Automatic Usability Assessment of Multimodal User Interfaces Based on Ergonomic Rules

Adrian Stanciulescu[1], Jean Vanderdonckt[1], Benoit Macq[2]
[1]School of Management (IAG), Université catholique de Louvain
Louvain-la-Neuve (Belgium)
E-mail: {stanciulescu, vanderdonckt}@isys.ucl.ac.be
Tel: +32 10/47{8349, 8525}
[2]Laboratoire de Télécommunications et Télétéction (TELE)
Université catholique de Louvain (UCL)
Louvain-la-Neuve (Belgium)
E-mail: benoit.macq@uclouvain.be
Tel: +32 10/472271
Web: http://www.similar.cc

Abstract. In the context of Open Interface project, the UsiXML language is currently evolving in order to encompass full multimodal user interfaces so that they are compliant with the tools produced by the OpenInterface platform and to go beyond multimodal web user interfaces which have been addressed so far. Therefore, the goal of this paper consists in extending the UsiXML language (www.usixml.org) with new functionalities that explicitly address the behavioral features required by the dynamic aspects of multimodal user interfaces, whether they are web oriented or not. For this purpose, a real-world large-scale case study is investigated so as to test the specification of multimodal user interfaces corresponding to the same tasks, but with different modalities, single or combined.

Keywords: multimodal interaction, navigational task instruction, usability evaluation rules.

1 Introduction

Information visualization [Card99] has become a successful paradigm for human-computer interaction. Numerous interface techniques have been proposed and an increasing number of empirical studies describe the benefits and problems of information visualization [Beazr90, Scha96, Horn99, Chen00]. Zoomable user interfaces have been extensively discussed in the literature on information visualization. Zoomable user interfaces organize information in space and scale, and use translation and zooming as their main interaction techniques [Perl93, Bede96]. Research prototypes of zoomable user interfaces include interfaces for storytelling [Drui97], Web browsing [High98], and browsing of images [Comb99, Bede00]. However, few empirical studies have investigated the usability of zoomable user

interfaces, and the results of those studies have been inconclusive. While zoomable user interfaces have been discussed since at least 1993 [Perl93], no definition of zoomable user interface has been generally agreed upon.

In zoomable user interfaces, space and scale are the fundamental means of organizing information [Perl93, Furn95]. The appearances of information objects are based on the scale at which they are shown. Most common is geometric zoom, where the scale linearly determines the apparent size of the object. Objects may also have a more complex relation between appearance and scale, as in so called semantic zooming [Perl93, Fran94], which is supported in the zoomable user interface toolkit Jazz [Bede00]. Semantic zooming is commonly used with maps, where the same area on the map might be shown with different features and amounts of detail depending on the scale. The second main characteristic of zoomable user interfaces is that the information space is directly visible and manipulable through translation and zooming. Translation changes the area of the information space that is visible, and zooming changes the scale at which the information space is viewed.

Usually, translation and zooming are controlled with the mouse or the keyboard, so that a change in the input device is linearly related to how much is translated or zoomed. In contrast, interfaces for combined spoken and pen-based input may be particularly effective for interacting with dynamic map systems, although multimodal interfaces that recognize two or more naturally-integrated input streams are still by and large in the planning stages. Although research relevant to the design of multimodal systems that incorporate speech and pen input is beginning to emerge [Ovia94a, Ovia94b], the problem of how to optimize such interfaces for map displays has received little attention. [Ovia96] shows that within a visual-spatial domain such as maps, clear performance advantages exist for supporting multimodal human-computer interaction. In comparison with the speech-only input to a map, combined use of pen and voice actually was faster, less error-prone, and input involved less complex linguistic expressions to be reorganized and parsed. With respect to relative efficiency, time required to complete map-based task was shorter during multimodal than speech-only input, primarily because location can be designated more precisely, rapidly, and with less effort and error using the pen.

In this article, we will address the followings:

- Automatic usability assessment of multimodal applications that enable navigational tasks over large scale images.
- Users interact directly with the information space by translation and zooming
- Interaction modalities are graphical (keyboard, mouse), vocal (speech input) and tactile (finger, stylus pen)
- Composing elements of navigational tasks are specified sequentially or in an order-independent manner.

2 Context of the work

2.1 OpenInterface platform

Description. Numerous multimodal laboratory prototypes embedding innovative modalities have been developed since R. Bolt's seminal demonstrator [Bolt80]. While scientific understanding and empirical knowledge of multimodal interaction have burgeoned, very few devices in everyday life, such as mobile phones, are multimodal. The OpenInterface project (www.openinterface.org) aims are:

1. To provide an open source platform for the design and rapid development of multimodal prototyped applications as a central tool for a truly iterative user-centered process
2. To ground the development of multimodal interactive systems in a scientific understanding of multimodal interaction. This will be achieved through reusable software components in the platform that are directly inferred and defined from theoretical results on multimodality
3. To turn the results into industrial standards by way of the platform.

