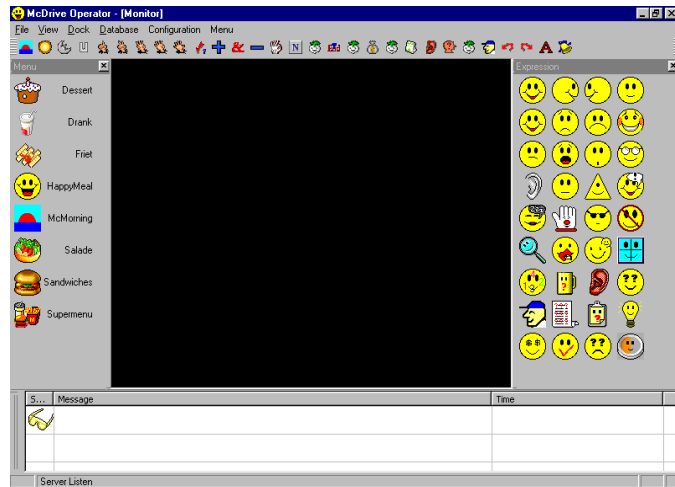
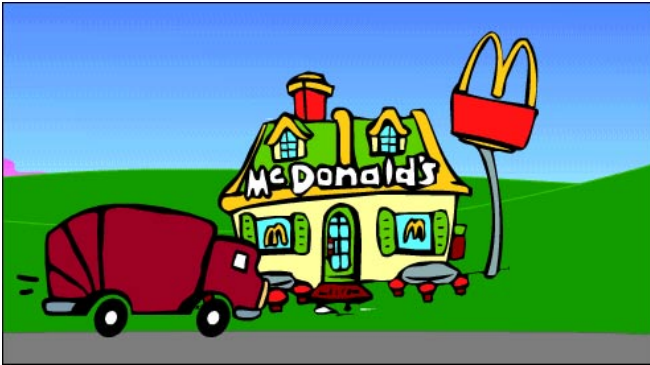


Multimodal McDrive System



Master's thesis of Yun Pang

Delft University of Technology
Faculty of Information Technology and Systems
August 2002



Graduation Committee:

Drs.dr. L.J.M. Rothkrantz
Prof.dr.ir. E.J.H. Kerckhoffs
Ir. A. Wojdel
Prof. dr. H. Koppelaar (chairman)

Pang, Yun

Master's thesis, August 2002
"Multimodal McDrive System"

Delft University of Technology, The Netherlands
Faculty of Information Technology and Systems
Knowledge Based Systems Group

Keywords: Artificial intelligence, multimodal, agent, smiley

Abstracts

This thesis describes my graduation project **Multimodal McDrive System (MMS)** at the Knowledge Based Systems (KBS) group of the Delft university of Technology.

Project

This project is aimed to build a multimodal intelligent system to replace the human operator of McDrive. McDonald's is the largest and best-known global foodservice retailer in the whole world. It has more than 30,000 restaurants in 121 countries. There are 1.5 million people working at McDonald's. McDrive is one of the branches of McDonalds fast food chains. As we all know the human power is very expensive. In order to reduce the human costs we want to build a system to replace the operators of McDrive who take orders from the customers. This automated system uses speech recognition technology to communicate with the customers. This issue has been discussed in their works of Farhaad Mohamed-Hoesein and Ramya Ramaswamy. To make this system works better we want to make it multi-modal. This system can talk to the customer, understand what the customer says and give the right response. To improve the customer understanding this system gives not only audio feedback but also visual feedback such as text and graphics. The graphics may be a picture or a flash movie. Our goal is to reduce the manpower cost, but in the meanwhile we must maintain the service level. It means this system needs to think and behave like a human being. We, human being, can communicate with each other using different ways. Often no talk is needed, just a gesture or a facial expression is enough and we can understand each other perfectly. We want this system also having such emotion expressions. We build a human wizard to express these feelings.

The final system will be automatic but first we need to build some prototypes for testing. There will be three prototypes that have different control modes: manual, semi-automatic and automatic. At this moment a manual prototype is built. In this prototype the customer can only use text as input because of the time and money limitation. The operator needs to produce the response manually through a special keyboard. This keyboard has three parts: menu, commandos and expressions. The keyboard is also needed in the semi-automatic prototypes. Only this keyboard is more intelligent. It is minimal - only the necessary buttons will be dynamically generated with the current condition and environments. The automatic prototype can generate the response automatically, but the operator can take the control at any time. The final goal is an automatic system, which can replace the human operator completely.

Acknowledgements

First I would like to thank my supervisor Leon Rothkrantz. He directives and gives me great advices during my graduation project. I also would like to thank Mr. P. Stathis for his help of installing software.

Also thanks the McDrive for the material.

Summary

The first chapter Introduction gives a general introduction of the McDrive project. First the reason why the Multimodal McDrive system is needed is explained, followed by what kind of feedback the Multimodal McDrive system will generate; and then there is a short description over the three prototypes. At last it gives an overview of the thesis structure.

The rest of the chapters is divided into two parts: the theory part and the design and implementation part. In the theory part we describes what kind of research work we have done. It contains five chapters in which we give an introduction of the multimodal system, smiley, Microsoft Agent, Poser and the parsing techniques. The last chapter of the theory part does a test of OZ to check the feasibility of the MMS system. The design and implementation part describes the design and implementation of the MMS manual prototype. At the end of this part the project is evaluated and some conclusions and

recommendations are presented. The appendixes function as supplementary of the two parts. Some tables and examples are taken in the appendixes.

We have done the case study in the second chapter. It starts with the overview of the McDrive order process and the goal of the MMS system. Section 2.2 analyses the available dialogues and finds the model of a general dialogue. And then BNF is explained in section 2.3. The following two sections analyse the basic operator and customer prompt sets and put them in BNF form. The last section states what needs to be done and what requirements the system needs to satisfy.

The subject of the third chapter is the multimodal system. Section 3.1 and 3.2 briefly outlines the multimodal system and the multimodal interface. The next two sections discuss what kind of multimodal inputs and outputs are needed in the case of McDrive. This chapter is closed with some examples about the multimodal data presentation.

Smiley and wizard are used to express the operator's feelings. Section 4.1 shows why smileys are popular in the nonverbal communication. In section 4.2 the functions of smileys are discussed and also the use of smileys in the dialogues. How can we design and implement a wizard is the subject of the last sections. Section 4.3 discusses the choices of the wizard. The next section gives a brief overview over the theory behind the wizard, Microsoft Agent. This chapter closes with some introduction over the necessary tools for the wizard building, Microsoft Agent Editor and Poser.

The fifth chapter discusses the parsing techniques and the brainstorming process is described. The first choice is a Prolog parser, the reason why we choose this parser and the communication between the interface and the Prolog parser are described in this section. And then we choose an XML Parser and give an introduction of its theory. At last an example is given to show how to use a VB parser to convert the input into a XML string.

The last chapter of the theory part is a test of OZ. A dialogue is simulated to see whether our design is feasible. A timeline of a dialogue is drawn with the text and graphical feedbacks.

The seventh chapter is the first chapter of the design and implementation part. It describes an outline of the structure of the system model. Section 7.1 discusses how the operator and customer interfaces communicate with each other. Section 7.2 and section 7.3 shortly address the operator and customer interfaces.

In chapter 8 we focus on the data management of MMS system. Section 8.1 discusses the data classification and the location it will be stored. In section 8.2 we do some research on the coding possibilities. Section 8.3 covers the overview of the tables of database McDrive and the functions of their managers. The database Customer is briefly mentioned in the last section.

The ninth chapter describes the keyboard building process from the first text keyboard until the end product. Section 9.1 describes the design of the text keyboard and the possible options. The graphical keyboard and its characters are discussed in section 9.2. The last section gives an overview of the keyboard structure and takes menu board as an example to explain how the menu board is generated.

The tenth chapter begins with an overview of the operator interface layout. The individual windows are introduced with their generation and functions. Section 10.2 focuses on the functions of the server.

The eleventh chapter studies the customer interface. The customer interface is composed of four individual windows. Section 11.1 gives a brief description over the customer interface layout and the individual windows. The function of the client is shortly addressed.

In the twelfth chapter the manual prototype is tested and a simulated dialogue process is outlined. At last it gives a short evaluation.

The thirteenth chapter discusses the results of our study and draw some conclusions. It gives a brief overview of what we have done and what the system feedback is. The chapter is closed with some recommendations and suggests for the future work.

Appendix I gives an overview of the rules of menu items. A short smiley dictionary is included in the appendix II. Appendix III covers some examples over how the smileys are used in the dialogues. The wizard actions are outlined in appendix IV. Appendix V gives a description of the fields of the database tables.

Table of contents

Abstracts	i
Table of contents	v
List of Tables	vii
List of Figures.....	ix
Chapter 1 Introduction.....	1
Part I. Theory	3
Chapter 2 Case Study	5
2.1 Problem Definition.....	5
2.2 Dialogues Analysis.....	6
2.3 Backus Naurus Form	9
2.4 BNF of McDrive Menus	9
2.5 BNF of System Prompts	11
2.6 BNF of Customer Prompts.....	11
2.7 Requirements Specification.....	13
Chapter 3 Multimodal System.....	15
3.1 Multimodal System.....	15
3.2 Multimodal Interface.....	15
3.3 Unimodal / Multimodal Input in MMS.....	18
3.4 Multimodal Output in MMS	19
3.5 Multimodal Data Presentation.....	20
Chapter 4 Smiley and Wizard.....	23
4.1 Smiley.....	23
4.2 The Functions of Smileys	24
4.3 Wizard Design.....	25
4.4 Microsoft Agent.....	27
4.5 Build the Wizard.....	29
Chapter 5 Parsing	35
5.1 A Prolog Parser	35
5.2 XML Parser.....	37
5.3 VB Parser.....	39
Chapter 6 Test of OZ.....	41
6.1 Simulated Dialogues	41
Part II. Design and Implementation	47
Chapter 7 Structure of Model.....	49
7.1 Communication between Operator and Customer Interface	49
7.2 Operator Interface	50
7.3 Customer Interface.....	50
Chapter 8 Data Management.....	53
8.1 Where to Store?.....	53
8.2 Data Coding.....	54

8.3 Database Management	56
8.4 Database Customer.....	66
Chapter 9 The Design of Operator Keyboard.....	67
9.1 First Design -- Text keyboard.....	67
9.2 Second Design --- Graphical Menu Keyboard.....	69
9.3 Last Design: A Manual Keyboard for the Operator Interface.....	72
Chapter 10 Operator Interface.....	77
10.1 Interface Layout.....	77
10.2 Server Functions.....	80
10.3 The Order Process.....	82
Chapter 11 Customer Interface.....	85
11.1 Interface Layout.....	85
11.2 Client Functions.....	87
Chapter 12 Simulation and Test.....	89
12.1 A Simulated Dialogue.....	89
12.2 Analysis.....	93
Chapter 13 Conclusion and Recommendations.....	95
Reference	99
Appendix.....	101
Appendix I Menu rules.....	103
Appendix II Smiley faces.....	107
Appendix III Smiley face used in dialogues.....	111
Appendix IV Wizard actions.....	117
Appendix V Database	119

List of Tables

Table 2.1: A dialogue between the first operator and the customer	6
Table 2.2: Formatted dialogue between the first operator and the customer	8
Table 2.3: The simulated dialogue between the system and the customer	8
Table 2.4: Questions in category “ask for additional information”	8
Table 2.5: Questions in category “ask for additional order”	9
Table 3.6: Applying the five combination schemas to the four combination aspects	17
Table 4.1: Some Smiley examples	24
Table 4.2: Dialogue example with smiley	25
Table 4.3: Some wizard actions	30
Table 4.4: The possible mouth animation images	31
Table 9.1: Keyboard set of operator interface	70
Table 9.2: Keyboard set of operator interface	70

List of Figures

Figure 2.1: The McDrive outside ordering process.....	5
Figure 2.2: Flowchart of dialogue between human operator and customer	7
Figure 2.3: Category Drink and its Subcategories and the Menu items	10
Figure 3.1: A model for the identification of basic processes in human-computer Interaction [MIAMI, 1995]	16
Figure 3.2: Architecture of multi-modal user interfaces. Adapted by Roope Raisamo from [Maybury and Wahlster, 1998]	16
Figure 3.3: Schema for modality combination with the same characteristics	17
Figure 3.4: A simulation of the McDrive environment.....	19
Figure 3.5: Some frames of flash movie “Ask drink size”	20
Figure 3.6: Flash movie “End”	20
Figure 4.1: Some characters.....	26
Figure 4.2: Microsoft Agent Character Editor	29
Figure 4.3: Build a new character with Character Editor.....	30
Figure 4.4: The transition between the animations	32
Figure 4.5: Some expressions made by Poser.....	33
Figure 4.6: Mimic can be used to generate speech animation for Poser	34
Figure 5.1: The parsing process of VB + Amzi! Prolog	36
Figure 5.2: Communication between DM and prolog program	37
Figure 5.3: How an input string can be converted to an order	38
Figure 6.1: The timeline of a dialogue example.....	45
Figure 7.1: The communication between the Server and the Client	49
Figure 7.2: The communication between the Server and the Client	49
Figure 7.3: Structure of Operator Interface	50
Figure 7.4: Structure of Customer Interface.....	51
Figure 8.1: The data locations.....	54
Figure 8.2: EAN 13-code	55
Figure 8.3: McDrive 7-code.....	55
Figure 8.4: Database and its managers.....	56
Figure 8.5: Database Manager	57
Figure 8.6: Add a new table with the database manager.....	57
Figure 8.7: Menu Manager with the 1st record.....	59
Figure 8.8: Add a new record to Menu table using Menu Manager.....	60
Figure 8.9: Delete a menu record.....	61
Figure 8.10: Search a record	62

Figure 8.11: Add a new category	62
Figure 8.12: Relation between field Attr_type (Menu) and field Type (Attribute).....	63
Figure 8.13: The alias manger.....	64
Figure 8.14: Add a new alias.....	64
Figure 8.15: Information overflow of Expression.....	65
Figure 9.1: The first design of a text keyboard.	67
Figure 9.2: The second design of a text keyboard.....	68
Figure 9.3: The layered approach of a menu item.....	68
Figure 9.4: The attribute options of the menu item MacShake.....	68
Figure 9.5: The Graphical customer keyboard.....	69
Figure 9.6: The graphical operator keyboard.....	70
Figure 9.7: Meta layer of keyboard.....	71
Figure 9.8: Alias: 1) Move mouse to button “drie”; 2) Click button “Drie”.....	72
Figure 9.9: The menu keyboard	73
Figure 9.10: The process of generating Menu Keyboard.....	74
Figure 9.11: 1) Click drop- down button “Coca-cola”	74
Figure 10.1: Operator Interface screen snapshot.....	77
Figure 10.2: Interaction between the windows of operator interface and McDrive database	78
Figure 10.3: The order window.....	79
Figure 10.4: The status part.....	80
Figure 10.5: Configuration of screen capture and expression board.....	80
Figure 10.6: A flow chart of a part of the ordering process	83
Figure 11.1: Customer Interface screen capture.....	85
Figure 11.2: Graphic Board of Customer Interface.....	86

Chapter 1 Introduction

McDonald's is the largest and best-known global foodservice retailer with more than 30,000 restaurants in 121 countries. McDrive is one of the branches of McDonalds' fast food chains. McDonald's opened the first restaurant in The Netherlands on August 21, 1971 in Zaandam. Today Holland has 193 McDonald's restaurants. The first McDonald's restaurant with McDrive was opened in 1987 in Huis ter Heide. This event caused a lot of people to visit the restaurant. From that moment McDrive has made success in the Netherlands. There are 1.5 million people working at McDonald's in the whole world and in the Netherlands 15000 people. Because the employee salaries represent one of the largest component of the enterprise cost the attention is focused on the manpower costs.

The reason that we take McDrive as our test is that the customer service of McDrive is relative simple. The popularity of McDrive thanks a lot to the combination of the service speed and the easy order manner. The order process of McDrive is as follows, the customer drives to the ordering booth, the operator takes order from the customer and tell him to collect his order at the collect window, then the customer drives to the second window, get his order, pays for it, and drives away. Our goal is to replace the operator at the ordering booth with a multimodal intelligent system – **Multimodal McDrive System (MMS)**. To accomplish this objective while maintaining a high level service to the customers, MMS is desired to think and behaves just like a human operator.

The MMS is multimodal so that the customers can have different ways to communicate with the system. The ideal is the customer speaks to MMS, but due to the limitation of the speech recognition technology the speech input of the customer is out of consideration at this period, the user has to input his request / order with text in this work. In order to make the customer understands MMS better, MMS gives not only audio feedback but also visual feedback like text and graphics. The picture of the menu item or a flash movie will be shown. And the text version of the MMS response will also appear on the screen. We want to make the customers feel more comfort, make them feel that they are not just facing a cold screen. The MMS needs to behave more human-like. We, -as human being-, can speak, gesture, smile or etc. Often no talk is needed, just a gesture or an expression is enough, and we can understand each other perfectly. So we build a human wizard who has all kinds of expressions such as smile, angry, confused and etc. This wizard can also do some actions such as nod, turn left, move left and etc. When the MMS welcomes the customer, the wizard appears and smiles; when MMS give the positive confirmation, the wizard nods; when MMS cannot understand the customer, the wizards becomes impatient. In one word this wizard can express right feelings at the right moment just like a human being. The wizard occupies only a part of the screen, so the wizard is relative small. To make the customer can see the expression better, we use graphical smiling faces as auxiliary. The smiling face gives the same expression as the wizard.

In the real life there will be only a customer screen that shows the customers the MMS response. Everything is automatically. But in the testing phase there are two interfaces needed- the operator interface and the customer interface. We build first prototypes that have an operator- and a customer-interface and make them communicate with each other to simulate the customer and operator dialogues. The operator- and customer-interface will be installed separately on two computers that are not in the same room. But these two computers are connected through the Internet. A human operator will sit behind the operator interface and controls it. The customer is behind the customer interface and gives the customer inputs. There will be three operator control modes: manual, semi-automated and automatic. For each mode we build a prototype. In the manual prototype the operator has to produce the response manually through a special graphical keyboard. This special keyboard is generated dynamically from the McDrive database at each time MMS runs. It contains three parts: Menu, Commandos and Expressions. The operator generates a code sequence by clicking the buttons. This code sequence will be transferred to the customer-interface and interpreted by the customer-interface. Picture, movie, text and expression will be shown and the response will be spoken. Then a

text box will appear so that the customer can type his input. In the semi-automated prototype a more intelligent keyboard will be used. Only the current necessary buttons will be dynamically generated, that is the right button at the right moment. The communication between the two interfaces is the same as those in the manual prototype. For the automatic mode the operator interface can do everything automatically. But we want the operator to be able to take control over at any time. On the basis of these prototypes a multimodal automatic system will be developed to replace the human operator at the ordering booth.

In the next chapters we'll define the problem and analyze the dialogues between operators and customers to get the minimal operator- and customer prompt set. Then we elaborate the designs of the customer- and operator- interface and make implementations of the manual prototype. This thesis is divided into two parts: the research part and the design and implementation part.

Part I. Theory

Chapter 2 Case Study

This chapter contains the description of the McDrive customer ordering process. The goal of analysis is to point out what the MMS system is aimed to replace. The requirements over the MMS system are outlined in different areas.

2.1 Problem Definition

McDrive is a branch of McDonalds' chains. It is different from the common McDonalds' restaurants. Instead of lying in the busy streets McDrive lies mostly on the highways outside the city. The customers visit McDrive in their cars. The customer can get out of car and go inside to order the food at the bar. Or he can drives to the ordering booth, orders from his car and drives away after collecting the menus. Most customers prefer the outside ordering because they are in a hurry, they don't have so much time to sit down and eat. The whole process is as follows: The customer drives to the ordering booth where a human operator greets him, takes the order, gives price on the display screen and sends the order to the kitchen. The customer will drive to the pickup-window to pay and collect the menus.

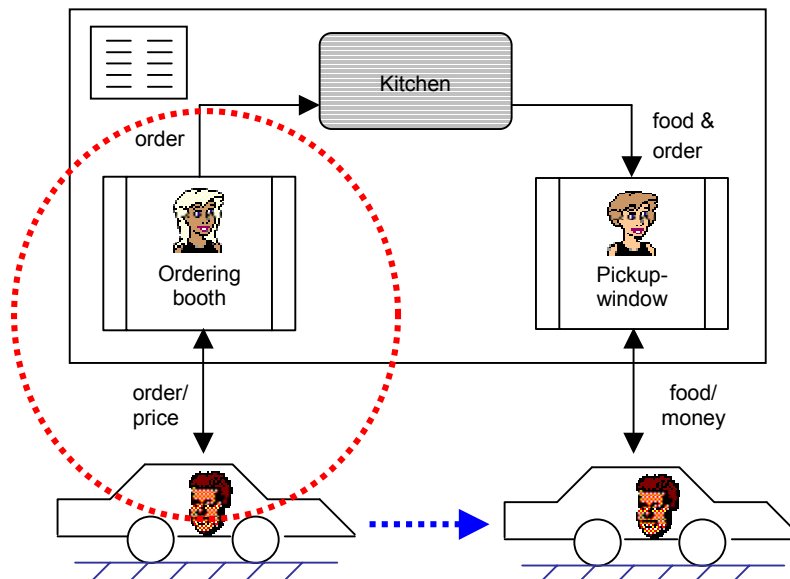


Figure 2.1: The McDrive outside ordering process

The above figure is a Data Flow Diagram (DFD) of the outside ordering process of McDrive. It gives us a picture of the data stream between the customer, the two operators and the kitchen. It is the operator of ordering booth who we want to replace with our system MMS, we call this operator as the first operator. In all McDrives round the world the menu items are quite similar except that the languages used are different. And the McDrive menu is relatively simple because the customers often have to eat in their cars. Many customers have a usual menu that they always order. They know what they want. For the others there is a big billboard with menu items available. So the ordering process is easier to be simulated. To replace the first operator we need to analyze first what his task is. Our approach is corpus based, we observed the human operator and recorded many dialogues between the customer and the operator. The investigation shows the following actions of the human operator:

- Greets the customer and takes the order.
- Gives the customer price and shows it on the display screen.
- Tells the customer where he can collect the menu.
- Sends the order to the kitchen.

Our aim is to build a system that can do all these things. When the system cannot accomplish his task, for example in the case of misunderstanding, it will give up and tell the customer to order inside.

2.2 Dialogues Analysis

The main task of the first operator is to take orders from customers, which is done through talking. Our prototypes are built to simulate communications between operator and customer. We need to analyze the operator and customer dialogues to track the regular patterns. Here we use the 200 dialogues between customers and operators that were used for the Talk McDrive project. These dialogues were recorded at different time during a day, with different operators. So they can represent the common dialogue patterns. The language used is Dutch. We try to translate them in English. But because the system is written in Dutch so most of time we just leave them in Dutch. There are totally 485 distinct words used from which we want to find a minimal set of customer and operator prompts. The human-computer dialogues is not for hundred percent the same as the human-human dialogues. For example, the system needs to confirm the order that the customer has given because the speech recognition rate is not high enough. The human-human conversation is natural and it can use any word. But here we want to define a basic system and customer prompts sets so that the number of basic words / sentences can be limited. That means if a word occurred just only once in the dialogues it would not be included in the minimal prompt set unless it is a menu item. First let we take a look at a dialogue example:

Table 2.1: A dialogue between the first operator and the customer

Operator	Good morning. Can I have your order please?
Customer	Hello. One Mac Deluxe with a big milkshake.
Operator	Which flavor do you want?
Customer	Vanilla and with fries sauce.
Operator	4 euros 75 and please drive to the first window.

From this example we can see this McDrive operator begun the dialogue with a welcome message and asked the customer to give his order. The customer gave his order and the operator asked additional information over the order. At last the operator gave the customer price and let hem to collect his order from the first window.

The dialogues can be very short or very long depending on the customer's order. From all the 200 dialogues recorded we can see that the shortest dialogue recorded had only 3 utterances: the operator greeted the customer; the customer gave an order and then the operator gave immediately the price of the order. The longest dialogue consisted of 11 turns where 11 of the utterances were just junks. The analysis of all these dialogues shows us that there are indeed some patrons. The average dialogue consists of 5 adjacent pairs. The McDrive operator starts almost always with a greeting and ends with the price and collecting window. The operator can ask questions depending on the completeness of an order. The operator can either ask whether the order is complete when there is no additional information needs to be achieved or give directly price. We sort the first operator and customer prompts into the following 6 pair categories:

Greet - Greet

The operator greets the customer. For example "Good morning, this is the test prototype for the McDrive system." The customer greets the operator. For example "Hello."

Ask for order – Give order

The operator asks the customer to order. For example "Would you please give your order after the beep?" The customer gives order. For example "I'd like to have two BigMac Menus."

Ask for additional information – Give additional information

Menus like “BigMac Menu”, “coffee”, “salad” and so on have apart attributes such as “drink”, “fries sauce”, “milk”, “sugar”, “dressing” and etc. If the customer doesn’t give information about these attributes as he orders then the operator has to ask questions about them. For example when a customer orders a “salad natural” the operator needs to ask, “What kind of dressing do you want?” The customer can answer, “Thousands islands, please.”

Ask for additional order – Give additional order

Some questions of this category seems just to be as the same as questions of category “ask for additional information”. But for the purpose of customer prompt parsing we put hem into a separate category. For example when operator asks: “Would you like nuggets sauce with your nuggets?” and the customer can reply: “Yes.” We put questions that ask whether the order is complete also in this category.

Give price and end - End

At last the operator gives price of the order and tells the customer where to collect his menus. For example operator says: “It is in total 10 guilders 45, please drive to the first window to collect your menus.” The customer says: “Thanks.”

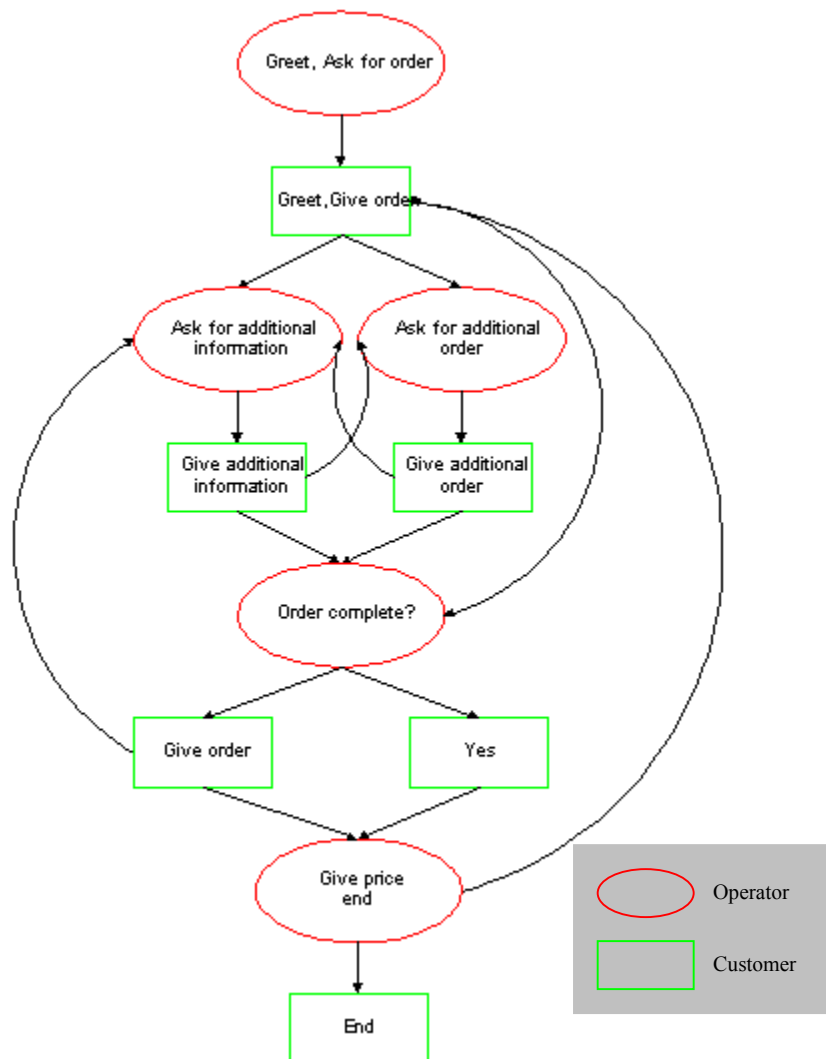


Figure 2.2: Flowchart of dialogue between human operator and customer

Give information – ask for information

Sometimes the customer wants to know more information about the menus. For example the customer asks: “What kind of sauces do you have?” the answer of operator would be “mustard, barbecue, zoetzuur, cherry.”

The mentioned dialogue example (see table 2.1) can be formatted as

Table 2.2: Formatted dialogue between the first operator and the customer

Operator	Greet, Ask for order
Customer	Greet, Give order
Operator	Ask for additional information
Customer	Give additional information
Operator	Give price and end

A common dialogue consists of utterances of the first 5 pair categories. Verification is little used by a human operator. In our prototypes the system has to ask confirmation when the customer gives order. The system acts like a human operator but it is still not a real human operator. The system can interpret the customer prompt wrong. By adding verification we can find the mistake in time. It cost more time but double security can be achieved. So a new category – **verification** is added in the prototypes. And if the system cannot understand the customer it will ask the customer to repeat himself. This applies when the system cannot hear what the customer says. After three times failure the system will give up. So for the system-human dialogues this kind prompts needs also to be added. The simulated system-customer dialogue of the example should like this:

Table 2.3: The simulated dialogue between the system and the customer

System	Good morning, this is the prototype of the McDrive system. Can I have your order please?
Customer	Hello. One Mac Deluxe with a big milkshake.
System	You want one Mac Deluxe with a big milkshake, is it correct?
Customer	Yes.
System	Which flavor of milkshake do you want?
Customer	Vanilla and with fries sauce.
System	You want vanilla and fries sauce, is it correct?
Customer	Yes.
System	4 euros 75 and please drive to the first window.

We summarized the questions belonging to the category “ask for additional information” and “ask for additional order” in the tables of below.

Table 2.4: Questions in category “ask for additional information”

<i>Menu</i>	<i>Question</i>
Supermenu	
Fries / Nuggets/	Which kind of fries sauce do you want?
Milkshake	Which flavor do you want?
Salad	Which salad dressing do you want?
Chocomel	Do you want your chocomel cold or hot?

