Multi-Modal-Method: A Design Method for Building Multi-Modal Systems

Hideo Shimazu and Yosuke Takashima
Information Technology Research Laboratories.
NEC Corporation
4-1-1 Miyazaki, Miyamae, Kawasaki, 216
Japan
{shimazu, yosuke}@joke.cl.nec.co.jp

Abstract

This paper describes Multi-Modal-Method, a design method for building grammar-based multi modal systems. Multi-Modal-Method defines the procedure which interface designers may follow in developing multi-modal systems, and provides MM-DCG, a grammatical framework for multi-modal input interpretation. Multi-Modal-Method has been inductively defined through several experimental multi-modal interface system developments. A case study of a multi-modal drawing tool development along with Multi-Modal-Method is reported.

1 Introduction

This paper describes Multi-Modal-Method, a method for building grammar-based multi-modal systems.

The motivation behind this research is that defining such a method is necessary for building next generation interfaces. We believe multi-modal interface is one of the advanced interface beyond present graphic user interfaces (GUI) such as Windows and Macintosh. Although there has been significant research on multi modal systems (Allgayer 1989; Cohen 1989; Cohen 1991; Hayes 1987; Kobsa 1986; Wahlster 1989), these systems have been built as task-specific expert systems, focused on the application of the ideas. Although a number of methodologies have been formulated to build present GUIs by software scientists and consulting firms, they are not applicable to multi-modal system development, because the underlying principles are different between present GUI and multi-modal systems. Thus, we had to develop our own design methodology, optimized for multi-modal systems. We used the first grammatical framework for multi-modal systems, Multi-Modal Definite Clause Grammar (MM-DCG) (Shimazu 1994). Then, the Multi-Modal-Method was inductively defined based on several cases of grammar-based multi-modal system development.

2 Multi-modal Processing vs Event-driven Programming

Multi-modal interface is one of the advanced interface beyond present GUIs. Present GUIs are integration of *object oriented computing* and *event-driven programming*.

One of the most important innovations in computer programming during the past decade has been the development of "Object-oriented" computing. Viewing software components as if they are physical objects, characterizable via class/subclass relations based on simple features and/or how functions of the objects differ, is a powerful metaphor. The programmer can now imagine complex systems as built up of these simpler objects, much as a child builds a large structure out of simple building blocks or an architect arranges a functional, yet aesthetically appealing edifice from components such as wooden beams and metal girders. Thinking of the computer screen, the windows on that screen, and even the bits in those windows as simple objects composed together into a powerful editor has been an extremely compelling vision for interface designers. In fact, object-oriented programming has become a cornerstone of interface design, and the dominant metaphor in interface programming systems.

However, some recent systems have gone beyond objects for dealing with interface development. This is because, especially in window-based systems, some types of interface components do not fit well when viewed as "objects." Thinking of the mouse as a physical entity for the programmer to use makes perfect sense, but viewing a "mouse click" as an object seems less compelling. Similarly, other actions, such as sketching with a light pen, scanning a document, or speaking a sentence cannot be though of as physical entities, but rather must be viewed as "events" which occur on an object. Thus, for example, tools on Windows like Visual Basic have been leaning toward a programming methodology that allows not only

objects, but also event-based programming.

It is our contention that while event-based programming is a step in the right direction, it does not go far enough. In particular, we claim that it is the order of events in a sequence that is critical. This is especially true in a multi-modal interface where events may be coming from a set of different computational devices, each running separately. In such an interface, a mouse click, a spoken utterance, a drawing with a light pen, and some typed commands may have to be integrated into a single input. The ordering of the input events is clearly a critical factor in understanding the meaning of such inputs, and "parsing" such a string requires a more principled approach than simply expecting an application to handle the plethora of diverse inputs in all their forms.

The major purpose of this paper is to define a framework and design methodology for a computing model which can interpret a set of events, particularly in the area of multi-modal interface design. In the next section we describe this idea more fully and develop a simple example.

Fusion \ Use of Modalities	sequential	parallel
combined	ALTERNATE	SYNERGISTIC
independent	EXCLUSIVE	CONCURRENT

Figure 1: Nigay and Coutaz's multi-modal system categorization

3 Understanding Event Streams

Nigay and Coutaz (1993) divided multi-modal systems into four categories. They are defined by two independent features; *fusion* and *use of modality*. "Fusion" covers the possible combination of different types of data. the absence of fusion is called "independent" whereas the presence is referred to as "combined". "Use of modalities" expresses the temporal availability of multiple modalities. This dimension covers the absence or presence of parallelism at the user interface. "Parallel use" allows the user to employ multiple modalities simultaneously. "Sequential" forces the user to use the modalities one after another. In this paper. we deal with the "synergistic" category, the most difficult among the four categories.

