Towards Multimodal User Interfaces Composition Based on UsiXML and MBD Principles

Sophie Lepreux[1], Anas Hariri[1], José Rouillard[2], Dimitri Tabary[1], Jean-Claude Tarby[2], and Christophe Kolski[1]

[1] Université de Valenciennes et du Hainaut-Cambrésis, LAMIH – UMR8530, Le Mont-Houy, F-59313 Valenciennes Cedex 9, France
[2] Université de Lille 1, Laboratoire LIFL-Trigone, F-59655 Villeneuve d'Ascq Cedex, France
{sophie.lepreux,anas.hariri,dimitri.tabary, christophe.kolski}@univ-valenciennes.fr, {jose.rouillard,Jean-claude.tarby}@univ-lille1.fr

Abstract. In software design, the reuse issue brings the increasing of web services, components and others techniques. These techniques allow reusing code associated to technical aspect (as software component). With the development of business components which can integrate technical aspect with HCI, the composition issue has appeared. Our previous work concerned the GUI composition based on an UIDL as UsiXML. With the generalization of Multimodal User Interfaces (MUI), MUI composition principles have to be studied. This paper aims at extend existing basic composition principles in order to treat multimodal interfaces. The same principle as in the previous work, based on the tree algebra, can be used in another level (AUI) of the UsiXML framework to support the Multimodal User Interfaces composing. This paper presents a case study on the food ordering system based on multimodal (coupling GUI and MUI). A conclusion and the future works in the HCI domain are presented.

Keywords: User interfaces design, UsiXML, AUI (Abstract User Interface), Multimodal User Interfaces, Vocal User Interfaces.

1 Introduction

The reuse is an important issue in software design, and by extension in interactive software design [2]. Means to support reuse have evolved in the meantime, from modularity to component-based development via object development. So the reuse issue brings the increasing of web services, components and others techniques. These techniques allow reusing code associated to technical aspect (as software component). The reuse can be applied to several steps of the development cycle with the support of three types of component: (1) the code components have a small granularity and are used at the development time; (2) the design components (as proposed by [1]) are used to reuse the known solutions at the design time; (3) the business components have a large granularity and are specific to the domain, they are defined at the

J. Jacko (Ed.): Human-Computer Interaction, Part III, HCII 2007, LNCS 4552, pp. 134–143, 2007.

analysis step. They can be associated to a task in the domain. A goal composition based on tasks was studied by to facilitate the reuse [7]. As these business components can integrate technical aspects with HCI, the composition issue appears.

The Model Based-Development (MBD) appears as a solution adapted to the reuse, the User Interface Definition Language (UIDL) named UsiXML (USer Interface eXtensible Markup Language) respects the MBD principles [8]. This language allows defining the User interface from four levels defined by the CAMELEON Project. UsiXML proposes four steps to define the user interface (cf. Figure 1). The Tasks & Concepts level describes the interactive system specifications in terms of the user tasks to be carried out and the domain objects of these tasks. An Abstract User Interface (AUI) abstracts a Concrete User Interface (CUI) into a definition that is independent of any interaction modality (such as graphical, vocal or tactile). A CUI abstracts a Final User Interface (FUI) into a description independent of any programming or markup language in terms of Concrete Interaction Objects, layout, navigation, and behavior. A FUI refers to an actual UI rendered either by interpretation (e.g., HTML) or by code compilation (e.g., Java).

Multimodality appears as a new technology adopted in the current inhomogeneous environments where several types of users work in different states and interact with a multitude of platforms. Multimodality tries to combine interaction means to enhance the ability of the user interface adaptation to its context of use, without requiring costly redesign and reimplementation. Blending multiple access channels provides new possibilities of interaction to users. The multimodal interface promises to let users choose the way they would naturally interact with it. Users have the possibility to switch between interaction means or to multiple available modes of interaction in parallel.