Nugget	Which nugget sauce do you want? How many chicken do you want, six, nine or twenty?
Donut	Do you want Cappuccino or cinnamon?
Milkshake	What kind of flavor do you want?
Croissant	What kind of jam do you want?
McFlurry	What kind of flavor do you want?
Sundae Ice	What kind of sauces do you want?

Table 2.5: Questions in category “ask for additional order”

<i>Menu</i>	<i>Question</i>
Supermenu/ Complete breakfast	You get a drink with your menu. What do you want to drink?
Drink	Do you want a small, medium or large drink?
Coffee/Cappuccino / Espresso	Do you want sugar? Do you want milk?
Order	Anything else?
Fries	Do you want fries sauce?

The system has different ways to ask all these questions. For example the system would like to know the flavor of milkshake, the system can also ask “strawberry, banana, chocolate or vanilla?” Because these questions are preformatted, we can bundle the same questions together. When the system needs to ask a question the system choose randomly one from its bundle.

2.3 Backus Naurus Form

We use Backus Naurus Form (BNF) to specify the syntax of the dialogue prompt sets. The following notions will be used:

Terminal symbols – these are symbols that we actually use when writing or speaking in the language.

Non-terminal symbols – each non-terminal symbol names a particular class of phrase.

Start symbol – this is the non-terminal symbol that names the principal class of phrases.

Production rule – these specify how phrases are composed from terminal symbols and other phrases.

In this thesis we use capital words in bold style for non-terminal words to distinguish them from terminal symbols. Because the prototypes are developed in Dutch we write also the BNF rules in Dutch.

2.4 BNF of McDrive Menus

The McDrive menu has the following categories: SuperMenu, Sandwiches, Salads, Fries, Drink, Happy Meal, McMorning and Desserts. In each category there are some menu items. For example in Supermenu category there are BigMac Menu, McChicken Menu, McNuggetsKip Menu and etc. For some menu items there is additional information necessary except the quantity of the menu item ordered. Some menu items such as the SuperMenu items are combinations of several menu items. For example when you order a BigMac Menu, you get a BigMac sandwich, a medium soft drink which you can choose yourself and a medium size portion fries. Of course if you want a big size, that is also ok, only you need to pay more. The operator needs to know from you what kind of drink you want. And he also needs to find out what kind of fries sauce you want for your fries. We call all these extra information **attributes**. There are two attributes for a Fries menu item: Size and Fries Sauce. Attribute Size has three possible values: small, medium and large. Medium is the default value. The customer can choose from three kinds of Fries Sauce: fries sauce, ketchup and mayonnaise. The menu items in

the same category may have different attributes. We take Drink menu items as example: Coco-cola has three size: small, medium and large. The attribute of coco-cola is Size. The other soft drinks have the same attribute. You can order a large coca-cola, but you cannot say that you want a large coffee. Similarly it is very common to have coffee with milk and sugar, but it is strange if he wants milk and sugar in his cola. We can say these two drink items don't have the same attributes. The chocomel have also a different attribute- temperature. You can order it cold or warm. There are four flavors milkshake. The other drinks such as milk don't have any attributes. If your order milk the operator will not ask you how would you like your milk. We can also see that the menu items don't have the same number of attributes. Cola have only one attribute, coffee has two and tea doesn't have any attributes. So in the Drink category there are five kinds of menu items with different attributes. We put hem into subcategories so that all the menu items in the same subcategory have the same attributes. The Drink category is divided into four subcategories.

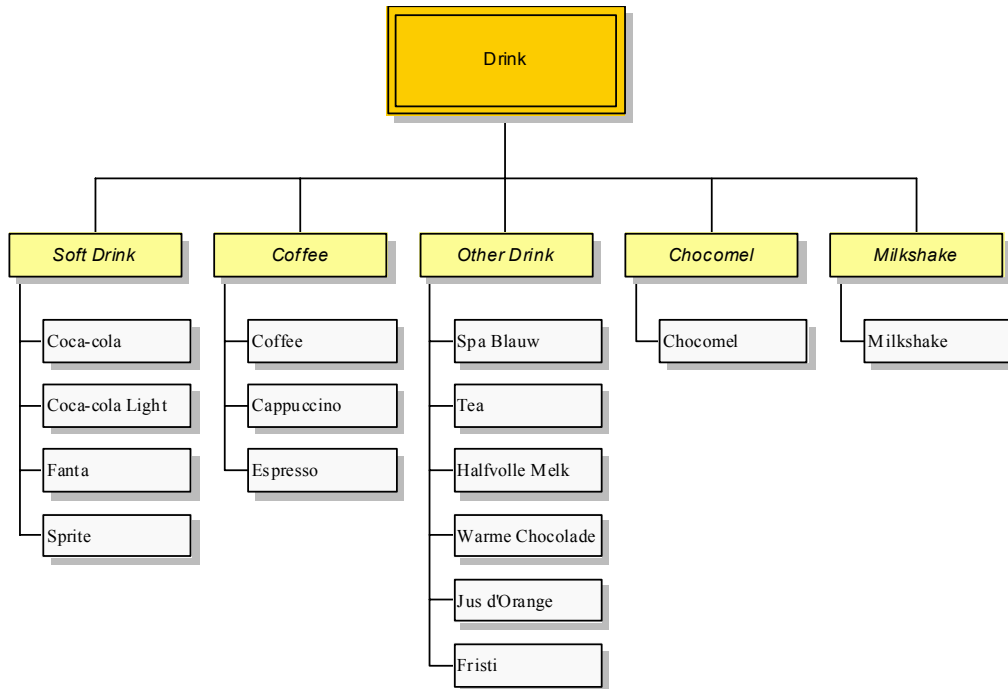


Figure 2.3: Category Drink and its Subcategories and the Menu items

We put each subcategory in a separate rule. And every attribute has its own rule. Here is the example of the BNF of the drink menu. The BNF of other menus see appendix.

<i>Category Drink</i>	
Frisdrank	::= Coca-cola Coca-cola Light Fanta Sprite
Otherdrank	::= Spa blauw Jus d'Orange Halfvolle melk Warme chocolade Thee Fristi
Chocomel	::= Chocomel
Chocomel_t	::= koud warm
Koffie	::= Koffie Cappuccino Espresso
Koffie_melk	::= melk
Koffie_suiker	::= suiker
Shake	::= Milkshake MacShake

Shake_smaak	::=	aardbei banana chocolade vanilla
--------------------	-----	--

2.5 BNF of System Prompts

OPEN	::=	GREETING OPENING
GREETING	::=	Goede morgen Goede middag Goede avond
OPENING	::=	Dit is het test systeem voor Auto-Mac. Geeft uw bestelling na de biep. De menu items kunt u in de lijst vinden.
CONFIRM	::=	CON1 BESTELLING CON2
CON1	::=	U heeft
CON2	::=	besteld. Klopt het?
ASK_EXTRA_ORDER	::=	EXTRA_ORDER_1 EXTRA_ORDER_2 EXTRA_ORDER_3 EXTRA_ORDER_4 EXTRA_ORDER_5
EXTRA_ORDER_1	::=	U krijgt drank bij uw menu. Wat wilt u drinken?
EXTRA_ORDER_2	::=	Wilt u frietsaus hebben bij uw friet?
EXTRA_ORDER_3	::=	Wilt u nuggetsaus hebben bij uw nuggets?
EXTRA_ORDER_4	::=	Wilt u een klein, middel of groot beker hebben?
EXTRA_ORDER_5	::=	Zo is uw bestelling compleet?
MISVERSTAND	::=	MIS_1 MIS_2 MIS_3
MIS_1	::=	Ik heb u niet begrepen.
MIS_2	::=	Ik heb u niet verstaan.
MIS_3	::=	Uw bestelling is te lang, Ik heb u niet begrepen.
MIS_4	::=	Helaas, het is mij niet gelukt u te begrijpen. U kunt het beter bij de balie proberen. Tot ziens.
ASK_EXTRA_INFO	::=	EXTRA_INFO_1 EXTRA_INFO_2 EXTRA_INFO_3 EXTRA_INFO_4
EXTRA_INFO_1	::=	Welke frietsaus wilt u hebben?
EXTRA_INFO_2	::=	Welke smaak van milkshake wilt u hebben?
EXTRA_INFO_3	::=	Welke nuggetsaus wilt u hebben?
EXTRA_INFO_4	::=	Wat voor salade dressing wilt u hebben?
END	::=	DANK REKEN PRIJS GULDEN HAAL
DANK	::=	Bedankt voor uw bestelling.
REKEN	::=	Het is
EURO	::=	euros.
HAAL	::=	Dan mag u doorrijden naar het tweede loket.

2.6 BNF of Customer Prompts

OPEN	::=	GREETING GIVE_ORDER
-------------	-----	----------------------------

GREETING	::=	Goede morgen Goede middag Goede avond
GIVE_ORDER	::=	SUBJ ORDER ORDER PLEASE
SUBJ	::=	Ik wil graag
PLEASE	::=	Alstublieft
GIVE_CONFIRM	::=	AGREE DENY DENY GIVE_ORDER
AGREE	::=	Ja.
DENY	::=	Nee Nee, bedankt.
GIVE_EXTRA_INFO	::=	WAT NUMBER WAT NUMBER PIECES WAT GIVE_EXTRA_INFO PLEASE GIVE_EXTRA_INFO DANK
WAT	::=	FRITE_SAUS SHAKE_FLAVOUR NUGGET_SAUS SALADE DRESSING
GIVE_EXTRA_ORDER	::=	WAT_DRANK F_SAUS_ORDER N_SAUS_ORDER DRANK_SIZE ORDER_COMPLET
WAT_DRANK	::=	DRANK NUMBER DRANK NUMBER STUK DRANK
F_SAUS_ORDER	::=	AGREE DENY DENY GEEN_F_SAUS
N_SAUS_ORDER	::=	AGREE DENY DENY DANK DENY GEEN_N_SAUS
DRANK_SIZE	::=	PORTIE NUMBER PORTIE DRANK_SIZE PLEASE
ORDER_COMPLETE	::=	AGREE NEE GIVE_ORDER
GEEN_F_SAUS	::=	Geen frietsaus, alstublieft.
GEEN_N_SAUS	::=	Geen nuggetsaus, alstublieft.

2.6.1 Rule of customers' order

The following rules are used to analyze the customer orders further. Only the menu items and its attributes are important, the other stuff like “I would like to”, “Please give me” and etc are all junks that we don't care. “stuks” is also junks because it is of no use to us. But its place is between “number” and the menu item, so we cannot neglect it. Many customers know what they want and also give the order in one sentence; the operator doesn't need to ask them for extra information such as the size of drink. If a menu item has its own attributes then it will get its own rule. Coffee has two attributes: milk and sugar. We cannot put it with other drinks together. Otherwise other drink like cola will have these two attributes too. There is another situation we need to consider. A customer can say “I want a large chocolate MacShake.” but he also can say “I want a MacShake, chocolate large”. We'll take Milkshake as an example to see how many ways the customer can order it. Milkshake has two attributes: Portion and Flavor. The customer wants order a big vanilla milkshake.

- Case 1: I would like to have a Milkshake.
- Case 2: I want a vanilla Milkshake.
- Case 3: I want a big vanilla Milkshake.
- Case 4: A big Milkshake please, vanilla.
- Case 5: Milkshake, please.
- Case 6: A milkshake, big, vanilla.
- Case 7: A milkshake please, vanilla, big.

From the example above we can see the customer can order the Milkshake in different ways. He can give all the information needed in one time or he can wait the operator to ask them. He can also put the attributes before the menu item or after it. The place of the two attributes can also be changed.

Sometimes the number of the menu is not given and the operator will take the default number –one- as granted. Here are some examples of those order rules. The words enclosed with “[]” are optional.

Shake_rule	=	[Numbers], [Stuk], [Size], [shake_smaak], shake, [Met], [Size], [shake_smaak], [Size]
Sandwiches_rule	=	[Numbers], [Stuk], Sandwiches
Frisdrank_rule	=	[Numbers], [Stuk], [Size], Frisdrank, [Comma], [Size]
Chocomel_rule	=	[Numbers], [Stuk], [Chocomel_temperature], Chocomel, [Comma], [Chocomel_temperature]

2.7 Requirements Specification

2.7.1 What will be done?

Our task is to design and implement a nonverbal computer interface for McDrive. The input of the system is text, because the speech recognition is not good enough yet. The output of the system is text, speech, and pictures of menu items, smiley faces and facial expressions of a wizard. The end product is a system with which the customer can order menus and the system will generate automatically the feedback.

- 1) Design an interface, what kind of feedback the customer will get. The screen will be divided into a number of sub-screens with fixed formats. A board for the verbal (text) reaction of the system; a board with the pictures/movies of the menu items; a board for the smiley faces and a board for the wizard.
- 2) Analyse existed/simulated McDrive dialogues and find out which feedback can be given: the text prompts and pictures of order and facial expressions of wizard.
- 3) Choose a tool to make a wizard who has facial expressions. The wizard will always be in the picture and show the right facial expressions at the right moment. There are diverse possibilities such as illustrators, regulators and so on.
- 4) Build a nonverbal dictionary, which is consisted of possible facial expressions.
- 5) Implement a manual prototype in which an operator- and a customer- interface will be built. The operator can generate the system response by using a special keyboard. The text answer and the belonging picture and smiley will be shown on the screen of the customer interface and the wizard will show the right facial expression.
- 6) To conclude what a good reaction is a Wizard of OZ study will be done; an automatic system will be simulated. Therefore a nonverbal keyboard with icons for facial expressions, answers and pictures is needed and by clicking the buttons the nonverbal feedback is generated. It looks like the system works automatically but in fact a human being give the feedback with the help of a special keyboard. A number of students will be asked to work as client and do some experiments and the results will be analysed.

2.7.2 Requirements

Before we begin to build the MMS system we need to decide what kind of requirements have to be satisfied. We'll look at the general requirements and other demands over function, security, presentation and so on.

General requirements of MMS

- Under normal circumstances The MMS is able to replace the human operator.
- Behave like a real person, that means have feelings.
- The customer cannot have the feel that they are speaking to a cold machine.
- It has to be a real time order.
- Menu picture need to be shown.
- Not only the spoken response the text need also be shown.
- Electronic dealing with the information.
- Has a memory of the menu -> to improve the understanding.
- No radical changes – the other department like the kitchen keep the same.
- The decentralized character of the organization needs to be maintained.
- Flexible system. Can be updated.

Functional requirements

- Input new menu category.
- Input new menu items.
- Edit existed menu items.
- Search the database.

Presentations requirements

- The system needs to be real time, the customer cannot wait half a hour for a reaction
- The capacity of the system needs to be big, the menu database is small but the customer database will be big.
- Simultaneous users has to be possible.
- After the central administrator changes the information the changes will be available in all the McDrive stores.

Security and privacy requirements

- Menu information only accessible for the employees of McDrive.
- Customer information only accessible for the related employees of McDrive.
- Only the administer can edit the database.
- Guarantee the privacy of the customers.

Reliability requirements

- Correction of information: no store- and /or transmission error.
- Consistence of information.
- Prevention of the loss of information – backup possibility.

Facilities wishes

No huge costs for the apparatuses and using the existed network facilities.

Availability requirements

- The system is under normal circumstances always available.
- The system can be used in every McDrive.

User-friendly Wishes

- Interface: conveniently organized, simple to use.
- Use of Windows.
- So much automatic possible and so minder handwork.

Chapter 3 Multimodal System

3.1 Multimodal System

A multimodal system supports interaction with the user through more than one modality, with respect to input and/or output, and with the capacity to interpret and/or generate with respect to the representation of content. Various systems have been developed recently combining speech (or text) interfaces with other modalities ([Maybury 1993]); CUBRICON, for example, combines speech with a graphics and mouse interface in the domain of mission planning; and Alfresco combines text with hypermedia for art exploration. In the general case, a multimodal system supports communication with the user through different modalities such as voice, graphics, and text. Nigay and Coutaz [1993] define multimodality in the following way: “Multimodality is the capacity of the system to communicate with a user along different types of communication channels and to extract and convey meaning automatically.” Both multimedia and multi-modal systems use multiple communication channels. Is Multimedia and Multimodal the same? No. The distinction intended is that a multimedia system is one, which uses different presentation media (e.g. text, graphics, video, speech) without a commitment to the underlying representation of the information presented. ***A multimodal system strives for meaning.*** For example, an electronic mail system that supports voice and video clips is not multimodal if it only transfer them to another person and does not interpret the inputs.

For reasons of efficiency of both storage and processing, individual specialized representations are usually used for information which is intended to be presented in each individual medium, and information can only be presented in a single medium. A “multi-modal” system is one that includes several inputs and output media, but is committed to a single internal representation language for the information. This permits the same information to be presented in any mode, chosen purely by rules that select that mode for that user at that point in task performance as being both sufficiently expressive and most efficient.

3.2 Multimodal Interface

Human-computer interaction is a field that has developed rapidly in the last decade. A common way to state the goal of getting rid of an interface is to call new interface prototypes “natural”. The term *natural user interface* is not an exact expression, but usually means that is as easy to use and seamless as possible. The user can fade out the feeling of using an interface by not attaching any interaction devices to the user and by designing the dialogue in a way that really is natural and understandable. Multi-modal interfaces combine many simultaneous input modalities and may present the information using synergistic representation of many different output modalities. The input modalities can be as simple as two pointing devices, or they may include advanced perception techniques like speech recognition and machine vision.

As spoken dialogue systems for simple information services begin to move into the area of technology, research interest is increasing turning to the integration of spoken dialogue interfaces with other modalities. Speech can compensate for some of the apparent limitations of a graphical interface and the graphical interface can compensate for limitations of speech by making immediately visible the effects of actions upon objects, and indicating through the display which objects (and by extension which actions) are currently salient for the system. Recent empirical studies have suggested that users not only prefer to interact multimodally, but that compared with a speech-only interface a multimodal interface can reduce performance errors, spontaneous disfluencies and task completion time. Multi-modal user interfaces are a strong candidate for being the next breakthrough in building better user interfaces. Multi-modal user interfaces need to be designed well in order to benefit from them. It has been found that when these interfaces are designed poorly they are neither better understood nor efficient. It is very important that human abilities are carefully analysed and understood when these

new interfaces are designed. Thus, careful consideration of human cognitive abilities and motor skills must be carried out.

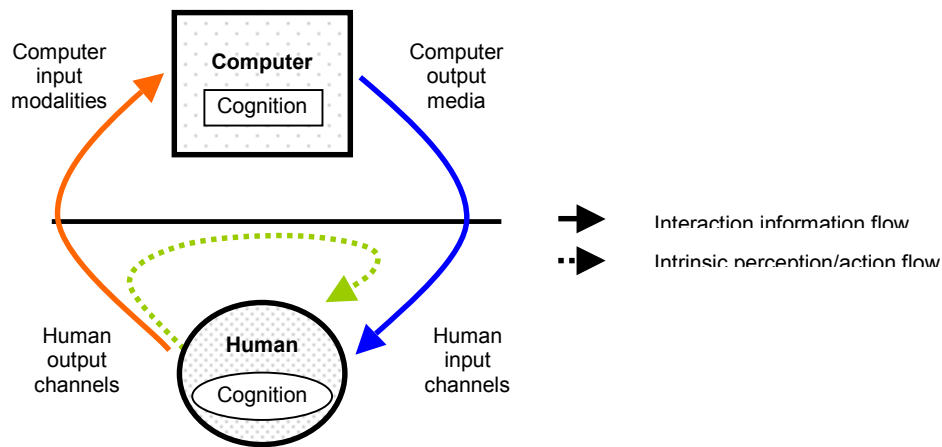


Figure 3.1: A model for the identification of basic processes in human-computer Interaction [MIAMI, 1995]

There are several problems in designing multimodal user interfaces. First, the set of input and output modalities must be selected right. This is not a trivial problem and depends greatly on the task the interface will be used for. Another, even greater problem is how to combine different input and output channels.

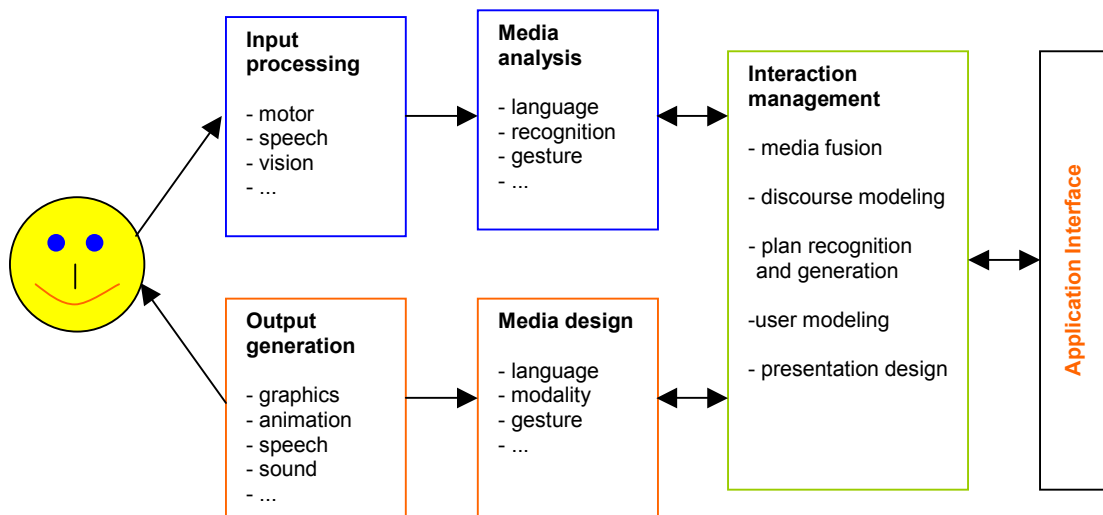


Figure 3.2: Architecture of multi-modal user interfaces. Adapted by Roope Raisamo from [Maybury and Wahlster, 1998]

It is important producing a sensible and useful user interface by integrating multiple input modalities, rather than building better speech and gesture recognizers alone. An interface supporting the kind of highly flexible interaction we envision must be capable of integrating information from both speech and non-verbal input sources to arrive at a correct understanding of complete multimodal events. Current speech and pen systems are still not very popular and did not live up to the promise of bringing the power of computing to the population at large. Their failure is partly due to users' frustrations generated by inadequate recognition performance, especially for handwriting recognition. Likewise users of speech-enabled systems quickly find out that there are tasks that cannot be

conveniently expressed by spoken commands but would be enormously simplified by the ability to point to or circle objects on the screen in addition to speaking commands.

The design of multi-modal interfaces requires the selection and the combination of multiple modalities. Such selection of atomic or composite output modalities can be performed:

1. by the designer while designing the system;
2. or by the user while using the system;
3. or by the system while running.

In case 2, we refer to the system as being *adaptable*. Case 2 must be related to case 1 because *adaptability* implies that the designer has previously selected a range of candidate modalities. In case 3 we call the system *adaptive (adaptivity)*.

For the output modality combinations we want to consider four different aspects:

1. Time: temporal combination
2. Space: spatial combination
3. Interaction language: syntactic combination
4. Semantic: semantic combination

When we have to deal with different modalities, there are various ways of combining them and interpreting their combination. There are 5 combination schemas:

- A. Distant modality combination,
- B. Combination with one point of contact,
- C. Combination with non-empty intersection,
- D. Combination with inclusion,
- E. Combination with the same characteristics.

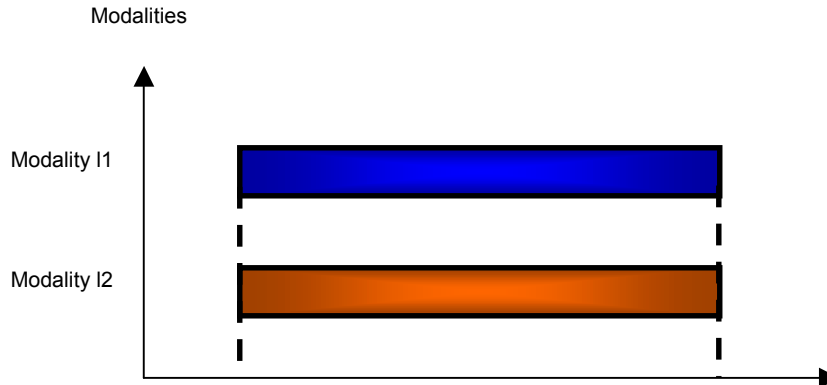


Figure 3.3: Schema for modality combination with the same characteristics

	A	B	C	D	E
Temporal	Anachronism	Sequences	Concomitance	Coincidence	Parallelism
Spatial	Separation	Adjacency	Intersection	Overlaid	Collocation
Syntactic	Difference	Completion	Divergence	Extension	Twin
Semantic	Concurrency	Complementary	Complementary and Redundancy	Partial Redundancy	Total Redundancy

Table 3.6: Applying the five combination schemas to the four combination aspects

For reasons of efficiency of both storage and processing, individual specialized representations are usually used for information which is intended to be presented in each individual medium, and information can only be presented in a single medium. A “multimodal” system is one that includes several input and output media, but is committed to a single internal representation language for the information. This permits the same information to be presented in any mode, chosen purely by rules, which select that mode for that user at that point in task performance as being both sufficiently expressive and most efficient.

A multimodal interface can benefit from the use of multiple modalities:

- Modality Synergy: Interfaces can benefit from modality synergy on both the input and output sides of system. On the input side, interpreting input that is conveyed redundantly in several modalities can increase interpretation accuracy. On the output side of a computer system, multimedia output is inherently more expressive than single modality output.
- Freedom of choice: Although the same task may be achieved with equal efficiency using different modalities, users may prefer one or another modality.
- Naturalness: offering multiple modalities to interact with a computer can be more natural to the human user.

The ease and robustness of human-human communication is due to extremely high recognition accuracy (using multiple input channels) and the redundant and complimentary use of several modalities. We make McDrive system multi-modal in the hoping that human-computer interaction can benefit from modeling several modalities in analogous ways.

3.3 Unimodal / Multimodal Input in MMS

How MMS generate the most appropriate interpretation for the incoming streams of multi-modal input? For McDrive system we use the customer speech as the input. The other possible inputs are the license plate of cars and the voice characteristic of customers. These possible inputs can be used to register the customers and their orders so that the new orders of the customers can be compared with the older ones. In this way the speech recognition can be improved. And the graphic input of the license plates using camera seems to be a better choice, even though sometimes members of a family use the same car. Identifying a license plate is much easier than identifying a voice. The registrations are put into a database that is checked every day. If a registration record has not been updated since a month ago, then it will be deleted. In this way we can keep the database compact so that the search/match time can be cut short.

Input devices can be divided into multiple groups, selected for functionality:

- Sound input (microphone)
- Image input (camera)
- Text input (keyboard)
- Others (sensors)

Machine vision is the observation of an environment using cameras. The camera is set up in full view of the number board of the car. The video camera is running before the car stopped and stopped when it can get a clear picture of the number board. It differs from image processing in that it extracts information from images that are relevant for a particular set of services. The basic services that machine vision can provide to HCI include detection, identification, and tracking. Detection determines the presence or the absence of an entity of a given type. For example, is there a number plate in the scene? Identification is recognizing what entity of the class is present in the scene, for example, that plate NW-46-LK is in the scene. Tracking is determining the location of an entity over time. Here we do not need this service.

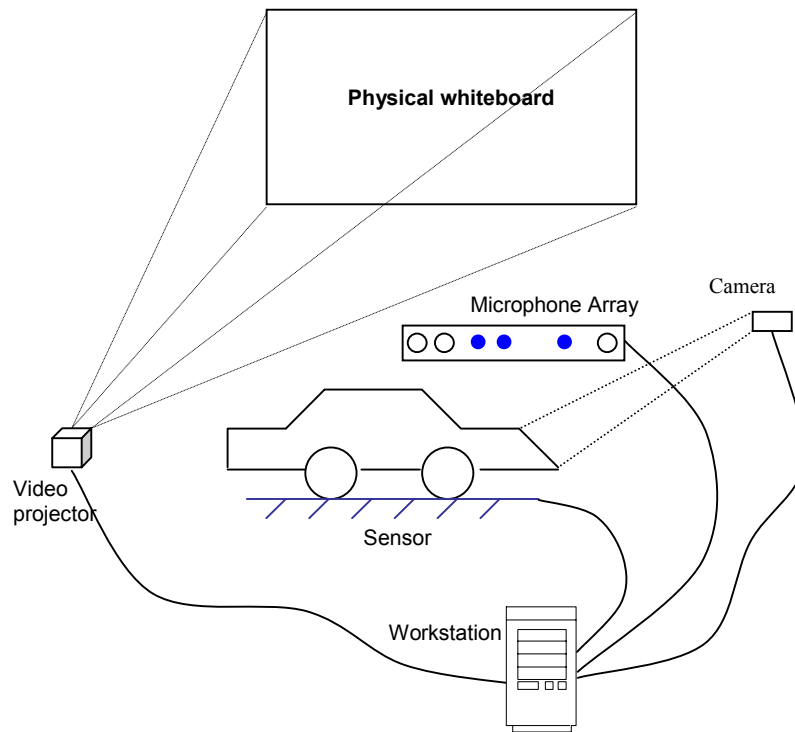


Figure 3.4: A simulation of the McDrive environment

We can place a pressure-sensitive sensor outside the McDrive window to detect new visitor. This sensor is connected to the computer. By detecting the state changes of the sensor, the system is able to figure out whether the visitor is entering, exiting, or standing on the threshold. Once the sensor has determined that the car of the new visitor has enter the area, the McDrive guide agent automatically appears on the screen.

3.4 Multimodal Output in MMS

Multi-modality has mainly been studied for input (from user to system) interfaces, by utilizing multiple input devices for exploiting several human sensory systems. In addition to the fact that fewer studies focus on output multi-modality, the related studies mainly investigate a single output modality such as speech synthesis and natural language text generation. We define an output modality as the coupling of a physical device with an interaction language. A system can be multi-modal without having several output devices. A system using the screen as the unique output device is multi-modal whenever it employs several output interaction languages.

Our definition of output is system-oriented. A user-centered perspective may lead to a different definition. For instance, according to our system-centered view, electronic voice mail is not multi-modal. It constitutes a multimedia user interface only. Indeed, it allows the user to send mail that may contain graphics, text and voice messages. It does not however extract meaning from the information it carries. In particular, voice messages are recorded and replayed but not interpreted. On the other hand, from the user's point of view, this system is perceived as being multi-modal: The user employs different modalities. (Referring to the human senses) to interpret mail messages. Under output modalities or computer output media, we understand the media (devices) and modalities (communication channels), which are used by computers to communicate with humans.