A simple example shows how difficult it is to understand synergistic user expressions. Consider the example of a child who is using a multimedia encyclopedia system which provides a mix of speech recognition (and language processing) and a mouse. The child states "Can this, do this," pointing at a picture on the screen and clicking the mouse during the first "this" and then choosing an item from a menu during the second. The system must realize that the first point is, say, a picture of a particular animal and the second is the menu item "fly." Somewhere, the system must create an internal representation of this query that conforms to some data (or knowledge) base query language. In the object-oriented metaphor, some sort of central application object is in charge, and must send messages to the screen, the mouse, and the voice system asking for input upon activation. This system then synthesizes that information and produces a query such as "[QUERY: Func-of <Object Dinosaur-bitmap-7><menu-item FLY >]" which it is programmed to answer.

Note, however, that as the central system object is in charge, it must send messages (or otherwise contact) the various modalities of interaction to be aware of the possibility of input. This can be arbitrarily hard, especially as we consider that the number of modalities will keep growing as user interface technology design continues. Even for this simple example the same query can be asked many ways: the child could speak "can a pteranodon fly?"; could choose from the menu "query-function," point at the dinosaur, and then mouse "fly"; could type to a command line "query-function PTD Fly"; or any other combination of these capabilities. The central object coordinating all these modalities must send appropriate messages at appropriate times to each of the drivers of the various devices, and then must synthesize the answers that are received.

Unfortunately, the situation is made even more complex by the fact that the system cannot extract all inputs and combine them in some simple manner. The sequence in which the inputs are received can be critical - that is, the "event stream" must be analyzed as an ordered set of events which determine the interaction. If the child says "Is this (points at elephant) bigger than this (points at pteranodon)?" then the system must recognize in which order the points and the anaphoric references occur. Simply recognizing the query concerning the elephant and pteranodon is not enough; we must understand (and process) them in the correct order.

The computational metaphor we prefer is not that of objects, but rather that of processing the stream of events in a grammatical manner. Thus, instead of having a central object initiating some sort of message passing, we view each of the individual interaction techniques as producing reports concerning the events which occur and the timing of these events (e.g., the mouse in the above scenario will simply report "<Mouse-Click :Xpos 300 :Ypos 455 :start 2700 :end 2735>.")

Using the example, "can this do this", we describe how sophisticate synergistic inputs should be processed more precisely. Figure 2 shows four

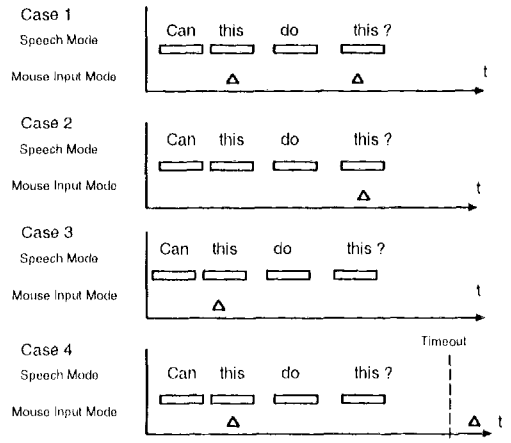

Figure 2: Four input timings for "can this do this"

timing cases of a user's input of the example. Each case should be processed in a different manner:

Case 1: There are two mouse inputs, and each of them matches corresponding speech input. Therefore, matching both inputs is easy.

Case 2: There is one mouse input which points at a specific animated object, "pteranodon". The input matches the first "this". The second "this", therefore, is interpreted as the last referred action.

Case 3: There is one mouse input which points at a specific action, "fly". The input matches the second "this". The first "this", therefore, is interpreted as the last referred animated object.

Case 4: There are two mouse inputs, one of which is input long after the first mouse input (for example, 1 minute after). In this case, the second mouse input is ignored because of *timeout* by the system. Only the first mouse input is interpreted. Therefore, case 4 is processed the same as case 2.

4 Multi-Modal-Method Design Process

The design process of the Multi-Modal-Method has seven steps.

Step 1: Task selection

A number of multi-modal interfaces have been developed. There are certainly several application fields in which multi-modal systems are applicable. They include: design and editing, presentation, information retrieval, and education.

Step 2: Mode and media selection

The number and type of modes and media should be determined. Generally, mode and media do not have a one-to-one correspondence. For example, although speech input and keyboard input use different media, they are treated as the same mode because they are used and interpreted identically.

Step 3: Corpus collection

The corpus of multi-modal expressions to the application is collected. This process is the same as that for natural language processing.