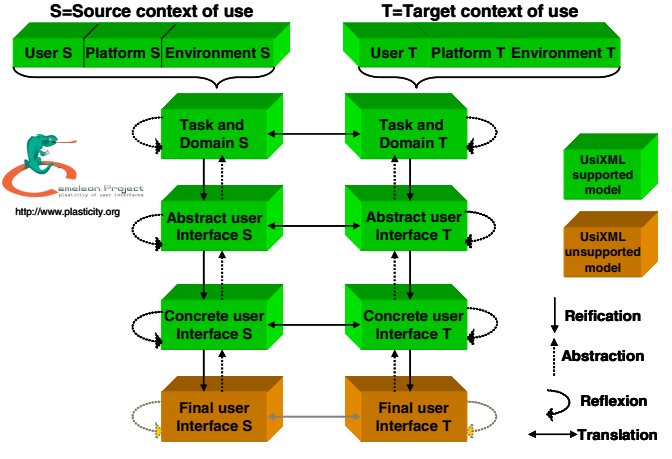


Fig. 1. The four abstraction levels used in the CAMELEON[1] framework

[1] http://giove.isti.cnr.it/cameleon.html

Since a few years, the W3C is working on this aspect and is publishing recommendations concerning a vocal interaction language based on XML, called VoiceXML, which allows describing and managing vocal interactions on the Internet network. VoiceXML is a programming language, designed for human-computer audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and DTMF (Dual Tone Multi-Frequency) key input, recording of spoken input, telephony, and mixed initiative conversations. Its major goal is to bring the advantages of web-based development and content delivery to interactive voice response applications [9, 10, 11].

The second section presents (1) the basic principles of our previous work on the Visual GUI composing based on UsiXML[2], and (2) the new rules to compose user interfaces and in particular multimodal user interfaces. In order to validate the proposed rules, a case study on a food ordering system will be the object of the third section. Finally the paper will conclude with the future works.

2 From GUI Composing to MUI Composing

2.1 Operators at the CUI Level for the GUI Composing

During a previous work, we have proposed composition rules to support GUI composition: each GUI is defined at concrete level of UI definition [4,5]. Since the UI is represented in UsiXML terms and since it is a XML-compliant language (cf. Figure 2), operations could be defined thanks to tree algebra.

In this work, the used notation is based on the data model defined by Jagadish and colleagues [3]. In this model, a data tree is a rooted, ordered tree, such that each node carries data (its label) in the form of a set of attribute-value pairs. Each node has a special, single valued attribute called ***tag*** whose value indicates the type of element. A node may have a ***content*** attribute representing its atomic value. Each node has a virtual attribute called ***pedigree*** drawn from an ordered domain. The pedigree carries the history of "where it came from". Pedigree plays a central role in grouping, sorting and elimination of repetitive elements.

They define a pattern tree as a pair P=(T, F), where T=(V,E) is a node-labelled and edge-labelled tree such that:

- Each node in V has a distinct integer as its label ($i);
- Each edge is either labeled `pc` (for parent-child) or `ad` (for ancestor-descendant);
- F is a formula, i.e. a Boolean combination of predicates applicable to nodes.

This pattern is used to define a database and to define the predicate used in the operations. This notation is adapted to documents specific to interface. Indeed, in the HCI case, the most important is the structure and not the content. For example, it is more important to know that the window has a box as sub-element than that the window has a height equal to 300. So the attributes are stored with the tag. A node is a tag with these attributes and their content. The pattern tree keeps coherent with the variant definition. Another point specific to the database is that the data are in several