In the McDrive system, user control is based on natural and intuitive dialogues using one single modality: voice. System feedback is provided in text, graphics, animations, sound and speech. Use of text and graphics, in addition to sound and speech, enhances system message feedback, and makes a natural user-machine dialogue, especially with natural spoken language. The McDrive system speech response will make use of pre-recorded speech for all fixed messages that are unlikely to be changed, and concatenation of speech units for variable information. Speech concatenation makes use of pre-recorded speech units that are stored in a dictionary. Responses in the form of a text string can be automatically generated, and the text string is used to locate the appropriate dictionary units for concatenation.

3.5 Multimodal Data Presentation

At one side we need to integrate all these medias, modalities and devices into a reliable, safe, secure McDrive system, so that the speech output and graphics /text output can be simultaneous. But at the other side we need to keep the McDrive system adaptable and open that means all of its output is generated on the fly and customized for the intended target audience and situation.

When the customer orders the picture of the corresponding menu item that he order will be shown on the display screen. Sometimes the operator needs to know more, for example which size of coca-cola the customer prefers. If we just put a picture of coca-cola, what the customer sees is only the cola. But the emphasis of the question lays on the size, not on cola self. Of course the customer can hear and see the questions but we want a more direct impression. The customer can understand what the operator asks at the first glance. We can use a picture of three cola's of different size. The picture is static and to make it more attractive to the customer we make some flash movies for these questions. Let's take the flash movie of asking coca-cola size as an example. Three coca-cola's appears on the screen one by one from small to large, and a hand points to them in turn, when the cola is pointed it jumps with a question mark on it. Here are some frames of the cola movie.



Figure 3.5: Some frames of flash movie “Ask drink size”



Figure 3.6: Flash movie “End”

As we discussed in chapter two the dialogues between the customer and the operator have some patrons. The operator welcomes the customer at the beginning and ends by telling him to get what he ordered at the pickup window. To make these predefined occasions more interesting we also made some movies. The McDonald's character Ronald plays a leading role in it. Ronald can be said as the symbol of the McDonald's. When customer comes, Ronald appears and walks towards. When the

order is completed, Ronald slides to the pickup window, jumps and turns over with a bag, and then Ronald slides away.

The other graphical stuffs like smiley faces and wizard we are going to talk about in the next chapter.


Chapter 4 Smiley and Wizard

The MMS is aimed to replace the human operator so it has to behave like a human being. People usually communicate using more than just words. We convey emotions, facial expressions, and other subtle body movements that help to add "flavor" to our sentences.

4.1 Smiley

Now more and more people use a little round yellow face to express some feelings or just something. This little face can be smiling, sad or angry. You can see it everywhere, email, SMS, even on the advertisements. They have a common name – Emoticon or Smiley as most people know. They are used to express emotions or convey facial expressions. In this thesis we'll use "Smiley". Smiley lets people know how you feel without saying "I am happy" or "I am bored". Smiley is graphical representations of words, thoughts, emotions or character using different combinations of standard ASCII characters. For example the smiling smiley :-)) consists a colon ":", a hyphen "-" and a right parenthesis ")" and the winking smiley ;-)) is composed of a semicolon ";", a hyphen "-" and a right parenthesis ")". Smiley is sideways because it is read by leaning your head down toward your left shoulder and looking at the characters sideways, reading from left to right. Isn't that :-)) a happy face on its side? The colon represents the eyes; the dash represents the nose, and the right parenthesis represents the mouth. They are used in e-mail messages and in chat, newsgroup and bulletin board postings and other forms of communication using computers. If you receive an email from a friend you may see a ":-))" on your email, which means, "don't take what I just wrote too seriously". If you go to a discussion forum you may see that there are some small icons like 😊 attached to the messages. And if you chat with others you can get some characters such as ":-0" on your screen. Nowadays it is also widely used on your mobile telephone. It is a way showing how a message should be interpreted (that is, the writer's mood). They offer sideways expression of some basic emotions that color and clarify a conversation that is not fact to face. In this way the messages are felt more like personal contact. Generally we can tell how the other feels simply by his facial expressions or other body languages or the voice tone. But it is impossible to see or listen to someone in a chat room or while reading an email unless there is a webcam or video camera. This makes it very difficult to realize what mood he or she may be in or how they are feeling at that moment. This is due to the impersonal nature of the Internet. Without face-to-face contact, it's difficult to convey that. Back in the early 70's Franklin Loufrani a journalist created a simple concept for France soir and other European newspapers, he displayed icons to communicate news and especially good ones. He gave this original icon the name of Smiley, it was published for the first time on Jan 1st 1972. Today it's impossible to read electronic mail or postings without encountering dozens of the little smileys.

There are hundreds of smileys used to convey emotions. There can be many variations; in fact everyone can create his own smiley. Someone may use ":-cu:" to express confused and the others may use ":-confused:" While there are no standard definitions for the smileys, there are some basic smileys such as the happy face :-)) ☺ and the sad face :-((☹ that are universal and recognized. If we type ":-))" in a word document then we get ☺ in place of ":-))". Word has automatically converted it. The administrators of BBS often provide the users a chance to add smileys to their messages. There are so many smileys and as we said there are no official rules about smileys, the users and readers cannot always understand what these smileys mean. And everyone has his own translation. But if there is a picture then everyone can know what the writer wants to express. So some BBS or chat room provide an additional function to transform your smiley characters into an actual picture of a smiley face so that the user needs only to type some keys and the system will convert it automatically to the real icon. In this ways the others are able to figure out what you feel or what you mean. There are also in some BBS directly pictures of smileys provided to the users. We can say that one person is angry, but how angry he is? He can be furious or just unhappy. There can be many graphical translations of angry.

Please look at the following Smileys: , , , , , all those pictures express one emotion, that is angry. We can always alter them by changing the eyes, the mouth, the nose, the colour and etc.

Although people always say that the customer is god. But in real life the human operator is like all of us, he is just a human being. There is not always smile and politeness. Sometimes he just has a bad day. Sometimes he feels impatient if he needs to repeat himself. When he can't understand the customer he feels confused. Sometimes he is tired after working long time. And sometimes he wants to have some movements. There needs more expressions than smiling.



4.2 The Functions of Smileys





The Smileys have the following uses.

- **Emblem:** here facial expressions are used to replace the text. For example a smiling face means happiness.
- **Illustrators:** they are complementary of the text
 - *Batons:* movements that accentuate one certain word, (indicate with one finger)
 - *Underlines:* movements which emphasize one certain word or part of sentence (nod one's head)
 - *Ideographs:* movements that emphasize line of thought (think)
 - *Kinethographs:* movements that sketch one human act (drink)
 - *Rhythmic:* movements that emphasize the rhythm or tempo
 - *Spatial:* movements that sketch one spatial relationship (one big or small cola)
 - *Deictic:* movements that point to a consultant/expert for example Ronald Donald
- **Affects utterance:** surprise, dazed, disapproval, and happiness
- **Regulators:** regulate the dialogue such as alignments
- **Adaptors:** barely aware movements that just to full an empty or to camouflage certain feelings (pull the mouth, yawn or scratch one's head)

We use the pictograms as code for the facial expressions of the wizard. In the following table we give some example. In the appendix there is a non-verbal dictionary composed of the pictograms as index and the facial expressions of the wizard. We try to use so many as possible smiley faces for the pictograms. But one important factor in the service of McDrive is the speed, the conversation between the operator and customer is rapid, the customer needs to see the menu board and listen to the operator, look at the wizards and at the same time he needs to tip the head to the left to see what the smiley means. It will be too much asked from the customer. So we use the real icons for the smileys so that the customer can directly know what the operator feels.











Table 4.1: Some Smiley examples

Smiley	Icon	Meaning	Category
8:)		Wizard	Illustrators
::)		Read	Illustrators
: -)		Smile	Affects utterance
: -o		Uh-oh	Affects utterance
: -(	Sad	Affects utterance
: '-(	Crying	Affects utterance

		Natural	Regulators
:~9		Lick	Adaptors
-I		Sleepy	Adaptors
;-)		Wink	Adaptors

First let us take the dialogue example which we shown in the previous chapter to see the use of the smiley faces. The other dialogue examples are discussed in the appendix.

Table 4.2: Dialogue example with smiley

Speaker	Prompts	Smiley
System	Good morning, this is the prototype of the McDrive system. Can I have your order please?	
Customer	Hello. One Mac Deluxe with a big milkshake.	
System	You want one Mac Deluxe with a big milkshake, is it correct?	
Customer	Yes.	
System	Which flavor of milkshake do you want?	
Customer	Vanilla and with fries sauce.	
System	You want vanilla and fries sauce, is it correct?	
Customer	Yes.	
System	10 guilders 45 and please drive to the first window.	 

4.3 Wizard Design

To make MMS more human like we want to design a wizard so that the customers can have a target to talk with. In the last few years, animated characters based either on cartoon-style drawings, read video, or geometric 3D-models have become increasingly popular in user interfaces. Human communication is fundamentally social. Successful communication involves more than recognition of words. The process of dialogue implies exchanging cues to signal turn-taking and understanding. In addition to non-verbal cues, a conversation involves a common context between the conversing parties. Using animated characters enables you to leverage this aspect of interaction. Add an interactive character that can show meaningful facial expressions and gestures, and you significantly expand the potential bandwidth of communication. Users will expect a character to conform to the same social, though not necessarily physical, rules they use when interacting with other people, even when they understand that the character is synthetic. Characters can improve conversational interfaces by providing cues like head tilts, nods, or shakes to indicate when the speech engine is in the listening state and when something is recognized.

To design a character we should first consider the profile of our target audience and what appeals to them as well as what tasks they do. We also need to consider our character's basic personality type:

dominant or submissive, emotional or reserved, sophisticated or down-to-earth; or perhaps you want to adapt its personality based on user interaction. A character's animations reflect its gender, age, personality, and behavior. The number and types of animations created for a character depend on what the character does and how it responds to different situations. Many popular cartoon characters are not realistic in their presentation. Characters can be based either on cartoon-style drawings, real video, or geometric 3D-models.

In the customer interface the wizard acts mainly as a presentation assistants, and in the operator interface his function is guiding. The tasks of the wizard is as follows:

- Express feelings. This is the main task of the wizard.
- Indicate the system's internal state: listening, understanding, uncertain, speaking etc.
- Visually refers to other stuffs on-screen graphics such as illustrations and flash movies.
- Suggest what the user should do.
- Check if the user makes the right choice (constraints).

For the model of the wizard of the McDrive system, we have a few choices:

- Use the ready-made character such as Genie, Merlin that Microsoft provides.
- Use Ronald or other McDonald's characters.
- Use a male or female.



Figure 4.1: Some characters

At the beginning we want to use Ronald as our model because every customer of McDonald's is familiar with him. We also considered the other McDonald characters such as Birdie and Grimace as the assists of Ronald. Because they are funny and familiar with the McDrive customers. But the laughing image made Ronald cannot have enough facial expressions. And we can see the mouse movements when Ronald speaks because of his thick lips. The cartoon character such as Genie and Merlin has the same problem so they are also out of our consideration. We decide to make a realistic human wizard. Now come the question "Should this wizard be a man or woman?" We prefer a woman as our wizard. One reason is that people works in McDrive are mostly women. The other reason is that the image of a woman is more affable. But will we use the whole body of a woman or only the upper part including the hands? The wizard of MMS is not like a guide in a museum. In a virtual museum, the wizard moves around to give a virtual tour. The museum wizard provides a full-body gesture interaction and the whole body needs to be presented. The wizard cannot walk around without legs. In MMS the facial expressions, the lip movements and the hand movements are important; the legs are not. And if we use only the upper part of the body, the wizard will occupy little place on the screen, and the wizard can be relatively bigger of format. And because it needs little memories, the system can run faster.

We call our wizard as "Lisa". She is a blond girl who has a white T-shirt on. Lisa takes place in the right part of the screen. The standard setting is 320*320 pixel. We can make her bigger or smaller according to the reality. The default language of Lisa is Dutch. Her task is to help the customers to place their orders and send them to the kitchen. Lisa has different facial expressions and movements under different conditions. Here are some examples. When Lisa greets the customer she smiles. If Lisa

cannot satisfy the demands of the customer, she says sorry and has a so-called sorry expression. Lisa put her hand near her ear when the customer talks. Lisa can also talk.

4.4 Microsoft Agent

Most people are familiar with Microsoft Office. One of the features introduced with Microsoft Office 97 was the Office Assistant, a cute little animation designed to help users get the most out of the application. It plays all kinds of cartoons in a small window, which change according specific actions. When you have problems, an animated paper clip jumps out and gives you some advice. This paper clip assistant is a multimedia assistant program that Microsoft supplies for Office program. Microsoft has made it easier with a new technology called **Microsoft Agent**. This technology allows you to incorporate Office Assistant-like characters into your own programs. It is an improve from the office assistant, the characters that Microsoft Agent supplies have not only interesting actions, they can talk with users through sound cards, microphone. These functions make computers to have more human glamour. For example, Microsoft Agent plays animations assigned to the Listening state when a user presses the push-to-talk listening key and animations assigned to the Hearing state when an utterance is detected.

Microsoft® Agent is a set of programmable software services that supports the presentation of interactive animated characters within the Microsoft Windows® interface. The characters can be used to introduce, guide, and entertain the user interface as interactive assistants. Microsoft Agent enables software developers and Web authors to incorporate a new form of user interaction, known as *conversational interfaces*, that leverages natural aspects of human social communication. The conversational interface approach facilitated by the Microsoft Agent services does not replace conventional graphical user interface (GUI) design. Instead, character interaction can be easily blended with the conventional interface components such as windows, menus, and controls to extend and enhance your application's interface. Microsoft Agent's programming interfaces make it easy to animate a character to respond to user input. Microsoft Agent represents a new generation past the original Office Assistants. Instead of living inside a small square with a frame, only the character, or Agent, itself is displayed. Animated characters appear in their own window that always appear at the top of the window z-order (that is, always on top), providing maximum flexibility for where they can be displayed on the screen. The Microsoft Agent software runs from the Agent server. The Agent server controls all of the Agents that are used on the computer at any point in time. The Agent server starts automatically the first time a character is requested, and it is stopped when the last character is unloaded. There are some agent characters such as Genie, Merlin available that we can see the picture in the figure 4.1. But Microsoft made it possible to everyone to build his own wizard.

An Agent is a character that, when active, will always overlay everything on the screen. The Microsoft Agent software also includes speech engines that allow the Agent to speak and to listen. A Windows-compatible sound card is necessary if you want the characters to speak or listen to you. It can speak through the use of the text-to-speech engine, often is the Lernout & Hauspie TruVoice text-to-speech engine used. In addition to spoken audio output, the Microsoft Agent interface supports textual captioning in the form of text output in cartoon-style word balloons. Words appear in the balloon as they are spoken. The balloon hides when spoken output is completed. Speech generated by some TTS engines can sound a bit unnatural and may be difficult to understand—many people prefer the natural-sounding voice provided by a wave audio (.wav) file. If a compatible Command and Control speech engine is installed, Microsoft Agent supplies a special window called the Voice Commands Window that displays the commands that have been voice-enabled for speech recognition. In this way it can also respond to spoken commands under your program's control. A menu with the commands can be used if a microphone is not available. Like the Office Assistants in Office 97, Agents are animated figures capable of expressing emotions such as boredom, confusion, happiness, and sadness. They can gesture at various locations on the screen and move around. They can also pop in from nowhere, or just roll up and disappear.

4.4.1 Microsoft Agent installation

System requirements

- Windows 9x, Windows NT 4.0(x86), Windows 2000, Windows Me, or later
- Internet Explorer version 4.0 or later
- A Pentium 100-MHz PC (or faster)
- At least 16 MB of RAM
- At least 1 MB free disk space for the core components
- An additional 2-4 MB for each character you install.
- An additional 32K for each language component (dll).

To control a character we first need to install some software.

- The Microsoft Agent core components - is the Server part of the technology and decides how the character moves, talks and listens.
- The Lernout & Hauspie TruVoice Text-to-Speech (TTS) Engine - provides speech output capabilities for Microsoft Agent so you can hear what the characters are saying through your sound speakers.
- The Microsoft Speech Recognition Engine (optional) -provides speech input capabilities for Microsoft Agent. This allows you to speak to characters through a microphone. However, Microsoft Agent must be specifically programmed to understand and interact with what you are saying. Microsoft Agent does not support dictation.
- The Speech Control Panel - enables you to list the compatible text-to-speech engines installed on your system and to view and customize their settings.
- Microsoft SAPI runtime binaries - The Agent speech interfaces are based on the industry standard Speech API (SAPI). This means that Agent can be hosted with speech engines provided by other vendors in other languages.
- A character which is put to the subdirectory \Program Files\Microsoft Agent\Characters
- An application that uses Microsoft Agent technology to get your Agent character talking and moving.

4.4.2 Agent Programming

This agent controls the actions of the wizard. The wizard is semi-automated. Its primary purpose is to execute presentation acts and talk to the customer. The lip movements and facial expression can improve the understanding. The wizard's behavior is not just designed by the directives in the script. The behavior of the wizard follows the equation:

Wizard behavior: = directives + self-behavior

There is a program needed to control the agent. Agent is a technology that includes a programming interface that can be coded from any language that supports COM, such as C++ or Visual Basic (VB). Here we give some VB examples over how to make the agent act.

- First the character is loaded with the Load method, passing it a character animation file name. This loads the character's data into the Agent Characters collection.
`Agent1.Load "my character", "genie.acs"`
- To make the character appear, the Show method is used, and the character reference used in the Load call needs to be specified.
`Agent1.Characters("my character").Show`
- Once visible, we can play a character's animation using the Play method, specifying the name of the animation we want to show.
`Agent1.Characters("my character").Play "Greet"`
- To make a character speak, use the Speak method, specifying the text to be spoken.
`Agent1.Characters("my character").Speak "Hello world!"`

4.5 Build the Wizard

But how can we build a character? What tools do we need? We can see that each action of a character is in fact an animation that is composed of a timed sequence of frames. Like traditional cell animation, each character's animation is made up of separate frames, each altered slightly, that when played sequentially create the illusion of motion. Each frame is composed of one or more bitmap images. A typical animation averages about 14 frames so that it plays for no more than six seconds. Microsoft provides the users a development tools **Microsoft Agent Character Editor** to build a character. We can use this editor to assemble, sequence, and time the images, supply other character information, and compile all the information into a final character file. But the images need to be created with other tools; the choice of the graphic tools depends on what kind of character we want to design. The character can have any kind of appearance you choose, from cartoon-style to realistic. We decide to design a human-like character. So we choose **Poser** as our image creating tools.

4.5.1 Microsoft Agent Character Editor

In the section we'll give a brief introduction over how to use the agent editor.

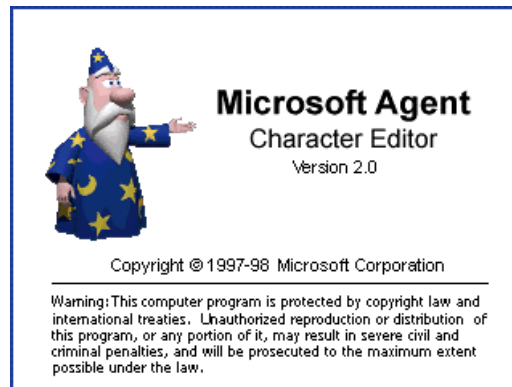


Figure 4.2: Microsoft Agent Character Editor

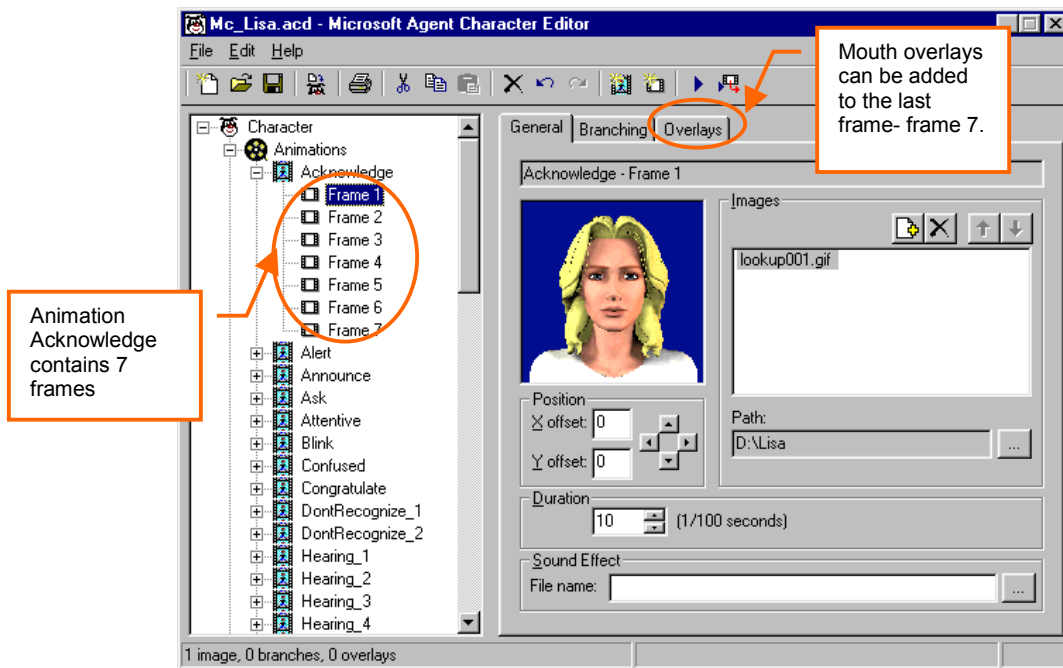
Determine the global animation settings

To build a character we need first to set the character's general information. A name is necessary and we can also specify a short optional description (256 characters) for your character in the Description text box. The server exposes what we enter in the Description text box to client applications. Every character has its own identifier. Here we choose Dutch as our default language. We can also define the character's default word balloon characteristics. In MMS the word balloon is not needed so we set it as invisible. Then we can go to the Properties page with the default settings for all animations by clicking the Animations icon in the tree. There we can alter the frame size, the default frame duration, and color palette settings. The animation frame height and width must remain constant throughout the entire character definition (that is, for all of that character's animations). We set our wizard as 320 * 320. The default voice setting can also be set. By default the character editor will use the 11th color in the palette. This is not always going to produce the transparency wanted. Before the images are added to the editor they need to be made "transparent", every image must have the same background, which would become the "transparent" color. Otherwise the built agent will have a background.

Creating a New Animation

To create a new animation, choose New Animation from the Edit menu or the New Animation button on the toolbar. This adds a new animation icon in the tree under the Animations icon and assigns the new icon a default name. Rename your animation by typing in the Animation Name field. Note that animation names within a character definition must be unique. Every animation is composed of frames. To create a new frame for the animation, choose New Frame from either the Edit menu or the toolbar. This adds a new frame icon to the tree under the animation icon, and displays three tabbed

pages. The General page includes controls that enable you to load and adjust an image for your frame. It also includes a display area for the frame's appearance. A frame can contain one or more images. To define an image for a frame, click the Add Image File button just above the Images list box. The Select Image Files dialog box displays, which allows you to select a bitmap image file. The four arrow buttons beneath the image in the Position box can be used to adjust the image's appearance within the frame. The Duration text box is to set the duration for the frame; that is, how long the frame will be displayed. If a frame has no image and zero duration, the frame will not be displayed when the animation plays. We can also define which frame plays next with the help of Branching. By default, the next frame played in the animation sequence is always the next frame in the z-order. However, by choosing the Branching page, you can set the probability for up to three other frames that the server may play. Animations can also loop indefinitely.



Here are some examples of the animations that Lisa supported and the whole list is attached in the appendix.








Table 4.3: Some wizard actions

Animation	Assign Prompts	Description	Supports Speaking	Sound Effects
Acknowledge	Give confirmation	Nods head	Yes	No
Alert	Give hints or warning	Raises eyebrows, open eyes bigger	Yes	Yes
Announce	Announce a price reduction action etc	Put both hands around the mouth	Yes	Yes
Ask	Ask for additional order/info	Raise left eyebrow and open the mouth partially	Yes	No
Attentive	Pays attentions to the customer	Raise eyebrows, open the	No	No

In addition, mouth overlays can be included to provide the visual images for lip-synced animation. The Microsoft Agent animation services display mouth animation frames on top of the last frame of an animation, also called the *speaking frame* of the animation. A character cannot speak while animating, so you only supply mouth images for only the last frame of an animation. The character editor and the text-to-speech engine will take care of making the mouth movements match the sounds.

In addition the Microsoft Agent Character Editor enables you to define seven basic mouth positions from closed to wide at full width that correspond to common phoneme mouth shapes shown in the following table. Mouth image files can be assigned to these standard mouth positions. In this way when Lisa speaks her face and mouth can also moves. So give it a realistic feeling.

Table 4.4: The possible mouth animation images

Mouth Position	Sample Image	Representation
Closed		Normal mouth closed shape. Also used for phonemes such as "m" as in "mom," "b" as in "bob," "f" as in "fife."
Open-wide 1		Mouth is slightly open, at full width. Used for phonemes such as "g" as in "gag," "l" as in "lull," "ear" as in "hear."
Open-wide 2		Mouth is partially open, at full width. Used for phonemes such as "n" as in "nun," "d" as in "dad," "t" as in "tot."
Open-wide 3		Mouth is open, at full width. Used for phonemes such as "u" as in "hut," "ea" as in "head," "ur" as in "hurt."
Open-wide 4		Mouth is completely open, at full width. Used for phonemes such as "a" as in "hat," "ow" as in "how."
Open-medium		Mouth is open at half width. Used for phonemes such as "oy" as in "ahoy," "o" as in "hot."
Open-narrow		Mouth is open at narrow width. Used for phonemes such as "o" as in "hoop," "o" as in "hope," "w" as in "wet."

For the character's spoken output, Microsoft Agent provides the choice of a synthesized, text-to-speech (TTS) voice or a voice that uses recorded sound files.

When we design an animation we need to consider how to smoothly transition from and to the animation. For example, we create an animation in which the character looks right, and another in which the character looks left. We want the character to animate smoothly from one position to the other. Although we could build this into either animation, a better solution is to define a neutral or transitional position from which the character starts and returns. In the Microsoft Agent Character Editor, we can specify a separate, complementary **Return** animation to return the character to the neutral position. The Return animation is typically no more than 2-4 frames so the character can quickly transition to the neutral position. Because every animation begins with the neutral position except for the Return animation and "Show" animation, so we delete the neutral frame from all the Return animation. In 1) of figure 4.4 we can see there is an interruption between the last frame of "Look Right" and the first frame of "Look Left". This interruption disappears when we add a 2- frame "Look Right Return" animation between them. When we call action "Look Left" after "Look Right" the character will automatically call "Look Right Return" first in order to go back to the neutral position. And then action "Look Left" will be done. We can define a Return animation by creating an explicit animation for this purpose. We can also create a Return animation using the exit branching we define within the animation. To assign a Return animation, select the animation in the tree, and select either the Return animation created or Use Exit Branching from the Return Animation drop-down list on the Properties page. Creating and assigning a Return animation has an added benefit: When the server gets a request to play another animation, it will attempt to play the Return animation for the last animation it played, if a Return animation is assigned. This ensures a smooth transition. If an animation begins and ends at the neutral position, there is no need to define a Return animation.

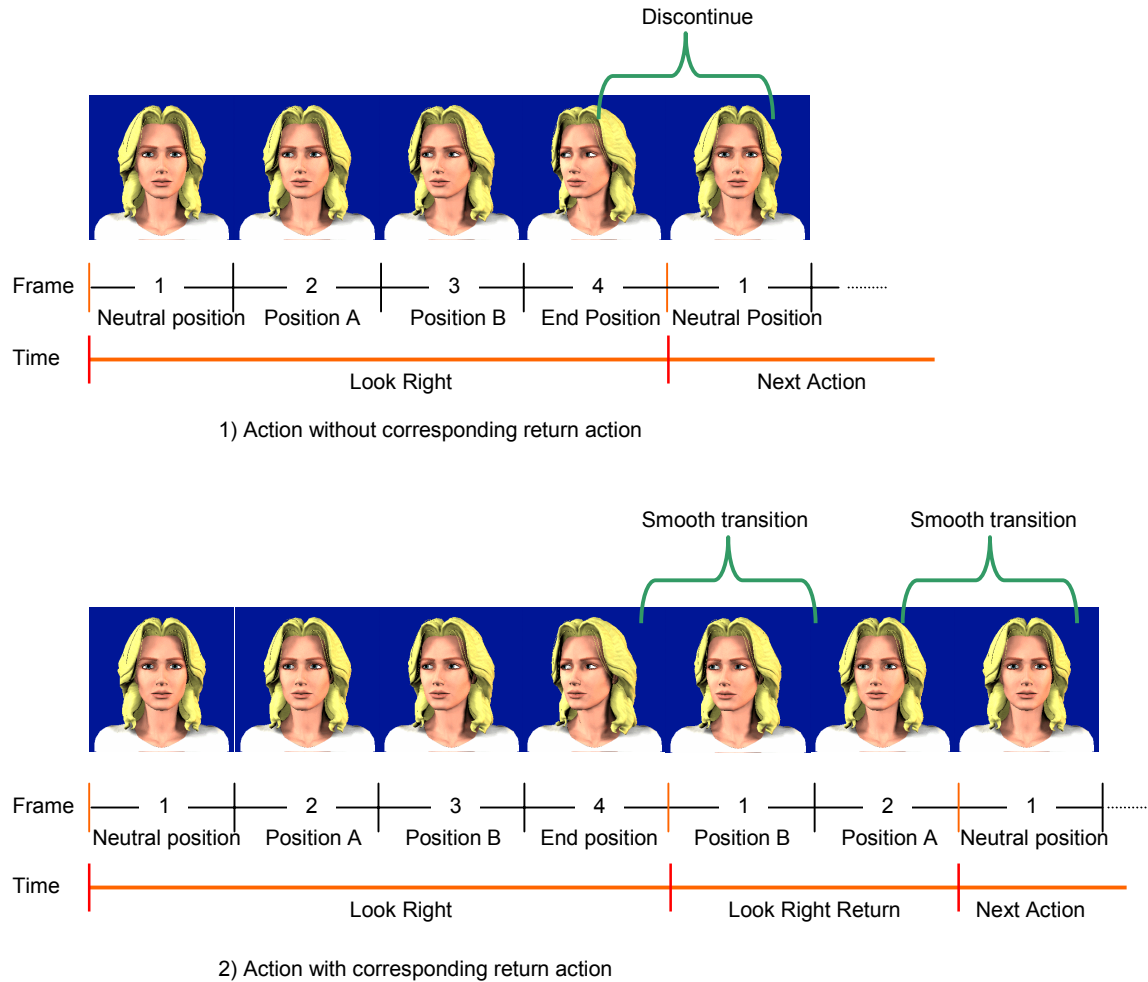


Figure 4.4: The transition between the animations

Assigning Animations to States

The Microsoft Agent animation services automatically play animations when the hosting client application uses certain methods. For example, when an application calls the `MoveTo` and `GestureAt` methods, the server automatically determines where the character is displayed and plays an appropriate animation. Similarly, Microsoft Agent automatically plays Idle animations when the user has not interacted with the character for several seconds. These conditions, when the server automatically plays animations on an application's behalf, are called states. However, for the server to know which animation to play, we must assign animations to these states. The Editor does not support creating additional states because states only apply to situations where the server must play an animation automatically on behalf of the client application. Thus, there is no benefit in defining your own state. If needed, animation can be played by explicitly using the `Play` method.