Components are the basic objects manipulated by the OpenInterface platform. Each one represents a bundled piece of software that provides a set of services/functionalities ranging from input devices driver, signal-treatment algorithm, network module, graphical interface, etc. To be able to manipulate a component, the OpenInterface platform requires the description of the interface of the component. This description is specified in CIDL description language (Component Interface Description Language). Once the CIDL is specified, the component can then be reused easily in any OpenInterface application. OpenInterface components can be composed together to create a network of components managing some advanced task. Such an inter-connection of components is called a pipeline. In order to be manipulated by the OpenInterface platform, a pipeline must be specified in the PDCL description language (Pipeline Description and Configuration Language). A PDCL description defines the components that are used in the pipeline and how they are connected together.

Advantages. The OpenInterface platform benefits from a set of advantages:

- Allow seamless integration of heterogeneous software. The platform will manage the translation and the communication of the data among the different programming languages using existing tools. The currently supported languages are C/C++, Java and Matlab, but support of other languages can be added rather easily
- Allows rapid prototyping of multimodal applications thanks to the bundled generic fission and fusion mechanism and the easy software connection.
- The delivered software is a reusable independent unit.

Shortcomings. However the platform suffers from a set of shortcomings:

- There is no graphical editor that enables designers to specify the components. Once a component is created it would be then easy to modify/replace it in order to evaluate the designed application

- There are no means to assess the usability of the systems at design time. For this purpose a new module should be implemented, and therefore the deployment of new internal procedures in the source code would be necessary. In this case it would be impossible to add/update/delete a usability criteria without modifying the code. This is an important shortcoming due to the lack of flexibility in the context of the continuous evolution of usability criteria. This evolution is also motivated by the appearance of new interaction modalities.

2.2 UsiXML language

Description. Many User Interface Description languages have been introduced so far that addresses different aspects of a UI: XUL, UIML, XISL, VoiceXML, X+V, TeresaXML. Depending on the perspective the UIDL may address portability, device independence, support for multiple computing platforms, user-centered design, iterative and incremental development to name a few. This proliferation results in many XML-compliant dialects that are not (yet) largely used and that do not allow interoperability between the tools that have been developed around them. Moreover, there has been little or no integration of the various models that address the needs of traditional graphical UIs and multimodal UIs, simultaneously.

In order to cope with the above challenges and others, we have selected and expanded UsiXML (User Interface eXtensible Markup Language – see www.usixml.org) with new functionalities that explicitly address the behavioral features required by the dynamic aspects of multimodal user interfaces, whether they are web oriented or not. UsiXML is structured according to the four basic levels of abstraction (Figure 1) defined by the Cameleon reference framework identified in [Calv03]. This framework is a reference for classifying UIs supporting multiple target platforms or multiple contexts of use in the field of context-aware computing and structures the development life cycle into four levels of abstraction: task and concepts, abstract UI, concrete UI and final UI. The identification of the four levels and their hierarchical organization is built on their independence with respect to the context in which the final system is used. Thus, the Task and Concepts levels is computation independent, the Abstract UI level is modality independent and the Concrete UI level is toolkit independent.

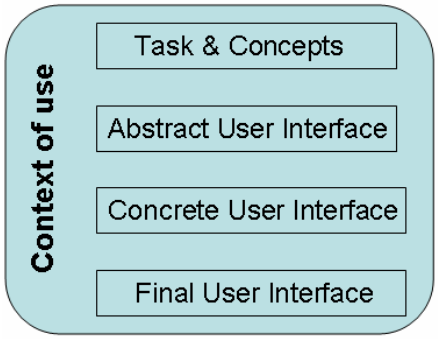

Fig. 1. Cameleon Reference Framework for multi-target UIs.

Advantages. The selection of the language UsiXML was based on the following motivations [Stan06]:

- Support for multimodal input/output: our ontology is based on a set of models that enable multiple input/output interaction modalities. These modalities are: graphical, vocal and tactile.
- Separation of modalities: the concepts corresponding to each modality are syntactically separated one from each other. This provides the developer with two advantages: (1) flexibility during the development process as the UI corresponding to each involved interaction can be specified separately and further combine them altogether, (2) reusability of the specification or part of a specification corresponding to an interaction modality in other applications that involve the use of the same modality.
- Support for CARE properties: the concrete model of UsiXML language is composed of concrete elements (i.e., graphical/vocal concrete interaction objects) that allow the support of CARE properties. For input modalities the supported properties are: assignment and equivalence, whereas the for output modalities assignment, redundancy and equivalence are supported.
- Extendibility to new modalities: the proposed ontology allows the extension with new types of interaction modalities thanks to the modularity of the framework where each model is defined independently of the others. This comes in the spirit of the principle of separation of concern adopted when modeling the language. Thus, at any time concepts belonging to a new modality could be connected to the already existing ontology. This is a principle that we would like to cover, but we are well aware of the fact that very complex interactions can not be supported.