[2] http://www.usixml.org

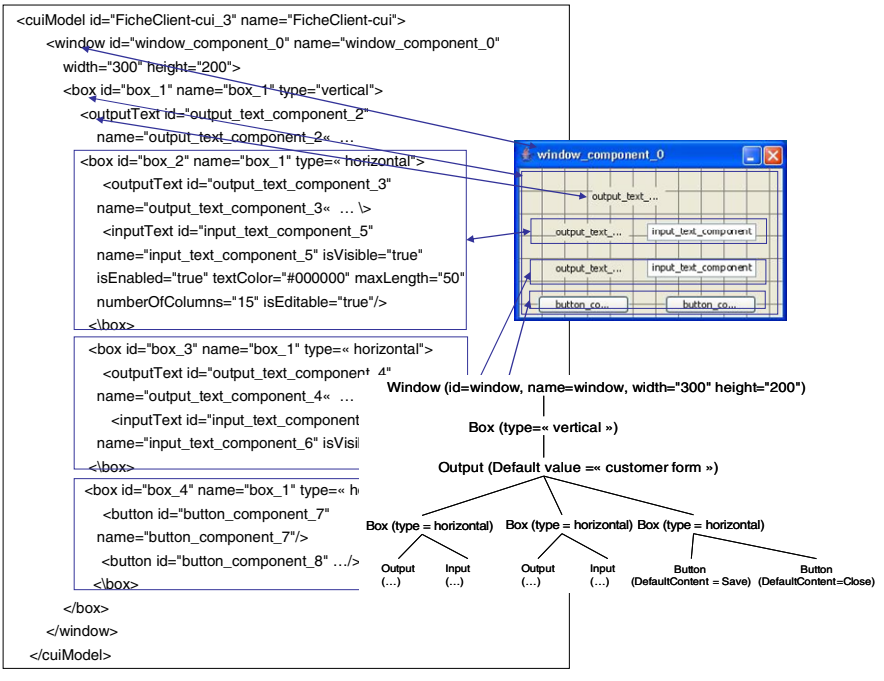


Fig. 2. User interface and its representation in UsiXML and as Tree

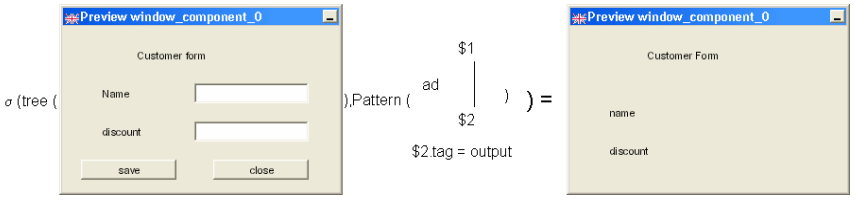


Fig. 3. Example of the *Selection* operator using to select the outputs of the input UI

data trees so the operators use a collection of data trees in input and output. In the HCI case, the input is one (for the unary operators) or two (for the binary operators) XML documents so one or two data trees. The proposed operators to manipulate the CUI model are Similarity, Equivalence, Subset, Set, Selection (cf. Figure 3), Complementary, Difference (Right or Left), Normal Union, Unique Union, Intersection, and Projection. These operations are logically defined on the XML tree and directly performed.

2.2 Adaptation of the Operators at the AUI Level for the Multimodal User Interfaces Composing

If the need is to compose user interface in *Difference* modality then we need to use the upper level as AUI. The same principle as in the previous work based on tree

algebra can be used in the other level of the UsiXML framework. The rules are proposed at the AUI level in order to allow the composition at level which is independent of the modality. A set of operators as *Fusion, Intersection*, and others are adapted to AUI model. An algorithm of Normal Union adapted to AUI model is proposed below:

Normal Union: The Union operation takes a pair of trees T1 and T2 as input and produces an output tree as follows.

Firstly, the root of the output tree T3 is created:

If (T1.$1.tag = T2.$1.tag = abstractContainer) then T3.$1.tag = abstractContainer

The node in which integrate the second tree is chosen.

If (subtree ti = subtree tj, ti ∈ T1, tj ∈ T2) then

If (relation (ti-1, ti) = relation (tj-1, tj)= order independency) then add (ti-1, order independency, ti, order independency, tj-1,) in T3.

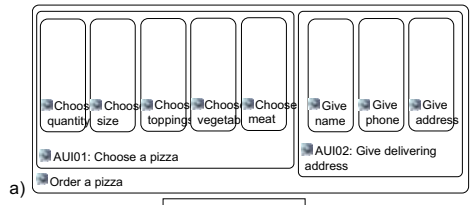
If (relation (ti-1, ti) = relation (tj-1, tj)= enabling) then add (abstractContainer AC1 in which added (ti-1, R1', tj-1), enabling, ti)…

3 A Case Study on a Food Ordering System on Internet

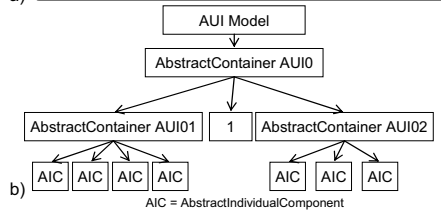
The case study developed in this section illustrates the Union operator presented in the previous part. Two applications are available. The first one is a multimodal application which aims at ordering pizza. The multimodal part which is available is "Choose a pizza". The CUI model and its FUI corresponding (XHTML+VoiceXML) are presented in the figure 4 (c, d), while the "Give delivering address" part is not available and can not content vocal modality (this point is discussed in conclusion). The second application is graphical which allows to ordering Chinese food; likewise CUI model and its FUI (in Java) corresponding to the "Choose Meal" task are realized with GrafiXML editor and presented in figure 4 (c, d, e). The AUI (Abstract User Interface) models of each application are developed (Figure 4a and 5a) with the IdealXML editor [6].