Build the character

Agent supports two different formats for compiled character files. With one format, all character and animation data is compiled into a single file. This format is primarily used when the character file is located on a local disk drive. A second format compiles each animation as a separate file and is used primarily in Web-based scenarios where animations are loaded from a server on demand. The Agent server also manages certain states of the character to make programming easier for clients. For example, when no animation has been played for four seconds, the server automatically places the character in an idle state, playing animations to keep the character from remaining in a frozen pose.

The idle state typically begins with simple animations such as breathing, eye blinks, or changes of gaze, attempting to model that of a real person patiently waiting for input. However, all of the animations assigned for these states are left to the character author to define.

4.5.2 Poser

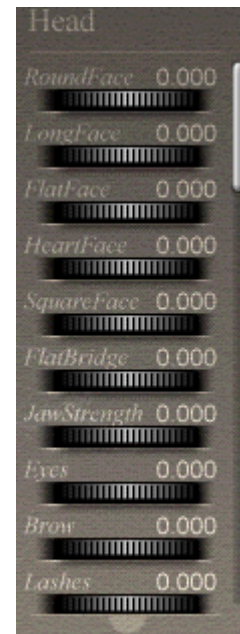
There are a lot of 3D animation software, such as Lightwave 3D of NewTek, Maya of Alias/Wavefront, **3D Studio MAX of AutoDesk**, Poser of MetaCreations and etc. Here we choose Poser as our character image creation tool. Poser is a 3D-character animation and design tool. Users can create images, movies, and posed 3D figures from a diverse collection of fully articulated 3D human and animal models. Libraries of pose settings, facial expressions, hand gestures, and swappable clothing are included as well. With Poser 4 we can dress a character up in different clothing, apply different hairpieces, use body morphs, etc.



Figure 4.5: Some expressions made by Poser

Happiness is expressed through a combination of bright eyes, upturned corners of the mouth, raised eyebrows, a lively stride, straight back, and raised shoulders and arms. The two occurrences are only fractions of a second apart, but nevertheless should be animated in that order. Although the human face is capable of hundreds of expressions, one can model approximately three dozen basic features to achieve most of them. Each of these have been divided into the following separate groups: brows, eyes, nose, mouth, and jaw. These often use sliding buttons that can generate various percentages of each facial movement. If the eye, mouth, nose, eyebrows, jaw, and so on movements are modeled separately, one can achieve a diversity of expressions and phonetic shapes for speech. Since the muscles on the left and right side of the face vary in strength, some of the features are modeled as separate left and right shapes. When modeling the following expressions, a hand held mirror becomes an indispensable tool.

Morphing is one basic technique of Poser animation. Poser is a 3D character animation and figure design tool. The standard Poser head morphs for face animations are: openLips, smile, frown, mouth O, mouth F, mouth M; tongue T; tongue L, brow down right/left, brow up right/left, worry right/left and blink right/left. With these morphs, it's possible to create phonemes in Poser to create realistic facial animations. A phoneme is a linguistic term for the position of the lips, teeth and tongue as they make sounds. Lip synchronous animations can be produced by using the phoneme morphs



4.5.3 Mimic

Lipsinc's Mimic is an add-on to Poser 3 and 4 that automatically create a lip-synching animated pose file for Poser. It provides speech animations by synchronizing the standard head morph parameters of Poser models with speech audio files. Imagine it, if we input some sound file "Mimic -- an affordable and effective lip-synching solution."



Figure 4.6: Mimic can be used to generate speech animation for Poser

The image above shows the working window. The top section is where the speech can be added as a .Wav file. Below that you can type in the text version of your speech and tell it if you want facial movement. With the text version being slightly better. Comma is used to separate actions. Also within the text section are 4 boxes, leaving these checked will make the figure blink, move his/her head, eyebrows and eyes. The program seems to know just when to put in a raised eyebrow or a flick of the head. You can also tell Mimic to use fewer keys and change the frame rate to match the one your using in Poser. The bottom section is simply the path where you want to store your pose file and below that the final button "Generate pose" and a launch Poser button.



Bring in the .wav file using the **Sound** button. The wave pattern of your voice file should now be visible. You can listen to the sound file using the **Playback** controls. You could now process your sound file, but you can get better accuracy by including text of the voice. Check the **Use Text** box and you can either bring in a text (.txt) file or type the text into the text dialog box. You can direct the output anywhere you want with the output dialog, but we recommend leaving the default. This way the Poser program can find the output file as you will see in the next step. Now press the **Generate Pose File** button and you are done. If you ever tried to manually do a lip synch, you will soon realize how powerful this Mimic program is! Once Poser is running and a figure is in the scene it's just a simple matter of applying the mimic pose file. This won't adversely affect any animation you created on the figure body; it will only change the head section. Next click the play button and the figure is talking complete with the .Wav file. Now select **Library>Poses>Mimic** and the pose file just created in the previous step. Answer **Yes** to the 'Do you wish to add frames...' popup. We can see that this text produces a 22 frame poses.

Chapter 5 Parsing

Our eventual goal for the MMS system is to allow users to express requests in natural conversational English, without any need to learn a specialized command language. The system should be able to extract information from them and generate the corresponding responses. The dialogue manager should act as the brain of the system, deciding how to respond to the customer requests. Because the limitation of the speech recognition technology the customer has to use text input in the prototypes in place of using speech input. The analysis of the customer prompts remains the same; only the speech recognition is left out. But in the way we can use the parsing method provided by the speech recognition software; we have to do the parsing by ourselves. We have been tried a few programming languages to write a parser and in the following sections we'll discuss them all.

5.1 A Prolog Parser

In the beginning we used only Visual Basic to build the application. One of the most important functions of the application is to generate response from the customer prompts. It is in fact a reasoning process that uses a lot of rules and data. For a logical programming language a reasoning application is relatively easy implemented. But in Visual Basic (or Delphi) the reasoning module looks like spaghetti with complex if-then-else statements because of the lack of procedure in the rules. The customer can say anything in any order. The customer prompt can be a question, an order, a conformation or something else. There are also so many menu items. To find the right response we need to compare the customer prompt with these data one by one. For example there is a simple customer prompt "I want a hamburger." which matches the rule "prompt= <order> <number> <burger>". But the Visual Basic is not rule-based programming language. We need to compare "I want" with all data to know it is a value of <order> and we need to do the same to "a" and "hamburger". There will be many if-then-else statements and for-loops. This method is not only inefficient but also not flexible. If we want to add a new menu item or add a new value to <order>, we need to change the code and rebuild the application. Of course we can put the data in a grammar file or a database, but then we need to read all the data into the array variables at the beginning each time the application runs. So we decided to find out whether we could make use of a logical programming language. And we find the Amzi! Prolog software that can work with Visual Basic.

PROLOG stands for PROgramming in LOGic. It has been used for applications such as expert systems, natural language, and intelligent databases, different from conventional procedural programming. The expressiveness of Prolog is due to three major features of the language: rule-based programming, built-in pattern matching, and backtracking execution. The rule-based programming allows the program code to be written in a form that is more declarative than procedural. This is made possible by the built-in pattern matching and backtracking, which automatically provide for the flow of control in the program. Together these features make it possible to elegantly implement many types of expert systems. Prolog originated from attempts to use logic to express grammar rules and formalize the parsing process. Prolog has special syntax rules that are called definite clause grammars (DCG). DCGs are a generalization of context free grammars. A context free grammar is a set of rules of the form: "sentence \rightarrow nounphrase, verbphrase", where nonterminal is a nonterminal and body is a sequence of one or more items. Each item is either a nonterminal symbol or a sequence of terminal symbols. The meaning of the rule is that the body is a possible form for an object of type nonterminal. Definite clause grammar has been used extensively for the development of complex natural language understanding programs. Prolog has the capacity to load definite clause grammar rules (DCG rules) and automatically convert them to Prolog parsing rules. Here is an example:

BNF rule:

```
GIVE_ORDER ::= SUBJ ORDER | ORDER PLEASE
```

DCG rule

```
give_order([X|Y]) --> subj(X),order(Y).
```

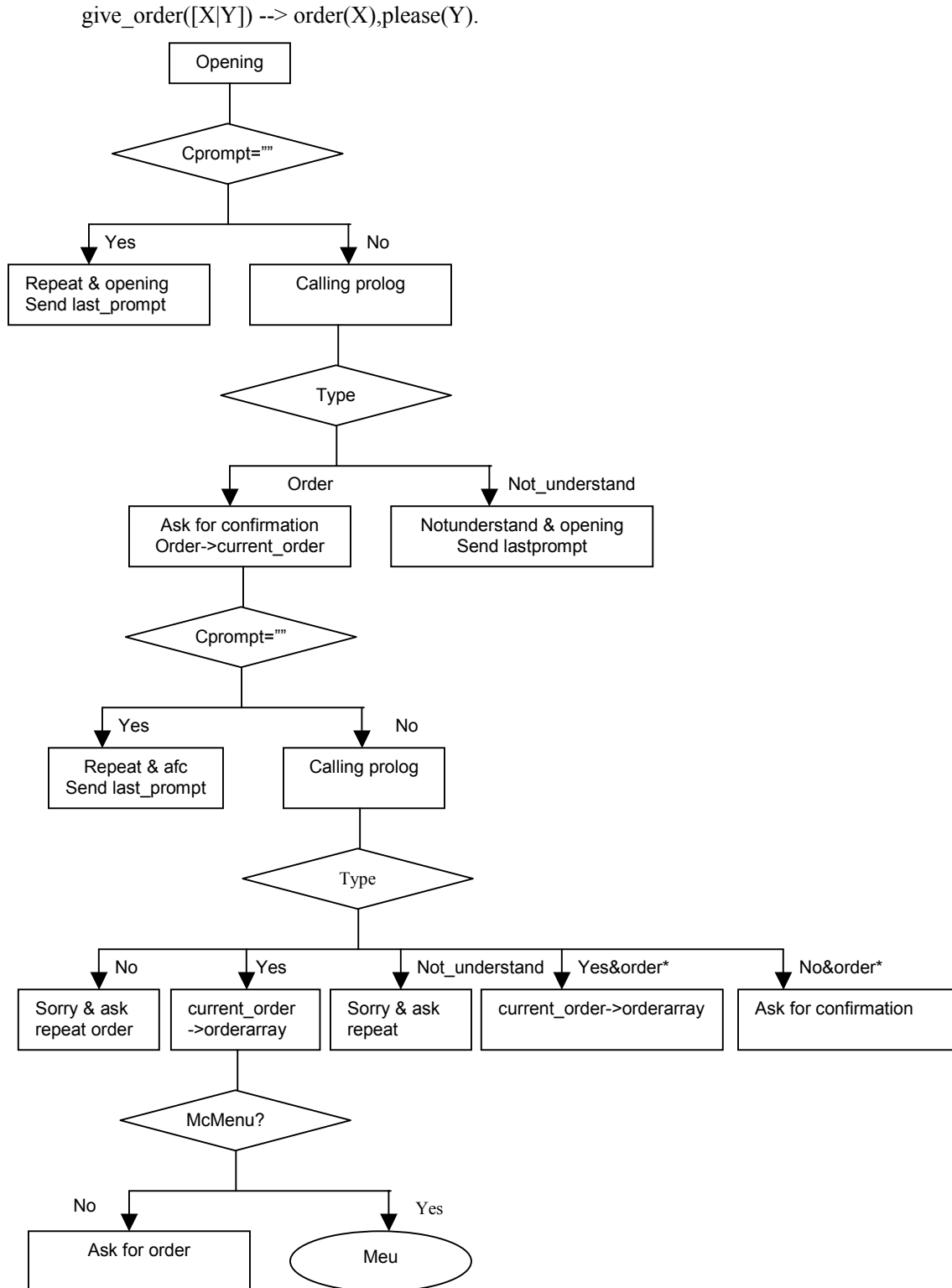


Figure 5.1: The parsing process of VB + Amzi! Prolog

We used the following tools to build the client-server McDrive system. Visual Basic (or Delphi) is used for the GUI (Graphical user interface); Amzi! Prolog was used to define the rules for analyzing

the customer prompt and Microsoft Access was used for the database of rules and data. With the help of Amzi! Prolog + Logic Server we could integrate the rule-based components in our VB application. And in this way the update of the application would be easier. The server part of the McDrive system was composed of three components: the Visual Basic user interface, the Access database of data, and the Amzi! Prolog logic base contains rules that define relationships based on the database. Visual basic user interface can get easy access to the Prolog logic base through Logic Server API and the Prolog rules can reason directly over the database tables because of the Logic Server ODBC extension.

The flowchart above describes the process of the parsing and the interaction between VB interface and Amzi! prolog base. We use the Dialog Manager (DM) to control the conversation process. It decides what the state of the server is and when to call the Amzi! prolog program. The beginning state of server is Opening and the server sends welcome message and asks for order. Depending on the response the prolog program will be called or the server enter the repeat opening state and send the client the last prompt.

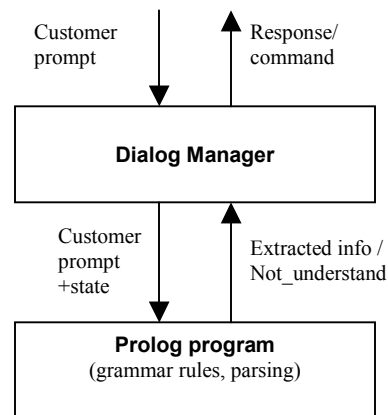


Figure 5.2: Communication between DM and prolog program

In the prolog program the customer prompt is defined by the definite clause grammar. If the customer prompt is recognized, the prolog program will extract useful information further and return them to the DM; otherwise “type not_understand” will be returned. In the DM we use variables as “state”, “repeat”, “friteordered” and so on to express the context. In the different context we will call different rules of prolog program.

5.2 XML Parser

Later we decided The Extensible Markup Language (XML) would be a better choice for parsing. XML is the universal format of structured documents and data on the web. It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification. XML allows the flexible development of user-defined document types. It provides a robust, non-proprietary, persistent, and verifiable file format for the storage and transmission of text and data both on and off the Web.

XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language. A data object is an XML document if it is well formed and a well-formed XML document may in addition be valid if it meets certain further constraints. XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure. The

Document Type Definition (DTD) can be used to describe the markup (the elements and other constructs). DTD defines the valid syntax of a class of XML documents. That is, it lists a number of element names, which elements can appear in combination with other ones, what attributes are available for each element type, etc. A DTD uses a different syntax (namely Extended Backus Naurus Form) from that used by XML documents. However, the design and construction of a DTD can be complex and non-trivial, so XML also lets you work without a DTD. DTDless operation means you can invent markup without having to define it formally, provided you stick to the rules of XML syntax. To make this work, a DTDless file is assumed to define its own markup by the existence and location of elements where you create them. When an XML application encounters a DTDless file, it builds its internal model of the document structure while it reads it, because it has no DTD to tell it what to expect. There must therefore be no surprises or ambiguous syntax: the document must be 'well-formed' (must follow the rules).

Today there are two widely accepted APIs for working with XML: the Simple API for XML (SAX) and the Document Object Model (DOM). Both APIs define a set of abstract programmatic interfaces that model the XML Information Set (Infoset). The DOM models the Infoset through a hierarchy of generic nodes that support well-defined interfaces. Due to the DOM's tree-based model, most implementations demand that the entire XML document be contained in memory while processing. SAX, on the other hand, models the Infoset through a linear sequence of well-known method calls. Because SAX doesn't demand resources for an in-memory representation of the document, it's a lightweight alternative to the DOM.

We need to convert the input string to the XML string before we can use the XML parser. The input string contains useful information like “one”, “BigMac” and so on that we want to extract and add to the XML string. We don't care about the other stuff such as “Ik wil graag”, “keer”. We use the pattern match and replace of regular expression to do this. We write for every type menu a pattern and then check whether the pattern appears in the input. If so we replace the elements of the matched part by adding the tags such as <number></number> and add them to the desired XML string, the matched part will be deleted from the input string, and then the next pattern will be checked in the remained string. Otherwise we check direct the next pattern.

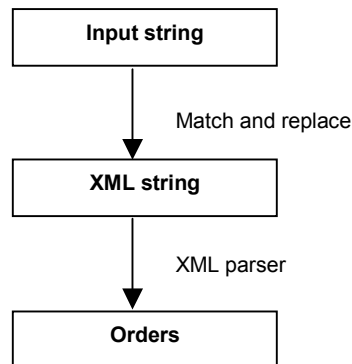


Figure 5.3: How an input string can be converted to an order

For example there is an input string “I would like to have four times BigMac, one fries with double fries sauce and three small milkshake vanilla.” The XML string will be as follows:

```

<orders>
  <sandwiches>
    <menu>BigMac</menu>
    <number>four</number>
  </sandwiches>
  <fries>
    <menu>fries</menu>
    <number>one</number>
    <size></size>
  
```

```

        <f_saus>
            <menu>fries sauce</menu>
            <number>double</number>
        </f_saus>
    </fries>
    <shake>
        <menu>milkshake</menu>
        <number>three</number>
        <size>small</size>
        <m_flavor>vanilla</m_flavor>
    </shake>
</orders>

```

But unfortunately we still have to face the parsing problem. The input of the XML parser has to be formatted. It can construct the order that will be transferred to the kitchen, but it cannot be used for parsing.

5.3 VB Parser

We have to go back to Visual Basic and try to analyse the customer prompt without outside help. This time we make use of RegExp. The result will be added to a XML string.

```

Private Sub Dialog_Initiate()
    ... ..
    sandwiches = "(BigMac|Quarter Pounder|McChicken|Groenteburger|Fish'Filet" & _
        "|Cheeseburger|Hamburger|McNuggetsKip|McDeluxe|Mac Deluxe)"
    numbers = "(een|twee|drie|vier|vijf|zes|zeven|acht|negen|dubbele|twintig)"
    stuk = "(stuk |stuks |maal |keer |kop |kopje)?"
    ... ..
End Sub

Function analyse_order() As Boolean
    stranswer = customer_prompt
    Set objRegExp = New RegExp
    objRegExp.Global = True
    objRegExp.IgnoreCase = True
    res = res + "<order>"
    check_supermenu 'check all the menu categories to find out if there is a match
    check_sandwiches
    check_frisdrank
    ....
    If order_number > 0 Then
        analyse_order = True
    End If
    res = res + "</order> " ' added to the XML string
End Function

'check sandwiches
'sandwiches_rule = [number], [stuk], sandwiches
'          1          3
'number and sandwiches occupies field 1 and 3; only these two fields matter

Private Sub check_sandwiches()
    Dim sandwiches_rule As String
    sandwiches_rule = numbers & "?\s?" & stuk & sandwiches ' numbers optional
    objRegExp.Pattern = sandwiches_rule
    Set matches = objRegExp.Execute(stranswer)
    order_number = order_number + matches.Count
    If matches.Count > 0 Then
        For Each objmatch In matches
            res = res + objRegExp.Replace(objmatch, "<sandwiches><name>$3</name> &_"

```

```
        <number>$1</number></sandwiches>") & vbCrLf 'added to XML string
    stranswer = objRegExp.Replace(stranswer, "")
Next
End If
End Sub
```

Chapter 6 Test of OZ



We use the Wizard of Oz technique (WOz) to test our design of the multimodal McDrive system. The basic idea of a WOz system is the modeling of a system or system behaviour which is not yet or only partly available by a human (the hidden "wizard") and to hide this fact from the user. By analysing the performed operations, the user's needs can be identified in advance that may lead to a better design of the final system.

In WOZ experiments, users believe they are interacting directly with the implemented McDrive system, but in fact a human "wizard" intercepts the user's commands and causes the system to produce the appropriate output. In this experiment there is no voice input, only the keyboard input. The user and the human "wizard" sit both behind a computer in two rooms. These two computers are connected by Internet or intranet. On the bottom of both screens there is a special designed keyboard that they can use to make sentences. On this menu keyboard there are the icons of menu items and of the minimal prompt set. The human "wizard" welcomes the user by clicking the icon welcome. On left side of the upper part of both screen a welcome video plays, and at the same time a wizard appears on the right side and says "Good morning / afternoon / evening. This is the test system of Auto-Mac. Please give your order after the bell." And then the user can give his/her order by clicking the menu keyboard.

6.1 Simulated Dialogues

6.1.1 Dialogue example 1:

Operator: *Goede avond. Dit is het test systeem voor Auto-Mac. Geeft u bestelling na de piep. De menu items kunt u in de lijst vinden*

- The operator clicks Goede Avond button  and test system button  on the operator keyboard.
- In the prompt part of the operator interface, these two icons appear.



- In the prompt part of the customer interface, because the customer keyboard does not have test system button, so in the prompt part a text memo and an icon appears.



- In the graphical part of the customer interface, first the flash movie Opening displays and then the MsAgent Character Genie flies to the center and make a deep bow and says “Hello, I am Genie”.

Customer: *Goede avond. Ik wil graag een Cheeseburger Menu.*

- The customer clicks in order on Goede avond button, IK button, een button and Cheeseburger menu button of the customer keyboard.
- The old operator prompts (text) are writing to the history window on both sides.
- In the prompt part of the customer interface, a row of icons appears.



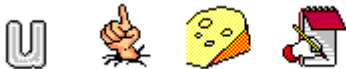
- In the prompt part of the operator interface, a row of text and icon appears.



- In the processing part of operator interface, Genie speaks.
- In the processing part of customer interface, Genie listens.

Operator: *U heeft een Cheeseburger besteld, klopt het?*

- Genie tells the operator to confirm the order.
- The operator clicks on U button, een button, Cheeseburger button, and besteld button.
- In the prompt part of operator interface, a row of icons appears



- In the prompt part of customer interface,



- In the graphical part of customer interface, a photo of cheesburger appears. A jumping question mark can be added.

Customer: *Ja.*

- Genie tells the customer to only answer yes or no.
- Customer clicks Ja button
- In the prompt part of the customer interface



- In the prompt part of the operator interface, text Ja appears.

Operator: *Wilt u frietsaus hebben bij uw friet?*

- Genie reminds the operator to ask the customer if he wants fries sauce.
- The operator clicks Fries, and then clicks plusmin button.
- In the prompt part of the operator interface, icon plusmin appears.



- In the prompt part of the customer interface, text “Wilt u frietsaus hebben bij uw friet?” appears.

Customer: *Ja.*

- Genie tells the customer to only answer yes or no.

- Customer clicks Ja button
- In the prompt part of the customer interface



- In the prompt part of the operator interface, text Ja appears.

Operator: *Welke frietsaus wil u hebben?*

- Genie tells the operator to ask the customer what kind of fritesaus he wants.
- The operator clicks Fries and then clicks whatsaus button.
- In the prompt part of the operator interface, icon whatsaus appears.



- In the prompt part of the customer interface text “Welke fritesaus wil u hebben?” appears.
- In the graphical part of the customer face, the name of the fritesaus twinkles.

Customer: *Ketchup, alstublieft.*

- Genie tells the customer to only choose from the sauce.
- Customer clicks on Ketchup button and then alstublieft button.
- In the prompt part of the customer interface, a row of icons appears
- In the prompt part of the operator interface, an icon and a text label appears.

Operator: *U heeft een ketchup besteld, klopt het?*

....

Customer: *Ja.*

...

Operator: *U krijgt drank bij uw menu. Wat wilt u drinken?*

- Genie tells the operator to ask what kind of drink the customer wants.
- Operator clicks on whatdrink button
- In the prompt part of the operator interface, a whatdrink icon appears
- In the prompt part of the customer interface, a text label appears.
- In the graphical part of the customer interface, a cheeseburger menu appears, and at the side of a jumping beker a question mark twinkles. And at the bottom all the drink names appears too.

Customer: *Een Coco-cola, alstulieft.*

- Genie tells the customer to only choose from the drinks.
- Customer clicks on een button, coco-cola button and alstublieft button.
- In the prompt part of the customer interface, a row of icons appears
- In the graphical part of the customer interface, a coco-cola appears.
- In the prompt part of the operator interface, two icons and a text label appears.

Operator: *U heeft een coco-cola besteld, klopt het?*

...

Customer: *Ja.*

...

Operator: *Wilt u een klein, middel of groot beker hebben?*

- Genie tells the operator to ask the beker size.

- ...
- In the graphical part of customer interface, a little beker, a middle beker and a big beker appears in order. Then a hand moves to point them in order and at the same time on the top a question mark twinkles and moves with the hand. It can also be Genie point to the bekers.



Customer: Groot, alstublieft.

- ...

Operator: Zo is uw bestelling compleet?

- ...

- In the graphical part of customer interface, all the menu items that the customer ordered appear.

Customer: Ja.

Operator: Bedankt voor uw bestelling. Het is... euros. Dan mag u doorrijden naar het tweede loket.

- ...

- In the graphical part of the customer interface the flash movie ending plays.



Text/speech	Operator Graphics	Wizard	Smiling face	Customer Text
Good morning, what do you want to order?				
				Hello, I want to two BigMac Menus.
You want two BigMac menus, is it right?				
				Yes.
What kind of fries sauce do you want?				
				Ketchup, please.
You get a drink with your menu. What would like to drink?				
				Cola, please.
What size do you want?				
				Medium, please
It is ... euros; please drive to the first window.			 	

Figure 6.1: The timeline of a dialogue example

Part II. Design and Implementation

Chapter 7 Structure of Model

In this part we are focusing on the design of the implementation of the manual prototype. The brainstorming process is also described so that the reader can have a picture how the prototype is build step by step.

7.1 Communication between Operator and Customer Interface

For the manual prototype there will be two interfaces built: the operator- and customer- interface. The Server and the Client. The Server and the Client are connected through the Internet. In this version there will only be a one-to-one relationship between the Server and the Client. In the future we hope that one Server can deal with more Clients.

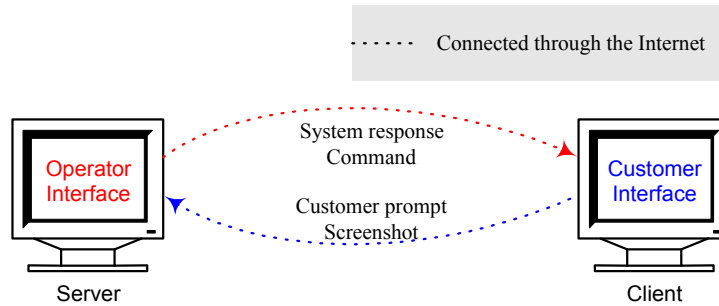


Figure 7.1: The communication between the Server and the Client

Every computer connected by the network has an own IP (Internet Protocol) address. In order to communicate with others we need to know the IP address of the Server. The Server application will provide this address and the Client application will keep it in its configuration file. If the Server application is moved to another computer then we only need to find out what the new IP is and change the customer configuration file. LANs (Local area networks) are privately owned networks within a single building or campus of up to a few kilometres in size. LANs are widely used to connect personal computers and workstations in company offices and factories to share resources (e.g., printers) and exchange information. The Internet means a specific worldwide Internet that is widely used to connect universities, government offices, companies, and of late, private individuals.

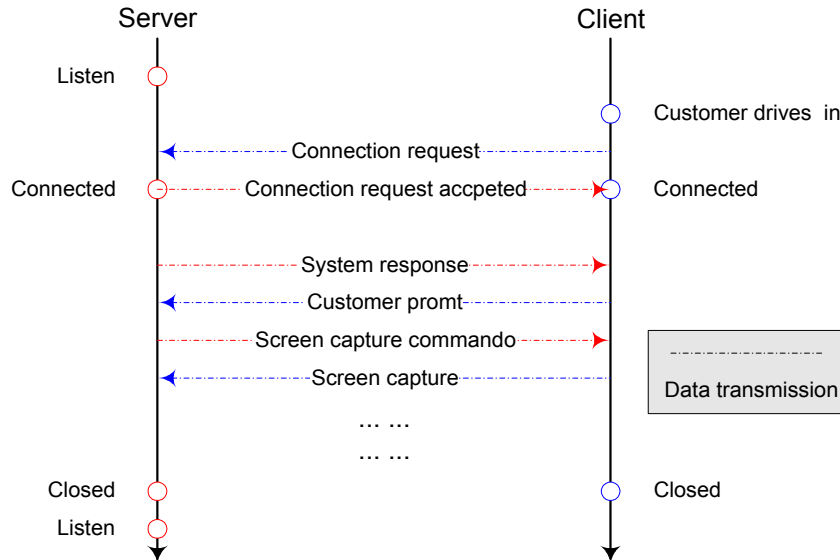


Figure 7.2: The communication between the Server and the Client

7.2 Operator Interface

The Operator Interface is used to generate the system response and control the dialogue. In the manual prototype a special keyboard has to be build to generate the system response. This keyboard can produce not only the text feedback but also the graphical feedback. The operator can also make the wizard show the right facial expression at the right moment with the help of this keyboard. The operator will sit behind the Server and cannot see the customer and the customer interface. He has only the customer prompt to tell whether the order process is normal or something is wrong. So we want the operator to be able to view the customer interface. We don't want to put a video camera in the front of the Client. So the screen capture of the customer interface can give us a clue what's going on there.

