Usability Framework for the Design and Evaluation of Multimodal Interaction

Jaeseung Chang and Marie-Luce Bourguet

Dept. of Computer Science

Queen Mary, University of London

Mile End Road, London, E1 4NS, U.K.

{ jac4;mlb }@dcs.qmul.ac.uk

ABSTRACT

The design and evaluation of multimodal interaction is difficult. For designers in industry, developing multimodal interaction systems is a big challenge. Although past researches have presented various methodologies, they have addressed only specific cases of multimodality and failed to generalise their methodologies to a range of applications. In this paper, we present a usability framework for the design and evaluation of multimodal interaction. First, in the early phase of multimodality design, elementary multimodal commands are elicited using traditional usability techniques. Second, based on the CARE (Complementarity, Assignment, Redundancy, and Equivalence) properties and the FSM (Finite State Machine) formalism, the original set of elementary commands is automatically expanded to form a more comprehensive set of multimodal commands. Third, this new set of multimodal commands is evaluated in two ways: user-testing and error-robustness evaluation. This framework acts as a structured and general methodology both for designing and evaluating multimodal interaction. We expect that it will help designers to produce more usable multimodal systems.

Categories and Subject Descriptors

D.2.2 [Software Engineering]: Design Tools and Techniques – *Evolutionary prototyping, User interfaces*. H.1.2 [Models and Principles]: User/Machine Systems – *Human factors*. H.5.2 [Information Interfaces and Presentation]: User Interfaces – *Evaluation/methodology, Prototyping, Theory and methods*

General Terms

Design, Experimentation, Human Factors

Keywords

Multimodality, interaction design, modelling, usability evaluation, finite state machine

1. INTRODUCTION

The design and evaluation of multimodal interaction is difficult. Although multimodal interaction is expected to provide more usable ways of communicating with various systems through natural modalities of interaction [9], it is still difficult to build usable multimodal systems. Designing multimodal interaction is especially challenging for the following reasons:

First, finding the optimal combination of multiple modes is not easy. The appropriateness of multimodal choice is influenced by many factors such as the nature of the tasks and the various user contexts. A certain modality choice, which is well suited in a particular environment, can be very inappropriate in other contexts [11]. Unfortunately, comprehensive multimodal corpora, which reflect diverse user circumstances and could help selecting appropriate combinations of modalities, are not yet available [11].

Second, multimodality is usually error-prone [1]. As multimodal systems tend to utilise natural channels of communication, they are widely exposed to human errors and ambiguity. It has been shown that redundant multimodal inputs can mutually disambiguate each other [10]. However, building error-robust multimodal interaction models, which enable mutual disambiguation, is not trivial.

In this paper, we present a usability framework for the design and evaluation of multimodal interaction, which provides designers with a practical methodology to overcome the above issues. In the next section, we describe how FSMs (Finite State Machines) can help us model various multimodal commands. From section 3, we introduce our usability framework, which is based on the FSM formalism.

2. FSM MODELLING

To date, a number of academic approaches have proposed reliable methodologies to design multimodal interaction, e.g. architectural approach [5], frame-based approach [8], statistical approach [11], and grammatical approach [13]. The common limitation of these approaches, however, is that they have addressed only specific cases of multimodality and failed to generalise their methodologies to a range of applications. On the other hand, the CARE (Complementarity, Assignment, Redundancy, and Equivalence) properties [6] offer a more general framework to describe and reason about various multimodal scenarios and modality combinations. Although CARE does not directly offer practical solutions to designers [2], the prominent generality of CARE should be noted.

At this point, we claim the usefulness of the FSM formalism as a modelling tool for multimodality design. The usefulness of

the concepts of FSM's 'state' and 'action' as an interaction modelling technique has already been reported in previous work such as [12] and [15]. As FSMs can represent multiple states and also identify multiple actions which provoke the change of states in a certain system with only a single or a few diagrams, we can easily describe and understand the structure of multimodal interactions using FSMs (Figure 1).