The goal is first to obtain a multimodal application allowing to order Chinese food or pizza, and second to reuse the "give delivering address" task from the Chinese food application.

In order to apply the tree algebra operators, the definition needs to be adapted. While the operators applied to the CUI model need to know the structure of the user interface, the operators applied to the AUI model must in addition taking account the relationships. In our example, the tree representations of the two AUI (figure 4a and 5a) are presented in the figure 4b and 5b. In this representation, the relationships are generalized in R1 and R2 in order to treat different possibilities.



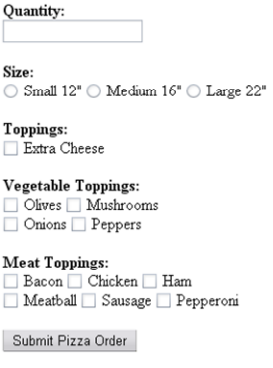
a)



b)

```
<outputText id="Ask_for_pizza_quantity"
          name="Ask_for_pizza_quantity" defaultContent="Quantity:"/>
     <inputText id="pizzaQuantity" name="quantity"
          defaultContent="1"
          voice_prompt="How many pizzas would you like?"
          voice_event_help="Say a number between one and twenty."
          voice_event_nomatch="Sorry I did not understand you."
          voice_event_noinput="You have to pronounce a quantity of
          pizza."/>
         <voice_quantity:grammar>
                  <![CDATA[
                      #JSGF V1.0;
                      grammar pizza_quantity;
                      public <quantity> = 1 | 2 | 3 | 4 | 5 | 6 | 7 |
          8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20
          ;
                  ]]>
         </voice_quantity:grammar>
     </inputText>
<outputText id="Ask_for_pizza_size" name="Ask_for_pizza_size"
          defaultContent="Size:"/>
<group voice_prompt="What size would you like?">
          <radioButton id="radiobuttonSize1"
          name="radiobuttonSize" defaultContent="small" />
          <radioButton id="radiobuttonSize2"
          name="radiobuttonSize" defaultContent="medium"/>
          <radioButton id="radiobuttonSize3"
          name="radiobuttonSize" defaultContent="large"/>
          <voice_quantity:grammar>
                  <![CDATA[
                      #JSGF V1.0;
                      grammar pizza_size;
                      public <size> = small | medium | large ;
                  ]]>
          </voice_quantity:grammar>
</group>
```