After the customer orders the normal operator will transfer the order to kitchen. What has the customer ordered? We cannot transfer the whole dialogue to the kitchen and ask them to read it and find out what the order is. The order should be automatically generated at the end of the dialogue. In the manual prototype the operator has to enter the order by himself so for this purpose we have an order form. In the manual prototype the order process takes more time than normal because the operator have to click the buttons to generate the response. So we have a dialogue window to help the operator remember what have been said. We try to use as more as possible windows. The user can always close or open these windows. It gives the user more flexibility.

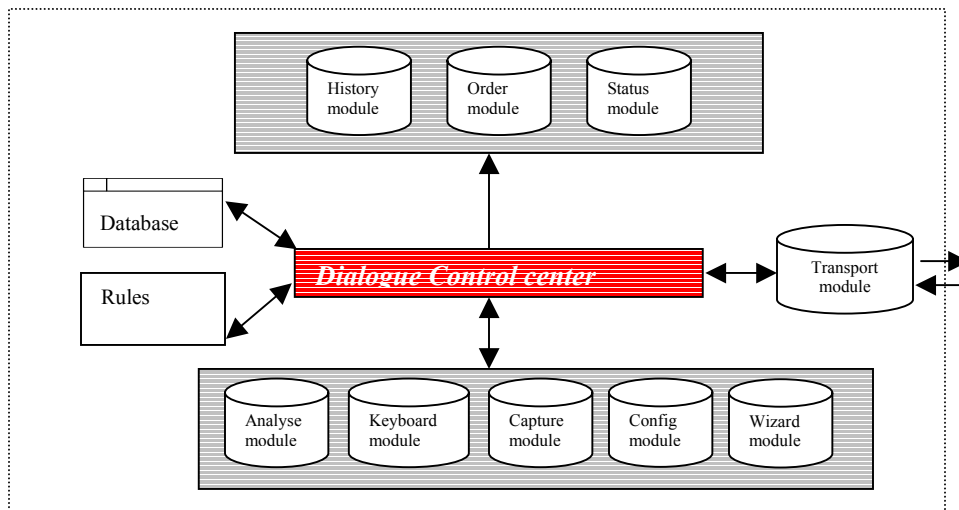


Figure 7.3: Structure of Operator Interface

7.3 Customer Interface

The customer Interface shows the customer the system feedback. There will be text feedback, pictures of menu items, movie over the ordering, smiley faces. And a wizard will change her facial expression according the circumstances. At he beginning we wanted to build a customer keyboard too. But later we thought a text input window is enough and text input is more similar with the speech input. All these feedback needs to be shown on the same screen at the same time. So we split the screen into 4 parts and each part is an individual. When a code sequence of the system response is received the customer interface searches the database McDrive and shows the results on the screen. After all the feedbacks are shown a textbox pops up and the customer can type in the new request or answer. The following figure is an design of the customer interface.

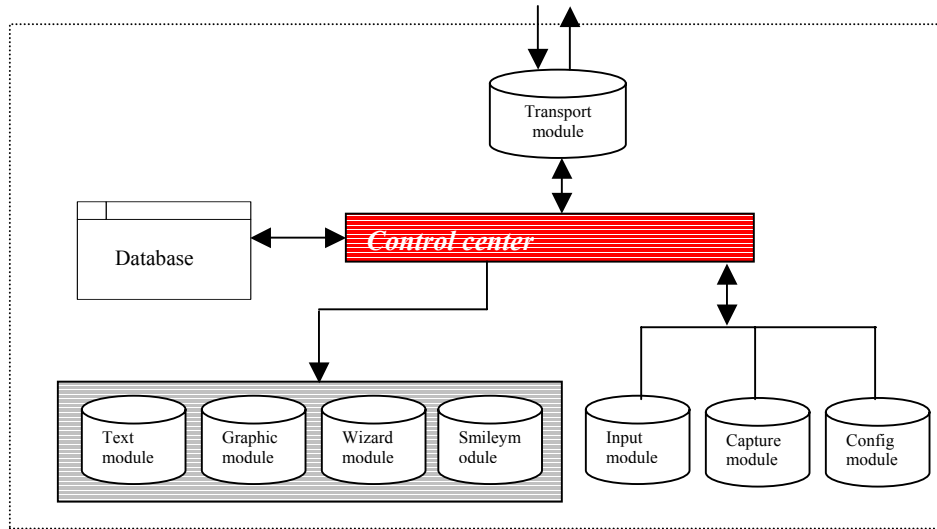


Figure 7.4: Structure of Customer Interface

Chapter 8 Data Management

In the first place let us see what kind of data do we have? In the chapter 2 Case study we have analysed the available dialogues and got a minimal set of operator prompts. This set will be used as a basis for the constructing of the system response. We can express ourselves using different words and sentences; the result is almost the same. For example “Good morning” and “Hello” are all can be used to greet someone, we say “Hello” is an alias of “Good morning”. We want the system response can be more flexible, not like a robot can only say, “Yes sir, No sir.” So beside the minimal set a supplementary set that contains all kinds of alias of the prompts in the minimal set is necessary. We call this set Alias. The MMS have also to behave like a human, therefore we build a wizard with different expressions and smileys. Now the speech recognition works not so good. If we know what the customer wants to say then it will be easily recognized. Almost everybody has his own favourite menu and many of them always order the same menu. As we know most people won’t change their cars for a few years so the license plate will also not change. If we keep the license plates and the corresponding orders, then when customer arrives we can find out what the common order of this customer is by comparing the car plates.

8.1 Where to Store?

We’ll use two databases: one is used for the customer information the other is for the McDrive information. All the McDrive sell the same McDrive menu, no matter it is located in Rotterdam or Amsterdam. But each McDrive has its own customer group. A customer who lives in Rotterdam maybe won’t visit a McDrive in Amsterdam in his whole life. We can say that the customer information is localised and we’ll use a local database Customer for every McDrive, which stores the license plate and order of the frequent visitors. Database Customer will be discussed in the last section. Here we’ll focus on the more regular information such as those of menu items. Therefore a central database will be used.

The data that we have are of different format. There are picture of menus, flash movies, information about menu items, wizard action and etc. How do we deal with them? If we put these data direct in the program code, the program will lose its flexibility. In that case everything is fixed, we cannot make a single change without editing the program codes. No, that is not what we want. Our system needs to be a dynamic program where the system manager can update these data at any time. In this way the operator keyboard can be built dynamically, otherwise we can only have a pre-built keyboard and any changes will not be reflected on the keyboard. The pictures shown to the customers will also be out-of-date. The solution is to store all these data in a database and they’ll be captured from this database when they are needed. For example when the customer orders a BigMac Menu the database will be searched to find the picture of the menu item. *The question is that do we need a multi-modal database?* A multi-modal database is just like a multimedia database. There are some type database can be used to store graphic stuffs. *Jasmine* is a real multimedia database but unfortunately it is commercial. **Microsoft Access** and **SQL Server** can both store pictures. We choose Microsoft Access because it is simpler than SQL Server and easier to maintain. SQL Server is often used for big programs with huge quantity records. So for us Microsoft Access is a better option.

From the first section we can see we can divide the data into two categories according their stability: regular and nonn-regular data. The minimal operator set and its supplementary set Alias belong to the first category. The smileys and wizard actions are also the regular data. The customer information is not regular. Some customers only passed by and won’t come back in a few years. Such information needs to be updated frequently otherwise the database will be huge.

First we have our minimal set of operator prompts. The prompts in this set are not the same. Prompt “Bigmac Menu” is a menu item and has a picture, but prompt “and” is a connection word and there is

no picture for it. So we want to divide this set further into the following 3 subsets and each subset has its own table in database “McDrive”.

- **Menu:** This subset contains all the McDrive Menu items. For example “BigMac” and “Coca-cola”
- **Attribute:** Many McDrive menu items have more than one attribute. This customer would like a small Coca-cola, the other one maybe prefer a large one. We can say menu item Coca-cola has an attribute – Size, and Size can have different values. All the attributes and the possible values belong to this subset.
- **Commando:** We put all the other prompts in this subset; no matter it is a sentence or just a word. Sentence like “Good morning, this is McDrive test system” is a commando and the connection word “and” is also a commando.

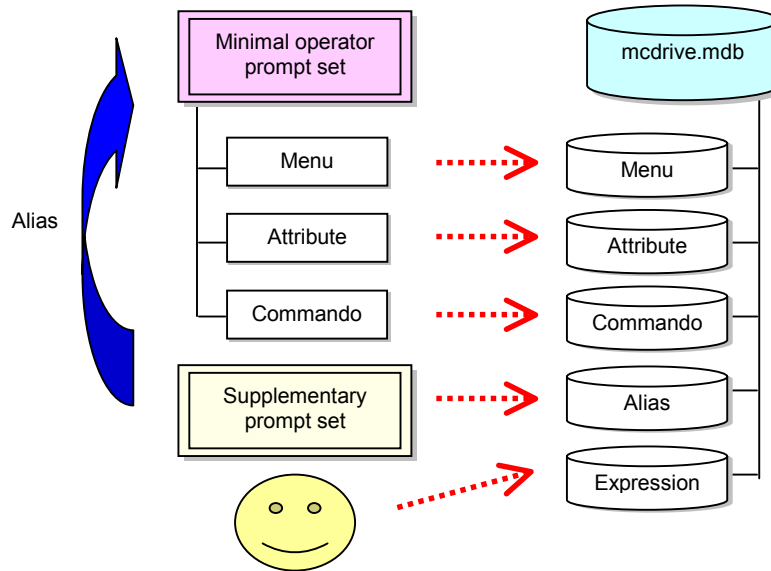


Figure 8.1: The data locations

The reason why we don't put attribute values together with menu items in a table is that otherwise it would be too chaotic. Most menu items have one or two attributes, which in turn also have some possible values. In this way a menu item will have a number of entries that is equal to the multiplication of the value quantities. Let's take French Fries as an example; this menu item has two attributes-size and sauce. Size can be small, medium and large. There are three kind sauces available. So if we put menu item and attribute together there will be nine entries for French Fries necessary. The supplementary set is put in an apart table named “Alias”. In this set there are no subsets. There is also a table named Expression for the smileys and the corresponding wizard actions.

At the beginning we used Data Access Object (DAO), but later it seems not to be a very good choice. Our application is a server-client application; there is a possibility that remote database access is required. We have two choices: Remote Data Object (RDO) or ActiveX Data Objects (ADO). ADO was introduced in 1996 by Microsoft to be a high-level interface to provide ease access to data stored in a wide variety of database sources. ADO can be used with a variety of programming languages including Visual Basic. And Microsoft will not develop RDO any more so we choose ADO.

8.2 Data Coding

Most information is stored in a database and captured from it when needed. This requires a lot of inquiry work of database. The operator generates response by clicking the keyboard and the message will be sent to the customer interface. But there is a problem, namely the database search. For example the operator wants to produce “*You want a BigMac?*” This message contains five parts “*You want*”,

“a”, “BigMac” and “?”. The database has to search these keywords and return the results. It is easy because these words are simple. But what if the operator wants to generate “*What kind of size do you want?*” This message has only one part but it is a long part. As we said in chapter three we made a movie for this question. The movie name has to be captured by search this long message. This is not clean. We want this question to have a code that will be used for capture. Only this code will be sent to the database and movie name can be returned by search this code. We call this code as an identifying code.

8.2.1 EAN 13-code

Nowadays codes are used to identify and/or classify an article or a service in everyday life, it serves not only the war any more. If you buy an article in a supermarket or in a store you can see there is a stripe barcode on the package. In Europe we use a 13 positions code- European Article Numbering (EAN) code. The structure of the EAN 13-code is described in the following figure. The EAN code system is originally developed for the uniform coding of the articles. It is applicable and uniform in the whole world and most important it is unique. The cashier of the supermarket or store needs only to scan the EAN code and the name and price of this article will be shown on the computer screen. Sometimes the producer or importer doesn't have subscribed to the EAN, so their products haven't got an EAN code. The cashiers can't scan the article, what do they do? Type the article name in? No, the name may be long, and there is always input error. Under this circumstance the store has its own code. The cashier has a numeric keyboard to type the code.

Structure of the EAN 13-code.													
Number positions	System code		Article serial number							Control digit			
	2 or 3		9 or 10							1			
Example: EAN 13-code in the Netherlands													
Position	System code		Article serial number							Control digit			
	13	12	11	10	9	8	7	6	5	4	3	2	1
Code	8...7		1	2	3	4	5	6	7	8	9	0	6

Figure 8.2: EAN 13-code

8.2.2 McDrive7-code

Of course we don't use EAN 13-code to identify our Menu items, it is only used as a reference. Our code system is only used in the McDrive system. It is a 7 positions code.

Structure of the McDrive 7-code.							
Number positions	Classify	Item serial number				Control digit	
	1	5				1	
Example: a McDrive Menu code							
Position	Classify	Item serial number				Control digit	
	7	6	5	4	3	2	1
Code	m	0	1	0	1	0	8

Figure 8.3: McDrive 7-code

We call it McDrive 7-code. There are the following rules to obey,

- The positions are numbered from the right (position 1) to the left (position 7);
- The code is numeric except the last position;

- The last position is a letter that classifies the code, for menu code it is “m”, for expression code it is “e” and etc. From this letter we can see what kind of data it is.

The base structure consists of three segments (from left to right: classification – item number - control digit). In the communication there can be some transport errors, how can we know something wrong happened? It depends on the control digit. The item number plus control digit is modulus of 10.

Every menu item, every attribute value, ... has its own code. If the customer orders a small fries with ketchup, there will be a coding sequence that consists three parts. Why don't we distribute every variant a code? There is only one code needed. At first we preferred this way. Strawberry Milkshake and vanilla Milkshake have different codes. But in this way the coding will be too complex. For example the happy meal is the composition of different menu items. These menu items have also attributes, nugget number can be 6,9 or 20; drink can be small, medium or large; fries sauce can be.... There will be too many combinations, which means too much codes. The current 7 position can't be enough. And the keyboard will also be too big. And because the menu items are often not ordered completely, not every people order a menu item in a sentence with all the details. The operator often needs to ask what size you want for your cola? A big one? The code will be changed all the time, although it would be easier for the kitchen. The information should be added step by step. After thinking about this one data one code is better.

8.3 Database Management

The records in the database change continuously. Add a new menu item or use a new picture. Of course we can open the database and update it directly. But the best way is to write a special program to do all these operations. For every table there is a table manager through which the records of this table can be viewed, added, deleted, updated or searched. These table managers deal only with records. We also build a database manager. This manager is global; it is used to show the properties of database and tables. Table and field can be deleted, added or renamed. But that is not of much use. The management of a specific table requires a specific manager. We have to build a new table manager when we add a new table. Unfortunately the program code needs to be changed.

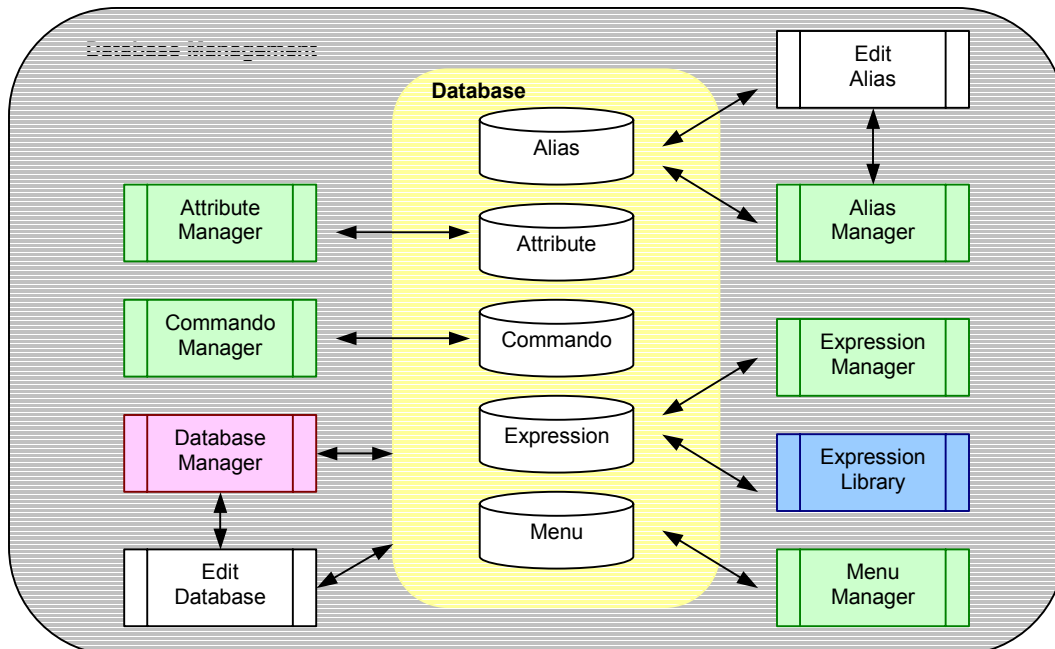


Figure 8.4: Database and its managers

In the following sections we will talk about all the tables and their managers in detail. The relations between them will also be discussed.

8.3.1 Database Manager

The Database Manager controls the database globally. The function of the database manager is shown as follows

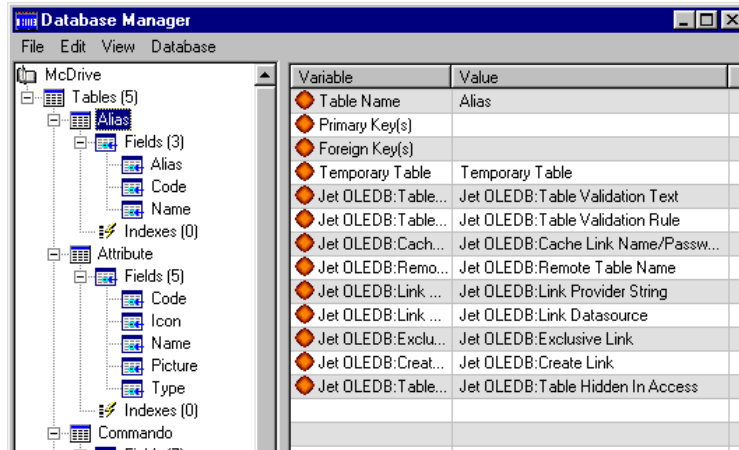


Figure 8.5: Database Manager

- Shows the property of the database and its tables.
- Add a new table.

A new window named “Add a new table” prompt out. The adding action requires us to provide not only the name of the new table but also the name and type of the first field. The new table cannot be empty. It needs to have at least one field.

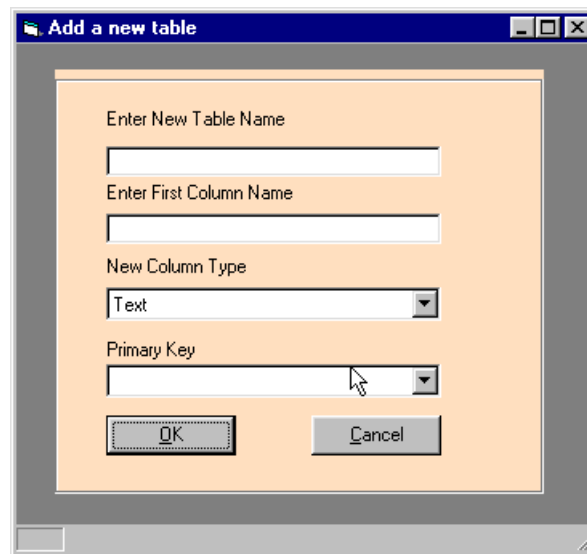


Figure 8.6: Add a new table with the database manager.

- Delete an existing table.
All the records in this table will also be deleted. Before finishing this operation a message box will pop out and ask for confirmation.
- Add a new field to an existing table.

Adding a new field requires us to give field name and type. Fields of Access database cannot be of any type. We only give the types are suited to Access database in the type list.

- *Delete an existing field from an existing table.*
- *Back up the database.*
- *Repair the database from the back up.*

Each table has its over manager. These managers allow user to add, delete, view or update a record. User can also search a record with code or name. Of course user can direct open Access database and then edit them, but we prefer do this work through programs. Some fields have certain values, we provide user with these values so that user need only to choose one of them. User can add new values of these fields too. For example in table Menu field Category have predefined values such as “Supermenu”, “Sandwiches” and so on, user can add a new value “Actionmenu” to this field.

Our designed keyboard has layers. Some menu item has attributes and we would like to have a new layer appeared with these attributes when we click these menu items. So we need to define a button that has simple and complex state. According field Attr_Type we can decide the state. Those menu items that don't have an attribute have an empty Attr_type field and their state are simple. Next question is how do we know which attribute belong to which menu item. Attributes have different types and each type have its own values. In table Attribute we have a field Type. But this field and field Attr_Type is not the same. Because some menu items of different categories can have the same attributes and some menu items have more than one type attributes. For example menu items of category Supermenu have two type attributes: Drink and Fries. Menu items of the same category can also have different attributes. Menu item Coffee of category Drink has attributes “milk” and “sugar”, other menu items of this category don't have such attributes. We can define relations between Attr_type and Type in a procedure, but in this way these relations are fixed and if we want to change them or add new relations we need to change program. We don't want to do this. So we put these relations in Attribute.ini. Those buttons with complex state have a small triangle at the right-bottom of button.

8.3.2 Management of Menu table

Our system is to help the customer to make orders. The information about the menu items is most important. Every menu item has a unique code beginning with “m”. Field “Name” is used to store the default name of menu item. Now the data in the minimal operator prompt set are distributed in the corresponding tables. But these prompts are only texts. The buttons on the keyboard cannot be text buttons because some prompts are too long to fit in. We need something to represent these prompts. Therefore we make for each prompt an icon symbol that the user can know what that is at the first glance. For example we use a milk box icon to represent milk. Beside these symbols we have also pictures of menu items, question movies that need to be stored. As we mentioned before, McDonald's classify their menu items into different categories. An extra field “Category” is therefore added. In a category there can be many menu items but a menu item belongs only to a category. The menu items of the same category can have different attributes. Can we just add an attribute field to store the attribute name or code? No, we can't. Because the number attributes are not fixed, a menu item can have two or more attributes. Of course there is a limit of the number of the attributes. We can add the max number attributes fields. But in the way for most menu items the attributes fields are in majority empty. So we just add an index field – “Attr_Type”. In fact we divide the menu items again despite the classification of McDrive self. According the attributes the menu items are divided into the following sets: Fries, Donut, Happy Meal, Happy Nuggets and etc. A set contains often less menu items than a category. Sometime there is only one item in a set. All the menu items in a set have the same attributes. The field “Attr_Type” contains only the name of the sets. The concrete details over the number and names of attributes are put in a configuration file “attribute.ini”. A donut belongs to the category “Dessert” but its Attr_Type is “Donut”. If a menu item doesn't have any attributes then this field is empty.

The **Menu Manager** is build for the maintenance of the Menu table. First let's see the structure of the Menu Manager. The title of the table is on the top and the status bar is at the bottom. The middle part of the manager is divided to two parts: the *display panel* and the *control panel*. All the fields of Menu table are shown in the display panel. We use textboxes and Combo box for the fields of type Text and picture boxes for those of type LongBinary. When we run Menu Manager the first record will be shown. We can see the textboxes and Combo boxes are grey. There are also no loading buttons for the picture and icon. We call the table is *locked*. These boxes are disabled so that the user can't change the record by clicking them and editing directly. Otherwise the only resume of type something wrong is to input the right information again. The user can add a new record by clicking the *new* button on the control panel and edit a record by clicking the *edit* button. Then those boxes become normal and the loading and cancel buttons for the picture boxes appears also. At the same time the control panel changes too. If the user clicks those boxes without click the edit or new button first then the status bar will show an alert message.

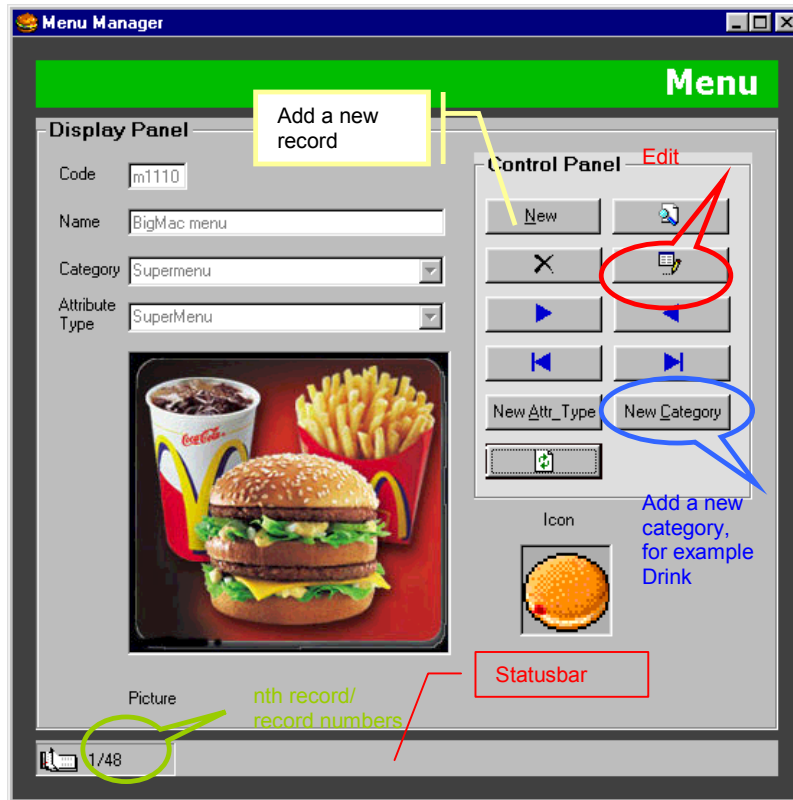
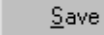
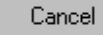


Figure 8.7: Menu Manager with the 1st record

The most important job of the Menu Manager is to edit records. But there are more functions than adding a new record.

- *Add a new record.*

We click the “new” button to add a new record. The database is now in the “adEditAdd” mode. The textboxes and the option combo boxes are unlocked. The textboxes are empty and ready for input. Clicking the arrow on the option box we can see a drop down menu with many options. These options are predefined on the basis of information that we have at this moment. For example the “Category” option box has the options such as “Sandwiches”, “Salad” and etc. New options can be added by clicking the button “New category”. The picture boxes are also cleared and now the loading and clear buttons are visible. We can load a picture with the load button and clear

the picture with the clear button. The control panel also changes, the buttons such as “search” hide and two new buttons “Save”  and “Cancel”  appear. If we change our mind we can always click the “Cancel” button to give up the adding. The manager will return to the old state and nothing will be saved. Otherwise we can save the record by clicking the “Save” button. But before the record written to the database the record needs to be validated. We use a function “*Validate_Data*” to do this task.

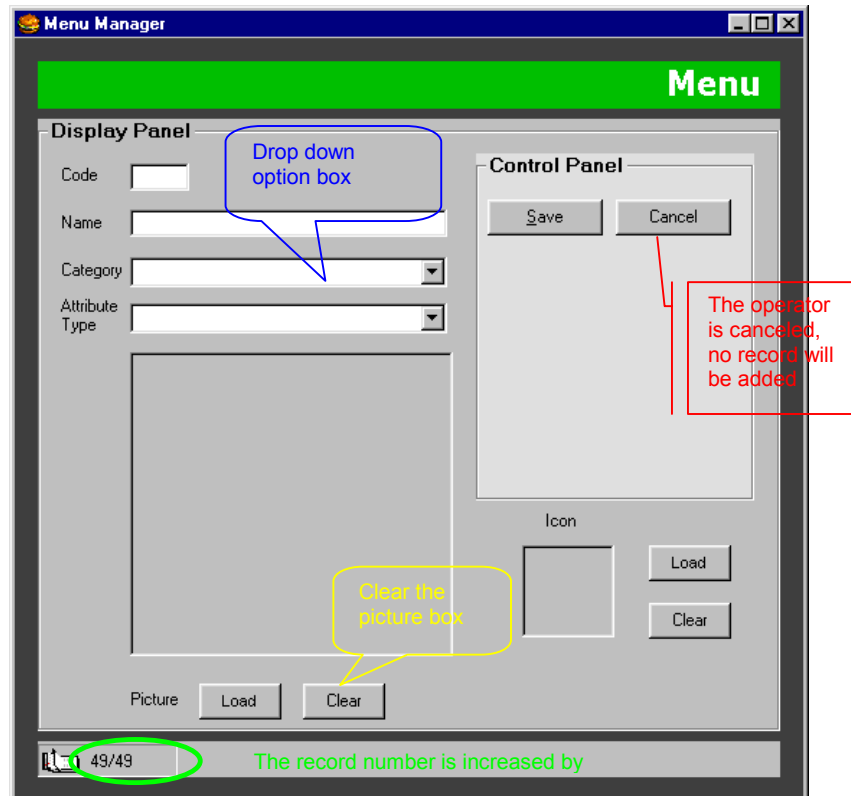


Figure 8.8: Add a new record to Menu table using Menu Manager

1. At the first place the code must be entered. Otherwise the system beeps and an alert message will be shown on the status bar. The focus is moved to this text box.

```

If txtCode.Text = "" Then
    Beep
    Show_State "Error: You need to enter Menu code!"
    txtCode.SetFocus
    Validate_Data = False
    Exit Function
End If

```

[1]

2. And then we’ll check whether the code begins with letter “m” or not. If not the same happens, only this time with a different warning message.

```

If Left(myCode, 1) <> "m" Then
    Beep
    Show_State "Error: Menu code must begin with m!"
    ...
    ...

```

3. Now we need to check the length of the code.


```

If Len(myCode) <> 5 Then
  Beep
  Show_State "Error: Menu code can only have 5 letter/digits!"
  ...
  ...

```

4. We want to add a new record. We need to check if the code of the new record exists or not. We find out by selecting the record with the new code. If there is no record found then the new code is unique. The function “Not_Unique” is as followings,

```


...
Set rstemp = New ADO.DB.Recordset
rstemp.Open "SELECT Code from Menu where Code =" & newCode & "'",
           adoConn, adOpenStatic, adLockOptimistic
If rstemp.RecordCount > 0 Then
  Not_Unique = True
Else
  Not_Unique = False
End If
rstemp.Close
Set rstemp = Nothing
...

```

5. The menu item must have a name.
 6. The menu item belongs to a category, this field cannot be empty.

Now the record is a good record. We can write the record to the database. The manager return to the lock state and the control panel renews.

- *Edit an existing record.*

We can edit an existed record. Click the edit button , the manager unlocks the boxes and display panel is just like adding a new record. Only now the boxes are not empty. All the data can be changed. The cancel operation is the same, the record keep unchanged. Before the changes are saved we need to check if the record still a validate record. The code doesn't need to be changed. After the validation the record is updated and the manager returns to the previous state.