Figure 1. FSM for multimodal interaction

Recently it has been shown in [1] that FSMs can be used to model multimodal interactions and evaluate their error robustness. For example, with FSM, an interaction can be perceived as a 'transition' from a 'source state' to a 'target state'. The transition is initiated by an 'event', and the 'event' causes an 'action' to reach the 'target state'. The simple idea of FSM can be easily augmented to represent complex situations of multimodality design and evaluation.

The following merits of FSMs should be noted. First, the FSM formalism is an efficient tool to represent complex aspects of multimodality. If multiple modalities are available and they appear in either a sequential or parallel way, an FSM can describe all possible combinations in a single diagram [3]. Second, FSMs enable us to rapidly prototype multimodal systems. Because FSMs are simple but logically structured to describe multimodality, we can easily decide which elements of a multimodal command representation should be added or removed from a system. Third, the availability of multimodal combinations can be easily adjusted by designers, forcing users to activate specific combinations of modalities (Assignment property) [3]. Fourth, as already introduced in [1], [2], and [3], FSMs are very useful to avoid recognition errors. Fifth, the CARE properties [6] can be easily described with a set of FSMs [4]. This means that we can overcome the less practical aspects of CARE and successfully apply its prominent generality to our framework. Sixth, FSM descriptions can be easily transformed into software code, using for example Java and XML [14].

The FSM formalism itself does not have the ability to prioritise modalities, because FSMs does not contain any information about each modality's temporal constraints or probabilistic value. However, the concept of FSM has been extended to probabilistic FSMs, which offer an effective methodology to cope with the ambiguity of multimodality by including probabilistic weight values inside each FSM. The application of probabilistic FSMs to modelling ambiguous multimodality has been proved to be effective in [7].

3. USABILITY FRAMEWORK

In order to provide designers with a more practical framework to design and evaluate multimodal interaction, we present six different stages of multimodal command design: (1) Text-based Elementary Multimodal Commands, (2) FSM-based Elementary Commands, (3) Augmented Multimodal Commands, (4) User-tested Multimodal Commands, (5) Error-robust Multimodal

Commands, and (6) Modelled Multimodal Commands. These six stages of multimodal commands constitute the overall structure of the framework (Figure 2).

Figure 2. Conceptual framework

3.1 Elementary Multimodal Commands

The first stage consists in using a variety of traditional techniques such as task analysis, lo-fi prototyping, focus group discussion, and expert review, to elicit a number of elementary multimodal commands. The focus here is on finding and writing down possible multimodal activities for specific tasks, as we would normally do when designing unimodal interactions. Let us take the example of dialling a call to a pre-defined speed dial number '1' with a multimodal mobile phone equipped with keypads, voice recognition, and touch-screen. Using possible user-scenarios, the task of the designer consists in writing down on paper possible action sequences to perform this task. At this point, there is no particular grammar or format for recording action sequences, and we do not need to consider the various issues linked to multimodality design.

Figure 3a. FSM translator

Figure 3b. FSM translator

Multimodal modelling with the usability framework starts when the previously elicited text-based commands are input into our framework through the "FSM translator" (Figure 3a), which is a program developed in Java for this research. On the left of the FSM translator, we set the names of the system and the task at

hand, and enter each elementary command as previously specified. These elementary commands are automatically translated into FSMs, and appear on the right screen of the system. When entering elementary commands, we can specify the relationships between sequential actions. For example, if an elementary command shows that either speech or touch-screen can be used, we can set the relationship as 'OR'. Otherwise, if it says that both speech and touch-screen must be used, we can set it as 'AND'. These 'OR' and 'AND' relationships are also translated into FSMs, reflecting the Equivalent and Redundant properties respectively, as described in CARE. The newly generated FSMs represent exactly the elementary multimodal commands, which were elicited from the previous stage. Once we "Save", we can finalise the first and second stages of the framework and obtain FSM-based elementary multimodal commands, stored in a text file (Figure 3b).

3.2 Augmented Multimodal Commands

The purpose of augmentation is to obtain a comprehensive set of possible multimodal commands, on the basis of the elementary multimodal commands obtained from the previous stage. Because even a limited number of modalities can give rise to a very large number of possible combinations, the process of expanding the original set of commands must be automated. We developed the "FSM generator", a program written in Java, which automates this process and avoid inefficient and manual modelling processes.

