Chapter 4

DESIGN OPTIONS FOR MULTIMODAL WEB APPLICATIONS

Adrian Stanciulescu and Jean Vanderdonckt
School of Management (IAG), Université catholique de Louvain
Place des Doyens, 1 – B-1348 Louvain-la-Neuve (Belgium)
E-mail: {stanciulescu, vanderdonckt}@isys.ucl.ac.be – Web: http://www.isys.ucl.ac.be/bchi
Tel: +32 10 47{8349, 8525} – Fax: +32 10 478324

Abstract The capabilities of multimodal applications running on the web are well delineated since they are mainly constrained by what their underlying standard mark up language offers, as opposed to hand-made multimodal applications. As the experience in developing such multimodal web applications is growing, the need arises to identify and define major design options of such application to pave the way to a structured development life cycle. This paper provides a design space of independent design options for multimodal web applications based on three types of modalities: graphical, vocal, tactile, and combined. On the one hand, these design options may provide designers with some explicit guidance on what to decide or not for their future user interface, while exploring various design alternatives. On the other hand, these design options have been implemented as graph transformations per-formed on a user interface model represented as a graph. Thanks to a transformation engine, it allows designers to play with the different values of each design option, to preview the results of the transformation, and to obtain the corresponding code on-demand.

Keywords: Design decision, Design option, Design rationale, Design space, Multimodal user interface, User interface extensible mark up language, Web interface.

1. INTRODUCTION

The combination of design options forms a *design space*. The *design space analysis* [9] represents a significant effort to streamline and turn the open, ill-defined, and iterative [13] interface design process into a more formalized process structured around the notion of design option. A design space consists of a *n*-dimensional space where each dimension is denoted by a single design option. For this space to be orthogonal, all dimensions, and therefore all their associated design options, should be independent of each other. This does not mean that a dimension cannot be further decomposed into sub-dimensions. In this case, the design space becomes a snowflake model. Design options often involve various stakeholders representing different human populations with their own preferences and interests. Consequently, design decisions often result from a process where the various design options are gathered, examined, and ranked until an agreement is reach among stakeholders. This decision process is intrinsically led by consensus since stakeholders' interests may diverge and by trade-off between multiple criteria, which are themselves potentially contradictory. Design options present several important advantages:

- When they are explicitly defined, they clarify the development process in a structured way in terms of options, thus requiring less design effort and striving for consistent results if similar values are assigned to design options in similar circumstances.
- Defining a design option facilitates its incorporating in the development life cycle as an abstraction which is covered by a software, perhaps relying on a model-based approach. Ultimately, every piece of development should be reflected in a concept or notion which represents some abstraction with respect to the code level as in a design option. Conversely, each design option should be defined clearly enough to drive the implementation without requiring any further interpretation effort.

On the other hand, design options also suffer from some shortcomings: design options could be very numerous, even infinite in theory. But in practice, it is impossible to consider a very large amount of design options because of several reasons: they are too complex or expensive to implement, they do not necessarily address users' needs and requirements, they are outside the designer's scope of understanding, or imagination, or background, their decision is not always clear and when they are decided, they may violate some other usability principle or guideline.

We believe that it is important to define such a design space for multimodal web applications because of several reasons: the languages in which they are implemented (e.g., XHTML, VoiceXML, X+V) restrict the amount of possible interfaces to obtain and directly set the CARE properties [6] to assignment and equivalence. In addition, the interaction styles [2] supported by these languages make them appropriate for certain types of applications (e.g., information systems), but totally inadequate for other types (e.g., air traffic control) [11]. Multimodal web applications typically combine three interaction modalities: graphical (e.g., a XHMTL web page in a web browser), vocal (e.g., a VoiceXML application through a multimodal web browser), tactile (e.g., a X+V application running on an interactive kiosk equipped with a tactile screen). These modalities could be combined together, thus multiplying the combination of design options which are specific to each modality and complexifying the entire design space. Sometimes, a design option which was estimated relevant for a particular modality (say the graphical channel) may become totally irrelevant when this modality is combined with another one (say with the vocal channel). The fusion/fission mechanism is generally the one implemented in the browser with which the multimodal web application is run. Independently of any implementation or tool support, having at hands a design space where a small, but significant, set of design options could be envisaged is a contribution which could be useful to any designer of a multimodal web application. This provision avoids designers to replicate the identification and definition of these design options, while leaving them free to consider other options or to overwrite existing ones.

After arguing for the importance of a design space for Multimodal Web User Interfaces (MWUIs) in this section, the next section reviews some efforts devoted to supporting the development life cycle of multimodal web applications and identifies the shortcomings to be addressed in this paper. Section 3 defines the so-called design space for MWUIs, channel by channel, in a way that is independent of any implementation. Section 4 explains one method for implementing this design space in MultiXML, an assembly of software modules for computer-aided design of MWUIs, based on UsiXML [10]. Section 5 exemplifies a case study developed according to the method supported by the MultiXML software. Section 6 summarizes the experience gained with this method and its associated software.

2. RELATED WORK

MWUIs are usually materialized as off-line or on-line applications for which we would like to address the following requirements, which have been mostly used in isolation in the state of the art, but not simultaneously: usage of models to produce the multimodal interface (e.g., [2]), description of these models with a specification language (e.g., DISL, D3ML [5], MXML–www.macromedia.com, RIML [17], UIML–www.uiml.org), XISL [8]), explicit design options for multimodal dialog (e.g., CARE properties [6]), task-based design of multimodal applications [4]).