Shortcomings. While the graphical interaction is well covered by UsiXML thanks to the numerous graphical concepts introduced and defined in [USIX06], the vocal interaction suffers from a series of shortcomings:

- The grammar element presents the user's instructions as a whole, without structuring it according to the composing elements of the instruction. For instance, in the context of user's instructions that follow the action-object-parameter paradigm, the composing elements are not split into chucks and assigned to a grammar sub-element. Consequently, the values of the instruction's components can not be reused.
- The grammar syntax does not enable to define the order of utterance of the composing elements: there is no manner of specifying an alternative between two or more utterances, a sequence of utterances or order-independent utterances. Consequently, all possible combinations between the elements of an instruction have to be explicitly specified in the grammar. This will result in a high number of possible combinations that will further increase along with the growing of the number of elements.
- There are no means to specify the cardinality of an uttered element: one user would like to assign just one action to a specific object of the UI, while some others two or more actions. For instance, "Translate top left" vs. "Translate and scale top left".

2.3 UsiXML for OpenInterface

Description. Figure 2 outlines the general schema for testing the usability of multimodal applications developed on top of Open Interface platform:

- Usability evaluation tests are conducted upon the Open Interface Applications
- A set of multimodal ergonomic rules will be abstracted from the results of the evaluation tests and gathered into a knowledge base
- Multimodal Open Interface Applications will be abstracted into UsiXML specification language
- A usability adviser tool will check the ergonomic rules upon the UsiXML specification in order to ensure a certain level of guidance during the development process of the multimodal application.

So far our knowledge base is composed of evaluation rules that address the requirements of GUIs mainly. This knowledge base will be further extended in order to cover multimodal applications that involve graphical, vocal and tactile UIs.

Advantages. The advantages of a usability adviser tool are significant:

- Real time guidance during the development life cycle when adding, deleting or modifying the components of a UI
- Evolutional knowledge base as it allows adding new ergonomic rules
- Portability of the knowledge base thanks to a single file that stores the existing set of rules
- Centralizes knowledge base that can be shared between multiple users connected to the same network
- Ergonomic rules are expressed in a formal language close to the natural language
- Learning the formal language for specifying the ergonomic rules is not mandatory as there is a rule specification assistant
- Multiplatform ergonomic rule validation as the evaluation is based on UsiXML, a platform independent language.

Shortcomings. However, we have identified a set of shortcomings with respect to the automatic usability evaluation tools:

- There is no existing tool that is capable of testing the usability of both web and non-web applications
- The ergonomic rules are most of the time expressed in a natural language, whereas the usability tools apply rules expressed in a formal language. But the natural language is much more complex and allows expressing more precise information then the formal one. Therefore, applying formalized ergonomic rules often results in some limitation with respect to the evaluation.

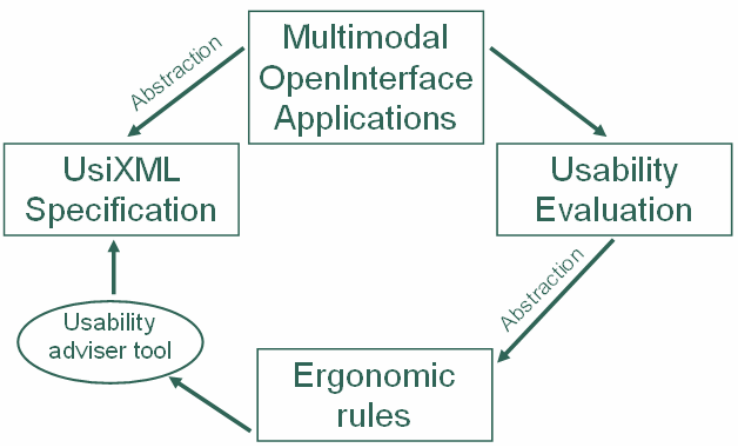

Fig. 2. Usability evaluation of OpenInterface applications based on UsiXML specifications.

3 Navigation in large scale images

3.1 Existing solutions

We are not aware of any technology that allows production of one piece, high-quality displays of arbitrary size. Proposed techniques typically involve combining multiple smaller displays into a single large display of high pixel count.

One common solution is to connect two or more computer monitors to a single computer, as supported by current operating systems, such as Microsoft Windows. In this setup, all connected monitors form a single logical display. This display allows large scale images to span multiple displays. However, [Grud01] observed that the visible gap between individual monitors discouraged users from having windows span multiple displays. His study suggests that users instead use additional monitors to separate windows belonging to different tasks. But, when navigating in large scale image there are no multiple windows to span as the image is displayed usually in a single window.

In order for large displays to overcome this effect, a substantial amount of research has been invested in the creation of seamlessly tiled display systems [Frun00]. Several solutions for avoiding the visible seams have been proposed, including the integration of up to four identical LCDs by butting them together into a single large LCD (a 9-megapixel display by IBM-http://domino.research.ibm.com/comm/pr.nsf/pages/news.20010627_display.html), the construction of video walls by assembling back projection modules with small borders - http://www.panasonic.com/business/medicalvideo/home.asp2, as well as a

series of research prototypes evolving around tiled hi-res projection displays [Here00]. Compound displays composed of multiple display units surrounding the user have been used in virtual reality, such as flight simulation [Mene00], and in immersive environments, such as the CAVE [Cruz92]. These proposed solutions are still cost-intensive, space-intensive, or both, which has prevented these technologies from reaching the mass market. Consequently, navigational techniques are required in order to go beyond the limits of nowadays technology.