Step 4: Corpus analysis

The collected corpus is analyzed. Each expression in the corpus should be analyzed based on the following criteria.

Economy: Does the expression save a user's labor? Each expression is examined as to whether it can save a user's labor when transferring his/her intention to the application system. For example, in a picture drawing tool, if a user is allowed to point at a specific object while saying "delete", he/she can save labor, because he/she does not have to change the mouse position from the canvas to a menu item at the menu bar area, and again from the menu bar area to the canvas.

Plausibility: Each expression is examined as to whether it is likely to be used in a real application. As described below, writing grammars for multi-modal interfaces requires much more effort than for single modal interfaces. Only frequently used expressions should be selected carefully. The speech mode is better for selecting an item among a large number of candidates, such as choosing a city name among all cities in the USA. On the other hand, a menu interface is better for selecting one among a small number of candidates.

The set of the selected expressions becomes the seed for the specification of the designed multi-modal system.

Step 5: Specification Design

The difficulty level of the interface design should be determined based on the analysis of selected corpus expressions. There are five difficulty levels of multi-modal input expressions (Table 1):

Level 1: Single mode input: Even in a multi-modal system, users often want to express their intentions with single modal expressions. For example, pointing at an existing object, then selecting "delete" from the menu.

Level 2: All mode inputs express identical contents: Each mode input expresses an identical content. For example, pointing at an existing object, then selecting "delete" from the menu, while saying "delete the rectangle".

Level 3: A combination of incomplete mode inputs complement each other: Each mode input does not expresses the contents by itself.

Each mode input complements other mode inputs; thus they express a single content. For example, pointing at an existing object, while saying "delete".

Level 4: Each mode input is contradictory: The contents generated from independent mode inputs are contradictory one another. For example, saying "delete the *circle*", while pointing at a *rectangle* object which hides the specified *circle* object on the screen. Contradictions are often solved by context analysis.

Level 5: A combination of mode inputs still lacks something: The contents generated from the combination of the interpretations generated from individual mode inputs are insufficient. For example, saying "move it here", while pointing at a specific point. The point should be unified with "here", and an object specified by "it" should be interpreted as the last referred object. This type of interpretation requires of context analysis.

It becomes more difficult to interpret expressions as the level increases. Especially, since level's 4 and 5 require tight integration with context analysis, interface designers should consider whether the application users really need these levels or not.

Step 6: Architecture Design

Any multi-modal system can have a multi agent architecture because each mode processing is easily mapped to an independent agent. There are two extreme types of architecture which manage the agents. One is blackboard architecture where agents exchange information using a shared memory called a blackboard. The architecture fits multi-modal systems whose multi-modal expressions are sophisticated and integrated with context analyses. The other is subsumption architecture where each agent acts rather independently. Information exchange paths between agents are limited. The architecture fits multi-modal systems whose multi-modal expressions are simple and stereotyped. Many actual multi-modal system architectures are combinations of these extreme architectures.

Step 7: Grammar rule writing

Each selected multi-modal expression is defined by the corresponding grammar rule to interpret it. The grammatical framework for the multi-modal expression should have the following functionalities:

(1) Modes should be interpreted equally and independently. If each mode is treated in the same manner as that of a natural language mode, syntax and semantics of inputs of each mode are defined with grammar formulation. Thus, complex multi-modal expressions can be defined declaratively and more easily.

(2) Mode interpretations should be referred to one another. Inputs of each mode should be interpreted independently. However, the interpretation of such inputs should be referred to by other mode interpretations. There are ambiguities which are solved only by integrating partial interpretations of related modes. For example, if a user states "this rectangle", pointing at a different type of object overlapping the rectangle object, the ambiguity of the object pointing must be solved by comparing the two mode interpretations.

(3) Mode interpretation should handle temporal information. Temporal information of inputs, such as input arrival time and the interval between two inputs, is important in interpreting multi-modal inputs.

Multi-Modal DCG (MM-DCG) supports these functionalities. MM-DCG is a superset of DCG (Pereira 1980); everything possible in DCG is also possible in MM-DCG. MM-DCG has two major extensions:

1. MM-DCG can receive arbitrary numbers of input streams, while DCG can receive only one. A single grammar rule in MM-DCG can allow the coexistence of grammatical categories, thus allowing for their integration.
2. In MM-DCG, each individual piece of input data is required to attach the beginning time and the end time as its time stamp. Using the time stamp, MM-DCG automatically calculates the beginning time and the end time of any level of instantiated grammatical categories generated during parsing. The translator of MM-DCG to Prolog predicates generates code which perform this task. [1]

Figure 3 illustrates an application written in MM-DCG.

