Methodology for the Development of Vocal User Interfaces

David Céspedes-Hernández[1], Juan González-Calleros[1], Josefina Guerrero-García[1]
[1]Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla, Av. San Claudio y 14 Sur, Puebla, México
Tel: +52222{229 5500}
dcespedesh@gmail.com, {juan.gonzalez, jguerrero}@cs.buap.mx, lilianavizz@hotmail.com

Jean Vanderdonckt[2], Liliana Rodríguez-Vizzuett[1]
[2]Louvain School of Management, Louvain Interaction Lab, Université catholique de Louvain, Place des Doyens 1, Louvain-la-Neuve, Belgium
jean.vanderdocnkt@uclouvain.be

ABSTRACT
Natural User Interfaces allow users to interact with systems similarly as they interact with people. Human communications occur, mostly, in an oral way, since personal dialogs to phone calls and more recently in complain or information systems; the tendency is to automate some of these activities so the user might complete tasks in a more efficient way. The necessity for having a methodology that supports the development of vocal interfaces is therefore taking interest on it. The objective for this sample paper is to establish a methodology and to describe a set of rules that might be used for developing a software tool to generate code for multiplatform vocal User Interfaces from models.

Categories and Subject Descriptors
H5.2 [**Information interfaces and presentation**]: User Interfaces – *Prototyping; user-centered design; user interface management systems (UIMS)*.

General Terms
Design.

Keywords
User Interface, Vocal Interaction, Model-Based User Interface Development, User-Centered Design.

1. INTRODUCTION
In the last few years, Natural User Interfaces (NUI) have become interesting for the scientific community. A NUI is defined as an interface that allows the user to interact with a system using as interaction techniques natural mechanisms, for example, the language used to order a pizza.

This paper is centered on the study of those NUIs that use verbal communication as interaction technique. Dialog systems are software designed with the ending of emulating the dialog of a human being with another one. Nowadays, some enterprises and institutions use this kind of system for giving information and automatization of services like information and reservation of airplane trips, meteorological information, food ordering and reservation of train trips.

MexIHC '12, October 03 - 05 2012, Mexico city, Mexico
Copyright 2012 ACM 978-1-4503-1659-0/12/10…$15.00.

By the nineties, web services with human voice support appeared, encouraging the creation of label based languages that assist the development of software systems capable of interacting orally with web pages.

Voice eXtensible Markup Language (VoiceXML) (http://www.w3.org/TR/voicexml20/) is the World Wide Web Consortium (W3C) standard format for human-computer vocal interaction applications. Its main utilities are speech tuning and recognition, but other features are dialog management and auditory feedback.

Microsoft Kinect SDK (Software Development Kit) (http://msdn.microsoft.com/en-us/library/microsoft.kinect.aspx) is a toolkit that along with other things allows the use of a microphone array with noise and echo cancellation, "beam formation" for identifying the audio source and integration with the API of Microsoft speech recognition (http://msdn.microsoft.com/en-us/library/system.speech.recognition). Its features enable that when using Kinect SDK and the API of Microsoft speech synthesizer (http://msdn.microsoft.com/en-us/library/system.speech.synthesis.aspx) it is possible to obtain applications that simulate vocal dialog between the user and the system.

Profile eXtensible HyperText Markup Language + Voice (XHTML+Voice) (http://www.w3.org/TR/xhtml1/) brings spoken interaction to standard World Wide Web (WWW) content by integrating a set of mature WWW technologies such as XHTML and eXtensible Markup Language (XML) events with XML vocabularies developed as part of the W3C Speech Interface Framework. The profile supports voice synthesis and speech recognition for command and control actions as for user dialog.

USer Interface eXtensible Markup Language (UsiXML) (http://www.usixml.org/) is a XML compatible label based language for describing User Interfaces (UIs) for multiple use contexts, such as Graphic User Interfaces (GUI's), Auditory User Interfaces (AUI's) and multimodal UIs. UsiXML is sustained by diverse tools that allow the exchange between applications with different interaction techniques, use modalities and computing platforms so the design of the interfaces remains regardless of the hardware platform.

The main purpose of this paper is to describe the design of a concrete model, as well as the mapping of an interpreter that allows the transformation of the code generated for VoiceXML, XHTML+Voice or Microsoft Kinect with Microsoft speech synthesizer so it might be used by UsiXML and vice versa.

