-count(preceding::option[../
@multiple='true'])+1))"/></xsl:attribute>
                                  <xsl:choose>
                                      <xsl:when test="ancestor::p[@align]">
                                          <xsl:attribute
name="glueHorizontal"><xsl:value-of
select="ancestor::p[@align]/@align"/></xsl:attribute>
                                      </xsl:when>
                                      <xsl:otherwise>
                                          <xsl:attribute
name="glueHorizontal"><xsl:value-of select="'left'"/></xsl:attribute>
                                      </xsl:otherwise>
                                  </xsl:choose>
                                  <xsl:if test="../@title">
                                      <xsl:attribute name="groupName"><xsl:value-of
select="../@title"/></xsl:attribute>
                                  </xsl:if>
```

```
                                    <xsl:attribute name="defautContent"><xsl:value-of
select="./text()"/></xsl:attribute>
                                    <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                                    <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                                    <xsl:choose>
                                        <xsl:when test="position()=number(../@ivalue)">
                                            <xsl:attribute
name="defaultState"><xsl:value-of select="true()"/></xsl:attribute>
                                        </xsl:when>
                                        <xsl:otherwise>
                                            <xsl:attribute
name="defaultState"><xsl:value-of select="false()"/></xsl:attribute>
                                        </xsl:otherwise>
                                    </xsl:choose>
                                </xsl:element>
                            </xsl:for-each>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:if>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
    <xsl:template match="input">
        <!--2.3template deal with the input element-->
        <xsl:element name="textComponent">
            <xsl:attribute name="id"><xsl:value-of
select="concat('textComponent_',string(count(preceding::input)+count(preceding::text())
+count(ancestor::text())+1))"/></xsl:attribute>
            <xsl:choose>
                <xsl:when test="ancestor::p[@align]">
                    <xsl:attribute name="glueHorizontal"><xsl:value-of
select="ancestor::p[@align]/@align"/></xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:attribute name="glueHorizontal"><xsl:value-of
select="'left'"/></xsl:attribute>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:if test="@title">
                <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
            </xsl:if>
            <xsl:if test="@size">
                <xsl:attribute name="size"><xsl:value-of
```

```
select="@size"/></xsl:attribute>
                </xsl:if>
                <xsl:if test="@maxlength">
                    <xsl:attribute name="maxlength"><xsl:value-of
select="@maxlength"/></xsl:attribute>
                </xsl:if>
                <xsl:if test="@type='password'">
                    <xsl:attribute name="isPassword"><xsl:value-of
select="true()"/></xsl:attribute>
                </xsl:if>
                <xsl:attribute name="defaultContent"><xsl:value-of
select="@value"/></xsl:attribute>
                <xsl:attribute name="isEditable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
            </xsl:element>
        </xsl:template>
        <xsl:template match="table">
            <!--3.1template deal with the table element-->
            <xsl:element name="table">
                <xsl:attribute name="id"><xsl:value-of
select="concat('table_',string(count(preceding::table)+1))"/></xsl:attribute>
                <xsl:attribute name="ySize"><xsl:value-of
select="@columns"/></xsl:attribute>
                <xsl:attribute name="xSize"><xsl:value-of
select="count(child::tr)"/></xsl:attribute>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:if test="@title">
                    <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
                </xsl:if>
                <xsl:if test="@align">
                    <xsl:attribute name="glueHorizontal"><xsl:value-of
select="@align"/></xsl:attribute>
                </xsl:if>
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:template>
        <xsl:template match="td">
            <!--3.2template deal with the td element-->
            <xsl:element name="cell">
                <xsl:attribute name="id"><xsl:value-of
```