3.2 Proposed solution

[Carr82] characterizes the instructions for controlling a robot based on two design variables: hierarchical structure and congruence. From the hierarchical point of view instructions could be hierarchical (verb-object-qualifier, e.g. MOVE ROBOT FORWARD) or non-hierarchical (verb-only, e.g. ADVANCE), whereas from the congruency perspective the possible values are congruent (meaningful pairs of opposites, e.g. RIGHT/LEFT) or non-congruent (non-symmetric pairs, e.g. GO/BACK). Usability studies performed over a set of 16 instructions show that the subjects disapproved non-hierarchical non-congruent form and gave the highest rating for hierarchical congruent form. This is because congruence helped subjects to remember the natural pairs of concepts and terms, whereas hierarchical structure enabled them to easily master the commands with only one rule of formation (verb-object-qualifier). Therefore, error rates and retention are dramatically lower for the congruent hierarchical forms.

In order to overcome the drawbacks identified in Section 3.1 and to ease the navigation in large scale images we propose the use of the 3X3 overlaying grid illustrated in Figure 3. The dimensions of the grid were selected for three different reasons: (1) enables the possibility of specifying congruent instructions thanks to symmetric pairs (Top /Bottom, Left /Right), (2) enables a hierarchical structure of the instruction (Y coordinate – X coordinate), (3) generates one single center column and center row enabling users to specify precisely the navigation instructions.

The small blue arrows indicate the possible directions of translation. The following notions were introduce and defined:

- **Visibility zone:** is the area of the image that is visible for the user.
- **Exploration zone:** is the complete area including the non-visible as well as the visible part of the image
- **Zoom factor:** is the level of magnification between two views. It is always greater or equal to 0.

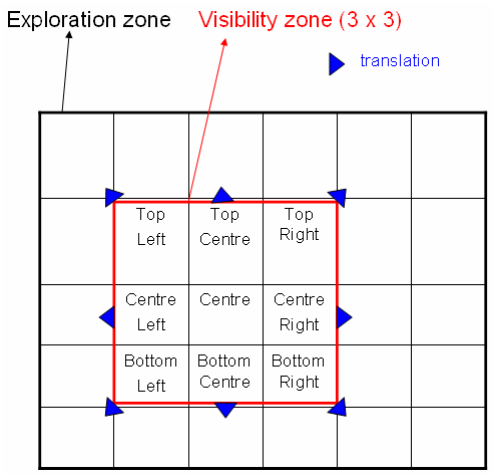

Fig. 3. Cell names for the visibility zone in 3X3 images

Figure 4 illustrates the visibility zone of a large scale image displaying the map of the Brussels city surroundings. After the execution of the task Translate Top Left, the visibility zone displays the area illustrated in Figure 5.

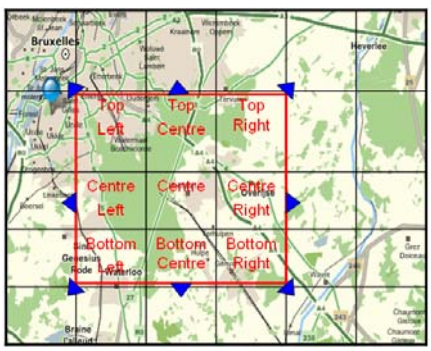

Fig. 4. Visibility zone before the execution of a translation task.

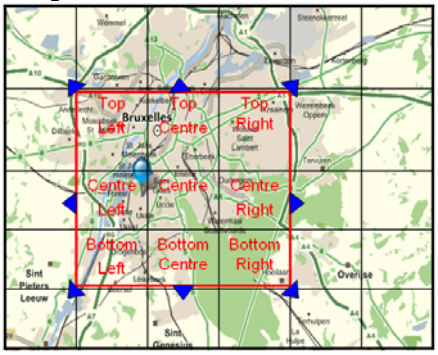

Fig. 5. Visibility zone after the execution of a translation task

4 Structure of the navigational task

4.1 Taxonomy of navigation tasks

[Plai94] presents a taxonomy of image browsers as a result of a comparison of techniques and features related to the presentation and operational aspects. This taxonomy allowed us to identify the specific navigational tasks when browsing large

scale images. Therefore, we introduce hereafter a taxonomy of navigational tasks that was adapted and improved in order to respond to the requirements of navigational tasks over 3X3 images. Moreover, the taxonomy has the advantage of being modality-independent:

- Translation: top / bottom / left / right / main diagonal up (Bottom Left) / main diagonal down (Top Right)/ second diagonal up (Bottom Right)/ second diagonal down (Top Left)
  - ✓ Translate without specifying a value: the user decides when to stop the translation
  - ✓ Translate by specifying the number of grid units
- Zooming: zoom in / zoom out
  - ✓ Zooming without specifying the zoom factor: it is inferred from the selected zoomed area and the size of the display window
  - ✓ Zooming with a zoom factor:
    - ❖ Fixed zoom factor: zoom by steps
    - ❖ Specified zoom factor: the value is specified by the user
  - ✓ Zooming in a single selected grid unit (e.g., zoom in Top Left / zoom in Centre, zoom in Bottom Right)
  - ✓ Zooming in multiple selected grid units:
    - ❖ Adjacent horizontal grid unit selection: zoom in Horizontal Top/ zoom in Horizontal Centre / zoom in Horizontal Bottom
    - ❖ Adjacent vertical grid unit selection: zoom in Vertical Left / zoom in Vertical Centre / zoom in Vertical Right
    - ❖ Diagonal grid unit selection: (e.g., zoom in Left Centre to Bottom Right / zoom in Centre to Top Right.