A representative example of a complete environment for producing several Multimodal Web Applications is MONA (Mobile multimOdal Next generation Applications – http://mona.ftw.at/index.html). It involves a presentation server for a wide range of mobile devices in wireless LAN and mobile phone networks that transforms a single MWUI specification into a graphical or multimodal UI and adapts it dynamically for diverse devices: WAP-phones, Symbian-based smart phones or PocketPC and PDAs. The application design process is based on use cases that allow to refine and validate the design of multimodal UI prototypes for each device. These prototypes are further submitted to a heuristic evaluation performed by evaluators with design experience.

ICARE [3] is a component-based approach for the design and development of multimodal interfaces, composed of elementary components that describe pure modalities. The composition components are: complementarity, redundancy, and equivalence. An editor was designed in order to allow users to graphically assemble components. It offers the possibility of automatic generation of the code supporting the CARE properties [6] in the fusion.

Teresa [4] automatically generates X+V MWUIs in the following way: the initial task model for the envisioned system is transformed into a system task model that is specific for a multimodal platform, that is in turn transformed into an abstract user interface, a concrete user interface, and the code of a final user interface. Designing such UIs implies the use of several parameters, with their associated values, but the authors do not identify a coherent and explicit set of design options that can be combined in a design space. In Teresa, all transformations are hard coded, embedded, and unique, whereas they are made explicit, thus visible and modifiable, executable and multiple in the present approach. All transformations implemented according to the expected design decision are written in the same specification language as for the models (i.e., UsiXML) and could then be modified.

MOST (Multimodal Output Specification Platform) [13] is a platform that allows the design of output multimodal systems (i.e., graphical, vocal and tactile modalities) based on a cycle model composed of three steps (i.e., analysis, specification and simulation). After identifying the necessary output interaction components (i.e., mode, modality and medium) in the analysis step, the specification step formalizes the results of the previous step based on a series of attributes and criteria assigned to each specific output interaction component. Further, depending on the current state of the interaction context, a behavioral model allows the identification of the most suitable output form that can be used in order to present each interaction component. The behavioral model is expressed under the form of a set of election rules that produces an adapted multimodal presentation.

The above described models and tools do not respond to all the requirements identified at the beginning of this section regarding an integrated set of design options for multimodal web applications. In the following sections we present our work that combines the design option requirements presented above into one single, systematic approach.

3. DESIGN OPTIONS FOR MULTIMODAL WEB USER INTERFACES

3.1 Design Options for Graphical UIs

Design options for graphical user interface are described according to the five parameters specified in Fig. 1.

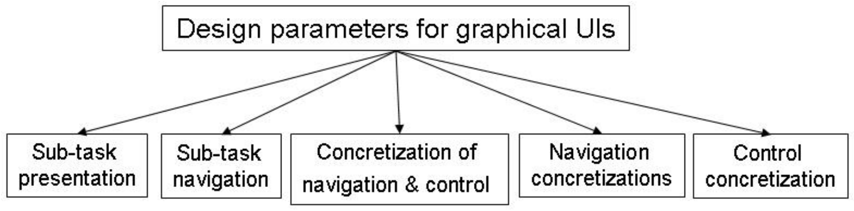

Design parameters for graphical UIs
Sub-task presentation
Sub-task navigation
Concretization of navigation & control
Navigation concretizations
Control concretization

Figure 1. Design parameters for graphical user interfaces.

Sub-task presentation parameter specifies the appearance of each sub-task in the final user interface. The possible values are illustrated in Fig. 2. The presentation of each sub-task can be either separated or combined. Separated presentation identifies the situation when each sub-task is represented in different containers (e.g., different windows), while the combined value identifies the situation when all sub-tasks are presented in the same container. In the last case, three different types of combinations are possible:

- *One by one*: only one sub-task is presented at a time (e.g., in combined box, in tabbed dialog box, in float window).
- *Many at once*: multiple sub-tasks are presented in the same time (e.g., in float window).
- *All in one*: all sub-tasks are presented in the same time (e.g., in areas with separators, in group boxes, in bulleted list, in numbered list).

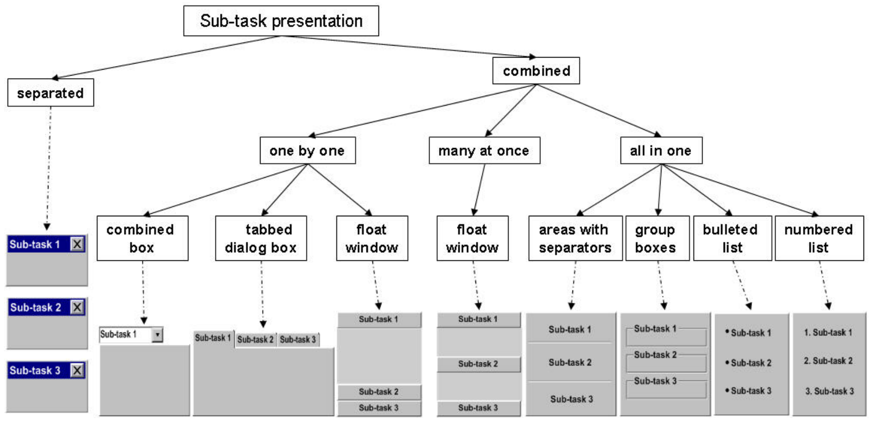


Figure 2. Final representation of sub-task presentation parameters.

Sub-task navigation parameter is an extension of the notation introduced in [16]. It specifies the way in which the dialog control is transferred from one sub-task to another. Fig. 3 illustrates the two possible values: *sequential* or *asynchronous*. The sequential navigation, also called synchronous, restricts the transfer of the dialog control only to a neighbor presented sub-task. The asynchronous type offers more flexibility in transferring the dialog control by eliminating the above restriction and allowing a transfer from any source sub-task to any target sub-task. Passing the dialog control from a source sub-task to a target sub-task implies two simultaneous actions: *deactivate* the container in which the source sub-task is executed (represented here with a yellow bulb) and *activate* of the container in which the target sub-task is executed (represented here with a red bulb).

