The CoGenIVE Concept Revisited: A Toolkit for Prototyping Multimodal Systems

Fredy Cuenca
Hasselt University – tUL – iMinds
Expertise Centre for Digital Media, Diepenbeek, Belgium
fredy.cuencalucero@uhasselt.be

ABSTRACT
Many specialized toolkits have been developed with the purpose of facilitating the creation of multimodal systems. They allow their users to specify certain tasks of their intended systems by means of a visual language instead of programming code. One of these toolkits, CoGenIVE, was developed in our research lab, and despite of its successful application in many internal projects, it gradually fell into disuse. The rethinking of CoGenIVE unveiled the existence of important gaps hindering a fuller understanding of these toolkits for rapid prototyping of multimodal systems. This paper aims to remedy some of these gaps with the proposal of: (a) the architecture of a toolkit for rapid prototyping of multimodal systems, (b) a scale for measuring the support for implementation provided by a toolkit, and (c) a classification of a representative set of existing toolkits.

Author Keywords
User interface toolkits; Visual languages;

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

A multimodal system is a computer system capable of collecting the information provided by the user through multiple input modes, combining these inputs in order to interpret the user's intent, and responding to the user through multiple output modes. Users can enter information into a multimodal system through speech, touch, handwriting, hand gestures or facial expressions; the system can respond the user with images, audio, synthesized voice or haptics. Its capability to decode user commands whose information is carried by several input signals is what distinguishes a multimodal system from a traditional WIMP system.

The implementation of a multimodal system is time-consuming and therefore expensive. Thus, several specialized toolkits have been developed with the purpose of facilitating the implementation of multimodal systems. These toolkits enable their users to specify certain functionality of the intended multimodal system by means of a visual language instead of programming code.

These so-called toolkits for rapid prototyping of multimodal systems started to be proposed since more than a decade ago, e.g. ICon [4], MEngine [1], OpenInterface [7], Squidy [9], HephaisTK [5]. During that period, we successfully developed a toolkit specifically designed to facilitate the development of multimodal virtual environments. Indeed, CoGenIVE [3] was internally used by many PhD students to create the interactive virtual worlds needed for their research projects. However, the lack of proper maintenance and its narrow scope made it fall into disuse, as did other toolkits. The analysis of this situation revealed that there are still gaps that hinder a fuller understanding of different toolkits for rapid prototyping multimodal systems. Some of these gaps include: (a) the lack of a broad description covering the structure and behavior of all the existing toolkits, which may be caused by the many differences among them, (b) the absence of a scale for measuring the support for implementation provided by a toolkit, which is pivotal to determine whether the proposal of a new toolkit advances the state-of-the-art or not, and (c) the shortage of cross-evaluations, which are always useful to classify apparently distinct elements, thus allowing their organized study.

The overarching goal of the research under discussion is the implementation of a visual language for modeling multimodal interaction, and of its supporting toolkit. This new language must rectify the shortcomings experienced with CoGenIVE's visual language: excessive notation, complex semantics, inability to model concurrency, and lack of underlying formalism. The first stage of this research is devoted to providing the theoretical background that fills the aforementioned gaps, this being the theme of the present paper.

TOOLKIT FOR RAPID PROTOTYPING OF MULTIMODAL SYSTEMS

A toolkit for rapid prototyping of multimodal systems consists of a framework and a graphical editor. The framework can be approached as a server offering some functionality to a client application. This application has to be developed without support from a toolkit. It has to implement the application-specific functionality of the intended multimodal system.

EICS'13, June 24–27, 2013, London, United Kingdom.

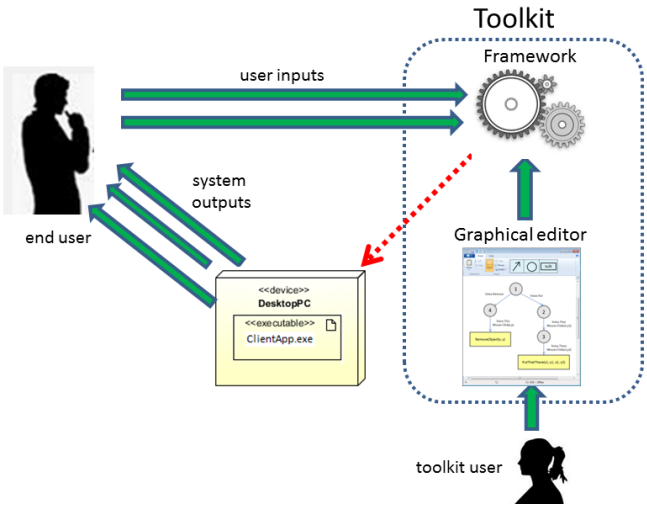

Figure 1. Architecture of a toolkit for rapid prototyping of multimodal systems.