- *View the existing records by moving around*

There are four buttons that used to move around- move forward, move backward, move to the first record and move to the last record. When it is moved to the first or last record a message will be shown on the status bar to tell now it is the beginning or end.

- *Delete an existing record.*

To delete a record we only need to click the “delete” button. Then a message box prompts and asks for confirmation. If the “Ja” button is clicked then this record will be deleted. Otherwise nothing happens.

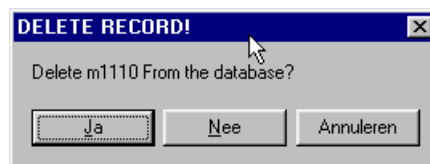


Figure 8.9: Delete a menu record

- *Search a record through the name or code*

Sometimes there are many records in a table. It costs too much time to look them one by one. If we know the code of name of the record we can use searching. If there are such records the first record with this code / name will be shown.

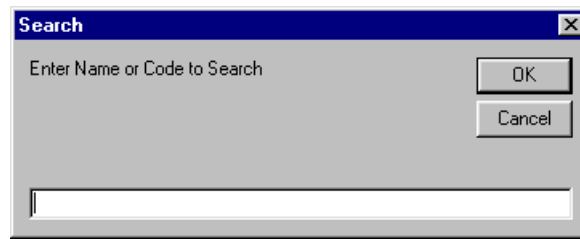


Figure 8.10: Search a record

- *Show the number of records.*

The number of the records and the index of the current record are shown in the left panel of the status bar.

- *Add a new category.*

The menus of McDrive are updated all the time. A new menu item maybe added with new attributes.

This “New Category” `New Category` needs to be checked if it exists already in the option box. If not then it will be added to end of list of the option box.

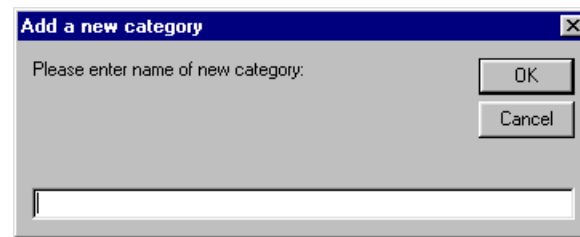


Figure 8.11: Add a new category

```


For i = 0 To cboCat.ListCount - 1
  If LCase(cat) = LCase(cboCat.List(i)) Then
    'stop if exists
    MsgBox "Category exists already!", vbCritical,
      "Add a new category"
  Exit Sub
End If
Next i
i = cboCat.ListCount
cboCat.AddItem attr, i

```

- *Add a new attribute type.*

A new menu item maybe added with new attributes. `New Attr_Type` is used to add a new attribute type. This action is just similar as the action of adding a new category.

- *Refresh.*

Refresh the contents by clicking .

We don't want to have too many keyboards, and just like we already said the attributes of a menu item can be found / **reference** with the help of the configuration file “attribute.ini”, so there is no need for an attribute board. When the user clicks a menu item icon on the menu board, a dynamic reference board will be built. alias, attributes, related question....

8.3.3 Management of Attribute table

This table is used to store the attribute values of the menu items. Remember the records are attribute values not the attribute self. For attribute “Size” there are three entries: small, medium and large. This table functions as a supplementary of the Menu table. It is related to the Menu table through the configuration file “**attribute.ini**”. Attribute table contains details over an attribute value such as name, code and icon. Field “Picture” exists but in the manual prototype it is not of much use. There is another field “Type”. But this field and the field “Attr_Type” of Menu table is not the same. Field “Type” contains the name of the attribute. Maybe we should use “Value” and “Name” as our field name in place of “Name” and “Type”. The operations of the **Attribute Manager** on this table are just like those on Menu table. So we will not discuss them.

The relations between these two fields and the configuration file “attribute.ini” is shown in the following figure. Attr_Type is marked with red and Type with green. The configuration file has a form like this:

```
[Attr_Type]
attribute1=Type
attribute2=Type
... ..
question1=
question2=
... ..
```

The number of attributes and questions are varied. “Fries” has two attributes and “Cola” has only one attributes. So we set both the maximum of attributes and the maximum of questions to 5. There can be at most 5 attributes and 5 questions for an *Attr_Type*. The system will search for the *Attr_Type* and returns the attributes and questions belonging to it.

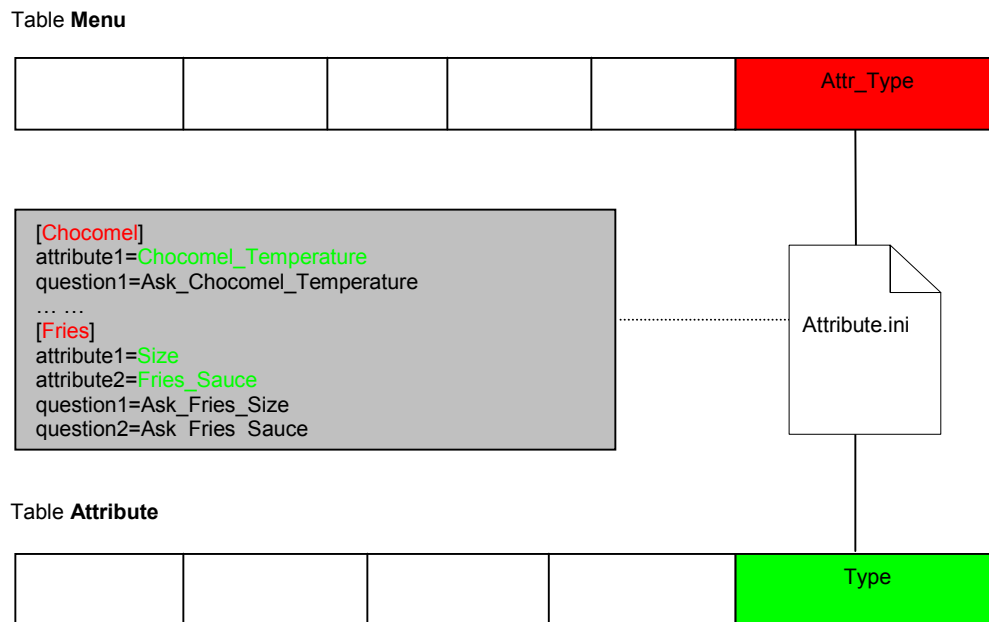


Figure 8.12: Relation between field Attr_type (Menu) and field Type (Attribute)

8.3.4 Management of Alias table

In this table there are three fields: Code, Name and Alias. Every menu item has its own name; of course there are different ways to call a menu item. Everybody has his own habit. For example, someone says always “MacShake” when he orders “Milkshake”. In the menu table only the default names prescribed by McDonald are used. In this case it is “Milkshake”. “MacShake” can be put in the

table Alias. The values of attributes can also have alias. “Small” and “klein” are aliases. The data in the Commando table has the most aliases. Take “Good morning” as example, “Morning”, “Hoi”, “Good day”, “Hi” means all the same. There can be different smiles, so the smiley faces also have alias. In one word the records of all the tables in the McDrive database can have aliases except those of the Alias table self.

The Alias Manager is not like the other table managers. At the top of the manager there is a toolbar with three buttons: Add, Edit and Delete. Initially these buttons are disabled. The middle part is divided into two parts. The left parts show the tables and record names. The right part shows the aliases of the chosen record. Each part has a status bar at the bottom. The left status bar shows the code and name of alias owner. The right status bar shows how many aliases there are. The space of these two parts can be adjusted.

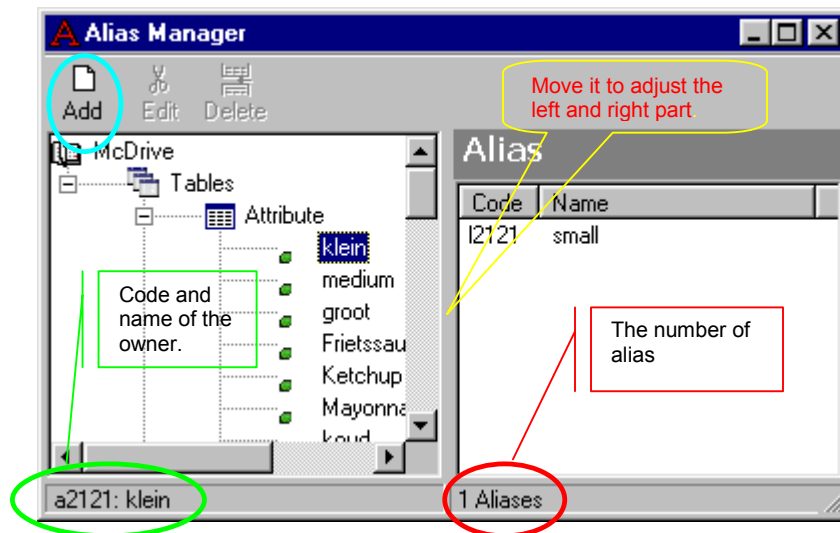


Figure 8.13: The alias manger

- Add a new alias.

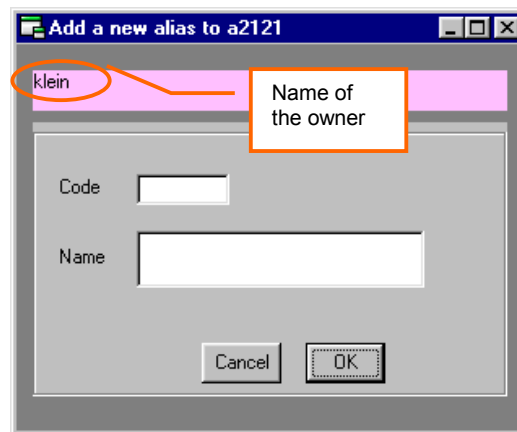


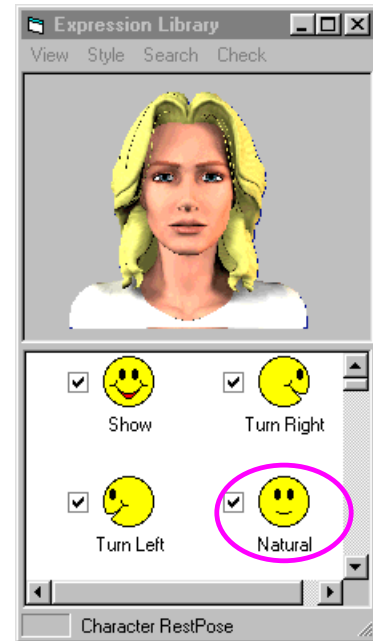
Figure 8.14: Add a new alias

- Edit an alias
- Delete an alias

8.3.5 Management of Expression table

The smiley faces are stored in this table. Each kind of expression has a default face, this smiley will have level one. The other similar smileys can be of level 2 or level 3. There are a lot kind expressions so we assign them with Show property according the frequency of use. The most often used smiley is the smiling smiley and its show Property is set to be true. You cannot imagine an operator cries to help the customer so the crying smiley is little used and its show property will be false. We divide the smileys also into different types on the basis of their use. Wizard is also used to express facial expressions. These facial expressions have a corresponding smileys. For example Wizard can smile; for this action there is a smiling face. Not every smileys has a related wizard action.

Expression Library is used to view expression and the corresponding wizard action if the action is available. We can add an expression by checking it. When the library is closed a message box will appear first to ask user whether changes will be saved to database or not. If user chooses “no”, then these changes will be reflected to keyboard but not saved, next time user will not see these changes. Expression Library provides user also option to check or clear all expressions, or just expression of level I and/or of level II.



The relationship between the Expression table, Expression Library, Expression Keyboard and Wizard is shown in the following figure.

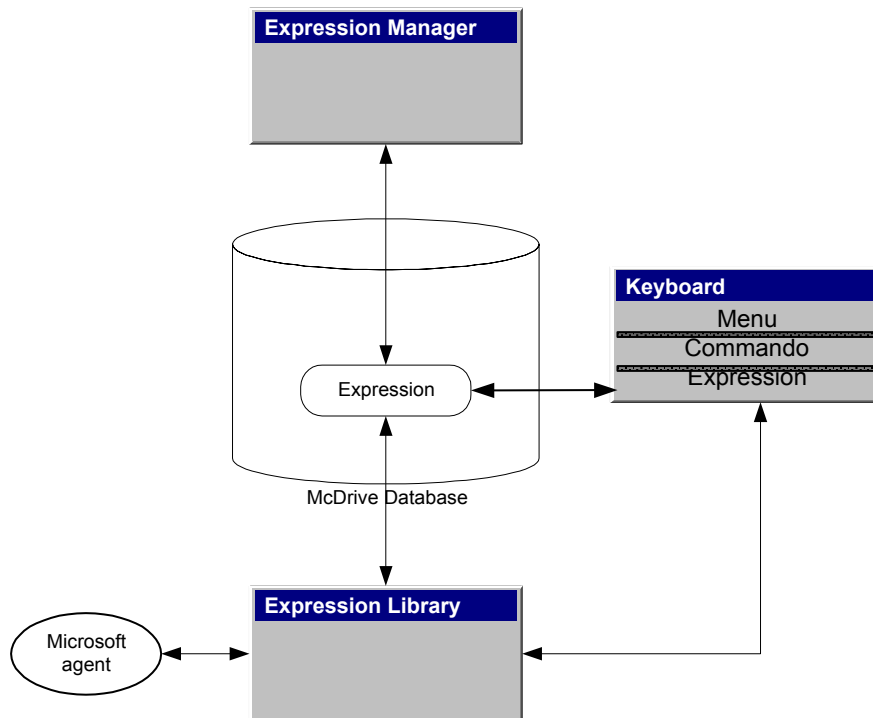


Figure 8.15: Information overflow of Expression

8.3.6 Management of Commando table

In this table there are all kinds prompts stored. It can be a connection word “and” and it also can be a question like “What size do you want?” As we have seen, fries have two attributes, namely “size” and “sauce”. If you order a fries, the operator wants to know extra information such as the size of the fries, what kind of sauces you want? So except the attributes there are some questions bounded to the menu items. We can’t put these questions in the table Menu either. Similar to attributes, we’ll encounter the problems such as how many fields of questions will be needed?

8.4 Database Customer

Some customers give always the same order, some customers give mostly of the time the same order and some others change their order frequently. So here we have a problem. How can we know which order is a fixed order? When a customer gives an order, we

1. Compare the new order with the content of field *Order1*. If they are the same, clear the field *Order2*, and update the field *Date*. Otherwise,
2. Compare the new order with the content of field *Order2*. If they are the same, clear the field *Order2*, put the new order in the field *Order1*, and updates the field *Date*. Otherwise,
3. Move the content of field *Order1* to field *Order2*, put the new order in the field *Order1*, and update the field *Date*.

We keep only the recent records, if a customer has not visited for 1 month, then the record will be deleted. Everyday we check the database and delete the out-of-date records. In this way we can keep the database compact.

Chapter 9 The Design of Operator Keyboard

In the manual and semi-automated prototypes the operator interface needs a special keyboard to generate the system multimodal prompts. These keyboards are built according to the BNF of the minimal prompt sets. The difference of these two keyboards is that the keyboard of the semi-automated prototype is a minimal keyboard. Only the necessary buttons are dynamically generated according with the current environments and conditions.

9.1 First Design -- Text keyboard

At first we made a text keyboard that can only generate the text response. What the customers see is only the text output. There will be no pictures, smiley faces, movies, or wizard. The purpose to build such a text keyboard is to test whether the system can produce the right response and the communication between the operator and customer is good or not. The keyboard is built on the basis of the dialogue analysis so it needs to contain a minimal prompt set. In the keyboard there are the following functions,

- Additional information
- Additional order
- Confirm
- Complete
- Misunderstanding
- Verification

Because with every sentence one of the functions will be used, we put them in the top of the keyboard. Over the placing of menu items there are a few options.

9.1.1 Option 1

For the menu items we list all of them on the keyboard so that it takes shorted time to look for the menu items. But there are too many of them even they are divided into different area according the menu categories of McDonald website. The attributes Number and Size are used frequently so they are added directly to the keyboard. Because the other parameters such as Salad dressing, Toppings and etc. are applied only for certain menu items, they will be displayed in submenus.

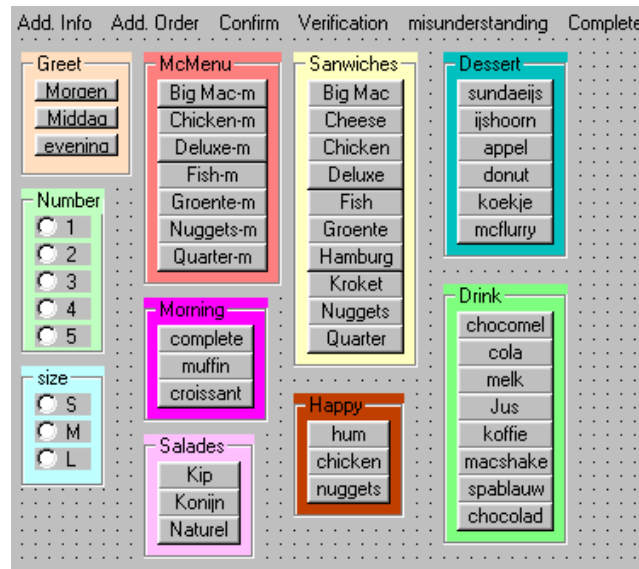


Figure 9.1: The first design of a text keyboard.

9.1.2 Option 2

In the previous design there is not much space left so this time we are trying to make the keyboard more compact through a layered approach. We use 5 drop down menu's that are located on the top of the keyboard: Dialogue, Number, Size, Menu and Functions. To open a new dialogue, just click menu *Dialogue* and choose *Morning* or *Afternoon* or *Evening* according to the time. The main part of the keyboard is used to display the attributes options of the menu items.

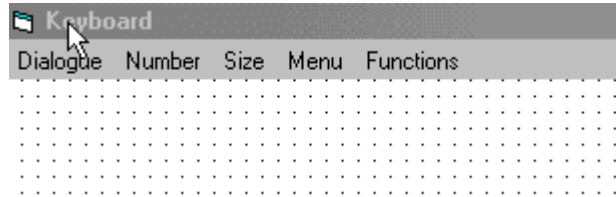


Figure 9.2: The second design of a text keyboard

All the menu items are put into submenu of menu *Menu*. To choose a menu item, the submenu must be selected first. It is not straightforward as the first design, but it saves the space. The default value of number will be set to one. If the customer asks two Big Mac menus, first click menu *Number* and choose 2, then click menu *Menu* to select *McMenu* and then select *Big Mac Menu*.

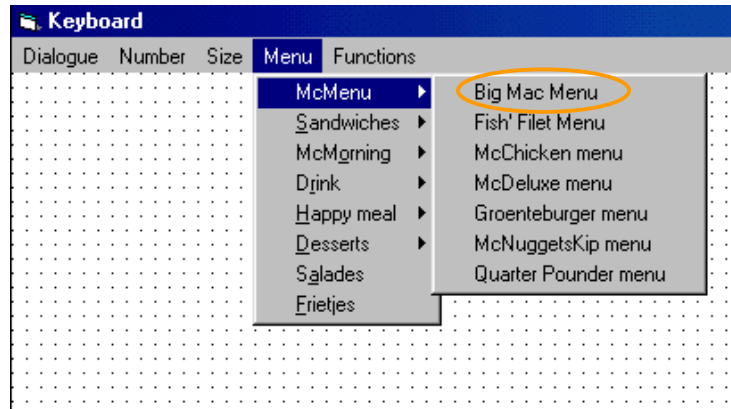


Figure 9.3: The layered approach of a menu item

If the customer asks for a milkshake/MacShake, then click *Menu*, choose *Drink* and then choose *MacShake*. In the downside part of the keyboard the attributes for MacShake will be displayed.

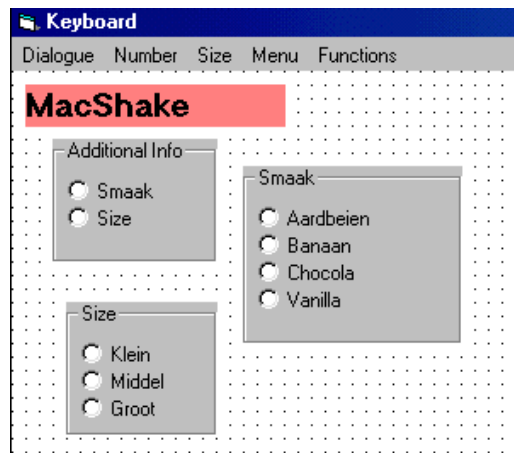


Figure 9.4: The attribute options of the menu item MacShake.

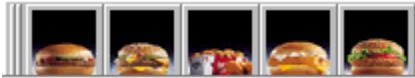
9.2 Second Design --- Graphical Menu Keyboard

Now we are trying to build two graphical keyboards of the menu items, one is for the customer interface; the other is for the operator interface.




9.2.1 Graphical buttons

For every prompt of the minimal prompt set we use a graphical button. For the menu items we have two choices:

1. We use the photo of the menu items to make the button. For example let us take the menu items of *sandwiches*



But there are many kinds of sandwiches such as Big Mac, Hamburger, Groetenburger and etc. When we reduce the format of the picture of the sandwiches to 32*32 pixel, all the sandwiches are similar to each other, it is difficult to find what is what at the first glance. So we must find some symbols to present these sandwiches.

2. We use a symbol to present the sandwich. For example we use an icon of vegetables  for Groenteburger and an icon of cheese  for cheeseburger and a fish icon  for Fish Filet. Then the user can see at a glance the difference and know what the symbols present. Of course we cannot find an appropriate symbol for every menu item. Here is the result.



9.2.2 Identical buttons

For the prompts that the customer and operator prompt set both contains we use same symbols. For example both prompt sets have prompt “een”, “twee”, “drie” and etc. So we use identical icons.



9.2.3 Compact keyboards

We try to make the keyboard as compact as possible. So put the menu items into a tab sheet according the categories.



Figure 9.5: The Graphical customer keyboard



Figure 9.6: The graphical operator keyboard

When the user clicks sandwiches, he can see all the menu items of category sandwiches. And the hint will appear when the mouse rolls over the icons.

9.2.4 Dynamic Keyboard

To make sure the user click the right button, we put only the necessary buttons on the keyboard. So in the order process, the keyboard is always changing. The used keyboard can be predefined or dynamic generated. We can produce a number of keyboards in advance according to the BNF of the customer / operator prompt sets.

Table 9.1: Keyboard set of operator interface

Prompts	Type
Open	Predefined
Ask for order conformation	Dynamic Generated
Ask for additional info	Predefined
Ask for additional order	Predefined
End	Dynamic generated

Table 9.2: Keyboard set of operator interface

Prompts	Type
Open	Predefined
Give order	Dynamic Generated
Give order confirm	Predefined
Give additional info	Dynamic Generated
End	Predefined

For example:

Operator: U heeft een cheeseburger besteld, klopt dat?

Customer: Ja.

In the operator interface, the operator keyboard is dynamic generated. We don't need put all the menu buttons on the keyboard. We put only the following buttons on the current keyboard.



In the customer interface, the customer keyboard is predefined. It has only two buttons:



In this way we can decrease the risk that the user chooses the wrong buttons.

9.2.5 Meta-layer

Some menu items can be accompanied with attributes. For example the customer can ask for a Big Mac menu with a medium Milk Shake and he wants *frietsaus*. There are some options to build this kind of keyboard.

Option 1: We let the user click category *Mcmenu* to choose *Big Mac menu*, and then click category *Drank* to choose *MilkShake*, and at last click category *Friet* to choose *frietsaus*.

This option costs commonly too much time. And it is inconvenient to the user. With this kind of keyboard the user may not know i.e. he can also choose the kind drink he wants.

Option 2: We use a meta-layer. When the user clicks on the *Big Mac menu* button, the keyboard automatically generates a meta-layer, which asks the customer for the possible attributes.



Figure 9.7: Meta layer of keyboard

9.2.6 Grammar check

There are some restrictions on the sentences that the operator and customer produce. For example when the operator ask what kind of milkshake the customer wants, the customer cannot answer with “frietsaus”. The answer of the customer is namely limited within “aardbeien, banana, chocolade, vanille”, otherwise the customer will get a wrong message. To prevent that this happens, we can use a number of methods:

1. Disable the other buttons, so that the customer can only click one of the four buttons.
2. Give the customer a hint in the processing panel. For example “you can only choose ...”
3. Dynamically produce these four choices in a special area.
4. Using a little guiding spirit that will fly to these 4 buttons, and draw a rectangle or just point to them.
5. Make the buttons twinkle.

We have to build flashed buttons by ourselves. Although they were produced but the flashed button takes too much space of memories, so they are left out.

9.3 Last Design: A Manual Keyboard for the Operator Interface

At the beginning we use one single keyboard to show the menus, commandos, attributes and the expressions. But it seems to be a little chaotic. And it is not flexible. If the operator don't want to have expressions, he cannot not let them disappear. So according the tables of the database, we use three keyboards: Commando Board, Menu Board and Expression Board. These expressions buttons use smiley face icons. For example button "smile" is 😊, and button "surprise" is 😮. The attributes are combined into the Menu Board, and the aliases are shown through Commando Board and Menu Board dynamically when they are needed. Here is an example When the operator moves the mouse to one of the commando buttons, that button will show the default prompt from the basic prompt set as tool tip, and if the operator clicks that button, a dropdown window will appear with the default prompt and all the alias of that prompt.



Figure 9.8: Alias: 1) Move mouse to button "drie"; 2) Click button "Drie"

There are two options – one is to use the menu pictures directly, the other is using a symbol to replace the menu picture. Which option is better? Consider the structure of the operator interface the menu keyboard cannot occupy too much space. When the picture is large we will not take a Big Mac as a chick burger. But when the picture is reduced to 360*360 pixels, it takes much more time to identify which one is which one. And for some menu items the pictures are the same. For example coca-cola, coca-cola light and tea have the same outward appearance, in other word the pictures are the same. It seems the first option is out of consideration. How about the second? How do we find a proper symbol for a menu item? This is not an easy task. Take the vegetable burger as example; we want to use a vegetable icon to represent this menu item. But when the user looks at the vegetable icon, he may image it is a vegetable burger, but he may also connect it with salad. So we need to reduce the error chance. But if the user knows in advance it is a sandwich, not a salad. We think there is no chance the user will take it as a salad. At this time the field Category is put to use. When we build the menu keyboard we put the menu items of the same category together.

Our designed keyboard have layers. Some menu item have attributes and we would like to have a new layer appeared with these attributes when we click these menu items. So we need to define a button that has simple and complex state. According field Attr_Type we can decide the state. Those menu items that don't have an attribute have an empty Attr_type field and their state are simple. Next question is how do we know which attribute belong to which menu item. Attributes have different types and each type have its own values. In table Attribute we have a field Type. But this field and field Attr_Type is not the same. Because some menu items of different categories can have the same attributes and some menu items have more than one type attributes. For example menu items of category Supermenu have two type attributes: Drink and Fries. Menu items of the same category can also have different attributes. Menu item Coffee of category Drink has attributes "milk" and "sugar", other menu items of this category don't have such attributes. We can define relations between Attr_type and Type in a procedure, but in this way these relations are fixed and if we want to change them or add new relations we need to change the program. We don't want to do this. So we put these relations in Attribute.ini. Those buttons with complex state have a small triangle at the right-bottom of button.

9.3.1 Menu Board

We take Menu Board as our example. In this section we'll discuss how this board is dynamically generated. Menu Board has a layered structure, there are two forms used, parent form and child form. When the program runs both forms are generated. The Menu categories are listed on the parent form. The Menu table of McDrive database is searched and find out all the menu categories and add them to the parent form. We don't see the menu item directly.

```
mysql = "Select Distinct Category from Menu"
```



Figure 9.9: The menu keyboard

- Menu Board is not pre-defined; there is no definite number of buttons. If there are only 5 menu categories then only 5 buttons are generated, not one button less nor one button more.
- Each button represents one menu category. We use both icon and text to make it clearer.
- The size of Menu Board can be changed by dragging the right side or the bottom. The mouse pointer will turn into an arrow W-E when the mouse moves to the right side or into an arrow N-S at the bottom. Then the user can click and drag it to change the width or height of Menu Board. At the same time the width of Order/Monitor Window or the height of Dialogue Window changes correspondingly.
- User can open or close Menu Board at any time. The Menu Board is docked to the left side of the interface. Its size can also be changed. The default size is loaded from the operator configuration file when the program runs.
- By clicking the icon or name of a menu category we get the child form shown with all the menu items in this category.

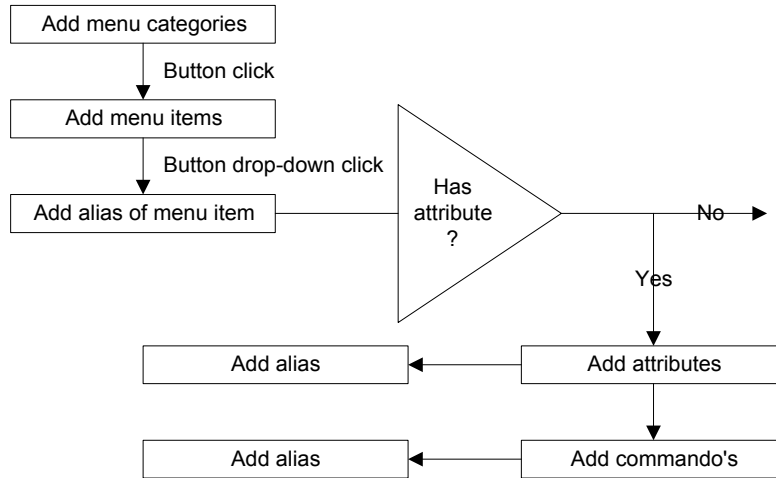


Figure 9.10: The process of generating Menu Keyboard

The above figure shows the process of generated menu keyboard. The table Menu of the database McDrive is searched to find all the menu categories and the results are added to the menu keyboard. From figure 9.5 we can see at this moment we have 8 categories. Before each category an icon is added to give a more direct impression. If a category is clicked then the menu items belonging to this category are added in a pop window by searching the menu items of which the field Category is the name of this category in the table Menu. If a menu item is complex then first it is checked whether it has aliases. It is done through searching the code of menu in the table Alias. The results will be added after the default name of the menu item. If the field Attr_type of a menu item is not empty then we can find the names of attributes with the help of the configuration file attribute.ini. The names of attributed are listed below the aliases. Click an attribute and the possible values of this attribute will be shown in the next layer by search the attribute name in the table Attribute. We have defined some questions for the attributes. If the customer asks for cola then the operator has to know what size the customer wants. This kind of questions can be predefined. And to make it for the user easier we would like to put these questions and attributes together with the menu item. The user doesn't need to search the entire interface for the corresponding questions. The belonging questions can also be found in the file attribute.ini and the question names will be added below the attributes. A question can be asked in different ways and all of them are stored in the table alias. And they will be shown in the following layer if a question is chosen. The figure 9.7 shows how to generate the prompt "small cola".