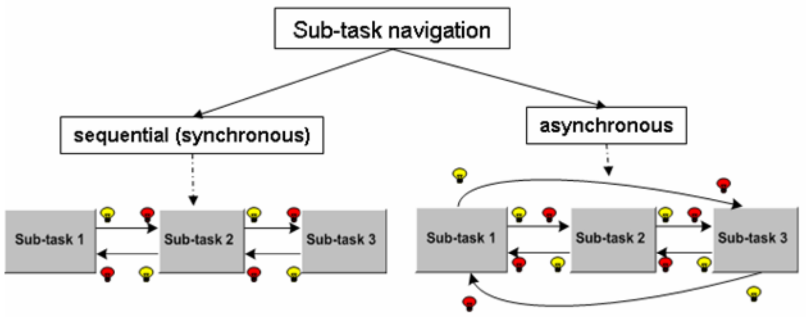
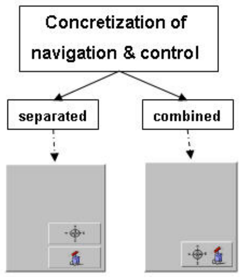


Figure 3. Types of navigation between sub-tasks.

Concretization of navigation and control is a parameter specifying whether the navigation and control are ensured by the same object. In Fig. 4, the *separated* value identifies the situation in which the control and the navigation between the sub-tasks are attached to different objects/logically grouped set of objects. When the same object ensures simultaneously both the navigation and control, the parameter is set to *combined*.



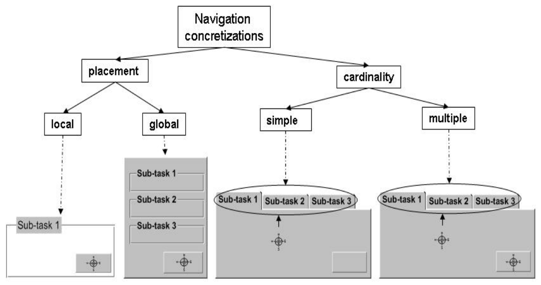


Figure 4. Navigation & control concretization. *Figure 5.* Types of navigation.

Navigation concretization parameter identifies the placement and the cardinality of the navigation objects/logically grouped set of objects that ensure the navigation. Fig. 5 illustrates the different values of this parameter.

There are two *types of placement* for the navigation objects are: *local placement* (specifies the existence of a navigation object attached to each presented sub-task) or *global placement* (a general object ensuring the whole navigation between the sub-tasks). The cardinality specifies the number of objects hosting the navigation. We have identified two *types of cardinality*: *simple* (the navigation is ensured by a single object like an OK button or a single logically grouped set of objects like the (NEXT, PREVIOUS) group of buttons) or *multiple* (the navigation is ensured simultaneously by two or more objects/logically grouped set of objects like a group of tab items in a tabbed dialog box and the (NEXT, PREVIOUS) group of buttons).

Control concretization parameter identifies the placement and cardinality of the control objects. Fig. 6 illustrates the different values of this parameter. There are two *types of placement* of the control objects: *local placement* (specifies the existence of a control object attached to each presented sub-task) or *global placement* (a general object controlling each sub-task). The cardinality specifies the number of objects that assure the control. We have identified two *types of cardinality*: *simple* (the navigation is ensured by a single object like an OK button or single logically grouped set of objects like a group of tab items in a tabbed dialog box) or *multiple* (the navigation is ensured simultaneously by two or more objects/logically grouped set of objects like a group of tab items in a tabbed dialog box and the (NEXT, PREVIOUS) group of buttons).

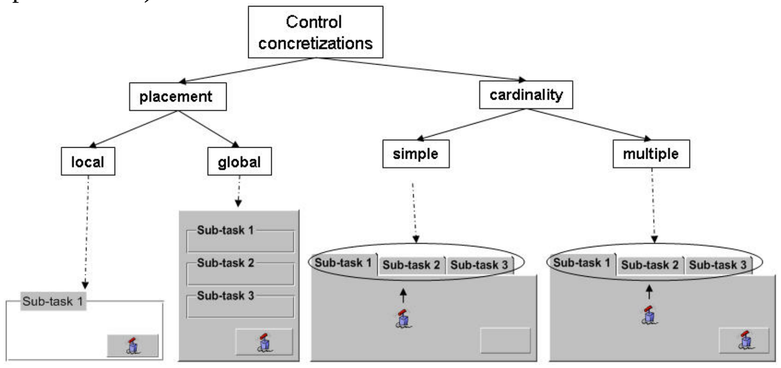


Figure 6. Types of control concretization.

In order to exemplify (Fig. 7) the design options for graphical UIs, we present a car rental system that is sub-divided into three sub-tasks: providing Personal information, selecting the Car's features and filling in the Payment information. The following values for the design options were set to:

- Sub-task presentation: is *combined all in one*. All sub-tasks are combined into their corresponding group box in the same window.
- Sub-task navigation: is *sequential*. Once the user has fulfilled the information required in the group box corresponding to sub-task 1, he can ac-

tivate only the sub-task 2 (see navigation *a*). From the group box associated to sub-task 2, he has two possibilities: returning to the sub-task 1 group box (see navigation *c*) or continue with the sub-task 3 (see navigation *b*). From sub-task 3 group box the user can activate only the sub-task 2 group box (see navigation *d*).