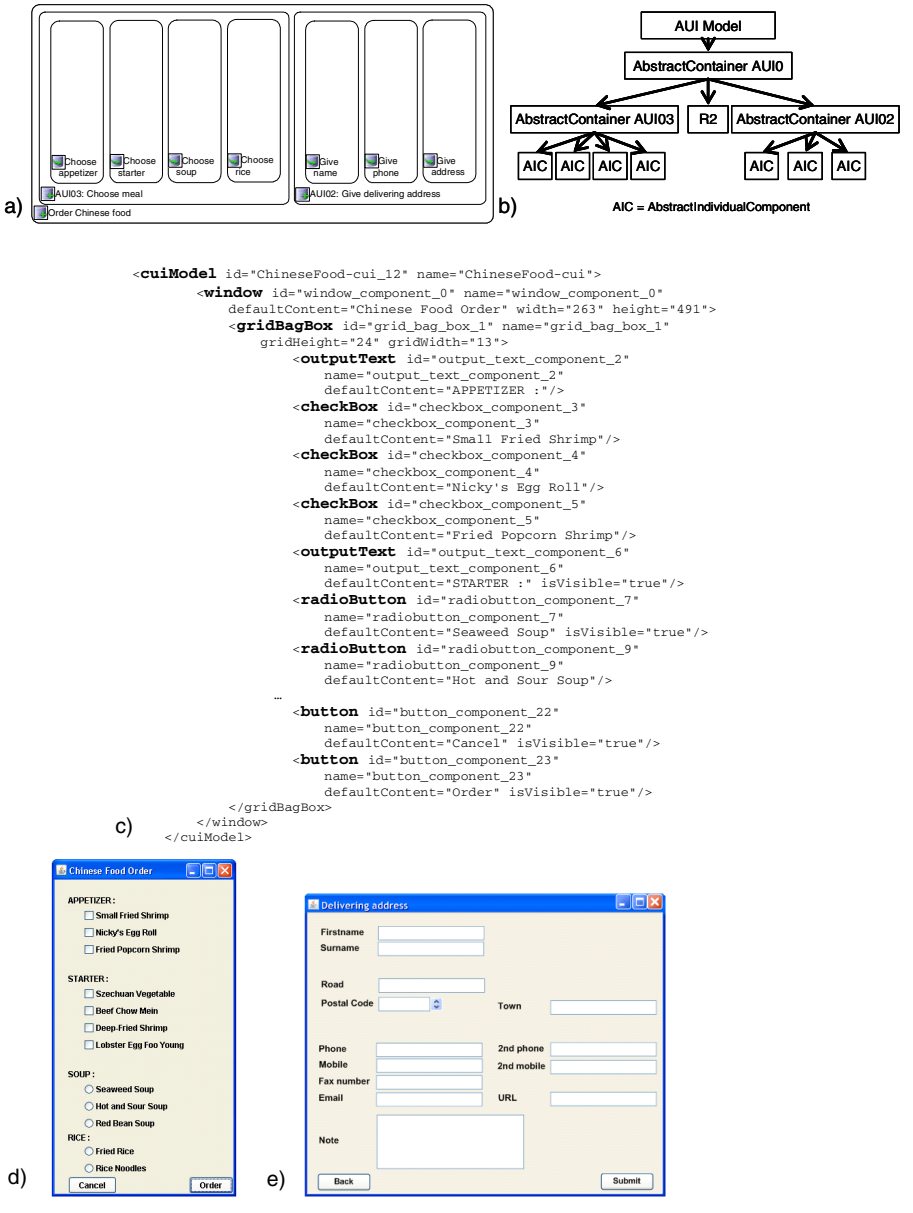
c)



d)

Fig. 4. (a) AUI model of Chinese food ordering application, provided by IdealXML, (b) AUI model in tree representation (c) CUI model extract and (d) FUI associated to the pizza ordering application (Multimodal application but the vocal part is not visible)

If R1 and R2 relationships are “order independency” then the union operator provides a new tree with the three AUIContainers : “Choose a pizza”, “Choose Meal” and “ Give delivering address” with the relationship “order independency “ between these three tasks. The “give delivering address” container (AUI3) was detected as repetitive element in the AUI models and as a result only one of them is reported in the resulting tree.




```
<cuiModel id="ChineseFood-cui_12" name="ChineseFood-cui">
    <window id="window_component_0" name="window_component_0"
        defaultContent="Chinese Food Order" width="263" height="491">
        <gridBagBox id="grid_bag_box_1" name="grid_bag_box_1"
            gridHeight="24" gridWidth="13">
            <outputText id="output_text_component_2"
                name="output_text_component_2"
                defaultContent="APPETIZER :"/>
            <checkBox id="checkbox_component_3"
                name="checkbox_component_3"
                defaultContent="Small Fried Shrimp"/>
            <checkBox id="checkbox_component_4"
                name="checkbox_component_4"
                defaultContent="Nicky's Egg Roll"/>
            <checkBox id="checkbox_component_5"
                name="checkbox_component_5"
                defaultContent="Fried Popcorn Shrimp"/>
            <outputText id="output_text_component_6"
                name="output_text_component_6"
                defaultContent="STARTER :" isVisible="true"/>
            <radioButton id="radiobutton_component_7"
                name="radiobutton_component_7"
                defaultContent="Seaweed Soup" isVisible="true"/>
            <radioButton id="radiobutton_component_9"
                name="radiobutton_component_9"
                defaultContent="Hot and Sour Soup"/>
            ...
            <button id="button_component_22"
                name="button_component_22"
                defaultContent="Cancel" isVisible="true"/>
            <button id="button_component_23"
                name="button_component_23"
                defaultContent="Order" isVisible="true"/>
        </gridBagBox>
    </window>
</cuiModel>
```