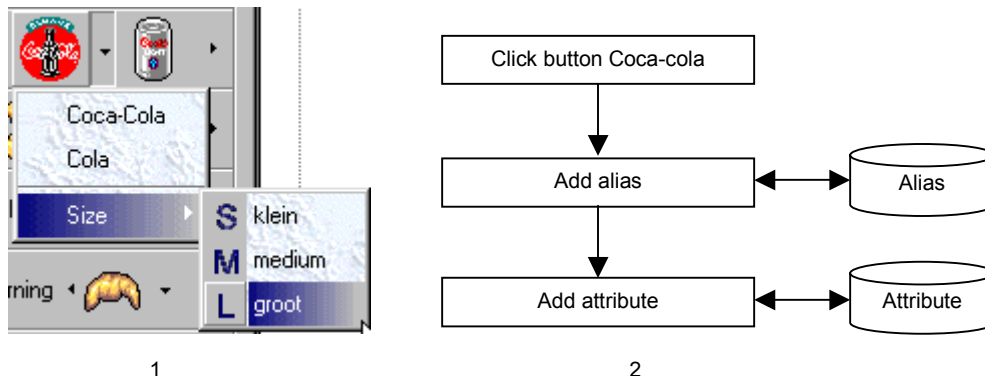


Figure 9.11: 1) Click drop-down button "Coca-cola"

2) The interaction between Menu Board and database

There are two kinds of buttons: default button and drop-down buttons. Their difference is that the drop-down button has an arrow on the right side and default button not. If we click the arrow a drop-down menu will appear. A menu item has a default button or a drop-down button depends on whether

it has attributes or not. If it has attributes then the button generated for this menu item will be a drop-down button, otherwise just a default button. In fact because a menu item may have alias we should make for all of them a drop-down button. But to make it clear to the user whether a menu item has attributes or not we make this difference. In this way the simple menu items can't have alias. But we can't satisfy both sides.

For semi-automated mode the keyboard is constructed with 3 toolbars: menu, commando and expression. On the menu toolbar not all the menu items buttons are available. Only the buttons of the menu items that the customer orders will be dynamically produced. In the same time the corresponding attribute values and questions buttons are generated automatically. The menu toolbar is update continuously, when new order comes buttons will be added and they will be deleted after the order is confirmed. The commando and expression toolbar keep the same.

Chapter 10 Operator Interface

10.1 Interface Layout

To make sure the McDrive system works correctly we want to first design a customer- and an operator- interface that we can manually control. Human interaction is characterized by a multiplicity of signals, which generate redundant and complementary information that makes human communication robust, flexible and natural. To endow McDrive interfaces with similar flexibility, robustness and naturalness, we want to develop multi-modal human- machine interfaces.

The task of operator interface is to control the customer interface with messages. The operator interface has three control modes: manual, semi-automated and automatic. In the manual and semi-automated mode the operator controls the system and produces response manually. Therefore a special keyboard with all kinds of buttons such as menu items, expressions and etc is needed. In the automatic mode the system everything works automatically, but he operator can take action if necessary. The operator keyboard is split in three parts: Commando, Menu and Expression. Each of them is an individual. The Commando Board is docked to the upper part of the interface and the Menu Board on the left and the Expression Board on the right. There is also a dialogue window that is docked to the bottom of the interface to help the operator remember has have been said. We can see there is a space left in the middle between Menu Board and Expression Board. The monitor window and Order Window will appear at this location in turn. When the operator is finished with his saying, the Monitor Board shows so that the operator can monitor the customer screen and see what's going on. When the customer is finished, the Monitor Board hides and the Order Boards appears so that the operator can fill the orders if he wants. These windows can show or hide by clicking menu View and choose the window. The other forms such as "Menu Manager" are not loaded when the program runs. The operator can open them through menus. This is a dynamic interface; the operator can change the place or size of a certain form except Monitor and Order. And the operator can close any of them. The interface size is read from the file "operator.ini" which contains all the settings.

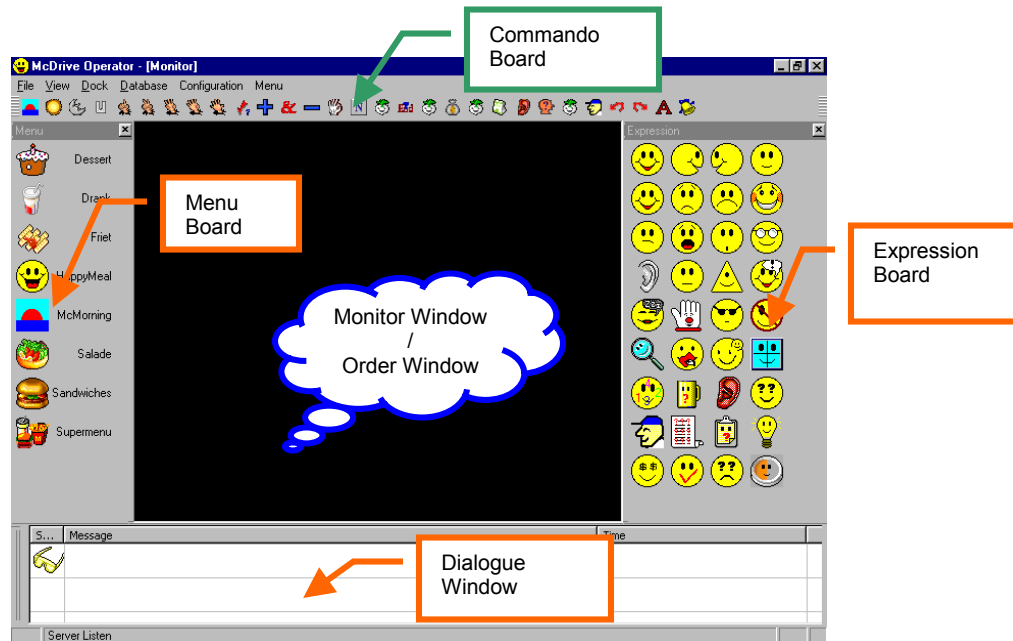


Figure 10.1: Operator Interface screen snapshot

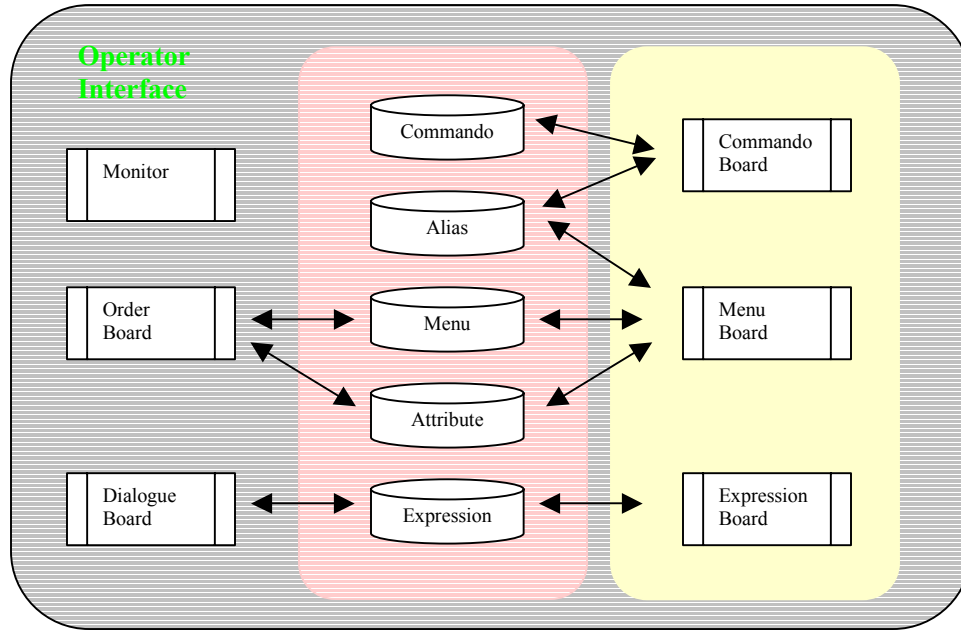


Figure 10.2: Interaction between the windows of operator interface and McDrive database

10.1.1 Menu Board

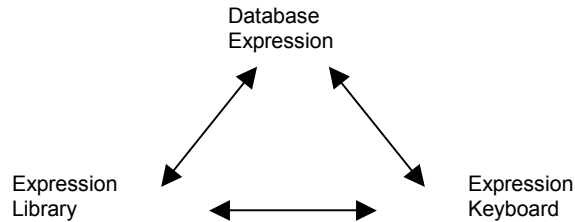
Menu Board has a layered structure. The manual and semi-automated operator use the keyboard to generate the response to the customer prompts. The keyboards in the manual and semi-automated mode are different. The manual keyboard is composed of three parts, text prompt, graphical and wizard. The buttons of the text and graphical parts have a one-to-one relationship; each button of the text prompt part has a related button in the graphical prompt part. For some text/graphical button a related wizard button exists, but not for all of them. Each button in the keyboard has an associated code and the related buttons have the same code. The code records are also kept in the client machine. When the button is clicked, it will generate a command beginning with his type (text/ pict/ wizz) and the associated code. For example the “hamburger” buttons of the graphical prompt part generate a command beginning with “pict” and the associated code “0201”. When the client receives this command, a picture of code “0201” will be displayed in the presentation part if such a picture exists. The operator needs to click the buttons in the order of text, graphical and then wizard. In this way on the client screen the text response and graphical picture, the wizard action will be displayed in the same order. We use an intelligent keyboard. When a text button is clicked, the related graphical button twinkles and the mouse moves automatically on that button. The same applies to the graphical button and the related wizard button. The operator can choose to click the twinkled button or other button. The intelligence of the keyboard shows also at the stage side. We don’t put all the buttons on a keyboard. On the contrary we use different keyboard at the different stage. In this way we can keep the keyboard impact and the operator doesn’t need to find a button in a button sea.

10.1.2 Commando Board

The commando Board is generated with the help of Commando table of McDrive database. In this table there are all kinds of prompts stored. We use a graphical button to represent these prompts. Such prompts may have many aliases, which can be shown by clicking the buttons. These are also some functional buttons such as “Undo”, “Redo”, “Send”. Sometimes the operator may click the wrong buttons by mistake, he can undo it by clicking the Undo button. It can step back to the beginning of this operator prompt sentence. The prompts that already sent to the customer cannot be undone. When the operator finishes his prompt he need to click “Bell” button send it.

10.1.3 Expression Board

There are many smileys stored in the expression table. The smileys will be added selectively to the Expression Board. There are a few options. We can choose only the level 1 smileys, or only the smiley which Show property is true. We can also choose level 2 smileys. The choice can be made by the configuration.



10.1.4 Dialogue Window

We want to keep a record of what the customer and operator say. We use an individual window to do this job. In this way the manual operator can view the prompts directly and then make decisions. It keeps track of the conversation between the operator and the customer. We use a list with three fields: time, speaker, prompt. When the operator clicks a commando button or a menu button the text will be added to the prompt field and the icon in the speaker field will change to the smiley the operator clicks. So in the operator row is always shown the current expression. When the customer says something, it will be added automatically in the list with the current time and the speaker in a new row.

10.1.5 Monitor Window

In the monitor window the captured region of the client screen will be displayed. The window sizes changes as the region size changes. The client captures the region at the capture interval, saves and sends to the server that loads the image at the same interval.

10.1.6 Order window

The order window is used to keep track of the orders. The menu item and the quantities are added when the customer confirms his order. It uses a tree structure.

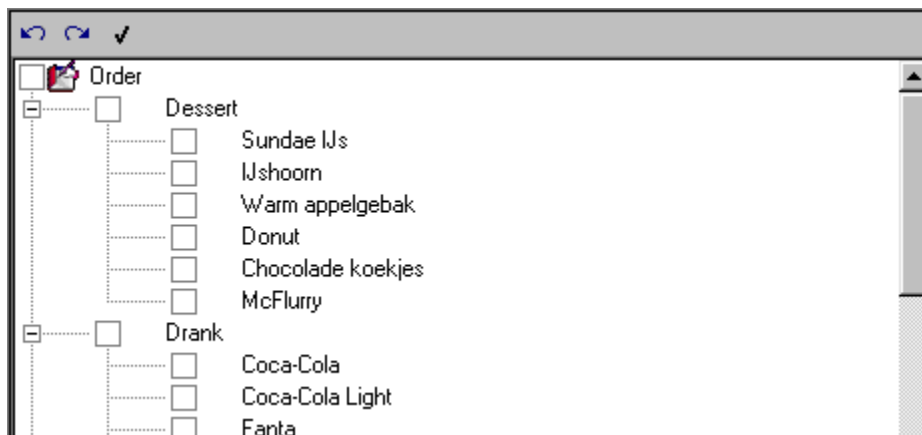


Figure 10.3: The order window

10.1.7 Status

The status part informs the operator about the process of the server. It is a long narrow window running across the bottom of the screen. This window contains an icon that indicates the state of McDrive system (ready, listening, or busy).

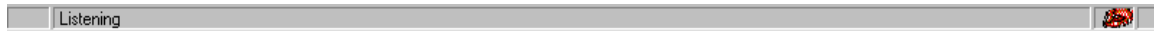


Figure 10.4: The status part

10.2 Server Functions

10.2.1 Configuration

We can adjust the server settings by clicking the Configuration menu.

- Screen capture configuration

The settings are the capture interval, capture region and scale parameter. The client will send the picture at the capture interval; the default interval is 2 seconds. The capture region has 3 choices, active window (default), desktop and the region with the coordinates. We don't want show a picture with the actual size of the captured region, such a picture takes more time to transport and occupies too much place on the server screen. According to the scale parameter the client reduces the capture region into a smaller picture. A region of 600*600 pixels can be reduced into a picture of 300*300 pixels according the default scale parameter 0.5. When the server changes the configuration, it will send a message to the client to change it too and to save it in the configuration-setup file of the client.

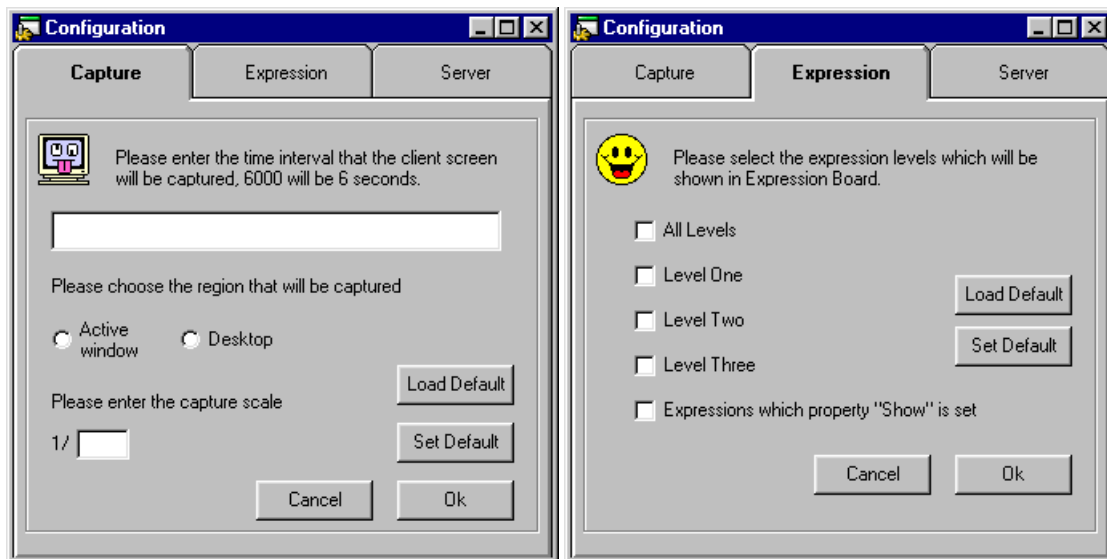


Figure 10.5: Configuration of screen capture and expression board

- Expression Board configuration

This setting decides which smiley can be added to the expression board.

- Show IP

Show the IP address and the local ports of the server machine so the client can set up the server configuration and make connections with the server.

- Configuration file

We use also a windows ini document. The windows ini document is special used to keep the initial information of the application and the environment information.

For example, there are two ini documents win.ini and system.ini in windows 3.1 which defined the mouse speed, shell and other settings in the windows environment as windows start. Ini document is a text document that can be edited with text editor such as Notepad. It has a specific form – it is composed of some sections, each section contains some key and the corresponding value. Take example of mcdrive.ini, its form is as follows,

The settings of server, capture and etc are written in the configuration file “McDrive.ini”.

```
[server]
servername=oemcomputer
serverip=127.0.0.1
serverport=2000
streamport=2001

[capture]
captureinterval=6000
captureregion=activewindow
capturescale=2
... ..
```

We can change the setting of mcdrive application through editing mcdrive.ini manually. We can update this setting by reload mcdrive.ini. At the beginning of running customer program the initial settings are loaded from the configuration file “McDrive.ini”. If the setting is changed in the application running, it will be saved into that configuration file.

```
INIPath$ = App.Path + "\McDrive.ini"
ServerName = ReadINI("server", "servername", INIPath$)
ServerIP = ReadINI("server", "serverip", INIPath$)
ServerPort = Cint(ReadINI("server", "serverport", INIPath$))
StreamPort = Cint(ReadINI("server", "streamport", INIPath$))
CaptureInterval = Cint(ReadINI("capture", "captureinterval", INIPath$))
frmMain.TimerCapture.Interval = CaptureInterval
CaptureScale = Cint(ReadINI("capture", "capturescale", INIPath$))
```

10.2.2 Real time remote client control

The most important function of the server is to real-time remote control the client. When the server receives the customer prompt from the client the server generate response/command and send them back to the client. The client will do tasks such as show pictures according to the responses and commands. The server has three control modes: manual, semi-automated and automatic. In the manual mode the operator needs to generate the text, graphical and wizard responses separately with the help of keyboard; in the semi-automated mode the operator can generate the different responses in one time with the help of keyboard; the server can self generate the responses automatically. In all these modes the operator can change settings at any time. The setting changes of the server and client are synchronized. When the server setting changes the client will be told to change them too.

10.2.3 Real time remote monitor client screen

The operator wants to monitor the client screen real-time. Our goal is that the operator can view the client screen in a monitor window on the server screen. The image in the monitor window changes as the client screen changes. To achieve this result the client has to save the captured region in an image file and send it to the server, then the server can load this file and show the image. The process will be repeated at an interval of certain seconds. We can set this interval for example 2 second then the monitor window will be refreshed at every 2 seconds. To save transport time the captured region will reduced into a smaller image. The operator can change the capture setting by clicking the “Monitor configuration” command.

10.2.4 Log document

The conversation between the operator and the customer will be kept in a log file. Each time the customer or the operator says something, it will be added automatically into the log file. The date / time and the speaker will also be added. We keep also the order log. Each order will be added automatically into the order log document after the confirmation.

10.3 The Order Process

The task of the system is to help the customer to give his orders. The system generates response on the basis of the customer prompt, the history and the current state. The system opens by greeting the customer and asking for order. If there is no answer heard then the system will repeat. The system will give up after two times failure. Otherwise the system analyses the customer prompt, if the system can't understand what the customer says it goes into the not_understanding state and repeat his question. If it is an order then the system will ask for confirmation. The system will take different actions on the basis of the answer of the customer. If the customer denies the order the system apologises and asks the order again. If the order is confirmed then it will be added to the order list. The system will ask additional information if the confirmed order has attributes. The figure 10.5 is a flow chart of a part of the ordering process.

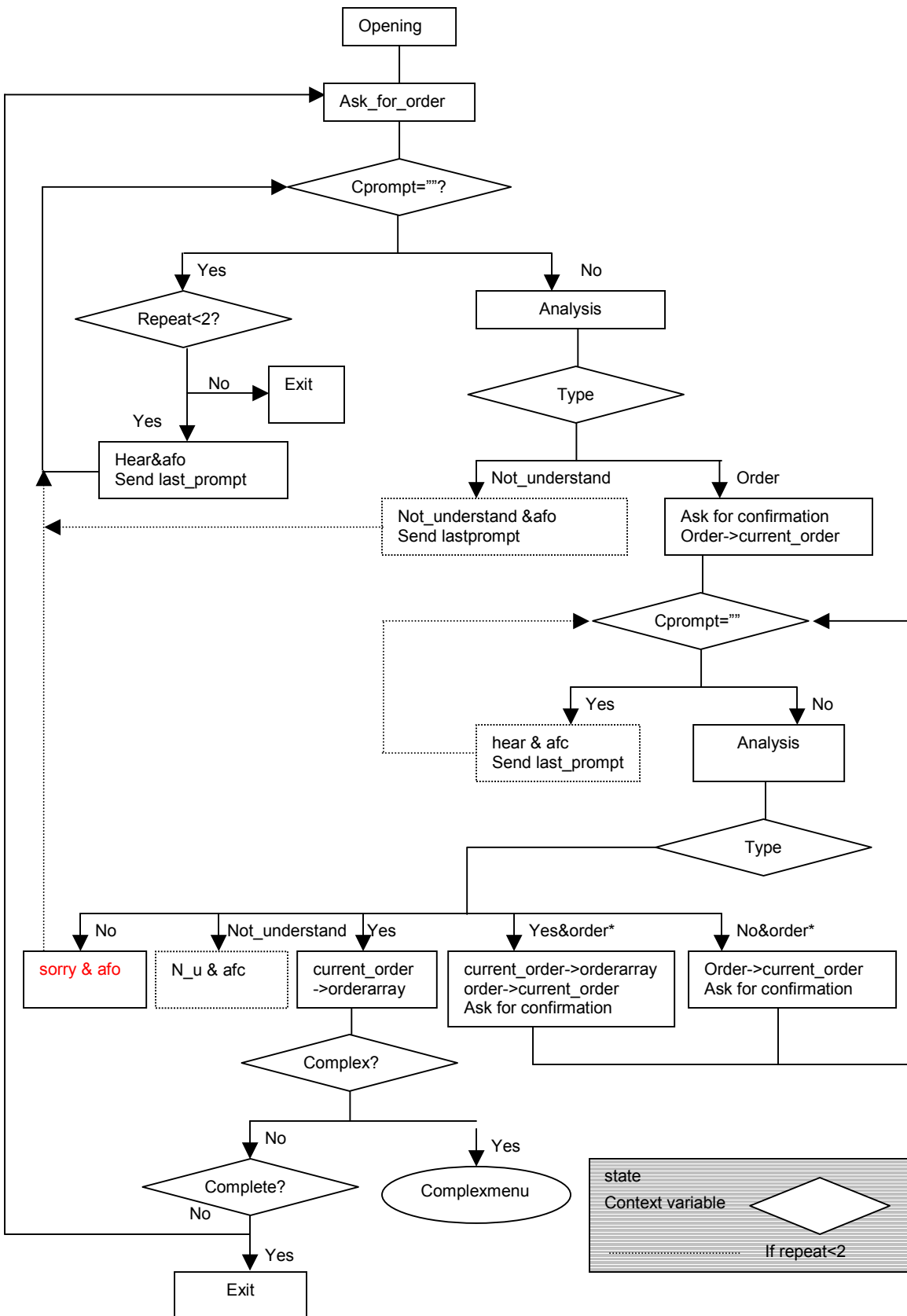


Figure 10.6: A flow chart of a part of the ordering process

Chapter 11 Customer Interface

The customer interface is used to show the customer system response. The feedback is text, pictures of menu items, movies of ordering and smiley faces. And there is also a wizard who shows the facial expressions.

11.1 Interface Layout

The customer interface is divided into 4 parts; each of them contains a separate window. These windows are Graphic Board, Wizard Board, Text Board and Smiley Board. Their place and size are fixed the customer cannot change them. The following figure shows the customer interface screen when the operator welcomes the customer. An opening flash movie plays in Graphic Board. The wizard Lisa drops to Wizard board and a happy smiley face appears in Smiley Board. In Text Board there is a welcoming message.



Figure 11.1: Customer Interface screen capture

11.1.1 Graphic Board

The client system let the customer to give his orders and show the picture of the menu items that the customer orders. In this version the customer gives his order using text. The server system asks the customer to confirm his order by showing the question with text in text board and menu items with pictures on the Graphic Board. The customer likes to order all he wants in one time. We can choose from two options: one is that we ask the customer to confirm one menu item once; the other is that the customer can give confirmation about different menu items. The first option simplifies the process when the customer denies the order. The second option cuts time of the dialog. When the server system is manual or semi-automated there is no problem, the operator can decide it self. For the automatic

server system we use the second option as default, the operator can change it at any time. For the first option the picture of the current menu item stays on the graphic board when the customer answers the confirming question. If the answer is a confirmation then the picture will be shrunk and added into one of the four cells of the graphic board. Otherwise a cross will be added on the picture. For the second option the picture of the current menu item will be displayed for about 5-10 seconds and then added into a graphic board cell. If the customer confirms it nothing will be changed. Otherwise a cross will be added on the menu item picture in the graphic board cell and the picture will be deleted. If the customer orders two hamburgers the hamburger picture will not appear two times, we just put a “2” on the right bottom part of the picture. The server system asks the customer to give extra information about the menu items with a flash movie that the choices are showed. The graphic board of the client system is divided into 4 cells that contain the picture of the ordered menu items. The flash movie and the picture of current menu item are displayed on the upper level of the graphic board when necessary.

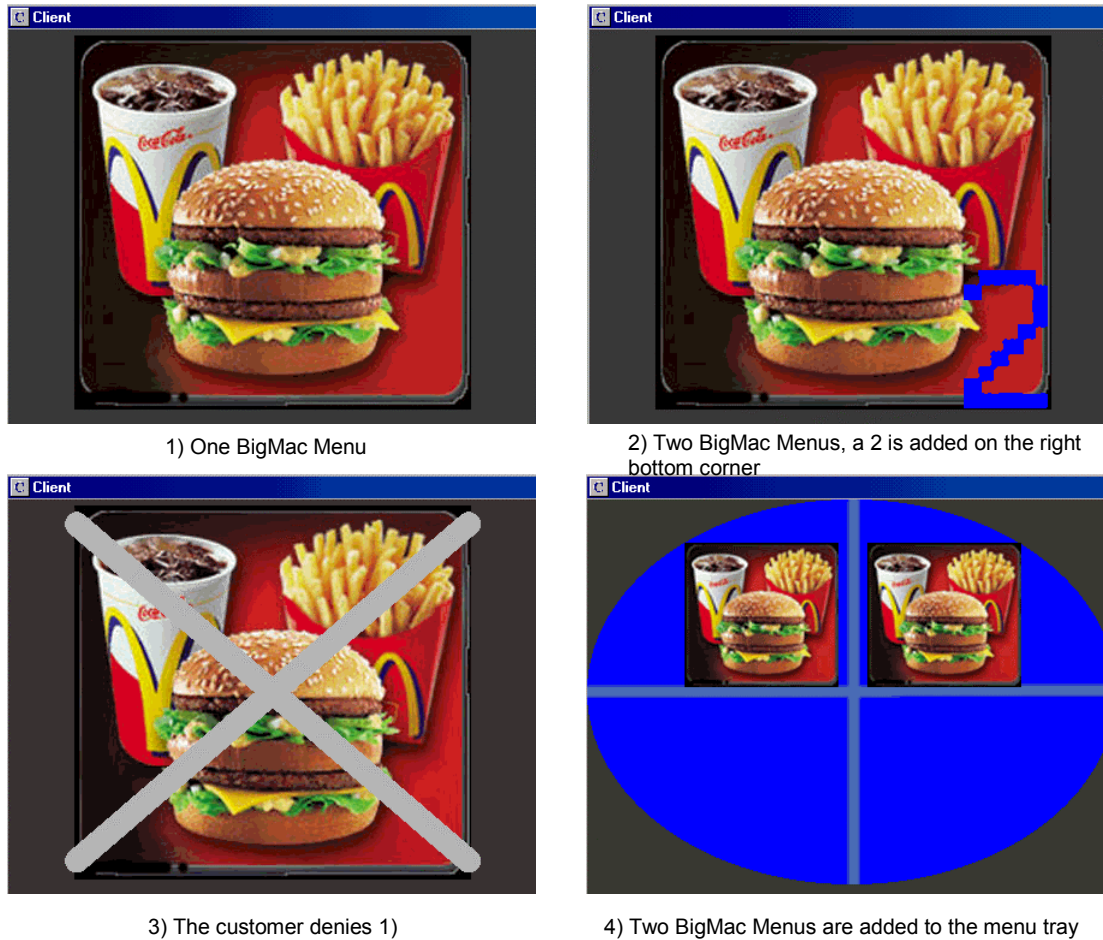


Figure 11.2: Graphic Board of Customer Interface

11.1.2 Wizard board

The place of the McDrive wizard is in the wizard board. We create a personage “Lisa” with Poser and Microsoft Agent Character Editor. We use Poser to produce different expressions for Lisa, each expression is a series of frames that are exported into a separate picture. An expression “smile” can have 6 frames from “default” to “smile”; in this way we achieve the continuity. Then we will add these 6 pictures to the action “smile” using Microsoft Agent Character Editor. When we give command “smile” to Lisa, Lisa smiles and her face changes according to those 6 pictures. Lisa is an agent character and she is in her own window that always appears at the top of the window. We put Lisa into the upper right part of the interface. When the client system receives an expression code it

will find the corresponding action name and commands Lisa do this action. When the customer speaks the wizard Lisa is listening.

11.1.3 Text board

In this board the server prompt is shown to the customer. The server sends in fact the codes to the customer. For every code except for which begins with 5 the corresponding text is searched and added to the text board. Only the current customer prompt is displayed.

11.1.4 Smiley board

Sometimes Lisa cannot good enough to express the situation. Smiley face has a supplementary function. When the operator clicks one of the smiley face buttons, the expression code will be sent to the client system. The client system loads the smiley face from the resource file using the code index.

11.2 Client Functions

11.2.1 Screen capture

The client needs to capture the screen and send the saved picture to the server when the server asks. The server will send also the desired capture region and interval to the client when necessary. The client keeps a configure document that records the capture settings.