Based on the study of several toolkits, the services usually provided by their frameworks include the recognition of user inputs, the identification of the user's intent, the identification of the system's state, and/or the dissociation of the system's response through multiple outputs. These services aim at extending the functionality of a client application so that it can handle multimodal input/output. The visual models depicted with the graphical editor of a toolkit are intended to specify how to intermix the services incorporated in the framework of a toolkit with the subroutines of a client application. These specifications are depicted by the user of a toolkit (who can also be the programmer of the client application), and are to be interpreted by its framework.

Once the framework and the client application are up and running, the end user is able to issue multimodal commands to an enhanced client application, which may not be originally capable of supporting multimodal interaction (Figure 1). Therefore, the client application is not the final system but a prototype: a partial implementation that needs to be supplemented by a toolkit so that its prospective users can have a means for experimenting, evaluating and/or redefining the intended system.

For illustrative purposes, consider a multimodal system whose users are allowed to utter a voice command 'zoom here' while touching a specific point on the screen to indicate the region to zoom in (left side of Figure 2). Prototyping such a system with the support of a toolkit entails the implementation of the GUI that the end user will interact with, and the subroutine(s) required to scale a specific region of this GUI. The particular behavior of the GUI and the specific scaling algorithms must be implemented (probably with a textual programming language) as part of a client application. This client application does not need to detect voice commands or touchscreen events. It neither has to verify the temporal co-occurrence of the speech input 'here' and the touch on the screen –required to zoom in on a region of the GUI. Both functionalities can be delegated to the framework through a visual model like the one shown on the right side of Figure 2.

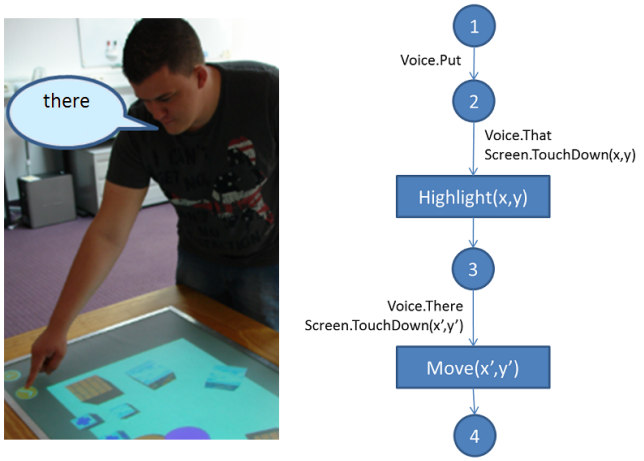

Figure 2. Left. End user interacting with a multimodal system. Right. Visual model used for specifying human-machine interaction.

This model specifies that the detection of the speech input 'zoom' followed by the simultaneous detection of the speech input 'here' and a touch on the screen will cause the execution of the subroutine *ZoomAt*, implemented in the client application. The use of visual models is due to the fact that their creation and maintenance is faster and easier than the edition of programming code.

A SCALE FOR MEASURING THE GAINS OFFERED BY A TOOLKIT

The more services a toolkit offers to its users, the less programming workload they will experience. Thus, the support for implementation provided by a toolkit depends on the functionalities that are pre-programmed in its framework. The study and testing of several toolkits revealed that the services commonly incorporated in their frameworks are:

Recognition of user inputs The framework incorporates software for recognizing a wide assortment of triggering events issued from several hardware devices.

Identification of the user's intent The framework can detect the occurrence of a multimodal command. Since the services offered by a system are requested through the issuing of multimodal commands, their detection reveal the user's intent. This process entails the recognition of patterns of events, i.e. sets of user events occurring in a particular order.

Identification of the system's state The framework can accurately determine the current state of the system after any arbitrary sequence of events. This permits prototyping systems that issue context-dependent responses.

Dissociation of the system's response The framework can concurrently launch several subroutines of the client application, and synchronize their execution.

The aforementioned functionalities are the ones used to describe the architecture of a multimodal system. Indeed, these match with the functions in charge of the recognizers, fusion

engine, dialog manager and fission component of a multimodal system respectively [2] [6].
We propose to express the support provided by a toolkit as a set of the aforementioned functionalitites. For instance, the support of CoGenIVE is {recognition of inputs, identification of the user's intent, identification of the system's state}, meaning that CoGenIVE releases its users from programming these functionalities. Then, the set containing all the possible combinations of the aforementioned functionalities is the scale of measurement we are proposing.