4.2 Reaching the structure

Based on the taxonomy introduced before, we started our research concerning the structure of a navigational task from the general structure of an instruction in ISs. According to [IBM92] natural languages typically have many more nouns than verbs, and a graphical UI typically contains more objects then actions. Just as the same verb can be applied to many nouns the same action can be applied to many objects, independent of the type of UI, be it graphical, vocal, etc. Therefore, the action/object paradigm is defined as a pattern for interaction in which a user selects an action and an object to apply it. But objects are usually endowed with features which help us characterize them. Therefore, in ISs these features were transposed into parameters assigned to objects. Consequently, the general structure of an instruction is composed of the following three elements:

$$Instruction := \{Action, Object, Parameter\}$$

These elements could have single or multiple cardinality or could be even optional depending on the context in which the instruction is used. Therefore, the general format of an instruction is:

Instruction:= Action 1, Action 2,…,Action n, Object 1 param11 param12….param1m, Object 2 param21 param22 param2p,….,Object r param r1 param r2…param rt

For a more clear expression of the general format we present hereafter its Extended Backus Naur Form (EBNF):

Instruction:= {{Action}- [Object] [{Parameter}]}-

An instantiation of an instruction is a particular concretization of all the elements specifying the instruction (Figure 6):

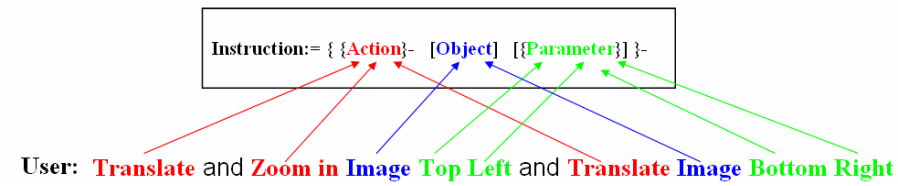

Figure 6. Instanciation of the general format of a navigational task

4.3 Working hypothesis

In this section we present the workspace in which we define the structure of the instructions specifying the navigational tasks and their instantiation on various combinations of interaction modalities. The workspace is defined by the following working hypothesis:

A. **Hypothesis 1: Action applied over 1 optional Object which has 2 Parameters:** is the particular format that will be considered for the instruction of a navigation task. The EBNF format of the instruction specified according to this hypothesis is:

Instruction:= Action [Object] {Parameter}

Where:

Action = user's command
[Object] = is optional and specifies that the action is applied over an image
{Parameter} = **{Param1, Param2}**
Param1 = specifies the X coordinate of the target image
Param2 = specifies the Y coordinate of the target image

Example: Figure 7 illustrate san instantiation of the above instruction, where the user is specifying a translation to the top left part of the image:

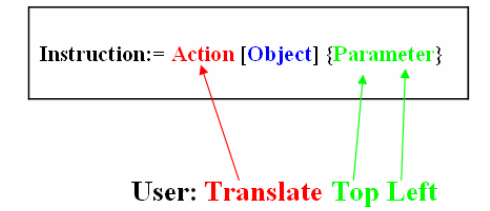

Fig. 7. Instantiation of the Action-Object command

B. Hypothesis 2: The Action-Object order: a software engineering paradigm

The action/object paradigm introduced in Section 4.2 has two instantiations depending on the order of specification of elements:

- **Action-object paradigm:** the user first selects an action and then selects the object. Once the user selected the action he can select the object over which he wants to apply it.
- **Object-action paradigm:** the user first selects the object and then selects the action. Once the user selected the object the system can then present a list of actions that can be applied to that object.

In order to better specify these paradigms in the context of navigational tasks we employ the LOTOS operators to introduce two binary operators and their corresponding semantics:

- >> **(sequential):** the operands on which applies have to be specified in the order of appearance. Once the first operand terminates to be specified the specification of the second one could be initiated
- |=| **(order-independent):** the operands on which applies do not have to be specified in a particular order. At the beginning both operands can be specified. However as soon as the first operand started to be specify, the entire specification has to be finished before enabling to specify the second operand.

After applying the introduced operators over the two paradigms the results are as follows:

- **Action** >> **Object:** the instruction has to specify first the action and then the object on which it applies
- **Action** |=| **Object:** the instruction doesn't have to specify the elements in a particular order.