2. STATE OF THE ART

The evolution of interactive systems reached a point where today's research is centered in the development of NUIs, this is evident from the observation of the tools and new technologies offered in the market. Gone are the days when the command line was the reigning interaction style and the beginning of the end of mouse pointer manipulated GUIs is arriving. This obeys at least two reasons, first the technological advancement that allows processing large amounts of data resulting on quick processing of natural interaction data and second that, when well designed, the nature of this techniques make the interaction easier and more intuitive. Since the user speaks instead of writing and listens instead of reading, the interaction appears in an easier way for him/her.

Speech is the most commonly used communication method by the human being, as consequence of this, voice synthesis for playing sound and speech recognition are a couple of highly developed knowledge areas. Taking this into account, experts in Human-Computer Interaction (HCI) predict that vocal interaction as well as other interaction modes within NUIs will be part of the future of software development.

The current challenge is not only in finding a methodology that supports vocal interfaces development but in creating technological tools for interoperability support considering different use contexts. It is hard to imagine a solution for vocal interfaces that cannot be adapted to other use contexts; this is why it has to be considered while realizing the methodology for development that it is not an isolated entity to solve a particular problem. Instead of this, it has to be considered a generality of options to be solved; this is the main reason that takes to think on a model based solution where methodological knowledge for vocal interfaces development gets consolidated.

As part of the UsiXML project, several tools had been developed. Those tools may be classified in its most as: editors, generators and interpreters. Each of these tools has its particular function.

The purpose of editors is to provide an environment for multiple platform interfaces creation, without requiring experience from the designer, so they can be exported to other languages or saved in UsiXML. As examples of UsiXML editors there are GrafiXML, VisiXML, SketchiXML, IdealXML, PlastiXML and ComposiXML.

Generators, as its name indicates, generate or produce code. This kind of tools work along with tasks and workflow models and take them as base for generating multiplatform UIs in an abstract as in a concrete level. KnowiXML, IKnowYou, UsiXML for Rich Internet Applications and UsiXML4All code generator are code generators belong to UsiXML Project.

The function of interpreters in UsiXML is to take a UsiXML compatible UI and render it for creating a functional interface besides from offering the possibility of resizing and setting constraints to adapt an interface to a determined platform (Interface Plasticity).

There are some other tools that have as main functionality the creation of adequate UIs from tasks, user and workflow models [6]; creating interfaces from others by applying reverse engineering [17] and for creating UIs with support of virtual reality [15].

The direct antecedent to this project is described as the definition of a code generating tool proposed in MultimodaliXML [14] where the objective of applying a set of XSL transformations over the specification of a concrete vocal UI model is mentioned, but it did not reach the implementation phase.

3. METHODOLOGY

3.1 Model Driven Approach

The development of virtual environments faces growing difficulties that had been confronted using the model based paradigm or MDA promoted by the Object Management Group (OMG), examples for this are [5], [12] and [3].

MDA is an OMG initiative that proposes to define a set of non-proprietary standards that will specify interoperable technologies with which to realize model-driven development with automated transformations. Not all of these technologies will directly concern the transformation involved in MDA. MDA does not necessarily rely on the UML, but, as a specialized kind of MDD (Model Driven Development), MDA necessarily involves the use of model(s) in development, which entails that at least one modeling language must be used. Any modeling language used in MDA must be described in terms of the MOF language to enable the metadata to be understood in a standard manner, which is a precondition for any activity to perform automated transformation [19].

The MDA paradigm incorporates a standard for the establishment of the components that integrate an Interactive system development methodology, these components are:

- Models. A set of models describe the elements that conform a UI: task, data model, user profile, graphic presentation and behavior. Models use UML class diagrams for capturing the abstraction of the modeled reality.
- Language. For specifying the models in some way the computer might process a User Interface Definition Language (UIDL) is required. That language allows designers and developers to exchange, communicate and share fragments of the specification so the software tools can operate on them. For this purpose, once the abstraction and the modeling is completed, UsiXML [13] will be used because it is supportive of the MDA. Its selection is based on the revision of existent documents [6] but the solution (model) is not only applicable to this language but to any other UIDL. The revision and comparison of several UIDLs has been realized and is accessible for its consult [7].
- Software. A methodology must be supported by software tools and its interoperability has to be ensured at least theoretically.
- Approach. This refers to paradigm used for giving an order to the UI development methodology steps. The process of design begins with a tasks model developed under a gradual approach for ending in the definition of the UI [2]. The approach is based on the Cameleon Reference Framework [1] which consider four development phases:

1. Tasks and concepts (T & C): describe user tasks; they are concepts referent to data models that are required for doing these tasks.
2. Abstract User Interface (AUI): define abstract containers and interaction individual components. Tasks are associated to

containers for its execution or to individual objects for its manipulation. An AUI is considered as an abstraction of a Concrete UI with respect to the interaction modality. At this level, the UI is composed mainly by the definition of the system inputs and outputs but does not define the modality to be used (graphical, vocal, tactile).