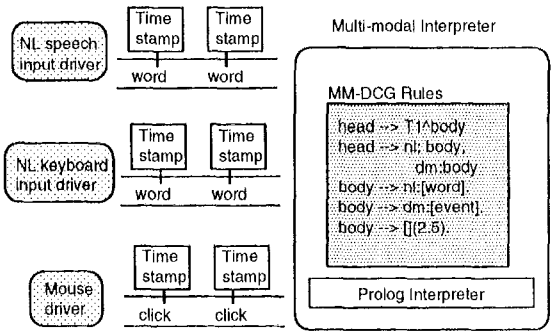

Figure 3: Multi-modal application written in MM-DCG

These processes form one cycle in the system evolution. Because of the increase in multi-modal expressions, the quality of the system improves as

[1] The details of MM-DCG are described in (Shimazu 1994)

Level	Characteristic	Example
1	single mode	pointing at an object, selecting "delete" from the menu
2	redundant	pointing at an object, selecting "delete" from the menu, while saying "delete the rectangle"
3	incomplete	pointing at an existing object, while saying "delete"
4	contradictory	saying "delete the *circle*", while pointing at a *rectangle* which covers the specified *circle*
5	lacking	saying "move it here", while pointing at a point.

Table 1: Five levels of multi-modal inputs

the cycle iterates. When the system reaches the mature stage, the system is released to end users.

5 Case Study

This section describes the design process of a multi-modal drawing tool along with the multi-modal-method. The following is the trace of the design process.

Step 1: Task selection Since there has been significant research on developing multi-modal drawing tool (Hiyoshi 1994; Nigay 1993; Vo 1993; Bellik 1993), the application field is promising.

Step 2: Mode and media selection In this experiment, we focused on only input modes. Input modes include speech, keyboard and mouse inputs. These input modes are synergistic. Output modes include pictures and text, but outputs are not synergistic.

Step 3: Corpus collection We collected about two hundred multi-modal expressions from potential users as instructions for the multi-modal drawing tool. The users had experience with using existing drawing tools.

Step 4: Corpus analysis The following are some of the results of the analysis of the collected corpus.

- Users want to use various mixed modes according to the situations they are dealing with.
- Users want to use abridged expressions, which causes integration of multi-modal interpretation and context analysis.
- Users want to handle existing objects as a set. For example, "Change the color of *all circles*.
- Users want to handle existing objects which are not shown on the display. For example, asking "how many rectangles are hidden out of the canvas?".
- Users want to use the mouse ambiguously. For example, saying "Delete this circle", while pointing at a point *away from but near* the *circle*. Such ambiguous pointing can be correctly interpreted only when multi-modal expression is allowed.

Step 5: Specification Design The analysis taught us that multi-modal drawing tools should support level's 4 and 5 (the most difficult levels) to meet ordinary users' requirements. The specifications were determined based on these requirements.

Step 6: Architecture Design Since the required specification is the most difficult synergy level, the architecture is blackboard architecture where each agent can exchange information in varying ways.

Step 7: Grammar rule writing After the analysis, about forty expressions were selected, and variations of each selected expression were also generated and added. Grammar rules were defined corresponding to each multi-modal expression. Figure 4 shows a part of the grammar rules written in MM-DCG. The rules define how to interpret an imperative sentence like "Delete this circle" with varieties of expressions. It allows the spoken utterance mode(speech stream), the type-in mode (keyboard stream), and the mouse pointing mode (mouse stream). Rules in the level 1 section define single modal expressions. In the level 2 section, whether different mode inputs express identical contents is examined. The combination of the verb_by_multimodal/1 clause and the second object/1 clause is an example of the level 3 expressions. In the level 4 section, select_right_meaning/3 enclosed inside curly brackets { and } is a Prolog predicate which determines the correct meaning using context analysis when different mode inputs generate contradictory meanings. Such a predicate is defined in a task-specific manner. In the level 5 section, find_appropriate_termt/2 enclosed inside curly brackets { and } is a Prolog predicate which finds an appropriate term using context analysis when the combination of generated meaning of all modes still lacks information. Such a predicate is also defined in a task-specific manner. A trivial heuristic rule example is "to use the most recently appeared term".