Due to the fact that the specification of the Object is not mandatory according to the Hypothesis A (by default the Object is the image to manipulate) the Action-Object paradigm reduces to Action-Parameters paradigm. Thus, two types of possible instructions for navigation task can be defined:

- Sequential instruction: the instruction has to specify first the action and then the parameters.

Instruction: = Action >> Parameters

- Order-independent instruction: the instruction doesn't have to specify the action and the parameters in a particular order.

Instruction: = Action |=| Parameters

Where the parameters are in number of two:

X = the coordinate on the Ox axis of the image
Y = the coordinate on the Oy axis of the image

C. Hypothesis 3: Equivalent navigational instructions:

A sub-instruction groups one or more elements of an instruction in a structure expressed as a whole which is specified by employing the same monomodal or multimodal interaction. The element(s) of a sub-instruction are delimited by braces.

An instruction is composed of one or more sub-instructions. Figure 8 illustrates all possibilities of grouping the elements of a navigational task in sub-instructions taking into account the working hypothesis (A) and (B) defined above. The resulting instructions are equivalent. In the context of the current research the **Equivalence operator (⇔)** defined between two instructions specifies that their corresponding instances generate the same result when executed by the system.

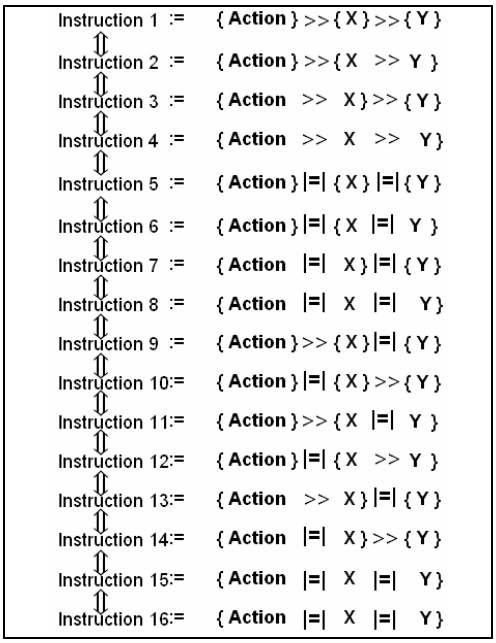

Fig. 8. Equivalent instructions of navigational tasks

5 UsiXML support for vocal interaction

5.1 Expanded model

Based on the shortcoming identified in Section 2.2 we expanded the UsiXML model in order to support the navigational task in vocal user interfaces. The following objects were introduced and defined:

- ***Grammar*:** is an enumeration, in a structured and compact form, of a set of utterances (i.e., words and phrases) that constitute the acceptable user input for a given vocalInput. The grammar can be internal (i.e., it is specified within the UsiXML document) or external (i.e., it is specified in an external file referenced from the UsiXML document by the defaultContent attribute). The attributes are:
  - ✓ *mainPart*: is the first part of the grammar that will be treated by the system.
- ***Part*:** contains other part elements or available input items. The attributes are:
  - ✓ *structure:* 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 are uttered in an order-independent manner).
  - ✓ multiplicity: indicate 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.
- ***Item*:** enables to specify a grammar input or to reference another part element. The grammar input is specified by the defaultContent attribute. The same attribute is used to specify the referenced part as a string containing the "#"symbol followed by the name of the part element. The attributes are:

5.2 Results

According to the expanded UsiXML model we present hereafter (Figure 9) the UsiXML specification for a vocal user interface that enables users to utter navigational task over large scale images. As specified by the *structure* attribute of the main part of the grammar, the elements composing the instruction have to be uttered sequentially in order to be recognized by the system.

```
<cuiModel id="CUI1" name="Concrete_Model">
  <vocalGroup id="VG1" name="vocalGroup">
    <vocalForm id="VF1" name="vocalForm">
      <vocalPrompt defaultContent="Please select the navigation task"/>
      <vocalInput currentValue="§var">
        <grammar version="1.0" language="English-UK" mode="voice" visibility="form" mainPart="instruction">
          <part id="instruction" structure="sequential" visibility="public" multiplicity="1-">
            <item defaultContent="#action"/>
            <item defaultContent="#XCoordinate"/>
            <item defaultContent="#YCoordinate"/>
          </part>
          <part id="action" structure="choice" visibility="public">
            <item defaultContent="translate"/>
            <item defaultContent="zoom"/>
          </part>
          <part id="XCoordinate" structure="choice" visibility="public">
            <item defaultContent="top"/>
            <item defaultContent="center"/>
            <item defaultContent="bottom"/>
          </part>
          <part id="YCoordinate" structure="choice" visibility="public">
            <item defaultContent="left"/>
            <item defaultContent="center"/>
            <item defaultContent="right"/>
          </part>
        </grammar>
      </vocalInput>
      <vocalFeedback defaultContent="You choice is §var"/>
    </vocalForm>
  </vocalGroup>
</cuiModel>
```

Figure 9. Expanded UsiXML specification for navigational tasks in VUIs

The expanded UsiXML specification for the description of vocal user interfaces for navigational tasks offers a set of advantages in terms of:

