Mobile Information Systems 5 (2009) 125–139
DOI 10.3233/MIS-2009-0077
IOS Press

XPL the eXtensible presentation language

A. Santangelo[a,*], A. Gentile[a], G. Vella[b], N. Ingraffia[b] and M. Liotta[a]
[a]*Dipartimento di Ingegneria Informatica, Palermo, Italy*
[b]*Research and Development Laboratory of Engineering Ingegneria Informatica S.p.A., Italy*

Abstract. The last decade has witnessed a growing interest in the development of web interfaces enabling both multiple ways to access contents and, at the same time, fruition by multiple modalities of interaction (point-and-click, contents reading, voice commands, gestures, etc.). In this paper we describe a framework aimed at streamlining the design process of multi-channel, multimodal interfaces enabling full reuse of software components. This framework is called the eXtensible Presentation architecture and Language (XPL), a presentation language based on design pattern paradigm that keeps separated the presentation layer from the underlying programming logic. The language supplies a methodology to expedite multimodal interface development and to reduce the effort to implement interfaces for multiple access devices, by means of using the same code. This paper describes a methodology approach based on Visual Design Pattern (ViDP) and Verbal Design Pattern (VeDP), offering examples of multimodal and multichannel interfaces created with the XPL Editor.

Keywords: Design patterns, verbal interaction, presentation languages, data access, multimodal interfaces for mobile systems

1. Introduction

In the past five years, an increasing number of technologies and applications have begun to focus on mobile computing and the wireless access to the web. Interface design research has been devoted quite exclusively to determine mechanisms to build easy to use and intuitive interfaces, so that users can perform a given task more smoothly and efficiently. This growing interest for multimodal interaction and mobile device motivated us to explore the creation of a design methodology aimed at providing users with multiple ways of interaction. A user can therefore choose between the available ways of interaction according to his preferences and needs, deciding whether to prompt verbally the system or to rely on traditional point and click.

The advantage of relying on multiple interaction modalities is twofold. First, usability is increased simplifying the relationship between tools and users, especially for processes related to complex technologies [15]. The weaknesses of one modality are offset by the strengths of another. For instance, on a mobile device with a small screen and keypad, a word may be quite difficult to type but very easy to say [1]. Secondly, accessibility features are expanded in terms of availability of services under different access scenarios. A well-designed multimodal application can thus be used by people with either permanent physical impairments or temporary, context depending ones. Such multimodal interaction is especially relevant for mobile devices where traditional user interface peripherals may not be available or appropriate to the needs and where multimodality improves the accessibility of pervasive services [2, 3].

*Corresponding author. Tel.: +39 091 7028529; Fax: +39 091 6598043; E-mail: santangelo@dinfo.unipa.it.

On the other hand, each modality is unfortunately tied to specific technologies and interface languages. No common standards are available for mobile interface design. Every newly available access device needs a full interface redesign. As a consequence, interface designers tend to avoid direct use of device-specific languages and rather rely on consolidated technologies.

To address this issue, this paper describes the eXtensible Presentation architecture and Language (XPL), a framework aimed at streamlining an interface design process that incorporates multichannel and multimodal capabilities and enables full reuse or software components. XPL allows interface description in a device-independent (XML-based) language and fully incorporates the concept of design patterns. Consolidated interface design patterns and schemas can this be reused with minimal effort in all those scenarios when the same use-case is re-occurring. The XPL framework does so by separating contents retrieval from publishing tasks while maintaining a clear distinction between client application and programming logic. All channel specific programming is embedded into a corresponding User Interface Design Patterns (UIDPs), so to access contents through a specific channel simply requires instantiating the corresponding pattern. Language scalability for newly available devices can be easily obtained by simply updating the channel library with the corresponding set of code translation routines.

Currently XPL supports two kinds of UIDP: Visual Design Pattern (ViDP) and Verbal Design Pattern (VeDP), respectively for visual and verbal interfaces. In addition, a Relation Pattern has been implemented, to provide synchronization between visual e verbal elements. Depending on contexts and situations, users can choose and alternate between the preferred interaction modality.

The paper is organized as follows: Section 2 gives an overview of related works; Section 3 gives an overview of XPL and its features; Section 4 describes XPL architecture and language constructs, along with some implementation examples; finally, some conclusive remarks are given in the conclusions in Section 5.

2. Background

2.1. Design pattern use and usability

In the last twenty years designing models have been based on usability and final users have increasingly obtained a central role giving life to the *User Centered Design* (*UCD*) *paradigm*. During the XPL design phase we have taken into account not only the usability from the users point of view but also from the developers point of view.

From the *user*'s perspective, usability is related to how XPL is able to meet the needs and expectations of users in interface building using knowledge already employed in interface designs.

From the *developer*'s perspective, usability is related to how XPL is easy to use and helps both novice and experienced designers to make the right design decisions and to prevent the designers from making the same mistakes over and over again taking into account the *design patterns* methodology. Christopher Alexander is the first one who applied this paradigm to architecture issues [7], stating that each pattern captures proven design knowledge providing solutions for recurrent problems coming up against several design problems which occur over and over again even in different contexts. The challenge with design patterns is to keep track of the core design solution originally devised by pattern creators that constitutes the valuable knowledge to share and reuse in future designs. In order to improve and optimize interface design that has to solve problems in simpler or more complex contexts, it is necessary to arrange design pattern languages into patterns groups and networks.