- Concretization of navigation and control: is *combined*. The (OK, CANCEL) logically grouped set of buttons assures in the same time the navigation between the group boxes as well as the control.
- Navigation concretization: is of type *global placement* because for all the sub-tasks there is a general logically grouped set of objects that assures the navigation (i.e., (OK, CANCEL) buttons) and has a *simple cardinality* because the navigation is assured by only one logically grouped set of objects (i.e., (OK, CANCEL) buttons).
- Control concretization: is of type *global placement* because for all the sub-tasks there is a general logically grouped set of objects that assures the control (i.e., (OK, CANCEL) buttons) and has a *simple cardinality* because the control is assured by only one logically grouped set of objects (i.e., (OK, CANCEL) buttons).

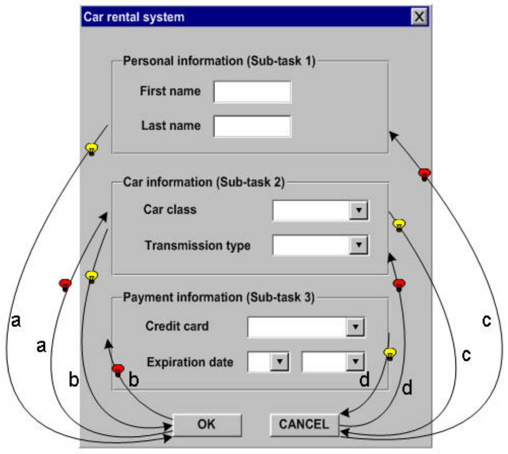


Figure 7. Sub-tasks presented into combined group boxes.

3.2 Design Options for Multimodal UIs

In order to facilitate the development process of multimodal web applications we introduce a set of design options based on four parameters illustrated in Fig. 8. These design options take into consideration the ergonomic criteria for the evaluation of human-computer interfaces presented in [1] and adapt them for the development of MWUIs. For simplification, we consider here only two interaction modalities, graphical and vocal, but other modalities can be taken into account later.

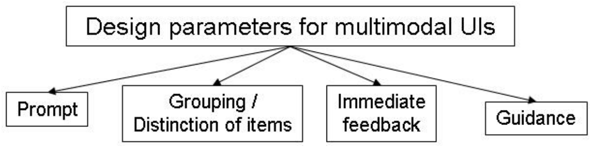

Design parameters for multimodal UIs
Prompt
Grouping / Distinction of items
Immediate feedback
Guidance

Figure 8. Design parameters for multimodal user interfaces.

For each parameter, we provide a definition and we identify a list of possible values (e.g., corresponding to each value we refer to [14]), associating in the same time the corresponding CARE properties described in [6]:

1. *Prompting*: refers to the interaction channels available in order to lead the users to take specific actions whether it should be a data entry or other tasks. The possible values are: *Vocal* (assignment), *Graphical* (assignment), *Multimodal* (complementarity or redundancy).
2. *Grouping/Distinction of Items*: concerns the organization of information items in relation to one another. This criterion takes into account the topology (location) and some structural characteristics (format) in order to indicate the relationships between the various items rendered, to indicate whether or not they belong to a given class, or else to indicate differences between classes. This is further decomposed into:
   a. *Input*: any information input from the user to the system. The possible values are: *Vocal* (assignment), *Graphical* (assignment), *Multimodal* (equivalence or complementarity).
   b. *Output*: any information output from the system to the user. The possible values are: *Vocal* (assignment), *Graphical* (assignment), *Multimodal* (complementarity or redundancy).
3. *Immediate feedback*: concerns system responses to users' actions. These actions may be simple keyed entries or more complex actions such as stacked commands. In all cases computer responses must be provided, they should be fast, with appropriate and consistent timing for any transaction. The possible values are: *Vocal* (assignment), *Graphical* (assignment), *Multimodal* (complementarity or redundancy).
4. *Guidance*: refers to the means available to advise, orient, inform, instruct, and guide the users throughout their interactions with a computer (e.g., messages, alarms, labels). We offer a more precise level of detail corresponding to the two possible types of interaction considered in this section (i.e., graphical and vocal). Thus, the graphical guidance is sub-divided into *textual* and *iconic*, while the guidance for the vocal interaction can be *acoustic* or based on *speech*. The guidance parameter is further subdivided into two parameters:
   a. *Guidance for input*: any guidance offered to the user in order to guide him with the input. The possible values are: *Textual* (assignment), *Iconic* (assignment), *Acoustic* (assignment), *Speech* (assignment), or *Multimodal* (by combining the previous values in a complementary or redundant way).

b. *Guidance for immediate feedback*: any guidance offered to the user in order to guide him with the feedback. *Textual* (assignment), *Iconic* (assignment), *Acoustic* (assignment), *Speech* (assignment), or *Multimodal* (by combining the previous values in complementary or redundancy).

We exemplify the above design parameters with a possible design decision for a multimodal text entry where the user has to input his/her name (Fig. 9). Then value of the prompt parameter is multimodal as the system indicates in a redundant way the task to fulfill by using two modalities: graphical modality (the label Name) and vocal modality used by the systems to invite the user to input his name (1). The guidance for input is of type iconic and is composed of two elements (the microphone icon and the keyboard icon) indicating the available interaction modalities. User's input is of type multimodal as it can be provided in an equivalent manner by employing either the graphical modality (the user is typing his name in the text entry), either the vocal modality (the user is uttering his name using the microphone (2)).The guidance for feedback is of type iconic and is ensured by the loudspeaker icon, indicating the vocal feedback. The feedback of the system to the user's input is of type multimodal as it is expressed by means off two redundant modalities: graphical (the user's typing result) and vocal (the system is uttering the result of the input recognition (3)). Table 1 summarizes a subset of all possible combinations of input and feedback design options for a text entry (for the complete table, we refer to [14]). Only the graphical and vocal interactions are taken into account, but the proposition might be extended to other types of interaction.