3. Concrete User Interface (CUI): the CUI defines an interaction modality and is composed by elements that describe it, Concrete Interaction Objects (CIOs) for defining the design widgets and the navigation through the interface. The CUI is computing platform independent and although it makes explicit the aspect and behavior of the UI, it is still a model only working for a particular environment. A CUI may also be considered as a reification of an AUI in the superior level and an abstraction of the Final UI with respect to the platform.
4. Final User Interface (FUI): corresponds to the operational elements, this means, the run time UI over a determined computing platform.

In order to support the modeling of the above seen levels, there are methods and transformation rules for different use contexts. The transformational development of the UI finds its motivations in the concept of heterogeneity of information systems. For this case, heterogeneity refers to the variety of use contexts for those which the UI was designed for. This heterogeneity makes a stand on the necessity of abstracting the pertinent details from specific contexts. It is possible to obtain specific representations from these abstractions. The advantage of accessing to those representations is the ability for reasoning about a unique model (task model) and to obtain many different UIs.

Models and transformations for models are expressed in UsiXML. Thanks to this language it is possible to develop and spread quickly a wide range of UIs for different computing platforms, with different interaction modalities and for diverse use contexts.

3.2 Establishment of the Methodology

For establishing the methodology, the three tools mentioned in the introduction are considered: VoiceXML, XHTML+Voice and Microsoft Kinect with Microsoft speech synthesizer. The first step consists on compare the elements that form each one of them. In Table 1 the resumed comparison of these elements is shown.

Voice-XML	**XHTML + Voice**	**Kinect with speech synthesizer**
<audio>	<audio>	Speak()
<prompt>	<prompt>	Prompt()+Speak()
<record>	<record>	Start()+Stop()
<field>	<field>	Start()+Stop()

Table 1. Summarized comparison between tools that support vocal interaction.

The second step consists of detecting the main components and analyzing their function, as in step 1, when a tool does not support a function qualified as important in an explicit way, it was necessary to join two or more methods or attributes for performing and establishing the bases for the methodology. Table 2 shows in a summarized manner how the analysis was made and the main functions were found. The full comparison consisted of a list of 63elements.

Once the comparison and the analysis are made, it is possible to propose a model that represents how vocal interaction could be applied and how vocal UIs can be developed by having the tasks' model. Figure 1 shows the proposed model for vocal interaction with context awareness.

The two main components of the model are the context model and Vocal Concrete User Interface (VocalCUI), the context model describes the three aspects of a context of use in which an end user is carrying out an interactive task with a specific computing platform in a given surrounding environment: context, platform and the user itself [20].

Voice-XML	**XHTML + Voice**	**Kinect with speech synthesizer**	**Function**
<audio>	<audio>	Speak()	Synthesizes Audio form a source
<prompt>	<prompt>	Prompt()+Speak()	Synthesizes a given message
<record>	<record>	Start()+Stop()	Records the audio input
<field>	<field>	Start()+Stop()	Waits for the user´s audio input

Table 2. Summarized comparison between tools that support vocal interaction with functional analysis.

A User model consists of a user stereotypes. A user stereotype is any set of users sharing similar characteristics. Stereotypes can be arranged in hierarchy. As so, a stereotype can be decomposed into sub-stereotypes. For this model, a modification was made to the user stereotype; cultural information is added as an element of the user because it plays an important role in vocal interaction by affecting the intonation, the language, and the grammar to be used for speech recognizing and for speech synthesizing.

A Platform model captures relevant attributes for each couple software-hardware platform and attached devices that may significantly influence the context of use in which the user is carrying the interactive task. This context model has been developed in Experiences with Adaptive User and Learning Models in eLearning Systems for Higher Education 3. A platform specification can consist of a series of physical hardware devices (hardware platform components), a series of software components (software platform), the characteristics of the network to which the platform is connected, the capability to support wireless (WapCharacteristics), and the capability of browsing web pages (BrowserUA) but for this purpose, the only elements considered are software and hardware platform.