To define different connection models among patterns, many pattern collections have been published [7] and some structures for pattern languages have been explored as in [8,9]. According to the evolution of pattern design in software engineering, the use of patterns in User Interface Design (UID), or related fields such as web design and GUI design, is slowly gaining momentum in practice. After initial investigations on the applicability of patterns for Interaction Design [13], actual patterns collections are now publicly available in [14–17]. In the rest of the paper UIDPs for visual interfaces are called Visual Design Patterns (VDPs). Early interests for User Interface Design Patterns (UIDPs) date back to 1994 [10], but a proper effective set of such patterns has not emerged yet. Some attempts have been made to create several different kinds of patterns for UID as suggested in [12].

2.2. User interface languages and accessibility

A key attribute of interfaces for final user accomplishment is the *accessibility*. In the computer field, accessibility notion is related to the capability that systems have to provide services and information according to the available technology knowledge. No discrimination to user with disabilities needing particular provisioning are allowed: assistive technologies and particular configurations are encouraged.

For instance, a software that meet various needs and preferences is helpful not only for disabled people but for many other users (for instance, think to people using a slow Internet connection, to people with temporary disabilities and to elderly people).

Many accessibility features are easily implemented if they are planned from the beginning of interface development, but fixing inaccessible interfaces can require significant effort. In the following paragraphs a set of languages for user interface development is described and discussed. Due to the enormous usage of the Web, there has been an explosion of ways to create user interfaces (UIs) for Web applications: *DHTML* [18], *XwingML* for traditional desktop applications, to describe documents; the Wireless Markup Language (*WML*) for mobile devices like cell phones with limited size displays 2; *SpeechML* and Voice Markup Language (*VoxML*), for voice-enabled devices like conventional telephones.

To enable user of alternative technologies to improve application accessibility, also for people with disabilities [5], a user interface language capable to manage multiple interfaces is needed. Such a language should also be extensible to interface technologies not available yet. Furthermore, another desirable property is to design interfaces using a language that offers a natural way to separate user interface code from internal logic one, as these section are customarily assigned to separate groups of designers. In this regard, UIML (User Interface Markup Language) is the earliest pioneer in user interface markup languages. It is an XML language that allows to describe the user interface in declarative terms and abstract it generating user interfaces for different platforms, like PDAs [19]. Other well known languages are XUL (XML User interface Language) [20]and its dialects for generic multi-platform applications logic available both on-line and off-line, AUIML [21], Thinlet and SVG (Scalable Vector Graphics) that is a standard markup language for graphics that can support rich, graphical user interface for web and mobile applications that includes support for vector/raster graphics, animation, embedded media, events and scripts that, allow to create rich user interfaces.

In the following section, the XPL eXtensible Presentation Language for user interface design based on visual and verbal design patterns is described in details. Purpose of this language is to overcome bounds presented in over described languages, considering for each channel the corresponding conceptual artifacts, linking related artifacts among different channels.

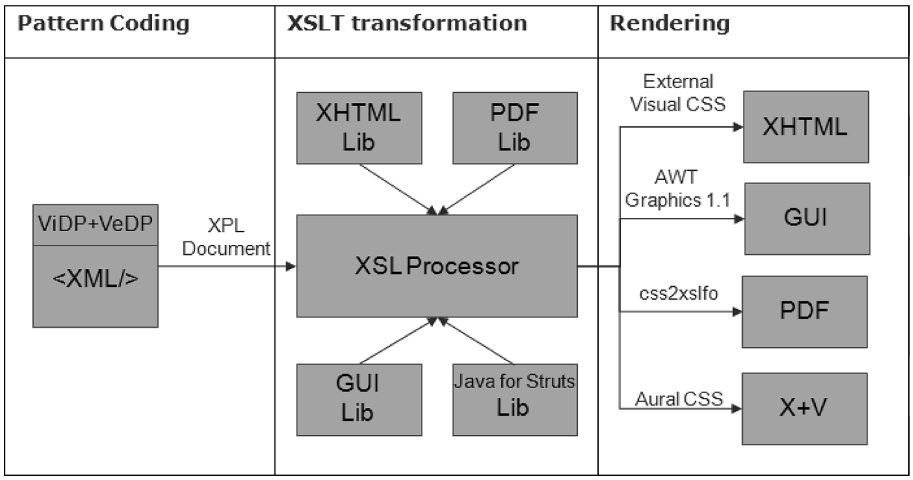

Fig. 1. XPL logic architecture.

3. XPL outline and feature

The eXtensible Presentation Language (XPL) is an XML-based User Interface Language (*UIL*) that aims at providing user interface designers with a methodology to design multimodal and multichannel interfaces using design patterns. Interface development is carried on by the XPL Editor, an ad-hoc editor based on the Eclipse Modeling Framework (EMF).

Designers, trough the XPL Editor, can create XPL documents, preview them, modify the properties through JFace propertysheets, and export a prototype, ready to be processed for business logic and binding.

XPL main features include i) multichannel development ii) multimodal interaction, and, iii) extensibility to include new features and capabilities. The next three sections describe these features.

3.1. Multichannel features

XPL has been designed to supply multi-channel features. Distinct stylesheets, called eXtensible Stylesheets Language for Transformations (XSLT), correspond to different devices. Channels which XPL can currently deal with, through ad-hoc libraries, are four: Web channel, both on pc and hand-held device; GUI (Graphical User Interface), printable format and Java Classes to be used with the XDPM (as shown in Fig. 1). With reference to the web channel, the XPL code becomes available for the web channel in a single step, where a set of presentation templates are applied recursively to the XPL source code transforming from XML to XHTML.