```
select="concat('cell_',string(count(preceding::td)+1))"/></xsl:attribute>
                <xsl:attribute name="xIndex"><xsl:value-of
select="count(../preceding-sibling::tr)+1"/></xsl:attribute>
                <xsl:attribute name="yIndex"><xsl:value-of
select="count(preceding-sibling::td)+1"/></xsl:attribute>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:template>
        <xsl:template match="fieldset">
            <!--3.3template deal with the fieldset element-->
            <xsl:element name="box">
                <xsl:attribute name="id"><xsl:value-of
select="concat('box_',string(1+count(preceding::fieldset)+count(ancestor::fieldset)+count
(preceding::optgroup)+1))"/></xsl:attribute>
                <xsl:if test="@title">
                    <xsl:attribute name="borderTitle"><xsl:value-of
select="@title"/></xsl:attribute>
                    <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
                </xsl:if>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:template>
        <xsl:template match="text()">
            <!--3.4template deal with the text content-->
            <xsl:element name="textComponent">
                <xsl:attribute name="id"><xsl:value-of
select="concat('textComponent_',string(count(preceding::text())+count(ancestor::text())+
count(preceding::input)+1))"/></xsl:attribute>
                <xsl:choose>
                    <xsl:when test="ancestor::p[@align]">
                        <xsl:attribute name="glueHorizontal"><xsl:value-of
select="ancestor::p[@align]/@align"/></xsl:attribute>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:attribute name="glueHorizontal"><xsl:value-of
select="'left'"/></xsl:attribute>
                    </xsl:otherwise>
```

```
                </xsl:choose>
                <xsl:attribute name="isVisible"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:attribute name="isEnable"><xsl:value-of
select="true()"/></xsl:attribute>
                <xsl:for-each select="ancestor::*">
                    <xsl:if test="local-name(.)='anchor'">
                        <xsl:attribute name="defaultHyperLinkTarget"><xsl:value-of
select="*/@href"/></xsl:attribute>
                        <xsl:if test="@title">
                            <xsl:attribute name="name"><xsl:value-of
select="@title"/></xsl:attribute>
                        </xsl:if>
                    </xsl:if>
                    <xsl:if test="local-name(.)='option'">
                       <xsl:if test="./@onpick">
                        <xsl:attribute name="defaultHyperLinkTarget"><xsl:value-of
select="@onpick"/></xsl:attribute>
                        </xsl:if>
                    </xsl:if>
                    <xsl:if test="local-name(.)='a'">
                        <xsl:attribute name="defaultHyperLinkTarget"><xsl:value-of
select="@href"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='b'">
                        <xsl:attribute name="isBold"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='em'">
                        <xsl:attribute name="isItalic"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='strong'">
                        <xsl:attribute name="isBold"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='i'">
                        <xsl:attribute name="isItalic"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='u'">
                        <xsl:attribute name="isUnderline"><xsl:value-of
select="true()"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='small'">
                        <xsl:attribute name="textSize"><xsl:value-of
```

```
select="8"/></xsl:attribute>
                    </xsl:if>
                    <xsl:if test="local-name(.)='big'">
                        <xsl:attribute name="textSize"><xsl:value-of
select="12"/></xsl:attribute>
                    </xsl:if>
                </xsl:for-each>
                <xsl:attribute name="defaultContent"><xsl:value-of
select="normalize-space()"/></xsl:attribute>
            </xsl:element>
        </xsl:template>
        <xsl:template match="img">
            <!--4.1template deal with the img element-->
            <xsl:element name="imageComponent">
                <xsl:attribute name="id"><xsl:value-of
select="concat('imageComponent_',string(count(preceding::img)+1))"/></xsl:attribute>
                <xsl:choose>
                    <xsl:when test="ancestor::p[@align]">
                        <xsl:attribute name="glueHorizontal"><xsl:value-of
select="@align"/></xsl:attribute>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:attribute name="glueHorizontal"><xsl:value-of
select="'left'"/></xsl:attribute>
                    </xsl:otherwise>
                </xsl:choose>
                <xsl:attribute name="hyperLinkTarget"><xsl:value-of
select="@src"/></xsl:attribute>
                <xsl:choose>
                    <xsl:when test="@vspace">
                        <xsl:attribute name="imageVertSpace"><xsl:value-of
select="@vspace"/></xsl:attribute>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:attribute name="imageVertSpace"><xsl:value-of
select="0"/></xsl:attribute>
                    </xsl:otherwise>
                </xsl:choose>
                <xsl:choose>
                    <xsl:when test="@hspace">
                        <xsl:attribute name="imageHorizSpace"><xsl:value-of
select="@hspace"/></xsl:attribute>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:attribute name="imageHorizSpace"><xsl:value-of
select="0"/></xsl:attribute>
```