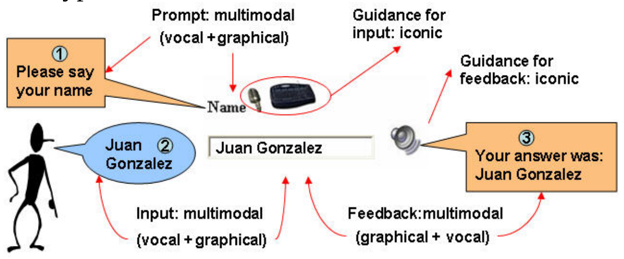
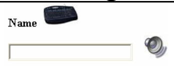
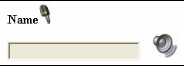
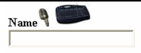

Prompt: multimodal (vocal + graphical)
Guidance for input: iconic
Guidance for feedback: iconic
① Please say your name
Name
Juan ② Gonzalez
Juan Gonzalez
③ Your answer was: Juan Gonzalez
Input: multimodal (vocal + graphical)
Feedback:multimodal (graphical + vocal)

Figure 9. A possible design decision for a multimodal text entry.

Parameters	**Rendering**

Input= graphical
Feedback= graphical and vocal
	

Input= vocal
Feedback= graphical and vocal
	

Input= graphical or vocal
Feedback= graphical
	

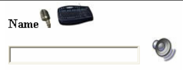
Input= graphical or vocal
Feedback= graphical and vocal
	

Table 1. Possible combinations of design options for input and feedback for text entry.

4. TRANSFORMATIONS BASED ON DESIGN OPTIONS

One of the main advantages of the design space introduced in the previous section is given by the fact that each design option composing the space is independent of any existent method or tool, thus being useful for any developer of multimodal UIs. Under these circumstances, it would be useful to provide an explicit support of the introduced design options. In order to support the development of computer-aided design of MWUIs based on the design options defined in Section 3, we consider MultiXML, an assembly of five software modules. By using MultiXML, we want to address a reduced set of concerns by limiting the amount of design options, thus making the design space more manageable or tractable [7]. Our support involves a transformational approach detailed in [15]. The method consists of a forward engineering process composed of four transformational steps illustrated in Fig. 10. To ensure these steps, transformations are encoded as graph transformations performed on UsiXML models expressed in their graph equivalent. All design options correspond to a class in UsiXML meta-model (e.g., the *tabbed dialog box* value corresponds to *tabbedDialogBox* class, the *feedback* parameter corresponds to the vocalFeedback class).

The five software modules of *MultiXML* tool are (Fig. 10): *IdealXML* tool, *TransformiXML* tool, *GrafiXML* tool (automatically generates graphical UIs (XHTML) from the UsiXML Concrete UI Model), *CFB (Communication Flow Builder) Generator* tool (generates XML code corresponding to the Communication Flow Builder file format by applying XSL Transformations over the Concrete Vocal UI specification of UsiXML), *XHTML+Voice Generator* tool (generates XHTML+Voice code by applying XSL Transformation over the Vocal and Graphical specification of the CUI Model).

In the last step the final graphical UIs are obtained by interpreting the code of the previous step. In order to obtain the vocal and multimodal final UIs, we are not generating the code using our own tools, but we are employing a series of IBM tools. Thus, *NetFront*, the multimodal browser, interprets the XHTML+Voice code and generates the final multimodal UI. For the vocal UI, we are recovering the result produced by CFB Generator tool into *IBM Communication Flow Builder* graphical editor integrated in *IBM Voice Toolkit*. The produced VoiceXML specification is interpreted with *IBM VoiceXML browser*.

5. CASE STUDY

In order to exemplify the design options, an on-line car rental system is described. The main task is decomposed into three basic sub-tasks: determine rental preferences (the user has to select a series of information, such as rental location, expected car features, type of insurance), determine car (the system will launch the search of available cars depending on the preferences established in the previous sub-task. Based on the search results, the user will select the car), provide payment information (the user provides a set of personal information, such as name and card details; then the system checks the validity of the card and finally, the user confirms the payment).