Heuristics to uncover framework's capabilities

The study and testing of several toolkits show that they all incorporate software for detecting the inputs coming from a myriad of hardware devices. This implies that their users do not have to implement algorithms for event recognition in their client applications. Rather, they can delegate the recognition of user inputs to the framework of a toolkit, as seen in Figure 1.
Regarding the identification of the user's intent, the detection of the system's state, and the dissociation of a response, these functionalities are not always pre-programmed in all the studied toolkits. Fortunately, it is possible to infer whether these services are provided by a toolkit or not, by examining its visual language.
The toolkits capable of detecting multimodal commands allow their users to specify composite events in their visual models. For instance, Figure 2 shows a composite event made up of two elements, *Voice.Here* and *Screen.TouchDown*, linked by a relation of simultaneity. The way a composite event is depicted, and the relations allowed among its constituent events, vary from toolkit to toolkit. In any case, the presence of composite events in a visual model is necessary to indicate the framework those sets of events whose perception (in a particular order) reveals that the user is requesting some service from the system.
The toolkits that can assume the responsibility of handling context-dependent human-machine dialogs always allow the depiction of the system's state in their visual models. This possibility enables users to specify multimodal systems that respond differently to the same command, depending on the state of the multimodal system. The model depicted in Figure 2 shows that both the perceptibility and response of the system to a multimodal command depends on its current state, e.g. once in state 2, the system will only respond to the designation of a target area, and will ignore other commands.
The dissociation of the system's response involves the concurrent activation of several synthesizers, and the coordination of their outputs. We found that toolkits whose visual models are variations of state diagrams (e.g. Figure 2) cannot be used to specify concurrency and synchronization. This limitation stems from the fact that state diagrams only experience one transition at a time, and thus only one subroutine can be executed in a given moment. However, toolkits capable of interpreting visual models based on Petri nets (e.g. PetShop [8]) allow for modeling both concurrent execution of subroutines and synchronization of events.

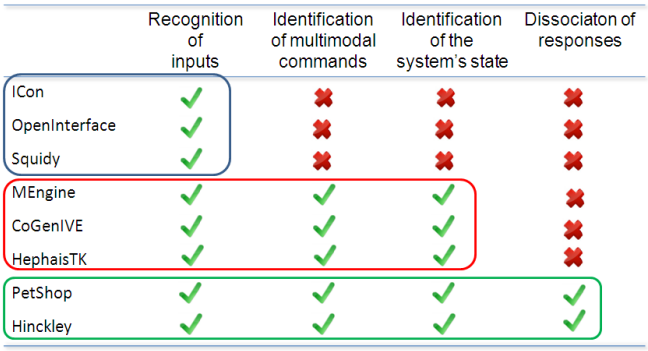

Figure 3. Checkmarks are used to indicate the services offered by different toolkits to their users. Under this criterion, toolkits can be clustered into three groups.

EVALUATION AND COMPARISON OF TOOLKITS

Cross-evaluations of toolkits for rapid prototyping of multimodal systems are few and difficult to conduct. This may be caused by the abundant differences among these toolkits: they offer different features, target different domains, operate with different programming paradigms and/or expect different skills from their users. However, despite of these numerous differences, some similarities can be observed when identifying the functionalities of their frameworks. The evaluation of several toolkits uncovered the existence of three classes of toolkits (Figure 3).
Toolkits in the first class are called *flow-based* toolkits. They incorporate software for event recognition, thus releasing their users from implementing this functionality. The visual models depicted with the editors of these toolkits resemble block diagrams. They specify the transformations experienced by the data flowing from the input devices to a client application. ICon [4], OpenInterface [7], and Squidy [9] are some examples of flow-based toolkits.
In addition, a second group of toolkits allow the specification of composite events and the depiction of the system state. Therefore, their users can delegate the detection of multimodal commands, and the execution of pertinent context-dependent responses to their frameworks. These toolkits are called *state-based* toolkits because of the resemblance of their visual models with state diagrams. Examples of this type of toolkits include MEngine [1], CoGenIVE [3] and HephaisTK [5].
Finally, the third class of toolkits also facilitates the dissociation of the system's response through multiple outputs. These toolkits use Petri nets as visual models. The tokens allow modeling concurrent activities, and the transition rule serves as a synchronization mechanism. These toolkits are called *token-based*, Petshop being [8] its most prominent example.
The clustering observed in Figure 3 suggests that the services a toolkit can offer to its users are restricted by the formalism (block diagrams, state diagrams or Petri nets) on which its visual language is based. This observation must be taken into account by those developing a toolkit for rapid prototyping, in order to avoid unwanted limitations of their intended toolkits after the implementation of its visual language.