The layout for the Web channel output mainly differs from that one for PDA because of the display size. XPL provides a solution to this issue passing some parameters, through the template attribute, to the XSL for the correct usage of the related Cascading Style Sheets to hide or reduce space with appropriate CSS classes for mobile device.

3.2. Multimodal features

Sometimes, the same technological solution can be helpful for different kind of disabilities. For instance, both blind people and people who cannot use their hands need full keyboard equivalents for

mouse commands in browsers and authoring tools, because they cannot use a mouse but they can use assistive technologies to activate commands supported by a standard keyboard interface. Moreover, a spoken output not only benefits blind users, but also Web users whose eyes are busy with other tasks; while captions for audio not only benefit deaf users, but also increase the efficiency of indexing and searching for audio contents on Websites.

The new paradigm of multimodal interface brings together visual and verbal activity during user-computer interaction. Verbal Design Patterns are design and development schemas intended as guides to realize homogeneous templates for the verbal interaction of a multimodal interface. Multimodal interaction can be achieved adding verbal interaction to visual interaction: in such case user can switch between one or the other modality according to his needs or complementing visual interaction with verbal interaction. Thus verbal information increases user experience during visual interaction. Starting from the analysis of multimodal languages recurrent problems and common activities have been identified as preliminary useful ideas for pattern definition. In addition, previous human–computer interaction (HCI) research has provided a low-level, domain-independent taxonomy of tasks that users can perform when compared with an interface. In particular, in [25] techniques for automatically creating multimedia presentations based on user goals are examined in detail. High-level presentation goals can be grouped into two intents: "inform" which simply conveys a presenter's message to a user and "enable" which helps the user to accomplish certain perceptual tasks such as search or verify. According to this theory, two main design issues have been identified in verbal interaction and the related solutions have been implemented in verbal patterns respectively called *Inform* and *Question*, described in the following paragraphs. By means of the extension, XPL has become a container of both ViDPs and VeDPs.

A key feature of a multimodal application is the relation among the several interaction modalities. Without any form of synchronization, in fact, user is not allowed to freely alternate different modalities: if he wants to use several interaction modalities, he is forced to provide each form of input and receive each form of output in an asynchronous way. For example, if the user has to interact with a radio-button, he can chose between the verbal interaction modality or the activation through the mouse click on the radiobutton but it is not implicit that these available capabilities are automatically synchronized: *ad hoc* mechanisms, called *Relations* element, have been implemented to manage the alternation of modalities.

Multimodality is a concept that allows user to move seamlessly between different modes of interaction, from visual to voice to touch, according to both changes in context and user preferences. Human-Computer communication includes two fundamental phases: the *synthesis* (Text-To-Speech, TTS) and the *recognition* (Automatic Speech Recognition, ASR). Research progress in Text-to-Speech are more ahead than in Automatic Speech Recognition for asymmetries inherent speech production and interpretation.

There are more proposed solutions for multimodal interfaces development: XHTML with scripting, XHTML with state transition networks or plan-and-rule based approaches, SALT and XHTML + Voice. We focus our attention on XHTML + Voice [20], called X + V for short, it is an XML based language that rises up from combination between XHTML (HTML with a syntax that conform to XML syntax) and a subset of VoiceXML [22] (W3C standard language, XML based, for pure verbal interface development). An X + V web page is a normal XHTML document with additional voice forms in the heading. These voice forms cover both speech recognition and text-to-speech mark-up. VoiceXML is a markup language which controls the verbal conversation flow and sets the context with the use of grammar file. The context is the set of legal utterances user can say which are defined in the grammar file. X + V can support the XML Form and the ABNF Form of the W3C Speech Recognition Grammar Specification [23]. For this application we have chosen the XML Form. Using X + V language, ASR and TTS technology are embedded together on the IBM Multimodal Browser which is a traditional web browser with additional

tools and engines for the verbal interaction. Control of interaction between the verbal and visual parts is made through XML Event, so that voice handlers can be invoked through a standard EventListener interface. An event can occur when the page is loaded, the user clicks on a button, or says something that the speech recognizer doesn't understand. As a result, a spoken message is prompted or a dialog is initiated with the user.

3.3. Extension methodology

A key feature of XPL is the possibility to extend the language, in terms of patterns, to extend the tranformations libraries according to the changes of the language, to extend presentation effects, incorporating third parties effects to keep always a fashionable presentation. In [25] XPL can be extended easily adding new XSL templates to libraries already built for different channels, scripting functions, java methods and CSS file. The analysis and the cataloguing of different applicative cases, stimulate extensions, that, should be carefully thought to be as much generic as possible and ordered in a patterns collection classified according to groups [25]. This ordered collection of patterns originated from experience is the best source for XPL extension, even if, since XPL has been published on Sourceforge.net, it is auspicable that the extensions are solicited from the open source community. Graphic library development is achieved with a single process of three steps, plus an optional fourth one: i) UIDP partition; ii) XPL pattern selection; iii) Pattern simplification; iv) (optional) Library completion. After the inclusion of multimodality feature, the library completion has to include two kind of xsl stylesheets: the visual and the verbal stylesheet. The bodies of the templates in common are in one file, while according to the interaction, a different header can be chosen automatically [24], or manually through the preferences set in the XPL Editor.