Grammar writers should understand that the number of grammar rules for multi-modal interfaces becomes much larger than for any single modal interfaces. If there are three modes; M_1, M_2, and M_3, and the numbers of grammar rules

```
% stream definition
active_stream(speech, mouse, keyboard)

% Level 1
imperative(meaning(Action, Object)) --> verb(Action), object(Object).
verb(Action) --> verb_by_menu(Action).
verb(Action) --> verb_by_multimodal(Action).
verb_by_menu(Action) --> menu(Menu_item, Action).
verb_by_multimodal(delete) --> (speech or keyboard):[delete].
menu(menu_item_3, delete).
object(Obj) --> noun_phrase(Obj).
object(Obj) --> pointing(Obj).
noun_phrase(Obj) --> article, noun(Noun), {attribute(type, Noun, Obj)}.
article --> (speech or keyboard): [this].
noun(circle) --> (speech or keyboard):[circle].
pointing(Obj) --> mouse:[button(left, loc(X, Y))],{attribute(location, (X, Y), Obj)}.

% Level 2
verb(Action1) --> verb_by_menu(Action1), verb_by_multimodal(Action2), {Action1 == Action2}.
object(Obj1) --> noun_phrase(Obj1), pointing(Obj2), {Obj1 == Obj2}.

% Level 3
% Level 4
verb(Action) --> verb_by_menu(Action1), verb_by_multimodal(Action2),{select_right_meaning(Action1, Action2, Action)}.
object(Obj) --> noun_phrase(Obj1), pointing(Obj2), {select_right_meaning(Obj1, Obj2, Obj)}.

% Level 5
imperative(meaning(Action, Object)) --> verb(Action), {find_appropriate_term(object, Object)}.
imperative(meaning(Action, Object)) --> object(Object),{find_appropriate_term(action, Action)}.
```

Figure 4: Grammar Description of "Delete this circle" Using MM-DCG

for each mode are; G_1, G_2, and G_3. Then, the total number of the multi-modal grammar rules is the sum of the grammar rules of any combination of these three modes. Thus, the total number, G_{total} is:

$$G_{total} = \sum_{M_1,M_2,M_3 \supseteq S} G_S \qquad (1)$$

The above steps took about two man month for the first cycle. The most time consuming steps were step 4 and step 7.

6 Conclusion

This paper described the multi-modal-method, a design method for building grammar-based multi-modal systems. The multi-modal-method defines the procedures which interface designers may follow in developing grammar-based multi-modal systems, and provides MM-DCG, a grammatical framework for multi-modal input interpretation. The multi-modal-method has been inductively defined through several experimental multi-modal interface system developments. A development process of a multi-modal drawing tool along with the multi-modal-method was also introduced.

Acknowledgements

We would like to thank Prof. James Hendler for his advice during this research and in writing this paper.

References

Allgayer, J., Jansen-Winkeln, R., reddig, C., andReithing N., "Bidirectional Use of Knowledge in the Multi-Modal NL Access System XTRA", Proc of IJCAI-89, 1989.

Bellik, Y, and Teil, D., "A Multimodal dialogue controller for multimodal user interface management system application: A multimodal window manager", Adjunct Proceedings of INTERCHI-93.

Cohen, P.R., Dalrymple, M., Moran, D.B., Pereira, F.C.N., et al., "Synergistic Use of Direct Manipulation and Natural Language", Proc. of CHI-88, 1989.

Cohen, P.R., "The Role of Natural Language in a Multimodal Interface", 1991 International Symposium on Next Generation Human Interface, 1991.

Hayes, P.J., "Steps towards Integrating natural Language and Graphical Interaction for Knowledge-based Systems", Advances in Artificial Intelligence - II, Elsevier Science Publishers, 1987.

Hiyoshi, M, and Shimazu, H., "Drawing Pictures with Natural Language and Direct Manipulation" Proc. of COLING-94, 1994.

Kobsa, A., Allgayer, J., Reddig, C., Reithing, Nl, Schumauks, D., Harbusch, K., and Wahlster, W, "Combining Deictic Gestures and Natural Language for Referent Identification", Proc. of COLING-86, 1986.

Nigay, L, and Coutaz, J., "A Design Space for Multimodal Systems: Concurrent Processing and Data Fusion" Proc. of INTERCHI-93, 1993.

Pereira, F., and Warren, D.H.D., p"Definite Clause Grammars for Language Analysis - A survey of the Formalism and a Comparison with Augmented Transition Networks", Artificial Intelligence, vol. 13, no. 3, 1980.

Shimazu, H., Arita, S., and Takashima, Y., "Multi-Modal Definite Clause Grammar" Proc. of COLING-94, 1994.

Vo, M.T., and Waibel, A., "A multi-modal human-computer interface: Combination of Gesture and Speech Recognition", Adjunct Proceedings of INTERCHI-93

Wahlster, W., "User and discourse models for multimodal communication", in J.W. Sullivan and S.W. Tyler, editors, Intelligent User Interfaces, chapter 3, ACM Press Frontiers Series, Addison Wesley Publishing, 1989.