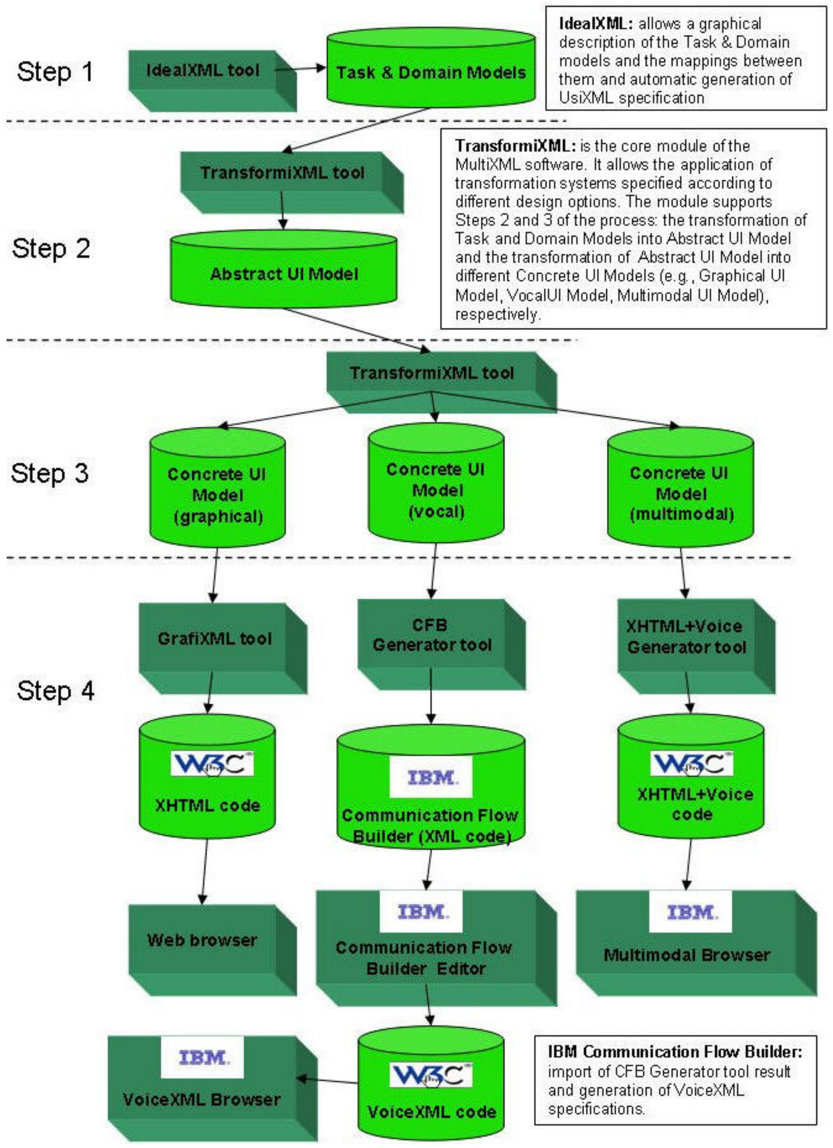

Step 1
IdealXML tool
Task & Domain Models
IdealXML: allows a graphical description of the Task & Domain models and the mappings between them and automatic generation of UsiXML specification
Step 2
TransformiXML tool
Abstract UI Model
TransformiXML: is the core module of the MultiXML software. It allows the application of transformation systems specified according to different design options. The module supports Steps 2 and 3 of the process: the transformation of Task and Domain Models into Abstract UI Model and the transformation of Abstract UI Model into different Concrete UI Models (e.g., Graphical UI Model, VocalUI Model, Multimodal UI Model), respectively.
TransformiXML tool
Step 3
Concrete UI Model (graphical)
Concrete UI Model (vocal)
Concrete UI Model (multimodal)
Step 4
GrafiXML tool
CFB Generator tool
XHTML+Voice Generator tool
XHTML code
Communication Flow Builder (XML code)
XHTML+Voice code
Web browser
Communication Flow Builder Editor
Multimodal Browser
VoiceXML Browser
VoiceXML code
IBM Communication Flow Builder: import of CFB Generator tool result and generation of VoiceXML specifications.

Figure 10. MultiXML software modules.

Based on the design options detailed in Section 3.1 we will illustrate, using different graph transformation rules, their applicability for the car rental system. For final graphical UI (Fig. 14) we consider the parameter *sub-task presentation* and *sub-task navigation*. Fig. 11 illustrates the transformation rule applied in order to generate a UI where the presentation of the sub-tasks is *separated in three windows*. For each top-level abstract container, a graphical container of type window is created.

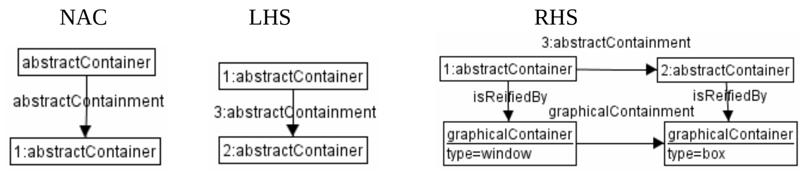


Figure 11. Create a window for each top-level abstract container.

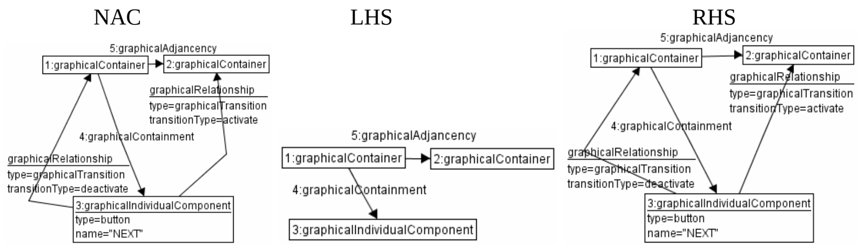
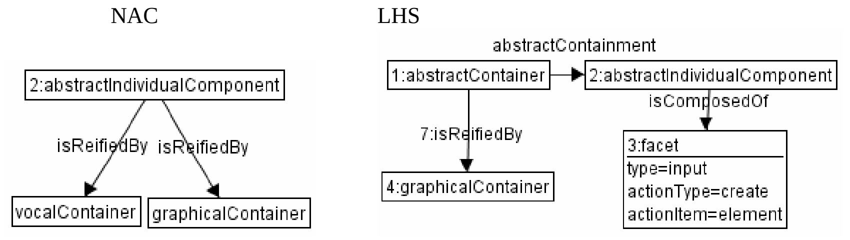


Figure 12. Endow the NEXT button with graphicalTransition features.

The navigation between the windows is of type *sequential* and is concretized in a *global placement* of the (NEXT,PREV) buttons identified on each window. The navigation is assured only by these two logically grouped objects, so the value of the cardinality parameter is *simple*. The transformation rule that endows the NEXT button (similar for PREV button) with *activate* and *deactivate* features is presented in Fig. 12. For a multimodal UI (Fig. 15), a transformation rule generates a multimodal text input that accepts the name of the credit card holder (Fig. 13). We consider the following design options values: *prompt* (graphical and vocal), *input* (graphical or vocal), *feedback* (graphical and vocal), *guidance for input* (iconic with microphone and keyboard icons), or *guidance for feedback* (iconic with speaker icon).