4. XPL design patterns

As previously anticipated, XPL incorporates visual design patterns (ViDP) and verbal design pattern (VeDP) for easy component reusability and multichannel features to address multiple way to access the same content, more detail are given below. Since UIDPs are consolidated practices in the user interfaces implementation, they allow easy reuse and knowledge sharing. Indeed, they provide a good start point for the definition of the language features. As a consequence, UIDPs can be combined with a model-based interface development process based on UIL able to support multi-device user interface creation [21]. In order to rationalize and reduce the information that allows to publish a UIDP, presentation features are divided into patterns with different levels of abstraction, as suggested in [7]. The partition itself is a reference to combine different UIDPs and, at the same time, to simplify and adapt them to the different channels. Another main feature of a user interface language should be that one to promote a high degree of accessibility: XPL makes no inherent assumptions that its output will be rendered visually or that input will come from standard mechanisms. Figure 2 shows the basic structure of an XPL document, the root element is the *xpl-document* that contains a *window* element for the visual part of the interface, a *verbal* element for the verbal interaction and if required a *relation* element, providing synchronization between visual and verbal elements.

4.1. Visual design pattern

In XPL, each pattern corresponds to an element that can be represented on a user interface. A window element contains one or more elements belonging to the *ApplicationGroup*. The *ApplicationGroup* is

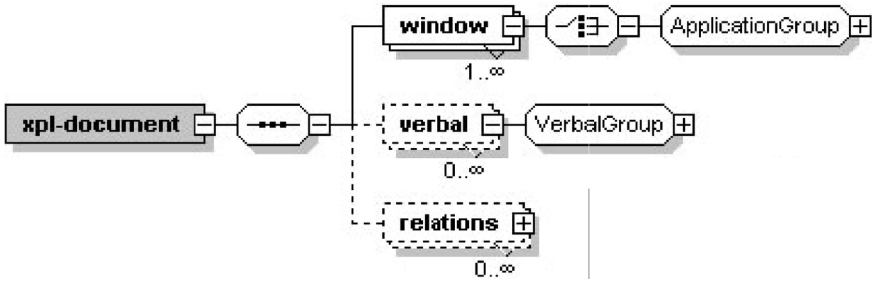

Fig. 2. The XPL root and its possible children.

a set of ViDP in which the application is represented. It is made up by simpler ViDP related to areas *toolbar*, *header*, *menu, help*, *breadcrumbs*, *metanavigation* and contents.

Contents can have elements coming from the *MacroGroup* and the *BasicGroup*. The *MacroGroup* is a set of elements containing two patterns: *login* and *form*. The *login* is a pattern represented as a form for the insertion of user personal data to access to private area. The *form* , is a pattern represented as a group of related contents used for the insertion or visualization of parameters and elementary elements. *BasicGroup*, instead, contains the basic patterns dedicated to the user interaction and to all the elements for the graphical representation, like *text*, *image*, *link*, *multimedia*, and *map* to specify some areas inside an image or an object, assigning a specific action to each areas.

XPL also provides a listener element to bind each event to the presentation of each data. The listener element manages only the *type* of check to do and the *name* of the method for the proper control. In particular, the pattern *EventGroup* only contains the *event* element used to associate a specific behaviour to the particular element in the graphic interface, as a response to the user interaction. The real work is made by the XSLT library holding the code of the method. Code can be maintained in either server-side logic (in the case of Thinlet transformation), or client-side logic (as either Javascript embedded into XHTML or included in external files, in the case of XHTML translation).

An *event* can optionally has one *handler* and zero or more *listener*. A *handler* calls the proper code through the attribute *callback* The type attribute is used to qualify a listener and to execute its code. A *listener* can also be defined giving to the *function* attribute the value of the name of a function in a scripting language associated to the graphic interface. For a more detailed description of implemented ViDPs see [25].

These XPL pattern has been used for the design of the user interfaces for a Web application, the Administrator Interface of a semantic P2P network, within the research project DISCoRSO (Distributed Information Systems for CooRdinated Service Oriented interoperability). This application aims to realize functionalities for the registration and search of Small and Medium Enterprises (SMEs) in a virtual district.

In this work in addition to the generally used patterns like window, form, header, groupbox, XPL is used as a mean to represent structured contents extracted from an ontology. If the business logic has to extract such information from a Web Ontology Language (OWL) document, the main aim of the language will be that one to represent it as a structured <menu> XML fragment and its nested children <menuitem>, that represent a concept or an instance of the ontology, that will be suitably transformed by XSLT.

4.2. Verbal design pattern

Verbal Design Pattern (VeDP) concept derives from the application of UIDP strategy to a verbal interaction and their implementation in XPL enlarge the language capabilities in the user interface design. VeDPs are designed in XML and provide multimodal contents in XHTML + Voice format applying ad-hoc XSLT transformations. The *verbal* element, shown in Fig. 2, can contain one or more elements belonging to the *VerbalGroup* which can contains one or more *Inform* and *Question* elements.

4.2.1. Inform, question and synchronization

The *Inform* VeDP implements a user interfaces design pattern which provides the user with spoken information using text-to-speech functionality. *Inform* is useful for welcome message, brief text reading and verbal description of visual content. These activity are grouped in a VeDP, called *Inform* just because the main function of this pattern is to inform verbally the user.

Inform VeDP is more useful for users with sight problems. For example, for partially sighted person which has difficulties getting the meaning of a graphical representation in a web page, the use of a verbal description associated to a visual element can make the page content much more easily accessible.

Command set which implements text to speech activity in the X + V language has been detected, complying Inform requirements.

The *Question* VeDP implements interaction between the user and the system, using both text-to-speech (TTS) and automatic speech recognition (ASR) functionalities of a multimodal interface. System utters a verbal question (TTS) to the user which can reply either using traditional instruments or pronouncing licit utterances included in the grammar file associated with the pattern. If the reply match with the grammar (ASR) the system goes on with the interaction, else it provides user help content for the navigation.