CONCLUSIONS

In this paper, we propose novel theoretical tools intended for improving the understanding of toolkits for prototyping of multimodal systems, and allowing their precise evaluation and objective comparison. Our proposal includes: (a) the architecture of a toolkit for rapid prototyping of multimodal systems, (b) a scale for measuring the support for implementation provided by a toolkit, and (c) a classification of a set of toolkits based on the support they offer.

In the proposed architecture, a toolkit for rapid prototyping of multimodal systems can be approached as a server intended to extend a client application with multimodal features.

Regarding the measurement scale, we proposed to measure the support for implementation provided by a toolkit in terms of the services it provides. The services used to evaluate the capabilities of a toolkit, are the ones that describe the architecture of a multimodal system, namely, the recognition of user inputs, the identification of the user's intent, the identification of the system's state, and the dissociation of a response through multiple output modes [2] [6]. A reference scale where the gains provided by a toolkit can be measured on is necessary to determine whether the use of a particular toolkit can lead to a higher reduction of the programming workload, which is the primary goal of these toolkits. Without such scale, it is also hard to assess whether new or improved toolkits are advancing the state of the art or not. In our opinion, this is a challenge that the community has not yet solved.

By comparing several toolkits, we noticed that they can be clustered into three groups called flow-based, state-based and token-based toolkits. Toolkits within each group do not only offer the same services to their users, but also exhibit resemblance in their visual languages.

FUTURE WORK

A potentially successful way to continue this research may consist of (a) the creation of a simple textual programming language for specifying composite events, (b) the creation of a simple textual programming language for specifying the execution of concurrent and synchronized actions, and (c) the development of a graphical editor that allows representing human-machine dialogs as state diagrams whose arcs will be annotated with the utterances of the languages mentioned in (a) and (b). The utterances of (a) will specify the composite events that will cause a system transition, and the utterances of (b), the actions to be performed during this transition. A state diagram enhanced with the languages (a) and (b) will give us the following advantages: First, the possibility to specify composite events with a textual notation rather than with a graphical one, will lead us to more concise models, which are probably easy to read, maintain, and extend. Second, the inability of CoGenIVE's visual language for specifying the concurrent execution of many subroutines will be overcome by annotating the arcs of a state diagram with the utterances of the textual language described in (b). Third, readers in general will not have to accomplish the undesirable task of inferring the semantics of CoGenIVE's visual language from informal descriptions of running examples. The semantics of a state diagram can be concisely and formally described with a set of mathematical formulas.

ACKNOWLEDGMENTS

This research was funded by the BOF financing of Hasselt University. We want to thank our UHasselt colleagues of the HCI group for the discussions about and feedback on this research.

REFERENCES

1. Bourget, M. Designing and prototyping multimodal commands. In *Proc. of INTERACT'03* (2003).
2. Bui, T. *Multimodal Dialogue Management - State of the Art*. PhD thesis, University of Twente, 2008.
3. De Boeck, J., Vanacken, D., Raymaekers, C., and Coninx, K. High level modeling of multimodal interaction techniques using NiMMiT. *Journal of Virtual Reality and Broadcasting 4*, 2 (2007).
4. Dragicevic, P., and Fekete, J. Icon: Input device selection and interaction configuration. In *ACM UIST 2002* (2002).
5. Dumas, B., Lalanne, D., and Ingold, R. Description Languages for Multimodal Interaction: A Set of Guidelines and its Illustration with SMUIML. *Journal of Multimodal User Interfaces 3*, 3 (2010).
6. Dumas, B., Lalanne, D., and Oviatt, S. Multimodal interfaces: A survey of principles, models and frameworks. In *Human Machine Interaction*, Springer Verlag (2009).
7. Lawson, L., Al-Akkad, A., Vanderdonckt, J., and Macq, B. An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. In *Proc. of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems EICS 09* (2009).
8. Navarre, D., Palanque, P., Ladry, J., and Barboni, E. ICOs: A Model-Based User Interface Description Technique dedicated to Interactive Systems Addressing Usability, Reliability and Scalability. *ACM Transactions on Computer-Human Interaction 16*, 4 (2009).
9. Werner, K., Raedle, R., and Harald, R. Interactive Design of Multimodal User Interfaces - Reducing technical and visual complexity. *Journal on Multimodal User Interfaces 3*, 3 (2010).