Figure 4. FSM generator – augmented FSMs

First, we import the elementary commands stored by the FSM translator to our FSM generator. Second, the FSM generator extracts multimodal seed FSMs from the imported data. These seed FSMs represent unimodal simple actions that constitute the basic elements of the previous elementary commands. With the previous example, we have six seed FSMs from two elementary commands. Third, the FSM generator generates augmented multimodal commands with the seed FSMs. The augmenting process does not merely consist in multiplying every seed with each other, but is based on the CARE properties. For example, from only two elementary commands, we obtained a total of ninety augmented commands (Complementarity: twelve commands, Assignment: six commands, Redundancy: thirty six commands, Equivalence: thirty six commands). Fourth, these augmented multimodal commands are stored in a new text file (Figure 4).

We should note that there are potential scalability issues when generating redundancy and equivalence multimodal commands, even with a small number of seed commands. In order to avoid being hampered by needlessly generated commands, redundancy and equivalence rules are applied to only one each sequence of multimodal actions.

3.3 User-tested Multimodal Commands

Automatically augmented multimodal commands need to be evaluated with real users. The evaluation primarily aims at

collecting user preference and behaviour related to multimodal choices. In particular, we concentrate on the following points when conducting user-testing:

First, real users' modality preference should be obtained. In the example above, it is important to know which modality users frequently choose among keypad pressing, voice utterance, and touch-screen selection. Second, users' multimodal combination preferences should be recorded. For example, users' preference to use the combination of speech and keypad inputs, in a redundant or equivalent manner, should be recorded for each command. Third, users' multimodal synchronisation patterns should be captured. For example, users can choose to say "Call" first and then activate "Speed dial 1" on the touch-screen. Alternatively, they can choose to make their selection on the touch screen prior to talking. These various synchronisation patterns should be recorded.

The evaluation can be conducted using traditional user-testing techniques, for example Wizard of Oz experiments [4], as multimodal interaction designs usually aims at producing completely brand new devices that have not yet been developed (Figure 5).

Figure 5. Wizard of Oz User Testing

In addition, if users demonstrate multimodal behaviours and commands that do not appear in the augmented set of multimodal commands, they can be added. The result of user-testing should be recorded as another set of FSMs, which will be the criteria for evaluation. We can then test the augmented multimodal commands set by filtering out inappropriate representations and prioritising the rest. Finally, the output from user-testing will be user-tested multimodal commands that include probabilistic weight values for each modality choice.

3.4 Error-robust Multimodal Commands

The second evaluation aims at making user-tested multimodal commands more robust. Although user-testing in the previous stage generates user-oriented multimodal commands, they are still error-prone, because some multimodal tasks, which can happen together but cannot be detected by user testing, may invoke unexpected errors.

Indeed, some multimodal choices may not work properly, when associated with other task commands. For example, in a set of commands which contains two user-tested multimodal commands such as "Make a call to speed dial 1" and "Make a call to the number 123-4567", the presence of the same task element (the number 1) is likely to cause the failure of one of these two tasks. As most systems normally run many tasks, it is

necessary to check multiple multimodal commands as a set of commands under the same system.

Because it is not possible for designers to manually verify the whole set of multimodal commands, our usability framework relies on the FSM formalism to automate the verification process. In the example above, the usability framework can detect similar or same task elements (the number 1) and automatically flag a possible problem. The detected elements will be automatically removed or altered.

The alteration can be done in two ways: First, we can change the order of the multimodal actions to differentiate between the tasks. Second, we can change the modality choices to let the two tasks disambiguate each other. In the above example, we can force users to use only keypads to make a call to normal numbers, but lead them to use voice commands for a speed dial.

Robustness evaluation and its improvement can be automated within our usability framework, from which we will get error-robust multimodal commands.

3.5 Modelled Multimodal Commands

The error-robust multimodal commands can be regarded as the outcome from the first cycle of iterative designing. However, we do not ignore the possibility of faultiness, even though these commands reflect real users' multimodal behaviour with error-robustness. For example, if the size of error-robust multimodal commands is too big or too small, we can consider the re-iteration of previous stages to guarantee the higher quality of designing.