This pattern has been called *Question* because it is the starting point of the interaction is the system question. Question VeDP is definitely more complete than the Inform VeDP, because user interaction, the main feature of an interface, is now taken into consideration. In this pattern implementation, utility of XPL language is more visible than in the previous pattern. X + V implementation of a scenario, which take into account all properties of the question pattern, produces a great number of row code. On the contrary XPL, which represent VeDP with the XML language, reduces considerably needed code and therefore developer effort.

Inside a Question there are three elements: the grammar element, that defines path and type for the associated grammar with location and type attributes; the variable element, which is used to manage verbal navigation between different pages through the javascript function defined in the attribute value; the suggestion element, which sets the attribute value with the utterance spoken to the user during navigation.

XPL has been used for the design of the user interfaces for a Web application, aims to realize functionalities for the offers aggregation of a Global Distribution Organization (GDO).

As shown in Fig. 3, to enable verbal interaction, VeDP <inform> patterns are used to navigate the application starting from the Home Page, while the VeDP <question> patterns read a series of links whose values are mapped in a grammar. In this case the question allows to choose among several answers, to activate javascript methods capable to manage different navigation path. Inside the <question> element, there is a <variable> element to invoke a javascript method, passing its parameters, and a <suggestion> element that helps the user during the navigation in case the system cannot match the user input or the user doesn't say anything.

On the other hand for the visual interaction the patterns used to build the interface are:a <header> that contains the logo of the project and a banner; on the left that shows a link to Create Demands

System: Welcome to this website. You can request a supplying for a certain product.

User: Help.

System: You can say offers list or fill in the form.

User: Fill in the form.

```
<?xml version= "1.0" encoding= "UTF-8"?>
<xpl-document template= "webSimple">
<verbal xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xpl="http://www.eng.it/xpl">
   <inform id= "welcome" style= "welcomestyle">
      Welcome to this website. You can request a supplying for a certain product.
   </inform>
   <question id= "navigate" style= "menustyle" name= "menu" value= "Click on
             create demands for new demands,click on received offers list to see
             all the offers received for one request or fill in the form to
             create a new demand">
        <grammar location= "gramm/menu.grxml" type= "srgs">
        </grammar>
        <variable name= "variable" value= "function">
        </variable>
        <suggestion name= "choose"
                    value= "You can say offers list or fill in the form">
        </suggestion>
    </question>
</verbal>
<xpl-document>
```

Fig. 3. Global Distribution Organization (GDO) example: Code, visual e verbal interaction.

<menuitem> and Received Offers List <menuitem>; and finally a <groupbox> that contains the central content of the page, in this case a form, which a user can "create a request" according to the choice done in the previous verbal interaction.

As said before, XPL supports both the visual and the verbal interaction and, moreover, an *ad hoc* XPL element has been introduced to synchronize the input given to and the output provided from the application whichever are the modalities they are offered to. This element is the *Relation* element a container of *referredTo* elements linking ViDPs and VeDPs, according to behaviours to be synchronized in the corresponding modalities. Elements contained in *Relation* provide a *static synchronization* among ViDPs and VeDPs. In this way it is possible to keep track of correspondence among visual and verbal elements. During the interaction between users and the application, a synchronization in run-time between the *Verbal* and *Visual Rendering* is enabled each time and the user can freely chose, for instance, to use his voice or a mouse for the interaction, and the output is always maintained coherent. The following example will give evidence for a meaningful example of a very effective alternation of interaction modalities.

In Fig. 4, Figs 5 and 6 is shown an example of XPL exploitation in the design of interfaces for the filling of a questionnaire related to nourishing familiar habits with a very effective alternation of interaction modalities. User interfaces resulting from this design permits users to interact with the application alternating each time verbal and visual interfaces. For instance, the user answers the question