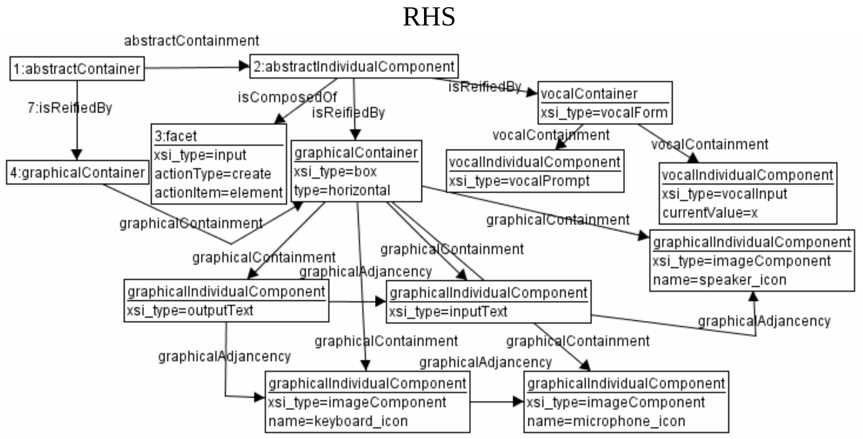
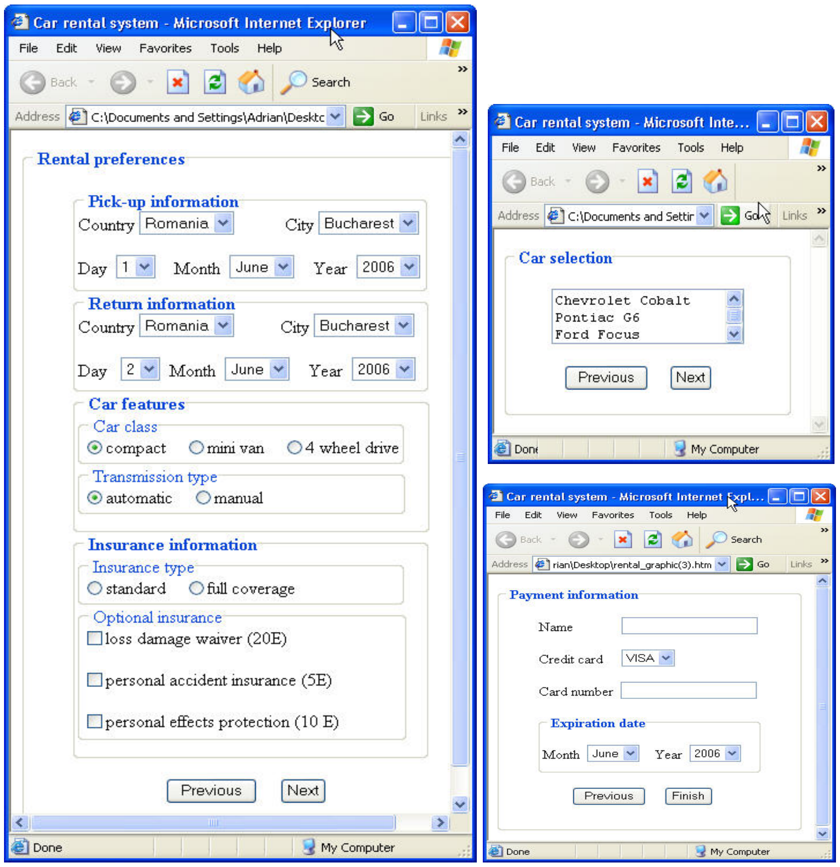


Figure 13. Generation of multimodal text input.


Car rental system - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search
Address C:\Documents and Settings\Adrian\Desktc Go Links
Done My Computer

Rental preferences

Pick-up information
Country Romania City Bucharest
Day 1 Month June Year 2006

Return information
Country Romania City Bucharest
Day 2 Month June Year 2006

Car features
Car class
[✓ compact] mini van 4 wheel drive
Transmission type
[✓ automatic] manual

Insurance information
Insurance type
standard full coverage
Optional insurance
☐ loss damage waiver (20E)
☐ personal accident insurance (5E)
☐ personal effects protection (10 E)

Previous Next

Car selection

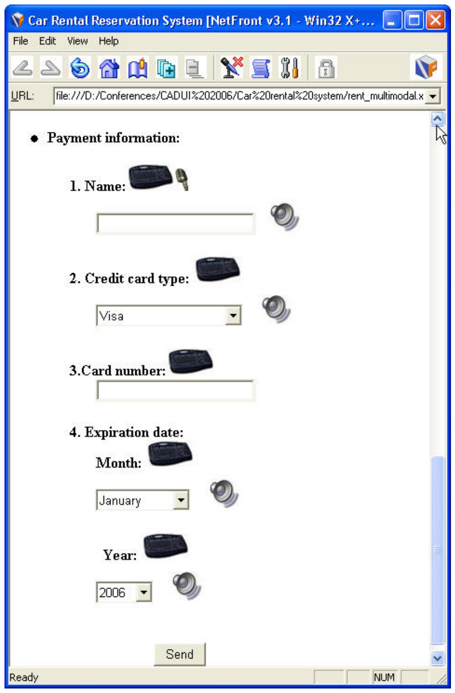
Chevrolet Cobalt
Pontiac G6
Ford Focus

Previous Next

Payment information
Name
Credit card VISA
Card number
Expiration date
Month June Year 2006

Previous Finish

Figure 14. Final graphical UI.



file:///D:/Conferences/CADUI%202006/Car%20rental%20system/rent_multimodal.x

- **Payment information:**

1. Name:

2. Credit card type: Visa

3.Card number:

4. Expiration date:

Month: January

Year: 2006

Figure 15. Final multimodal UI.