- ***Flexibility*:** all the composing elements of an instruction can be specified sequentially or asynchronously or a sequence of two elements can be combined asynchronously with the third one or an asynchronous combination of two elements can be combined sequentially with the third one. Table 1 illustrates the UsiXML specification of the main part of a navigational task grammar for all possible structures of elements composing an instruction. The predefined order of utterance is the **Action** followed by the **X Coordinate** and the **Y Coordinate**. Other structures can be specified by changing the order of appearance of the elements within the instruction (e.g., Y Coordinate, Action, X Coordinate). The notations in the table are as follows:

  A = Action
  X = X Coordinate
  Y = Y Coordinate
  >> = sequence of two items/parts
  |=| = order-independent between two items/parts

- ***Reusability*:** as a navigational instruction is split into atomic elements (i.e., action, XCoordinate, YCordinate) who's values are specified in particular part elements of a grammar, these values can be reused in other grammars independently of the type of sequence between them (i.e., sequential, asynchronous, or a combination of them).

Structure of the instruction	UsiXML specification
A >> X >> Y	<part id="instruction" structure="sequential" visibility="public" multiplicity="1-"> <item defaultContent="#action"/> <item defaultContent="#XCoordinate"/> <item defaultContent="#YCoordinate"/> </part>
A |=| X |=| Y	<part id="instruction" structure="asynchronous" visibility="public" multiplicity="1-"> <item defaultContent="#action"/> <item defaultContent="#XCoordinate"/> <item defaultContent="#YCoordinate"/> </part>
A >> (X |=| Y)	<part id="instruction" structure="sequential" visibility="public" multiplicity="1-"> <item defaultContent="action"/> <part id="sub-instruction" structure="asynchronous" visibility="public" multiplicity="1-"> <item defaultContent="#XCoordinate"/> <item defaultContent="#YCoordinate"/> </part> </part>
A |=| (X >> Y)	<part id="instruction" structure="asynchronous" visibility="public" multiplicity="1-"> <item defaultContent="#action"/> <part id="sub-instruction" structure="sequential" visibility="public" multiplicity="1-"> <item defaultContent="#XCoordinate"/> <item defaultContent="#YCoordinate"/> </part> </part>
(A |=| X) >> Y	<part id="instruction" structure="sequential" visibility="public" multiplicity="1-"> <part id="sub-instruction" structure="asynchronous" visibility="public" multiplicity="1-"> <item defaultContent="#action"/> <item defaultContent="#XCoordinate"/> </part> <item defaultContent="#YCoordinate"/> </part>
(A >> X) |=| Y	<part id="instruction" structure="asynchronous" visibility="public" multiplicity="1-"> <part id="sub-instruction" structure="sequential" visibility="public" multiplicity="1-"> <item defaultContent="#action"/> <item defaultContent="#XCoordinate"/> </part> <item defaultContent="#YCoordinate"/> </part>

Table 1. UsiXML specification for all possible types of navigational instruction structure

6 Conclusions and future work

As the taxonomy introduced in this paper suggests there are a large number of choices to make when designing or choosing the available navigational tasks and their concretization in a modality or another. Improved design based on controlled usability studies could improve the speed, error rates and subjective satisfaction. As the guidelines are limited and very few have been validated, the abstraction of the results of these studies into a knowledge base of ergonomic rules applied automatically by a tool over interface specifications seems to us a step forward for the improvement of the design process of multimodal interaction with large scale images. This is already confirmed by the successful results obtained with graphical user interfaces.

As future work we will address as well the tactile interaction which proved to be very effective especially in combination with the vocal modality. For this purpose we intend to expand the UsiXML language in order to respond to the requirements of

tactile interaction, as well. Furthermore, we will continue to improve the usability adviser tool and the knowledge base be adding new ergonomic rules related to the multimodal interaction.

Acknowledgments. 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) and of OpenInterface Project (www.openinterface.org), an open source platform for the rapid development of multimodal interactive systems.

References

[Bear90] Beard, D. B. Walker, J. Q. 1990. Navigational techniques to improve the display of large two-dimensional spaces. Behav. Inform. Techn. 9, 6, 451–466.

[Bede96] Bederson, B. B., Hollan, J. D., Perlin, K., Meyer, J., Bacon, D., Furnas, G. W. 1996. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. J. Vis. Lang. Comput. 7, 1, 3–31.

[Bede00] Bederson, B. B.,Meyer, J., Good, L. 2000. Jazz: An extensible zoomable user interface graphics toolkit in Java. In UIST'00, ACMSymposium on User Interface Software and Technology, CHI Lett. 2, 2, 171–180.

[Bolt80] Bolt, R.A., Put-that-there: Voice and gesture at the graphics interface, Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH'80 (Seattle, 1980), pp. 262–270.

[Card99] Card, S. K., Mackinlay, J. D., Shneiderman, B. 1999. *Readings in Information Visualization,* Morgan Kaufmann, San Francisco.

[Carr82] Carrol, J., Learning, using and designing command paradigms, Human Learning, 1, 1 (1982), pp.31-62.

[Calv03] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J., A Unifying Reference Framework for Multi-Target User Interfaces, Interacting with Computers, 15(3), June 2003, pp. 289–308.