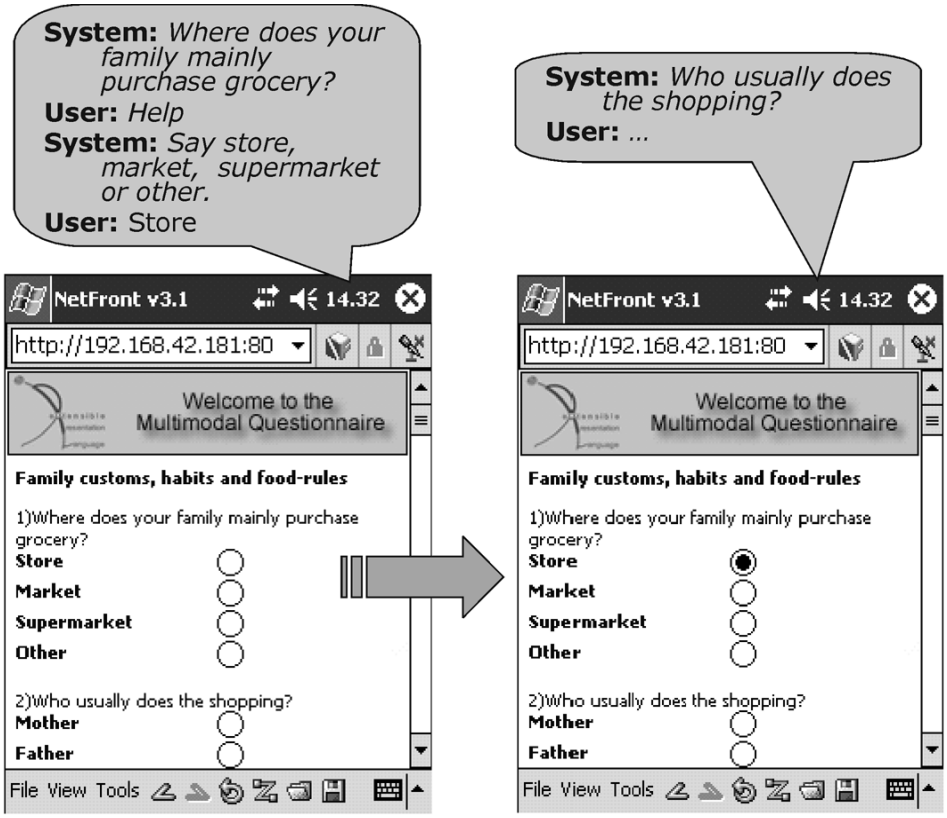

Fig. 4. Questionnaire example: Visual and verbal interaction.

using voice and the correspondent visual interface is simultaneously updated according to the answer provided.

In the Fig. 6 the arrows in the code and in Fig. 5 the sceenshot, show that the verbal pattern element variable "verbal_uno" inside the Question pattern is associated to the visual patterns named "visual_uno", by means of the ReferredTo element.

4.2.2. Verbal grid (*VeDP*)

The Verbal Grid Pattern is a new verbal pattern that formalizes table content access. It has been identified and defined according to accessibility and usability principles.

To this aim, a directed study on was conducted relying on visually impaired and blind users. Usually these users access a computer using a screen reader to hear what they can't see. Even if supported by such assistive software, they still have great difficulty in accessing table content. In fact, the screen reader fails to recognize any logical relationship between rows and columns, and is only capable of reading the table row or column wise.

Verbal Grid features are shown in Table 1, using the schema "Problem, When, Solution, Why, Examples" typical of a pattern description.

The study has been conducted in cooperation with the responsible of ASPHI (Avviamento e Sviluppo Per ridurre l'Handicap mediante l'Informatica) in Sicily who is an official validator of web sites accessibility for the public administration. Thanks to different meetings arranged with ASPHI members, the

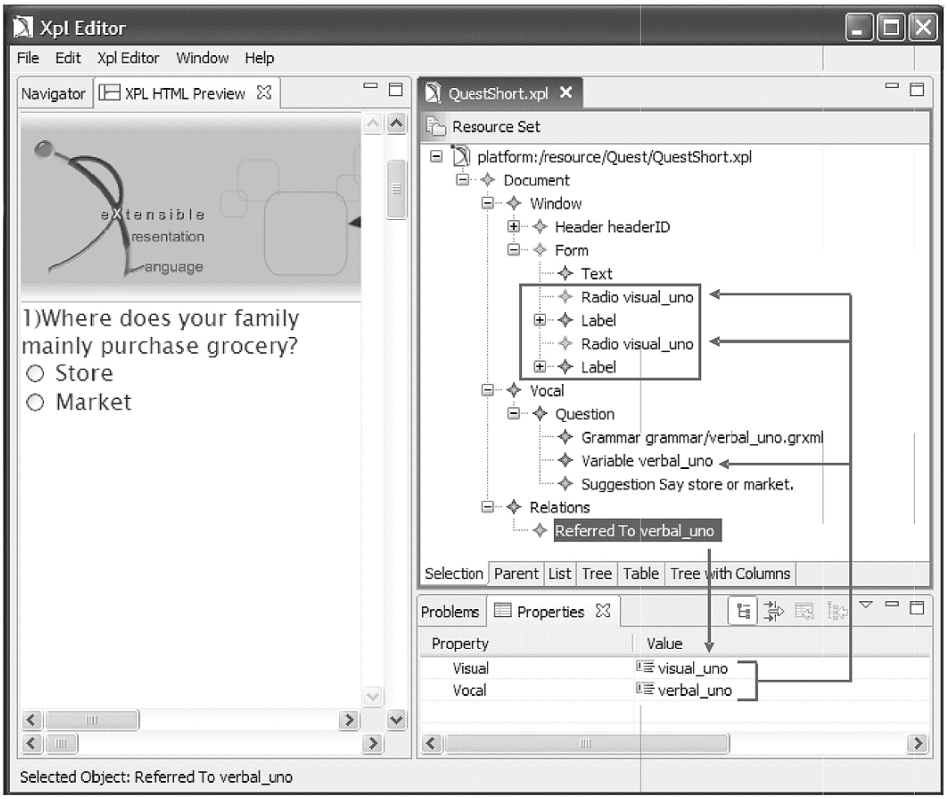

Fig. 5. Questionnaire example: Plug-in screen-shot during implementation.

table issue has been recognized and when the reading schema has been formalized they have tested our solution.

A conversational interface supplies tools to streamline data access in a table format. The data reading task is more structured because the system can ask user for keyword to select the appropriate information, according to the row-column relation. The table can be represented as a grid and the user can access to each node according to his preferences.

Two different kind of tables have been selected: basic data table and complex data table. The recursive use of this two schemas can represent every kind of table. The example selected for the basic data table study is about the detailed list of bank account activity. We have considered a data set with a low number of header column that data set permits a simpler reading approach. The used example is shown in Table 2.

The example used for the complex data table study is about the demands and offers of agricultural goods. This data set is more structured and contains a huge number of header columns that group information in data sub-set. The user has to give more information to reach the needed data. The used example is shown in Table 3.