```
                    </xsl:otherwise>
                </xsl:choose>
                <xsl:choose>
                    <xsl:when test="@align">
                        <xsl:attribute name="glueVertical"><xsl:value-of
select="@align"/></xsl:attribute>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:attribute name="glueVertical"><xsl:value-of
select="'bottom'"/></xsl:attribute>
                    </xsl:otherwise>
                </xsl:choose>
                <xsl:if test="@height">
                    <xsl:attribute name="imageHeight"><xsl:value-of
select="@height"/></xsl:attribute>
                </xsl:if>
                <xsl:if test="@width">
                    <xsl:attribute name="imageWidth"><xsl:value-of
select="@width"/></xsl:attribute>
                </xsl:if>
            </xsl:element>
        </xsl:template>
        <xsl:template name="transition_a">
            <!--4.2template deal with the a element-->
            <xsl:for-each select="//a">
                <xsl:element name="graphicalTransition">
                    <xsl:attribute name="id"><xsl:value-of
select="concat('textLink_',string(count(preceding::a)+count(preceding::anchor)+1))"/></
xsl:attribute>
                    <xsl:attribute name="type"><xsl:value-of
select="'open'"/></xsl:attribute>
                    <xsl:element name="source">
                        <xsl:attribute name="sourceId"><xsl:value-of
select="concat('textComponent_',string(count(preceding::text())+count(ancestor::text())+
count(preceding::input)+1))"/></xsl:attribute>
                    </xsl:element>
                    <xsl:element name="target">
                        <xsl:attribute name="targetId"><xsl:value-of
select="@href"/></xsl:attribute>
                    </xsl:element>
                </xsl:element>
            </xsl:for-each>
        </xsl:template>
        <xsl:template name="transition_anchor">
            <!--4.3template deal with the anchor element-->
            <xsl:for-each select="//anchor">
```

```
                <xsl:element name="graphicalTransition">
                    <xsl:attribute name="id"><xsl:value-of
select="concat('textLink_',string(count(preceding::a)+count(preceding::anchor)+1))"/></
xsl:attribute>
                    <xsl:attribute name="type"><xsl:value-of
select="'open'"/></xsl:attribute>
                    <xsl:element name="source">
                        <xsl:attribute name="sourceId"><xsl:value-of
select="concat('textComponent_',string(count(preceding::text())+count(ancestor::text())+
count(preceding::input)+1))"/></xsl:attribute>
                    </xsl:element>
                    <xsl:element name="target">
                        <xsl:attribute name="targetId"><xsl:value-of
select="child::go/@href"/></xsl:attribute>
                    </xsl:element>
                </xsl:element>
            </xsl:for-each>
        </xsl:template>
        <xsl:template name="transition_optionTextLink">
            <!--4.4template deal with the option element for linked textComponent-->
            <xsl:for-each select="//option[@onpick]">
              <xsl:if test="count(../option) &lt; 7">
              <xsl:if test="not(string(../@multiple)='true')">
                        <xsl:element name="graphicalTransition">
                            <xsl:attribute name="id"><xsl:value-of
select="concat('textLink_',string(count(preceding::a)+count(preceding::anchor)+count(pr
eceding::option)+1))"/></xsl:attribute>
                            <xsl:attribute name="type"><xsl:value-of
select="'open'"/></xsl:attribute>
                        <xsl:element name="source">
                            <xsl:attribute name="sourceId"><xsl:value-of
select="concat('textComponent_',string(count(preceding::text())+count(ancestor::text())+
count(preceding::input)+1))"/></xsl:attribute>
                        </xsl:element>
                        <xsl:element name="target">
                           <xsl:attribute name="targetId"><xsl:value-of
select="@onpick"/></xsl:attribute>
                        </xsl:element>
                </xsl:element>
              </xsl:if>
              </xsl:if>
            </xsl:for-each>
        </xsl:template>
</xsl:stylesheet>
```