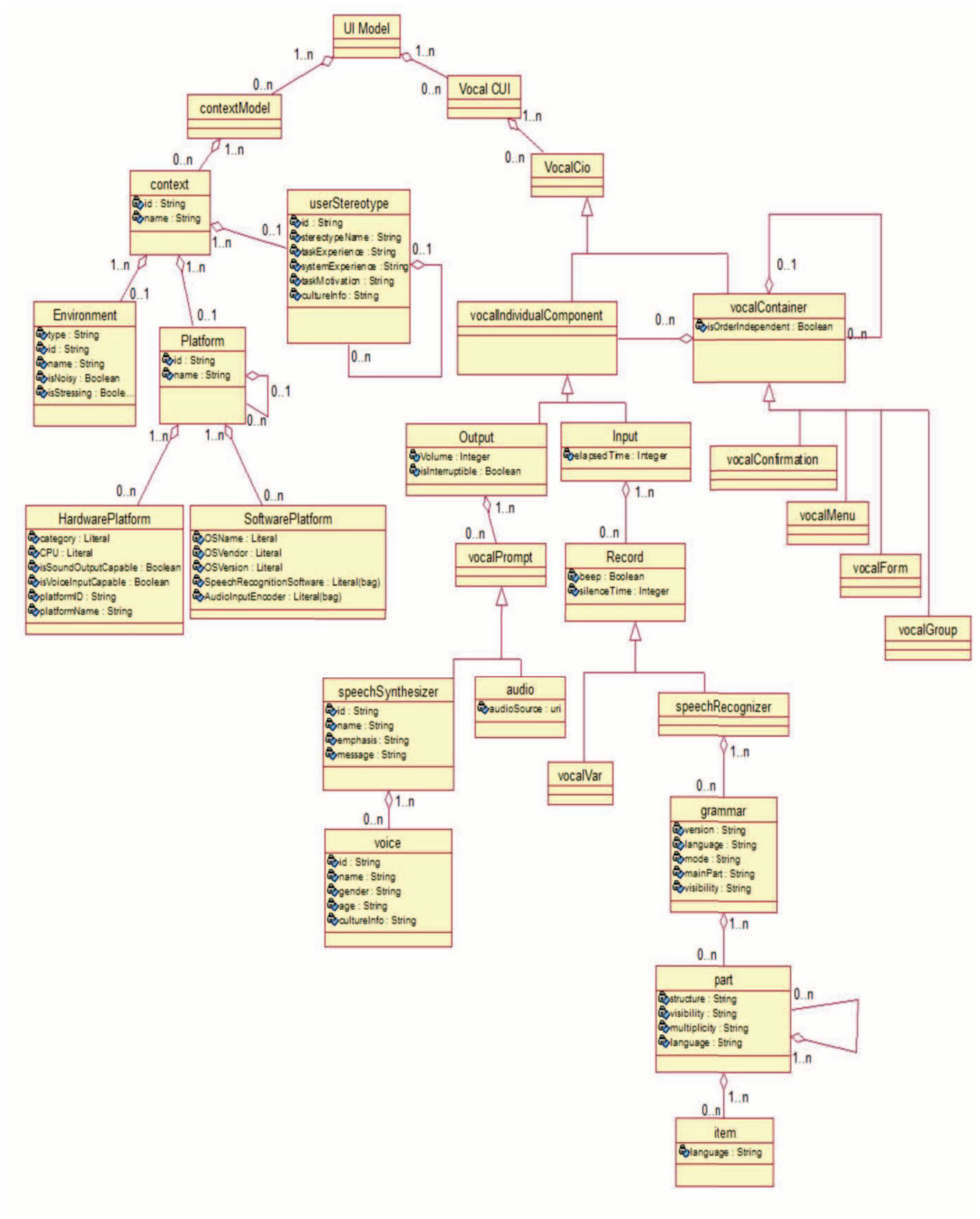

Figure 1. Vocal UI model with context awareness.

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 physical (e.g., lighting conditions), psychological (e.g., level of stress), and organizational (e.g., location and role definition in the organization chart). In the model presented, the environment model is summarized in order to present only the determinant characteristics for the type of interaction that it tends to represent.

3.3 Elements that Conform the Vocal UI Model

Vocal CUI is formed by Vocal Concrete Interaction Objects (VocalCIOs), which are divided into Vocal Individual Components and Vocal Containers.

VocalContainers represent a logical grouping of other containers or individual components and inherit the isOrderIndependent attribute which indicates if the inputs of the container can be filled in any order or not:

VocalGroup: is the root element of all vocalCIOs. A VocalGroup Acts as a basic container for all containers and components.