In the basic data table the interaction is performed using a macro-dialogue that manage in unique phase a set of sub-dialogues. Once visited the page, the system reads a row or a set of row after a single user choice, giving for each element the appropriate connotation, reading headings and the related cell. Blind users can access to a web-site that supplies home-banking services using a screen reader. When the user reaches the balance page the screen reader reads the whole table content, row by row. Using a

```
<xpl-document>

<window>
    <text>1)Where does your family mainly purchase grocery?</text>
    <radio id="store" name="visual_uno" value="store">
    </radio>
    <label control="store">
      <text id="label store" style="boldtext">Store</text>
    </label>
    <radio id="market" name="visual_uno" value="market">
    </radio>
    <label control="market">
      <text id="label market" Market</text>
    </label>
</window>

<vocal>
    <question id="verbal_uno" value="Where does your family mainly purchase ">
      <grammar location="grammar/verbal_uno.grxml" type="srgs"></grammar>
      <variable name="verbal_uno"></variable>
      <suggestion name="say store or market."></suggestion>
    </question>
</vocal>

<relations>
    <referredto vocal="verbal_uno" visual="visual_uno"></referredto>
</relations>

</xpl-document>
```

Fig. 6. Questionnaire example: XPL implementation code.

Table 1
VeDP grid description

Name	VERBAL GRID
Problem	User wants to access a data table by means of vocal interface
When	User wants specific details only, not just a mere scan-read row- or column-wise
Solution	The data table is decomposed in a sequence of microdialogues, in which questions a user may ask are built from the table headers, and table rows are the utterances spoken by the system in reply to those questions
Why	It enables navigation for either blind users or for users with their eyesight committed to other tasks (i.e. querying your checking account while driving)

Table 2
Detailed list of bank account activity

Description	Operation date	Value date	Reason	Account
Just for you formula fixed costs november	01.12.2006	01.12.2006	198	€ 4,00
Deposit	01.06.2006	10.06.2006	105	€ 200,00
Overall balance			€200	

multimodal browser the user can access to the balance page using a vocal command. By means of the conversational interface implemented by the Grid Pattern he/she can select what rows of the table he/she needs to listen.

In a complex table the rows and columns headers become vocal command. The user can single out a set of data from the whole table that will be navigable as a basic table. This situation supplies a kind of "dynamic" sub-dialogues created and activated according to the user preferences. A complex table, as shown in table III, is navigated from the screen reader as a piece of text, so the user will hear row by row the whole content. In our model the system will ask user to chose between the 2007 and the 2006. After this choice only a data sub-set related to the selected year will be available and the dialogue to describe this data sub-set will be activated.

Table 3
Demands-offers data table for agricoltural products

		Executed demands list				Received offers list		
		Demands code	Item	Status	Demands sending date	Offers code	Status sending date	Offers
2006	Smith	R/0001	Grapes	CLOSED	28/09/2006	O/0001	CONF	02/10/2006
	Wilson	R/0002	Cherries	CLOSED	06/08/2006	O/0001	CONF	05/09/2006
2007	Jefferson	R/0003	Potatoes	NEW	15/01/2007	O/0001	ACCEPT	25/02/2007
	Brown	R/0004	Apples	NEGOTIATION	18/02/2007	O/0001	NEW	01/03/2007

5. Conclusions

In this paper the XPL was presented, a framework aimed at developing multichannel and multimodal interfaces with full component reuse. This work demonstrates how well-designed interfaces can aid users with physical impairments to navigate web applications and gain full understanding of page contents. XPL allows developers to design once (universal designing) for all types of users and for different access channels, reusing components as their use-context allows. Tests conducted with blind testers by ASPHI validated the quality of this approach. We are currently involved into the development of additional verbal design pattern to expand on XPL capabilities and features.

Acknowledgements

This work is the result of an on-going research collaboration between Engineering Ingegneria Informatica S.p.A. and Università degli Studi di Palermo. On Engineering side, the work results are part of national research activities focused on methodologies and technologies for Internetworked Enterprise and Collaborative Working Environments. The research unit at DINFO has been partially funded by the Italian Ministry for University and Research (PRIN 2005 project, contract no. 2005103830_002), and by the Università degli Studi di Palermo (ex60% 2006 project, contract no. ORPA06ZT43). Authors are indebted with Dr. Ciro Arnone of ASPHI (Avviamento e Sviluppo Per ridurre l'Handicap mediante l'Informatica), for his tireless collaboration to testing and verification of design efficency.

References

[1] MobileHCI '05, Vol. 111, ACM Press, New York, NY, 2005, 243–246.
[2] M. Weiser, The computer for the 21st century, *Scientific American* **265**(3) (1991), 94–104.
[3] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen and E.-J. Malm, Managing context information in mobile devices, *Pervasive Computing* **2**(3) (2003), 42–51.
[4] J. Nielsen, Usability engineering, in: *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, G. Salvendy and M.J. Smith, eds, Elsevier Science Publishers, 1989, pp. 394–401.
[5] J. Brewer and A. Chuter, Policies Relating to Web Accessibility, http://www.w3.org/WAI/Policy/.
[6] W. Chisholm, G. Vanderheiden and I. Jacobs, Web content accessibility guidelines 1.0. World Wide Web Consortium, Recommendation WAI-WEBCONTENT-19990505, 1999.
[7] C. Alexander, S. Ishikawa and M. Silverstein, A Pattern Language: Towns, Buildings, Construction, Oxford University Press, 1977.
[8] S. Fincher, Analysis of Design: an exploration of patterns and pattern languages for pedagogy, *Journal of Computers in Mathematics and Science Teaching: Special Issue CS-ED Research* **18**(3) (1999), 331–348.
[9] K. Mullet, *Structuring Pattern Languages to Facilitate Design*, workshop Patterns in Practice, CHI2002, 2002.
[10] L. Bannon, A Pilgrim's Progress: From Cognitive Science to Cooperative Design, *AI and Society* **4** (1990), 259–275.