Fig. 5. (a) AUI model of pizza ordering application, provided by IdealXML, (b) AUI model in tree representation (c) CUI model extract and (d) FUI associated to the "Choose meal" sub task and (e) FUI associated to the "Give delivering address" sub task of the Chinese food ordering application (Graphical application); the (c, d, e) elements are provided by GrafiXML[3]

[3] GrafiXML is an editor associated to UsiXML available at http://www.usixml.org

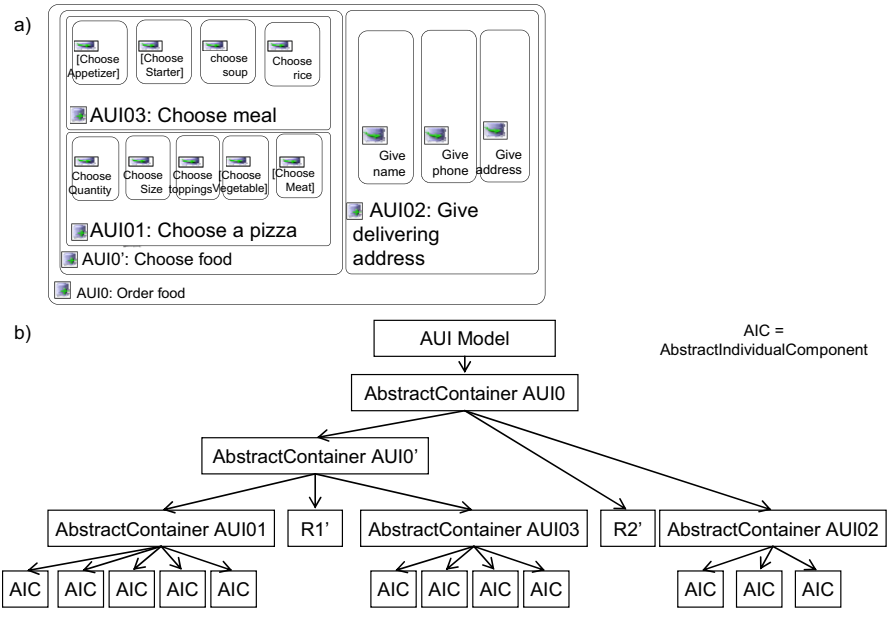


Fig. 6. (a) AUI model result and (b) its tree representation of Union operator

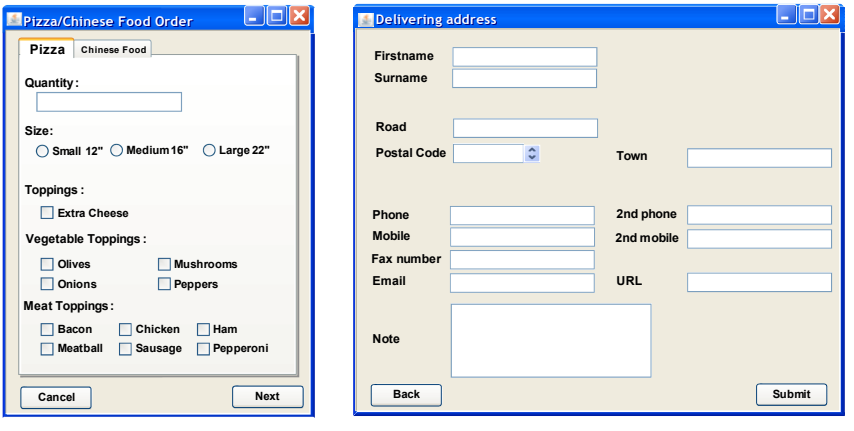


Fig. 7. Resulting FUIs generated from the AUI. In the first window, the first tab is a multimodal reuse from the first application while the second tab is the result from the second application (graphical can be reuse or multimodal must be generated). The second window, corresponding to the common task "Give delivering address", is graphical.

If R1 and R2 are « enabling » relationships, the union operator detects the common part as AUI02 "Give delivering address". As the previous relation is enabling then a new abstractContainer is created (AUI0' and named by the designer). The relation R1' is also chosen by the designer (here order independency) and R2' is enabling

relationship. The result in this case is presented in figure 6. The figure 6a shows the AUI model resulting while the figure 6b shows the tree representation associated to the result.