[Chen00] Chen, C., Czerwinski, M. P. 2000. Special Issue on Empirical Evaluation of Information Visualizations, Internat. J. Hum.-Comput. Studies 53, 5.

[Comb99] Combs, T., Bederson, B. B. 1999. Does zooming improve image browsing? In Proceedings of the ACM Conference on Digital Libraries (DL '99, Berkeley, Calif., Aug. 11–14). N. Rowe and E. A. Fox, A., Eds. ACM Press, New York, N.Y., 130–137.

[Cruz92] Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V., and Hart, J.C. The CAVE: audio visual experience automatic virtual environment. Commun. ACM 35(6): 64–72, June 1992.

[Drui97] Druin, A., Stewart, J., Proft, D., Bederson, B., Hollan, J. D. 1997. KidPad: A design Collaboration between children, technologists, and educators. In Proceedings of the ACM Conference on Human Factors in Computing

Systems (CHI ’97, Atlanta, Ga, Mar. 22–27). S. Pemperton, Ed. ACM Press, New York, N.Y., 463–470.

[Fran94] Frank, A. U., Timpf, S. 1994. Multiple representations for cartographic objects in a multi-scale tree - an intelligent graphical zoom. Comput. Graph. 18, 6, 823–829.

[Furn95] Furnas, G. W., Bederson, B. B. 1995. Space-scale diagrams: Understanding multiscale interfaces. In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI ’95, Denver, Colo., May 7–11). I. R. Katz, R. Mach, L. Marks, M. B. Rosson, and J. Nielsen, Eds. ACM Press, New York, N.Y., 234–241. Proceedings of the Ninth ACM Conference on Hypertext (Hypertext ’98, Pittsburgh, Pa., June 20–24). ACM Press, New York, N.Y., 58–65.

[Frun00] Funkhouser, T., and Li, K. On the wall large displays. IEEE Computer Graphics & Applications, 20(4), July/August 2000.

[Grud01] Grudin, J. Partitioning Digital Worlds: Focal and peripheral awareness in multiple monitor use. Proceedings CHI 2001, pp. 458-465.

[Here00] Hereld, M., Judson, I., Stevens, R., Introduction to Building Projection-based Tiled Display Systems IEEE Computer Graphics & Applications Vol. 20, No. 4, July/August 2000.

[High98] Hightower, R. R., Ring, L. T., Helfman, J. I., Bederson, B. B., Hollan, J. D. 1998. Graphical multiscale Web histories: A study of PadPrints. In In Proceedings of the Ninth ACM Conference on Hypertext (Hypertext ’98, Pittsburgh, Pa., June 20–24). ACM Press, New York, N.Y., 58–65.

[Horn99] Hornbaek, K., Frokjaer, E. 1999. Do thematic maps improve information retrieval? In IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT ’99, Edingburgh, Scotland, Aug. 30–Sep. 3). M. A. Sasse and C. Johnson, Eds. IOS Press, Amsterdam, The Netherlands, 179–186.

[IBM92] Object-Oriented Interface Design, IBM Common User Access Guidelines, Que publisher, 1992.

[Mene00] Menendez R. G., and Bernard J.E. Flight simulation in synthetic environments. Proc. 19th Digital Avionics System Conference, pp. 2A5/1-6 vol.1, 2000.

[Ovia94a] Oviatt, S., L., P. R. Cohen, and M. Q. Wang. Toward interface design for human language technology: Modality and structure as determinants of linguistic complexity. Speech Communication, 1994, 15(3-4), 283-300.

[Ovia94b] Oviatt, S. L., E. Olsen. Integration themes in multimodal human-computer interaction. In Proc. of the ICSLP, Acoust. Soc. of Japan, Yokohama, 1994, 551-554.

[Ovia96] Oviatt S.,L., Multimodal interfaces for Dynamic Interactive Maps, Proceedings of the Computer Human Interaction 96, April 13-18, 1996.

[Perl93] Perlin, K., Fox, D., 1993. Pad: An alternative approach to the computer interface. In Proceedings of the 20th Annual ACM Conference on Computer Graphics (SIGGRAPH ’93, Anaheim, Calif., Aug. 2–6). J. T. Kajiya, Ed. ACM Press, New York, N.Y., 57–64.

[Plai94] Plaisant, C., Carr, D., Shneiderman, B. (April 1994) Image Browsers: Taxonomy, Guidelines, and Informal Specifications IEEE Software, vol.12, 2 (March 1995) 21-32.

[Scha96] Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S., Roseman, M. 1996. Navigating hierarchically clustered networks through fisheye and full-zoom methods. ACM Trans, Comput.-Hum. Interact. 3, 2, 162–188.

[Stan06] Stanciulescu, A., A Transformational Approach for Developing Multimodal Web User Interfaces, M.Sc. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, June 2006.

[USIX06] UsiXML Consortium, UsiXML, a General Purpose XML Compliant User Interface Description Language, UsiXML V1.6.4, 1 March 2006. Available at http://www.usixml.org/index.php?view=page&idpage=6