[11] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*, Addison-Wesley, 1995.
[12] M.J. Mahemoff and L.J. Johnston, Pattern Languages for Usability: An Investigation of Alternative Approaches, Asia-Pacific Conference on Human Computer Interaction, Los Alamitos, *CA: IEEE Computer Society* (1998), 25–31.
[13] J. Borchers, A pattern approach to interaction design, conference proceedings on designing interactive systems: processes, practices, methods, and techniques. Brooklyn, NY, 2000, 369–378.
[14] D.K. van Duyne, J.A. Landay and J. Hong, *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*, Addison-Wesley Publishing, 2002.
[15] I. Graham, A pattern language for Web Usability, Addison-Wesley, Boston, US, 2003.
[16] M. van Welie, G. van der Veer and A. Eliens, Patterns as Tools for User Interface Design, *International Workshop on Tools for Working with Guidelines*, Biarritz, France, 2000, 313–324.
[17] J. Tidwell, Interaction Design Patterns, Conference on Pattern Languages of Programming, Monticello, Illinois, USA 1998.
[18] J.C.Teague, DHTML and CSS for the World Wide Web: Visual Quickstart Guide, Peachpit Press, 2001.
[19] M. Abrams, C. Phanouriou, A. L. Batongbacal, S. M. Williams and J. E. Shuster, *UIML: An Appliance-Independent XML User Interface Language, Computer Networks* **31**(11) (1999), 1695–1708.
[20] T. Cheng, XUL – Creating Localizable XML GUI, 15th International Unicode Conference, San Jose, California August, 1999.
[21] P. Azevedo, R. Merrick and D. Roberts, Ovid to auiml – user-oriented interface modeling, workshop "Towards an UML Profile for Interactive Systems Development (TUPIS)", York, UK 2000.
[22] G. Mori, F. Paternò and C. Santoro, Tool Support for Designing Nomadic Applications, International Conference on Intelligent user interfaces, 141–148, Miami, Florida, USA, 2003.
[23] Holm, Ivar, Ideas and Beliefs in Architecture and Industrial design: How attitudes, orientations, and underlying assumptions shape the built environment. Oslo School of Architecture and Design, 2006, ISBN 8254701741.
[24] R. Raccuglia, G. Vella, A, Santangelo, N. Ingraffia and A. Gentile, The eXtensible Dynamic Presentation Manager for content adaptation, International Conference on Complex, *Intelligent and Software Intensive Systems*, Barcelona, Spain, 2008.
[25] G. Vella, M. Liotta, A. Santangelo, N. Ingraffia and A. Gentile, XPL, a Presentation Language based on User Interface Design Pattern, 6th IEEE International Conference on Computer and Information Science (ICIS 2007), Melbourne, Australia, , July 11–13, 2007, pp. 285–290.

Antonella Santangelo received her Laurea degree "cum laude" in computer engineering from the University of Palermo in 2005. She is currently a Ph.D. candidate in computer science at DINFO Dipartimento di Ingegneria Informatica, University of Palermo, Italy. Her research interests are in the field of natural language processing and multimodal human-computer interfaces.

Antonio Gentile is associate professor at DINFO – Dipartimento di Ingegneria Informatica of the Universita‘ degli Studi di Palermo, Italy. He received the Laurea degree “cum Laude” in electronic engineering and the doctoral degree in computer science from the Universita‘ di Palermo, in 1992 and 1996, respectively. He also received the PhD degree in electrical and computer engineering from the Georgia Institute of Technology in 2000. His research interests include high throughput portable processing systems, image and video processing architectures, embedded systems, speech processing, and human-computer interfaces. Prof. Gentile is a senior member of the IEEE, of the IEEE Computer Society, and a member of the ACM and of the AEIT. He is currently serving as associate editor for *Integration – the VLSI Journal*.

Giuseppe Vella is a User Interface Designer at Engineering Ingegneria Informatica Research and Development Laboratory. He received a Laurea degree in foreign languages from the University of Palermo, Italy, in July 2001 and his Master degree in Digital Publishing for Internet from the ICT Training school of Ferentino. His main research results are human-computer interaction and eXtensible Presentation Language. He is author of several scientific publications concerning HCI and in particular the XPL for content presentation according to the Visual and Verbal Design Pattern methodology and XDPM, a technological infrastructure for content and presentation and adaptation.

Nunzio Ingraffia got a University Degree in Electronics Engineering with major in Computer Electronic in July 1997. He worked in various Software Companies, acquiring technical Know-how in different computer programming environments. He works in Engineering since November 2000, obtaining technical and management knowledge in software project in different business environments. He is member of NESSI Costa Committee to evaluate and promote the adoption of Standards on distributed architectures. Research coordination is related on user presentation language based on visual and verbal design pattern paradigm, information management framework founded on Semantic Web technologies, latent semantic analysis, business process modeling and management.

Marilia Liotta received her “cum laude” Laurea degree in computer engineering and the doctoral degree in computer science from the Universita‘ di Palermo, in 2001 and 2007, respectively. From 2002 to 2005 she has been a researcher in the R&D Lab of Engineering Ingegneria Informatica, working in the area of neural networks and data mining. Her research interests include information retrieval, semantic knowledge representation, semantic web technologies and human-computer interfaces. Today she works for Sicilia e-Servizi Spa in the designing of e-health systems.