VocalForm: enables a dialog whose purpose is to synthesize/collect data from the system/user.

VocalMenu: allows choosing among different items.

VocalConfirmation: requests from the user a confirmation of a previous input. It is composed of a vocalPrompt that solicits the confirmation followed by an input gathering the user's input. For instance, “Do you want to delete this file? Say Yes or No”.

VocalIndividualComponents are VocalCIOs contained in a VocalContainer. Two VocalIndividualComponents are introduced:

Output: is an object used for presenting data to the user. The *volume* attribute specifies the sound volume expressed in Db (decibel) while the attribute *isInterruptible* specifies if the output can be interrupted by a user’s utterance.

Input: is an object used to gather input from the user by speech recognition or audio recording. The *elapsedTime* attribute is the time frame expressed in seconds during which the user is allowed to utter the input.

Those individual components use objects for synthesizing and storing data, the Output object uses a vocalPrompt.

A *vocalPrompt* is an object that prepares data to be presented to the user; this object is as well divided onto two different objects depending on the source of the data:

- *SpeechSynthesizer*: is used to do a text-to-speech transformation of the message given in the message attribute. The *emphasis* attribute expresses the dominant tone according to which the vocalOutput will be synthesized: positive, negative, interrogative, exclamative. This object uses the object *voice* for establishing other parameters in order to complete the emission of the message. The *gender* attribute is used to specify if the voice to be played corresponds to a male, female or neutral individual. *Age:* is an attribute to determine the age of the *voiceFont to use* (e.g. child, teen, adult, senior). *CultureInfo:* First introduced in this model, this attribute represents the language or grammar level for presenting data to the user.
- *Audio:* is employed to play audio prerecorded files. The *audioSource* attribute specifies the URI of the audio file to be played or the name of the reference where the recorded file is stored.

The input individual component uses a *Record* object in order to gather the information that the user provides.

A *Record*: is an object used to record a vocal message of the user and when specified can be divided into a *vocalVar* and /or a *speech Recognizer*. If the *beep* attribute is set to TRUE, an acoustic beep is emitted by the system announcing the availability of the recording. If set to false (the default value) no beep is emitted and the user can start to record immediately after the prompt. The *silenceTime* attribute is the silence time period that determines the record to be stopped. It is expressed in milliseconds or seconds.

vocalVar: used to declare a variable. May use functions and conditions such as *setVar, resetVar, if, elsif and else.*

The *Speech Recognizer* is software that as its name says, realizes speech recognition, the speech Recognizer uses *grammar(s)* as well as *parts* and *items*.

Grammar is a structured and compacted enumeration of a set of utterances (i.e., words and phrases) that constitute the acceptable user input for a given input. The grammar can be internal (i.e., it is specified within the document) or external (i.e., it is specified in an external file). The *version* attribute indicates which version of the grammar specification is being used. The *language* attribute indicates according to which language the utterance has to be pronounced in order to be recognized by the system. The specification of the language takes the form of the couple: the name of the language followed by the country in which it is used (e.g.: English-UK). The *mainPart* attribute is the first part of the grammar that will be treated by the system. The *mode* attribute specifies the available interaction type. The default type is voice for voice-based interaction, whereas for phone-based interaction the value is dtmf. The *visibility* attribute specifies the visibility of the grammar. If set to document the grammar is active throughout the current document. If set to form (the default value) the grammar is active throughout the current *vocalForm*.

Part: contains other part elements or available input items. The *structure* attribute specifies how the user’s inputs should be uttered in order to be recognized by the system. There are three possible values: choice (i.e., the grammar items are alternative inputs), sequential (i.e., sequence of grammar items that have to be uttered one after another in the order of their appearance) or asynchronous (i.e., sequence of grammar items in which the items do not have any particular order of utterance). The *visibility* attribute specifies the visibility of the part component. If set to private (the default value) the part component can be used only by the containing grammar. If set to public the part component can be referenced by other grammars. The *multiplicity* attribute indicates how many times the enclosed items may be repeated. The default value is 1.

The multiplicity is defined as follows:

- X (where $X>0$): the items are repeated exactly X times.
- X-Y (where $0\leq X<Y$): the items are repeated between X and Y times (inclusive).
- X- (where $X\geq 0$): the items are repeated X or more times.