Once we are satisfied with the result, the final version of multimodal commands will be recorded as XML format for use in the development stage.

4. CONCLUSION

In this paper, we presented a usability framework to design and evaluate multimodal interactions. The predominant generality and extendibility of CARE properties and FSMs helped us build this framework. We can summarise several aspects of our usability framework as follows:

First, the framework is not limited to specific combinations of modalities, but can be applied to any multimodal scenario. Second, the operation of the framework does not require any prior knowledge about multimodality design. We can use traditional techniques such as lo-fi and hi-fi prototyping, focus group discussion, general usability testing, etc. Third, the concept of six different multimodal commands is easily understandable for designers. It means that our framework can also act as a rapid prototyping and iterative design tool for multimodal interaction. Fourth, because the framework uses FSMs to process multimodal commands and produce XML file as its result, the outcome can be easily translated into software code. We hope that the framework will raise the overall productivity of multimodality development.

So far, we have completed the implementation of the FSM translator and the FSM generator. The next step in our research is to build software systems for user-testing and robustness evaluation. We will then be able to verify the whole usability framework, using concrete scenarios and applications. The plan is to design a new multimodal mobile phone system, using the usability framework presented in this paper.

5. REFERENCES

- [1] Bourguet, M. L. How finite state machines can be used to build error free multimodal interaction systems. *The 17th British HCI Group Annual Conference*, 2 (2003) 81-84.
- [2] Bourguet, M. L. Designing and prototyping multimodal commands. *Human-Computer Interaction INTERACT' 03* (2003) 717-720.
- [3] Bourguet, M. L. Software design and development of multimodal interaction. *IFIP 18th World Computer Congress Topical Days* (2004) 409-414.
- [4] Bourguet, M. L. and Chang, J. Design and usability evaluation of multimodal interaction with finite state machines: a conceptual framework. To appear in *Journal on Multimodal User Interfaces 2008* (2008)
- [5] Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., and Smith, I. et al. QuickSet: multimodal interaction for simulation set-up and control. *Proceedings of the Fifth Conference on Applied Natural Language Processing* (1997) 20-24.
- [6] Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., and Young, R. M. Four easy pieces for assessing the usability of multimodal interaction: the CARE properties. *Proceedings of INTERACT' 95* (1995) 115-120.
- [7] Hudson, S. E. and Newell, G. L. Probabilistic state machines: dialog management for inputs with uncertainty. *Proceedings of the 5th Annual ACM Symposium on User Interface Software and Technology* (1992) 199-208.
- [8] Minsky, M. Minsky's frame system theory, *Theoretical Issues in Natural Language Processing. Proceedings of the 1975 Workshop on Theoretical Issues in Natural Language Processing*, 3 (1975) 104-116.
- [9] Nigay, L. and Coutaz, J. A design space for multimodal systems: concurrent processing and data fusion. *Proceedings of the SIGCHI conference on Human factors in computing systems* (1993) 172-178.
- [10] Oviatt, S. Mutual disambiguation of recognition errors in a multimodal architecture. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI is the Limit* (1999) 576-583.
- [11] Oviatt, S., Cohen, P. R., Wu, L., Vergo, J., Duncan, L., and Suhm, B. et al. Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions. *Human-Computer Interaction*, 15 (2000) 263-322.
- [12] Parnas, D. L. On the use of transition diagrams in the design of a user interface for an interactive computer system. *ACM/CSC-ER, Proceedings of the 1969 24th National Conference*, ACM Press (1969) 379-385.
- [13] Shimazu, H., Arita, S., and Takashima, Y. Multi-modal definite clause grammar. *Proceedings of 15th Conference on Computational Linguistics*, 2 (1994) 832-836.
- [14] Van Gurp, J. and Bosch, J. On the implementation of finite state machines. *Proceedings of the 3rd Annual IASTED International Conference on Software Engineering and Applications* (1999), 172-178.
- [15] Wasserman, A. Extending state transition diagrams for the specification of human-computer interaction. *IEEE Transactions on SW Engineering*, 11 (1985), 699-713.