6. CONCLUSION

We defined a design space for multimodal web applications based on design options having multiple design values which can be reused by any designer and developer in similar circumstances. Independently of that, these design options have been translated into parameters that govern design rules encoded as graph grammars. Each graph grammar is composed of graph transformation rules so as to perform each rule on the graph corresponding to the UsiXML specifications of the UI under study. A transformation engine could then process these rules. MultiXML introduces a method for structuring the development of multimodal web applications whose results are recuperated in IBM Multimodal Toolkit, from which final UIs can be generated for graphical (XHTML) and multimodal (X+V) applications. For vocal UIs (VoiceXML), the recuperation is ensured by importing a CUI in the Communication Flow Builder format supported by IBM Voice Toolkit.

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the SIMILAR network of excellence (http://www. similar.cc), the European research task force creating human-machine interfaces similar to human-human communication of the European Sixth Framework Programme (FP6-2002-IST1-507609). This research is fully funded by SIMILAR. We also thank Didier Magotteaux and Vincent Wauters for the IBM Belgium grant received with the IBM Multimodal Toolkit™, Software Modeler™, and WebSphere™, with which this research is conducted.

REFERENCES

[1] Bastien, Ch. and Scapin, D.L., *Evaluating a User Interface with Ergonomic Criteria*, In-

ternational Journal of Human-Computer Interaction, Vol. 7, 1995, pp. 105-121.

[2] Beaudouin-Lafon, M., *Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces*, in Proc. of the ACM Conf. on Human Factors in Comp. Systems CHI'2000 (The Hague, 1-6 April 2000), ACM Press, 2000, pp. 446-453.

[3] Bouchet, J. and Nigay, L., *ICARE: a Component-based approach for the design and development of multimodal interfaces*, in Extended Abstracts of the ACM Conf. on Human Factors in Comp. Systems CHI'2004, ACM Press, New York, 2004, pp. 1325-1328.

[4] Berti, S. and Paternò, F., *Migratory MultiModal interfaces in MultiDevice environments*, in Proc. of 7th Int. Conf. on Multimodal Interfaces ICMI'2005 (Trento, 4-6 October 2005), ACM Press, New York, 2005, pp. 92-99.

[5] Chevassus, N., *SNOW – Service for Nomadic Workers, A Framework for authoring and exmpliting multimodal documentation*, W3C Seminar, 21 June 2005.

[6] Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., and Young, R., *Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE properties*, in Proc. of 5th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'95 (Lillehammer, 27-29 June 1995), Chapman & Hall, London, 1995, pp. 115-120.

[7] Hoover, S. and Rinderle, J., *Models and Abstractions in Design,* Design Studies, Vol. 12, No. 4, October, 1991.

[8] Katsurada, K., Nakamura, Y., Yamada, H., and Nitta, T., *XISL: A Language for Describing Multimodal Interaction Scenarios*, in Proc. of 5th Int. Conf. on Multimodal Interfaces ICMI'2003 (Vancouver, 5-7 Nov. 2003), ACM Press, New York, 2003, pp. 281-284.

[9] Limbourg, Q., Vanderdonckt, J., and Souchon, N., *The Task-Dialog and Task-Presentation Mapping Problem: Some Preliminary Results*, in Proc. of 7th Int. Workshop on Design, Specification, Verification of Int. Systems DSV-IS'2000 (Limerick, 5-6 June 2000), Lecture Notes in Comp. Science, Vol. 1946, Springer-Verlag, 2000, pp. 227-246.

[10] 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 Conf. on Engineering for Human-Computer Interaction EHCI-DSVIS'2004 (Hamburg, July 11-13, 2004), LNCS, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 200-220.

[11] MacLean, A., Young, R., and Moran, T., *Design Rationale: the argument behind the Artifact*, in Proc. of the Conf. on Human Aspects in Comp. Systems CHI'89, pp. 247-252.

[12] Mueller, W., Schaefer, R., and Bleul, S., *Interactive Multimodal User Interfaces for Mobile Devices*, in Proc. of 37th Hawaii Int. Conf. on System Sciences HICSS'2004, Track 9 (Hawai, 5-8 January 2004), IEEE Comp. Society Press, Los Alamitos, 2004.

[13] Rousseau, C., Bellik, Y., and Vernier, F., *Multimodal Output Specification/simulation Platform*, in Proc. of 7th Int. Conf. on Multimodal Interfaces ICMI'2005 (Trento, 4-6 October 2005), ACM Press, New York, 2005, pp. 84-91.

[14] Stanciulescu, A., *A Transformational Approach for Developing Multimodal Web User Interfaces*, DEA thesis, Univ. catholique de Louvain, Louvain-la-Neuve, 12 June 2006.

[15] Stanciulescu, A., Limbourg, Q., Vanderdonckt, J., Michotte, B., and Montero, F., *A Transformational Approach for Multimodal Web User Interfaces based on UsiXML*, in Proc. of 7th Int. Conf. on Multimodal Interfaces ICMI'2005 (Trento, 4-6 October 2005), ACM Press, New York, 2005, pp. 259-266.

[16] Vanderdonckt, J., Limbourg, Q., Florins, M., *Deriving the Navigational Structure of a User Interface*, in Proc. of 9th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'2003 (Zurich, 1-5 Sept. 2003), IOS Press, Amsterdam, 2003, pp.455-462.

[17] Ziegert, T., Lauff, M., and Heuser, L., *Device Independent Web Applications- The Author Once - Display Everywhere Approach*, in N. Koch, P. Fraternali, M. Wirsing (eds.), Proc. of 4th Int. Conf. on Web Engineering ICWE'04 (Munich, 28-30 July 2004), Lecture Notes in Computer Science, Vol. 3140, Springer-Verlag, Berlin, 2004, pp. 244-255.