The *language* attribute indicates in which language the items have to be pronounced in order to be recognized by the system. The specification of the language takes the form of the couple: the name of the language followed by the country in which it is used (e.g., French-CA). If it is not specified, it inherits the value from the language attribute of the embedding grammar element.

Item: enables to specify a grammar input or to reference another part element. The language attribute indicates in which language the item has to be pronounced in order to be recognized by the system. The specification of the language takes the form of the couple: the name of the language followed by the country in which it is used (e.g.: French-CA). This attribute allows mixing multiple languages in the same grammar. If it is not specified, it inherits the value from the language attribute of the embedding part element.

4. CASE STUDIES

As a test for the proposed model and for a better understanding of it, graphical examples representing dialogs between the system (S) and the user (U) are shown:

The dialog in Figure 2, describes the fulfillment of the *Provide age* task by an end-user. A *vocalPrompt* is necessary for containing the other objects that are used in the interaction (input, output, vocalConfirmation). Considering the direction of the dialog from left to right, the first output is used to welcome the user and invites to input the age.

Then, an input object is necessary for gathering the information the user provides, in this case the age. This information may be saved into a vocalVar or recorded to a file so the system can access to it and by applying speech recognition (with the speech recognizer) make a supposition on what the user said. Later, the system provides a confirmation to the user by emitting an output conformed by the phrase *"Your answer was"* plus the synthesis of the word that it recognized. And finally as part of this confirmation the system uses an input so the user can confirm or deny.

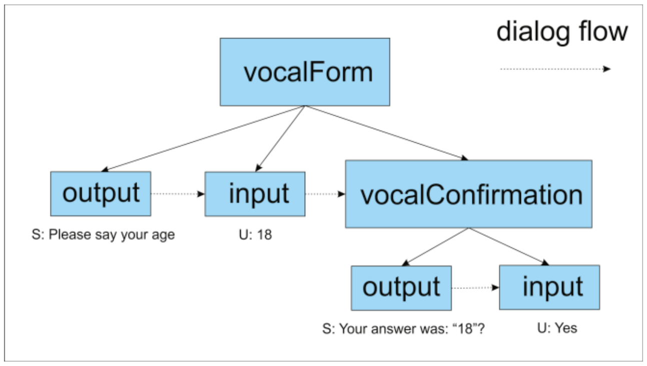

Figure 2. VocalCIOs involved in the fulfillment of provide age task.

In Figure 3, the dialog describes a vocal interaction application of a phone company where users can select among different options. There are two tasks involved, for the first one the user provides the name to the system and for the second one, s/he selects among three proposed options. As in the first example VocalCIOs are contained by a vocal Form. Considering the direction of the dialog from left to right, the first object presented is an output, where the system introduces and invites the user to say the name. After this is done, the system receives the input from the user and uses a vocalMenu container in order to present to the user the available options.

For this proposal, the system produces an output for each item or option and after presenting them, expects for the user to choose one of them. The user selects an option and realizes the input to the system; the system, as in the previous dialog recognize the selection, provides a confirmation and in this case, proceeds to a determined part of the software program that might be another vocal Form or other vocal container.

This examples, made evident the necessity of a set of rules or equivalences that determine how to generate code from the tasks or interaction models for any of the above mentioned languages.

In A Methodology for Developing Multimodal User Interfaces of Information Systems [14], a similar analysis is done but for the multimodal domain and in a higher abstraction level. Taking the principles explained there and adding the concepts seen in the proposed model, the following rules and equivalences are identified:

- *Output*: <prompt> or <audio> for VoiceXML and XHTML+Voice and prompt() and/or speak() for Microsoft speech synthesizer.
- *Input*: <field> or <record> for VoiceXML and XHTML+Voice and prompt() and start()-stop() for Microsoft Kinect. For recognizing the input, it is necessary to use a grammar with items on it. For VoiceXML and HTML+Voice this is implemented using the <grammar> and <item> fields, while in Microsoft Kinect this is done by the Microsoft speech recognizer and may also be configured.
- *vocalMenu*: a vocal menu can be divided into two different parts, one to present the set of options for the user: <prompt> or <audio> for VoiceXML and XHTML+Voice and prompt() and/or speak() for Microsoft speech synthesizer and the other one to gather the selection from the user: <field> or <record> for VoiceXML and XHTML+Voice and prompt() and start()-stop() for Microsoft Kinect. Functions and conditions (e.g. if, else, elsif, oneof, foreach) as well as variables may be used to organize, evaluate and execute user's choices.
- *vocalConfirmation:* a vocal confirmation is necessary when a selection is made by the user; to implement this an *output* and an *input* is necessary

Other structures and objects like combo boxes, radio buttons, check boxes and list boxes can be implemented following this set of 4 rules or equivalences.