In order to entirely reuse the existing application, the CUI model is always linked to the AUI model. Thus when the composition (the using of operator) is realized, the CUI parts stay available. The reification in the CUI model can be immediately operated. The result is also generated and is shown in the figure 7. Two windows correspond to the two containers: 1 - AU0': Choose a food and 2 - AUI02: Give delivering address. The tags allow to give the choice to the user between the food to choose: Pizza or Chinese food. The first window is Multimodal because generated in part from a multimodal CUI, whereas the second window is graphical because it is generated from the CUI specific to the graphical modality.

4 Conclusion

From previous work on GUI composing, we tried to apply the same principle of using tree algebra operators to compose multimodal user interfaces. This adaptation is realized at the AUI level to be independent to the modality. The example of the Union of two existing interfaces (one GUI and one MUI) was used to apply the union operator. The issue outlines that the previous work can not be used exactly in the same way so a new adaptation is necessary and proposed. The proposal is validated on the example of the case study. In the context of the case study, some limitations were identified during the automatic generation of multimodal applications development. First, VoiceXML applications are described by context-free grammars. Then, recognized vocabulary is limited. If vocal grammars are easy to prepare for input fields for which we already know all the possible values, it is more difficult even impossible to establish a grammar for open fields. For example, it is not possible to prepare an exhaustive grammar for the field "Name input", because, of course, all the possible responses could not be prepared for this field. Second, it is not possible to obtain synergic multimodal applications, because, we are limited by the X+V language [12]. Indeed, with X+V, the user can choose to use a vocal or a graphical interaction, but it is not yet possible to pronounce a word and click on a object simultaneously, in order to combine the sense of those interactions.

Our research perspectives concern the reuse of the CUI adapted to different modalities; for instance let us suppose two applications with similar tasks: if the first one is defined with a graphical CUI and the other one with a vocal + graphical CUI, how the first application (its CUI) could be transformed into a multimodal version?

Acknowledgement. The present research work has been supported by the "Ministère de l'Education Nationale, de la Recherche et de la Technologie», the «Région Nord Pas-de-Calais» and the FEDER (Fonds Européen de Développement Régional) during the projects MIAOU and EUCUE. The authors gratefully acknowledge the support of these institutions. The authors thank also Jean Vanderdonckt for his contribution concerning UsiXML.

References

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of reusable Object-Oriented Software. Addison Wesley, Massachusetts (1994)
2. Grundy, J.C., Hosking, J.G.: Developing Adaptable User Interfaces for Component-based Systems. Interacting with Computers 14(3), 175–194 (2001)
3. Jagadish, H.V., Lakshmanan, L.V.S., Srivastava, D., Thompson, K.: TAX: A Tree Algebra for XML. LNCS, vol. 2397, pp. 149–164. Springer, Heidelberg (2001)
4. Lepreux, S., Vanderdonkt, J., Michotte, B.: Visual Design of User Interfaces by (De)composition. In: Doherty, G., Blandford, A. (eds.) DSVIS 2006. LNCS, vol. 4323, Springer, Heidelberg (2007)
5. Lepreux, S., Vanderdonckt, J.: Toward a support of the user interfaces design using composition rules. In: Proc. of the 6th International Conference on Computer-Aided Design of User Interfaces, CADUI'2006, Bucharest, Romania, June 5-8, 2006, pp. 231–244. Kluwer Academic Publishers, Boston (2006)
6. Montero, F., Víctor López Jaquero, V.: IDEALXML: An Interaction Design Tool and a Task-based Approach to User Interface Design. In: Proc. of the 6th International Conference on Computer-Aided Design of User Interfaces, CADUI'2006, Bucharest, Romania, June 5-8, 2006, pp. 245–252. Kluwer Academic Publishers, Boston (2006)
7. Nielsen, J.: Goal Composition: Extending Task Analysis to Predict Things People May Want to Do (1994) available at http://www.useit.com/papers/goalcomposition.html
8. Vanderdonckt, J.: A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 16–31. Springer, Heidelberg (2005)
9. VoiceXML 1.0., W3C Recommendation, http://www.w3.org/TR/voicexml10
10. VoiceXML 2.0., W3C Recommendation, http://www.w3.org/TR/voicexml20
11. VoiceXML 2.1, Working Draft, http://www.w3.org/TR/voicexml21/
12. X+V, XHTML + Voice Profile, http://www.voicexml.org/specs/multimodal/x+v/12