5. CONCLUSION

In this paper, the establishment of the methodology for development of vocal UI settled the bases for a software tool so it might be possible for designers and non expert developers in the vocal UI area to design and create this kind of interfaces by only abstracting the components needed for it. Along with the transformation rules, the developer can do migration of already existing projects from one of the analyzed languages to one another or apply reverse engineering for this purpose.

Another important point that was mentioned is the aggregation of the context model to the vocal UI model, because in this particular case (vocal interaction), aspects like culture, the used platform

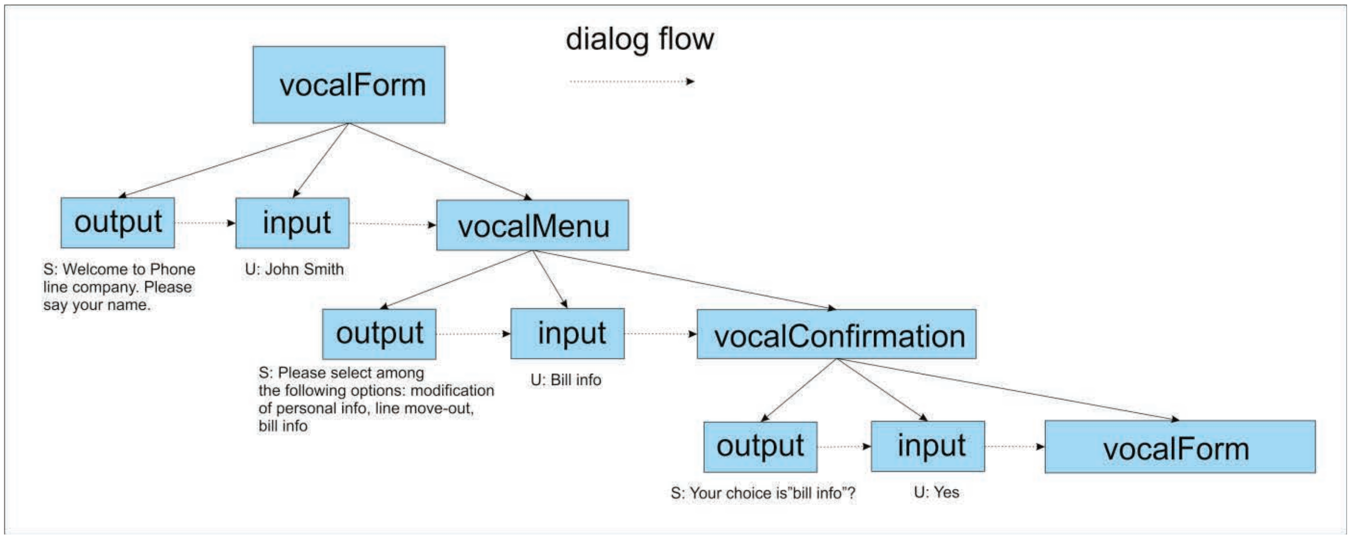
dialog flow
vocalForm
output
input
vocalMenu
S: Welcome to Phone line company. Please say your name.
U: John Smith
output
input
vocalConfirmation
S: Please select among the following options: modification of personal info, line move-out, bill info
U: Bill info
output
input
vocalForm
S: Your choice is"bill info"?
U: Yes

Figure 3. VocalCIOs involved in the Phone line company example.

and the environment itself represent an important factor in the success or not of the system performed.

In the near future, the objective is to implement the software tool that supports the model and transformation rules as well as apply a reverse engineering process to existing applications and recreate them for a multiplatform context.

6. ACKNOWLEDGMENTS

We gratefully acknowledge the support of the ITEA2 Call 3 UsiXML project under reference 20080026, the Mexican PROMEP/103.5/12/4367 project, and the Computer Sciences Faculty of the University of Puebla.

7. REFERENCES

[1] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, Vol. 15, No. 3, June 2003 289–308.

[2] Cuppens, E., Raymaekers, Ch., Coninx, K, A Model-Based Design Process for Interactive Virtual Environments, Proc. of Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2005 (Newcastle upon Tyne, 13-15 July 2005), Lecture Notes in Computer Science, Vol. 3941, Springer, Berlin, 2005, pp. 225-236.

[3] De Boeck, J., Raymaekers, C., Coninx, K. A Tool Supporting Model Based User Interface Design in 3D Virtual Enviroments.GRAPP 2008: 367-375

[4] Flor, T.: ”Experiences with Adaptive User and Learning Models in eLearning Systems for Higher Education” In: Journal of Universal Computer Science, volume 10 (2004)

[5] González-Calleros J., Vanderdonckt J., Muñoz Arteaga J., A Method For Developing 3D User Interfaces Of Information Systems. CADUI 2006: 85-100

[6] Guerrero, J., Vanderdonckt, J., Gonzalez Calleros, J.M., FlowiXML: a Step towards Designing Workflow Management Systems, Journal of Web Engineering, Vol. 4, No. 2, 2008, pp. 163-182.

[7] Guerrero-García, J., González-Calleros, J.M., Vanderdonckt, J., Muñoz-Arteaga, J. A Theoretical Survey of User Interface Description Languages: Preliminary Results. In Proc. of LA-Web/¬CLIHC'2009 (Merida, November 9-11, 2009), IEEE Computer Society Press, Los Alamitos, 2009, pp. 36-43.

[8] Laurent Bouillon, Reverse Engineering of Declarative User Interfaces, Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 21 June 2006.

[9] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Lopez, V.: UsiXML: a Language Supporting Multi-Path Development of User Interfaces. In: Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCIDSVIS’2004 (Hamburg, July 11-13, 2004). Springer-Verlag, Berlin (2005).

[10] Medina, J-L., Chessa, S., Front, A., A Survey of Model Driven Engineering Tools for User Interface Design Proceedings of 6th International Workshop on Task Models and Diagrams TAMODIA’2007 (November 7-9, 2007), Springer, Berlin.

[11] Molina, J.P., Vanderdonckt, J., González, P., Fernández-Caballero, A., Lozano, M.D., Rapid Prototyping of Distributed User Interfaces, Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006 (Bucharest, 6-8 June 2006), Chapter 12, Springer-Verlag, Berlin, 2006, pp. 151-166.

[12] Pellens, B., Bille, W., De Troyer, O., Kleinermann, F.: "VR-WISE: A Conceptual Modelling Approach For Virtual Environments", CD-ROM Proceedings of the Methods and Tools for Virtual Reality (MeTo-VR 2005) workshop, Gent, Belgium (2005)

[13] Schaefer, R., Steffen, B., Wolfgang, M., Task Models and Diagrams for User Interface Design, Proceedings of 5th International Workshop, TAMODIA'2006 (Hasselt, Belgium, October 2006), Lecture Notes in Computer Science, Vol. 4385, Springer Verlag Berlin, 2006.

[14] Stanciulescu, A., A Methodology for Developing Multimodal User Interfaces of Information Systems, Ph.D. thesis, Université catholique de Louvain, Louvain, Belgique, 2008.

[15] Stanciulescu, A., Limbourg, Q., Vanderdonckt, J., Michotte, B., Montero, F., A Transformational Approach for Developing Multimodal Web User Interfaces based on UsiXML, Working Paper IAG n°06/02, Université catholique de Louvain, Louvain School of Management, Louvain-la-Neuve, 2006.

[16] Stanciulescu, A., Vanderdonckt, J., Macq, B., Automatic Usability Assessment of Multimodal User Interfaces Based on Ergonomic Rules, Proc. of E-Mode Joint Workshop on Multimodal Interfaces 2007 (Paris, 27-28 September 2007), S. Praud (ed.).

[17] Thevenin, D., Adaptation en Interaction Homme-Machine: le cas de la Plasticité, Ph.D. thesis, Université Joseph Fourrier, Grenoble, France, 2001. Available online: http://iihm.imag.fr/publs/2001.

[18] Vanderdonckt, J., Model-Driven Engineering of User Interfaces: Promises, Successes, Failures, and Challenges. . In S. Buraga and I. Juvina, editors, Proc. of 5th Annual Romanian Conf. on Human-Computer Interaction ROCHI'2008, (Iasi, 18--19 September 2008), pages 1--10. Matrix ROM, Bucarest, 2008.

[19] 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.

[20] Vanderdonckt,J., Calvary, G., Coutaz, J., Stanciulescu, A., Multimodality for Plastic User Interfaces: Models, Methods, and Principles, in “Multimodal user interfaces: from signals to interaction”, D. Tzovaras (ed.), Chap. 3, Lecture Notes in Electrical Engineering, Springer-Verlag, Berlin, 2007, pp. 79-105..