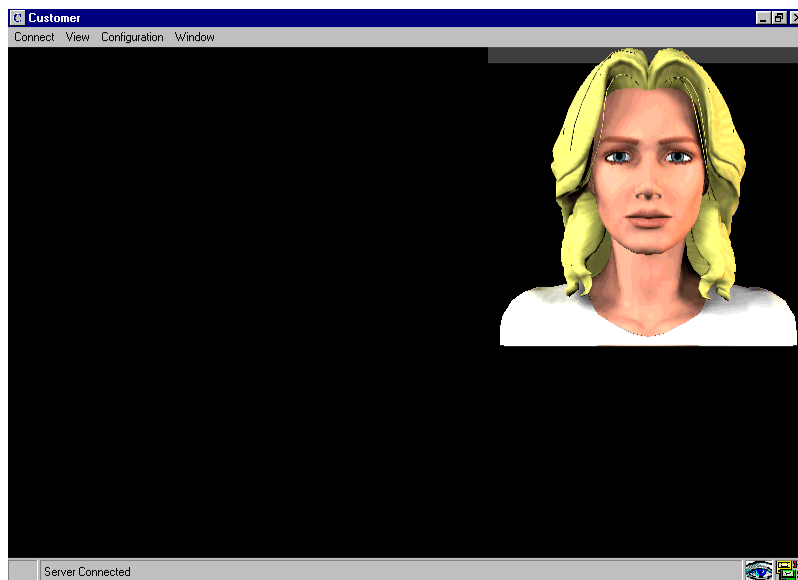
Chapter 12 Simulation and Test

Now we are going to simulate the McDrive order process and to find out whether the manual prototype can satisfy our demands. We call the computer where the operator interface is installed “Server”, the other computer with the customer interface “Client”. These two computers are connected through the Internet. The operator sits behind Server and has no view of the screen of Client.

12.1 A Simulated Dialogue

We have simulated some McDrive dialogues and one of them is as follows:


1. *Customer drives in.*

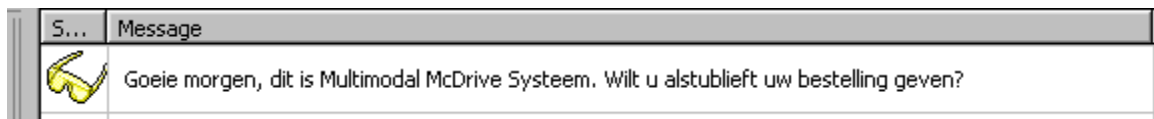



Client makes connection with Server. Server beeps to warn that new customer arrives. Wizard appears on the right part of the screen of Client.

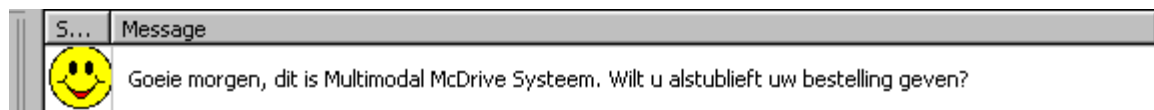
2. *The operator says: “Goeie morgen, dit is het Multimodal McDrive Systeem. Wilt u alstublieft uw bestelling geven?”*


Operator actions:

- The operator clicks the *Goeie morgen* button  on the Commando Board to generate the text response. And the text response is added to Dialogue Board.



- The operator clicks the *smile* smiley  on the Expression Board and the icon in the speaker column of Dialogue Board is changed to the smile smiley.



- The operator clicks the *Send* button  on the Commando Board and the system response is sent to Client. A new row with the customer icon is added to the dialogue window.

On the screen of Client:

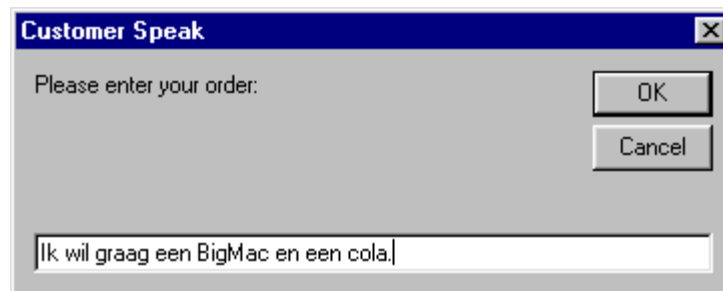
- Ronald walks out.
- Text response is shown.
- Wizard smiles.
- Smiling smiley is shown. (See the screen capture in figure 11.1)
- A textbox pops out to let the customer enter his order, the wizard listens and the smiley is changed to a listen smiley.



3. *The customer says: "Ik wil graag een BigMac en een cola."*

Customer action:

- The customer types in the order in the text box that pops up.



On the screen of Server:


- The order is sent to Server and added to the customer row of the Dialogue Board.

4. *The operator says: "U heeft een BigMac besteld, klopt het?"*




Operator actions:

- The operator clicks on the *U* button on the commando board, there are two aliases, and the entry "U heeft" is chosen. The text is added to the dialogue window.



- The operator clicks on the *een* button  on the commando board. The text is added to the dialogue window.
- By clicking the category *Sandwiches* a window with all the menu items of that category pops up and the *BigMac* button is chosen. BigMac is added to the dialogue window.



- The operator clicks the *klopt het* button  on the commando board.
- The *Ask* smiley  on the Expression Board is also clicked. The icon in the operator row is changed to the *ask* smiley.
- The operator clicks the *Send* button  on the commando board to send the system response. A new customer row is added in the dialogue window.

On the screen of Client:



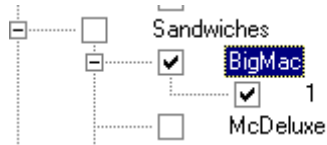
- The picture of BigMac Menu is shown.
- The text response is shown. The colour of the menu item BigMac Menu is red and the rest of the text remains yellow.
- Wizard raise one eyebrow and plays the ask action.
- The asking smiley is shown.
- Input textbox pops out.

5. *The customer says “Ja.”*

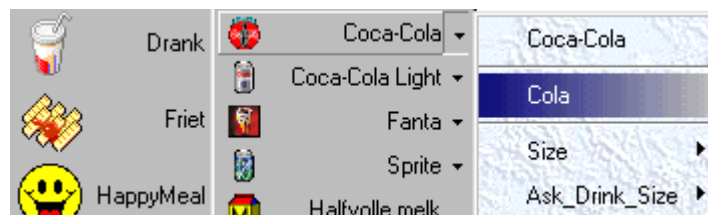
Same as 3.

6. *The operator asks “U heeft een cola besteld, klopt dat?”*

The operator checks the menu item Big Mac on the order Board.



Cola is the alias of Coca-cola. The operator needs to click the Drank button to see the menu items and then choose the alias by clicking the drop down arrow of coca-cola button.



7. *The customer says “Ja.”*

8. *The operator asks “Wilt u een klein, middel of grote beker hebben?”*

The ask_drink_size question is also listed in the drop down window of coca-cola. The operator can choose how he asks this question.



.....
9. *The operator says “Het is ... euros. Dan mag u doorrijden naar het tweede loket.”*

On the screen of Client Ronald slides to the second window, collects the order and slides away. Wizard disappears and the connection between Server and Client is closed.



12.2 Analysis

From the tests we can see that the simulated dialogue is longer than the normal dialogue between the customer and the human operator because the system has to ask confirmation of the orders. The communication between Server and Client works good and fast. The structures of the interfaces are reasonable. By using the individual windows the interfaces are more flexible. The graphical stuffs such as the pictures of menu items and the flash movies are nice. The wizard is life like and her facial expressions are realistic but more expressions can be added. The tables of the database McDrive are reasonable divided. The table managers make the database administration and record editing that can be found in chapter 8 very easy. The administrator can also add new tables or new columns. Only the managers are fixed and a new table means a new table manager has to be built. The keyboards are dynamically generated from the database and they can be edited by editing the database. The text feedback is clearer by using different colours for the menu items, attributes and other system prompts. It would be better if the text can be self-scrolling on the basis of the text length. The test shows it is better to choose the smiley first and then the other prompts.

Chapter 13 Conclusion and Recommendations

We have designed the Multimodal McDrive System (MMS) and implemented the manual prototype. This system is aimed to replace the first operator of McDrive. Through our tests we can see that the system can complete this job.

What we have been done?

This thesis'w work is about a multimodal extension of a McDrive system. In a common McDrive system the operator and client communicate by speech. We researched the possibilities to use multimodal interfaces. The operator can give feedback to the customer using graphics, animations and etc. And the client can interact with the operator / system using touch screen, icons etc. We designed and built a running prototype of the multimodal system. More in detail we performed the following activities:

- We have analysed hundreds of dialogues and defined a basic system and customer prompt sets.
- A wizard of OZ study has been done to see whether the design works.
- A Database McDrive is built, which contains the basic prompt sets, pictures of menu items, smileys and other information. The database is divided into 5 tables: Menu, Expression, Commando, Alias, Attribute. The picture of menu items and other information over the menus are put into table Menu. Attributes have their own table because not every menu item has attributes and the number of attributes is also different. The table Expression contains the smileys and the corresponding wizard actions. The supplementary prompt sets are added into table alias so that the operator can have different ways to ask the same question. We have built a manager for each table so that we can add, delete, search or edit records.
- An operator keyboard is built on the basis of the basic prompt sets and database McDrive. This keyboard contains 3 sub-boards: Menu, Expression and Commando. The operator can generate the system response by clicking the buttons. This keyboard is a graphical board and icons have been used for representing the menu items and other prompts. The keyboard is dynamically generated when the MMS runs. The keyboard has a layered structure. When a menu category is chosen all the menu items of this category are shown in a pop window and then if a menu button is clicked the alias, names of attributes and related questions will appear in the drop down window and then the values are shown in the next layer successively. There isn't enough space to show all the smileys on the Expression Board. Which one is going to be shown depends on his level or property Show.
- An operator and customer interface has been built. These two interfaces can be connected through the Internet. The interfaces are divided into some sub-boards. The operator interface contains keyboard (Menu, Expression, Commando), Order-, Monitor- and Dialogue Window. The customer interface is constructed of four parts: Graphical, Text, Wizard and Smiley. These four boards have the fixed format and location.
- Configuration files are used to keep the system settings.
- The two interfaces can communicate with each other in real time.

Implemented system feedbacks

The implemented system feedbacks are as follows:

- MMS generates text feedback. The response is shown at the left bottom part of the customer screen. The menu items, the attributes and the sub-menu items are marked with different colours.
- MMS generates graphical feedback. Pictures of menu items are shown when the operator asks confirmations. For some predefined questions there is a flash movie made to make the customer understand better. For example in the flash movie "ask drink size" three glasses of different size appear and jump one by one. The customer knows the drink size is asked without looking at the text.

- MMS gives a more realistic feeling by using a human wizard. The wizard shows when the customer arrives. The wizard can express many facial expressions such as smile, sad or angry. When the customer speaks the wizards listens.
- Smiley is used as an auxiliary of wizard. There are a lot of smileys collected and added to the table Expression of Database McDrive. The smileys are also listed in the appendix. For some smileys there is a corresponding wizard action. For example for the smiling smiley there is a smiling action of wizard.

The manual prototype of Multimodal McDrive System is a flexible and dynamic system. And the other parts of McDrive can remain the same.

Recommendation

Use more graphical stuffs. We have made some flash movies for questions over the drink size, nuggets number, opening and etc. These movies are made not only to make the system more funny and attractive but also give the customer information. Because the limitation of time and graphical material we just made a few flash movies. Maybe in the future we can make a flash movie for each predefined questions and prompts.

In the manual prototype we have only implemented a wizard for the customer. It is a presentation wizard and its task is mainly to show the customer in which mood the operator is. Maybe we can also build a wizard for the operator. When something is wrong the operator wizard will pop out and give advice. The operator can also call the help of the operator wizard by himself. It is an assistant wizard. Because the operator needs to monitor the client screen so it is better to use a different wizard. Maybe this time we can build a cartoon wizard just like the Microsoft character Genie or Peedy.

We Chinese say that men have seven emotions, namely, joy, anger, sorrow, fear, love, hate and desire. But this is only a general classification. For example, there are many kinds of joy. You can be a little happy, happy or very happy. A man smiles always about the same, but sometimes a smile is bigger than others and the mouth is more open and the eyes are narrower. We don't want a face editor to show the facial expression by adjusting the parameters of mouth, eyes and etc. That would be too complex and cost too much time. But we can add more animations to the wizard. Now we have only one animation for smile. We can make more animations like "smile degree 1", "smile degree 2" and etc. There is another way to achieve the different smiles. An animation of a wizard is composed of several frames. If we can control at which frame the wizard stops then we can get smiles form a little smile to a big smile. Unfortunately it is impossible at this moment because technical limitation.

In the manual prototype we also use a supplementary prompt set except the basic prompt set. The prompts from this supplementary set can be added to the default prompts as alias. Because the limitation of space the smileys are divided into different levels and we can choose which level smiley we want to show. But if we can add the smileys from the other level to the default smiley as alias, when the operator clicks the default smiley all the aliases in the order of level is shown in a pop window. For example if the operator is smiling and he chooses level 3 then the smiling smiley level 3 will be shown and the smile of the wizard is of degree / level 3.

In the semi-automated prototype the flash buttons can be used for the keyboard. According to the context and the current customer prompt certain button can blink to get the operator's attention.

The customer screen is of fixed format and the size of the text window is also fixed. Sometimes the text feedback is too long to be shown in this window. We can make the text to be self-scrolling; if the text is too long then it will scroll by itself. The operator can adjust the scrolling speed.

As discussed in the chapter 5 the input of a XML parser needs to be formatted. We have to use VB to analyse and parse the customer prompts and put the result in a XML tree. In the future work the system can search this XML tree and decide what the system response is. Of course the system

response also relies on the current context. If the customer says “A cola, please.” a XML tree will be built with nothing between the tags <Size> and </Size>; then the system will generate the confirmation question based on this XML tree; after the customer gives confirmation the system will search the XML tree again and find the first empty value, that is Size; the system will ask the customer about the cola size. Now we have to fill the order form manually. In the future the system will search the XML tree at the end of dialogue and fill the order form on the basis of the results of searching.

When the system works total automatically a human operator can be used. He will monitor a few MMS and takes over the control when there is a problem.

Reference

- [1] Ferhaad Mohamed-Hoesein, Ali Shirzad (1998) *Talk McDrive An ASP Application*. Master thesis of Knowledge based system of TU Delft.
- [2] Renato De Mori (1998) *Spoken Dialogues with Computers*. McGill University, Montreal, Quebec, Canada and Universite d' Avignon, France.
- [3] Ray C. Dougherty (1994) *Natural Language Computing, An English Generative Grammar in Prolog*. New York University Linguistics Department.
- [4] T. Van Le (1993) *Techniques of Prolog Programming with implementation of logical negation and quantified goals*. University of Canberra.
- [5] Rogers S. Pressman, Ph.D. (1992) *Software Engineering A practitioner's Approach*.
- [6] (1998) *Using the Microsoft Agent Editor*. Microsoft Corporation.
- [7] *Poser 4 User Guide*. MetaCreations.
- [8] *Emoticons. What is an Emoticon or Smiley?* <http://www.muller-godschalk.com/emoticon.html>
- [9] www.smiley-faces.com
- [10] Roope Raisamo (1999) *Multimodal Human-Computer Interaction: a constructive and empirical study*. Academic Dissertation, the Faculty of Economics and Administration of the University of Tampere.
- [11] *Het EAN-Codesysteem in handel, industrie, transport en logistieke dienstverlening*. EAN-Nederland.
- [12] McDonald websites.
- [13] Kenji Mase *Aspects of Interface Agents: Avatar, Assistant and Actor*. ATR Media Intergration & Communication Research Labs, Kyoto, Japan.
- [14] Zsofia Ruttkey, Han Noot *Animated Chartoon Faces*. Centre for Mathematics and Computer Science, Amsterdam, The Netherlands.
- [15] Elisabeth Andre, Thomas Rist and Jochen Muller *WebPersona: A Life-like Presentation Agent for the world-wide Web*. German Research Center for Artificial Intelligence.

Appendix

Appendix I Menu rules

<i>Category SuperMenu</i>	
SuperMenu	::= BigMac Menu Quarter Pounder Menu McChicken Menu Groenteburger Menu Fish 'Filet Menu Cheeseburger Menu Hamburger Menu McNuggetsKip Menu McDeluxe Menu
NuggetMenu	::= McNuggetsKip Menu McNuggets Menu
Nugget_number	::= 6 9 20 zes negen twintig
Nugget_saus	::= mosterd barbecue zoetzuur kerry

<i>Category Fries</i>	
Friet	::= Patat French Frites frites friet
Friet_saus	::= frietssaus ketchup mayonaise

<i>Category Sandwiches</i>	
Sandwiches	::= BigMac Quarter Pounder McChicken Hamburger Fish'Filet Cheeseburger Groenteburger McNuggetsKip McDeluxe Mac Deluxe
Nugget	::= McNuggetsKip

<i>Category Drink</i>	
Frisdrank	::= Coca-cola Coca-cola Light Fanta Sprite
Otherdrank	::= Spa blauw Jus d'Orange Halfvolle melk Warme chocolade Thee Fristi
Chocomel	::= Chocomel
Chocomel_t	::= koud warm
Koffie	::= Koffie Cappuccino Espresso
Koffie_melk	::= melk
Koffie_suiker	::= suiker
Shake	::= Milkshake MacShake
Shake_smaak	::= aardbei banana chocolade vanilla

<i>Category Salad</i>	
Salade	::= Salade Ham Kip Salade Tonijn Salade Naturel
Salde_dressing	::= thousands islands blue cheese vinaigrette fijne kruiden

<i>Category McMorning</i>	
McMorning	::= Egg McMuffin Compleet Ontbijt Crossiant
Compleetontbijt	::= Compleet Ontbijt

Ontbijt_drink	::=	Jus d'Orange Koffie Cappuccino Espresso Thee Warme chocolade
Crossiant	::=	Crossiant
Crossiant_jam	::=	aardbeien kersen Zwarte bessen

<i>Category Dessert</i>		
Dessert	::=	Sundae IJs ijshoorn warm appelgebak chocoladekoekjes donut McFlurry
Sundae	::=	Sundae IJs
Sundae_saus	::=	chocolade caramel aardbeien nootjes
Donut	::=	Donut
Donut_smaak	::=	cappuccino kaneel
Mcflurry	::=	McFlurry
Mcflurry_smaak	::=	M&M's Douwe Egberts Cappuccino Crunch























<i>Category Happy Meal</i>		
Happy Meal	::=	Hamburger Happy Meal Cheeseburger Happy Meal McNuggetsKip Happy Meal
Nugget Meal	::=	McNuggetsKip Happy Meal























Number	::=	een twee drie vier vijf zes zeven acht negen dubbele twintig
Stuk	::=	stuk stuks maal keer kop kopje
Size	::=	klein middel groot medium kleine small large
Met	::=	en met en met , ,
Zonder	::=	geen zonder

Sandwiches_rule	=	[Numbers], [Stuk], Sandwiches
Frisdrank_rule	=	[Numbers], [Stuk], [Size], Frisdrank, [Comma], [Size]
Chocomel_rule	=	[Numbers], [Stuk], [Chocomel_temperature], Chocomel, [Comma], [Chocomel_temperature]
Koffie_rule	=	[Numbers], [Stuk], Koffie, [[Met], [melk suiker]]*
Otherdrank_rule		[Numbers], [Stuk], otherdrank
Friet_rule		[Numbers], [Stuk], [Size], friet, [Met], [Numbers], [friet_saus]
Dessert_rule		[Numbers], [Stuk], dessert
Donut_rule		[Numbers], [Stuk], [donut_smaak], donut, [donut_smaak]
Sundaeijs_rule		[Numbers], [Stuk], [sundae_saus], sundae, [Met], [sundae_saus]
Mcflurry_rule		[Numbers], [Stuk], [mcflurry_smaak], mcflurry, [Met], [mcflurry_smaak]



Salad_rule	[Numbers], [Stuk], salad, [Met], [salad_dressing]
Mcmorning_rule	[Numbers], Stuk , mcmorning
Compleetontbijt_rule	[Numbers], [Stuk], compleetontbijt, [Met], [Numbers], [compleetontbijt_drink]
Crossiant_rule	[Numbers], [Stuk], crossiant, [Met], [crossiant_jam]
Happymeal_rule	[Numbers], [Stuk], happymeal, [Met], [frisdrank], [Met], [frietsaus]
Shake_rule	[Numbers], [Stuk], [Size], [shake_smaak], shake, [Met], [Size], [shake_smaak], [Size]























Appendix II Smiley faces

Code	Smiley	Name	Wizard Action	Level
Emblem				
e5001		Show	Show	1
Illustrators				
e5101		Turn Left	Turn Left	1
e5102		Turn Right	Turn Right	1
e5103		Look Left	Turn Left	1
e5104		Look right	Turn Right	1
e5105		Move Up	Move Up	1
e5106		Move Down	Move Down	1
e5107		Move Left	Move Left	1
e5108		Move Right	Move Right	1
e5109		No	No	1
e5110		No	No	2
e5111		Screen		2
e5112		Alert	Alert	1
e5113		Alert	Alert	1
e5114		Ronald		2
e5115		Help		1
e5116		Screen		2
e5117		Write		1
e5118		Rthyme		2
e5119		Search	Search	1
e5120		Phone		1
e5122		Read		2



e5123		Listen	Hearing_1	1
e5125		Earphone		2
e5126		Keep silent		2
e5127		Alert	Alert	2
e5201		Big or small		1
e5202		Window		1
e5203		How many		1
e5204		Ask		1
e5205		Information	Suggest	1
e5207		Order complete?		1
e5208		Order too long		1
e5209				
e5210		Money		1
e5211		Question		2
e5212		Yes	Yes	1
e5213		Ask again		1
e5214		Sauce or not		1
e5215		What kind of drink		1
e5216		Number		2
e5217		Human Operator		1
e5218		Order		2
e5219		Order		2
e5220		Drink		2

Affects utterance

e5401		Smile	Smile	1
e5402		Laugh	Laughing	1

e5403		Sad	Sad	1
e5404		Feel wronged		1
e5405		Blush	Sorry	1
e5406		Uncertain	Uncertain	1
e5407		Surprise	Surprise	1
e5408		Uh-oh	Uh-Oh	1
e5409		Toothy		2
e5410		Smirk		2
e5411		Tease		2
e5412		Tooth		2
e5413		Suspicious	Suspicious	1
e5414		Humph		2
e5415		Snarl		2
e5416		Eager		2
e5417		Sad	Sad	2
e5418		Confused		1
e5419		Feel complacent		2
e5420		Angry		2
e5421		Sour		2
e5422		Bashful		2
e5423		Cry		2
e5424		Tease		2

Regulators







e5601		Natural	RestPose	2
e5602		Natural	RestPose	1
e5603		Time		1

Adaptors








e5702		Lick		2
e5704		Karaoke		2
e5705		Sleepy		1
e5706		Wink		2
e5707		Sleeping		1
e5708		Cool		2
e5701		Yawn		1

Appendix III Smiley face used in dialogues






1.

McDrv	Goedenavond welkom bij Mac Donald's, mag ik uw bestelling, alstublieft?	
Klant	Ja hoor, ik wil een McChicken.	
McDrv	Een McChicken. Dat was uw bestelling?	
Klant	Ja	
McDrv	Dan wordt het 5 gulden en een dubbeltje. Dan mag u doorrijden naar het eerste raam alstublieft.	
McDrv	Sorry dat ik u zo lang heb laten wachten, maar u mag uw bestelling geven hoor?	








2

Klant	Ik wou graag een cheeseburger	
McDrv	Ja	
Klant	En twee keer een McChicken Menu.	
McDrv	Twee keer?	
Klant	Ja, eentje met sinas en eentje met cola.	
McDrv	Ja	
Klant	En allerbei met dubbele frietsaus.	
McDrv	Dus vier in totaal?	
Klant	Ja, dat was het.	
McDrv	Dus wat op het scherm staat dat klopt nu?	
Klant	Ja.	
McDrv	Dan wordt het 22 gulden 65 en mag u doorrijden naar het eerste raam alstublieft.	









3

McDrv	Goedenavond welkom bij Mac Donald's, mag ik uw bestelling, alstublieft?	
Klant	Een Cheeseburger Menu.	
McDrv	Wat wilt u daarbij drinken?	
Klant	[ah] een milkshake vanille.	
McDrv	En had u nog frietsaus bij de friet van het menu gewild?	
Klant	Twee maal.	
McDrv	Dan wordt het 8 gulden en 95 centjes. Mag u doorrijden naar het eerste raam alstublieft.	







4





McDrv	Goedenavond welkom bij Mac Donald's, mag ik uw bestelling, alstublieft?	
Klant	Ik wou twee Mac Deluxe/	
McDrv	Is uw bestelling zo compleet?	
Klant	Wat zegt u!	
McDrv	Is uw bestelling zo compleet?	 
Klant	Ja hoor.	
McDrv	7gulden50 en dan mag u door naar het tweede raam alstublieft.	 

5












McDrv	Goedenavond welkom bij Mac Donald's, mag ik uw bestelling, alstublieft?	
Klant	Mag ik een Big Mac Menu met een extra hamburger.	
McDrv	Wat wilt u daarbij drinken?	
Klant	Cola.	
McDrv	En frietsaus bij de friet?	
Klant	[stilte]	
McDrv	Wilt u frietsaus bij de friet?	 
Klant	Nee hoor geen.	
McDrv	Het wordt 11,45. Mag u door naar het tweede raam alstublieft.	 

6



McDrv	Goedenmiddag welkom bij Mac Donald's, mag ik uw bestelling?	
Klant	Ja goedenmiddag ik wou een happy Meal een hamburger.	
McDrv	Wat wilt u daarbij drinken?	
Klant	Een Jus d'orange. Ik wou een Fish'Filet zonder kaas.	
McDrv	Sorry mevrouw wat zegt u?	
Klant	Eeneh broodje Fish'Filet met vis maar zonder kaas.	
McDrv	Ik geef em door aan de keuken.	
Klant	En dan wil ik ook nog graag eeneh salade met [ah] aardappel en kip.	
McDrv	Salade is of met ham en kalkoen of met kip?	
Klant	[ah] doe dan maar met kip.	








McDrv	Wat wilt u voor dressing erbij?	
Klant	Wat heb je allemaal?	
McDrv	Fijne kruiden, Bluecheese, *** of finitraite.	
Klant	Doe maar fijne kruiden.	
McDrv	Frietsaus bij de friet?	
Klant	Ja	
McDrv	16 gulden 30. U mag door naar het tweede loket.	

7







McDrv	Goedenavond welkom bij Mac Donald's, mag ik uw bestelling, alstublieft?	
Klant	Ja [eh] mag ik twee Happy Meals met stukjes kip.	
McDrv	Wat wilt u daarbij drinken?	
Klant	[eh] een Mac Deleuxe een medium patat frietjes.	
McDrv	Wat wilt u drinken bij de Happy Meals?	
Klant	[eh] twee chocomel	
McDrv	Wilt u frietsaus bij de menu's?	
Klant	Ja [eh] dubbel bij de Mac Deluxe en [eeeeh] bij de Happy Meals ook.	
McDrv	Anders nog iets?	
Klant	Ik kan niet zien wat [eh] er staat niets op de film.	
McDrv	Ik ga het voor u opnoemen. Ik heb voor u staan twee Happy Meals met KipNuggets chocmel en dubbel frietsaus en medium friet een Mac Deluxe.	
Klant	[eh] met dubbel frietsaus ook.	
McDrv	Sorry?	
Klant	Vier frietsaus in totaal.	
McDrv	Ja [eh] nee in totaal heeft er zes.	
Klant	Zes?	
McDrv	Ja bij alles dubbel u wilde ook bij de McNuggets althans bij de Happy Meals dubbel frietsaus.	
Klant	Is goed zo.	
McDrv	Ja, dan wordt het 23 gulden 40 en dan mag u doorrijden naar het tweede raam alstublieft.	

8





McDrv	Goedenavond welkom bij Mac Donald's, mag ik uw bestelling, alstublieft?	
Klant	McChicken Menu	










McDrv	Wat wilt u daarbij drinken?	
Klant	Cola	
McDrv	Frietsaus?	
Klant	Ja.	
McDrv	Ander nog iets?	
Klant	En [eh] McNuggetsKip 6 stuks	
McDrv	Wat voor saus?	
Klant	Wat is er?	
McDrv	Zoetzuur, mosterd, kerrie of barbecue?	
Klant	Doe maar zoetzuur.	
McDrv	Was uw bestelling zo compleet?	
Klant	Ja hoor.	
McDrv	13 gulden 90 u mag door naar het tweede raam alstublieft.	

9

McDrv	Een hele goedenavond welkom bij Mac Donald's, mag ik uw bestelling?	
Klant	Ja [eh] doe maar [eh] vier cheeseburgers, twee grote patat, zes kuipjes en [eh] doe maar [eh] KipNuggets 9 stukjes.	
McDrv	Wat voor saus wilt u daarbij?	
Klant	[ehm] monsterd.	
McDrv	En hoeveel frietsuas wilde u in totaal?	
Klant	Zes.	
McDrv	Ok. Wordt het 25 gulden 65 u mag door naar het tweede raam alstublieft.	
Klant	Ooh [eh] doe maar [ehm] een grote cola zonder ijsblokjes.	
McDrv	Wordt het 29 gulden 20 en u mag door naar het tweede raam alstublieft.	

10

McDrv	Goedenavond, mag ik uw bestelling alstublieft?	
Klant	Goedenavond, ik wou een *** met stukjes kip en een sinas en [ah] Cheeseburger Menu met cola	
McDrv	Een Cheeseburger Menu met cola?	
Klant	Ja en [ah] een Quater Pounder Menu met sinas en 6 stukjes kip.	
McDrv	Met mosterd, kerrie of frietsaus?	
Klant	En [ah] heeft u geen zoetzure?	

McDrv	Helaas meneer anders had ik hem wel opgenoemd. Komt vanavond binnen.	
Klant	Doe maar kerrie.	
McDrv	Wat wilt u bij de [ah] [brr] effe kijke hoor bij de *** drinken?	
Klant	sinas	
McDrv	En [ah] wilt u nog frietsaus bij de friet?	
Klant	Ja	
McDrv	Hoeveel?	
Klant	Drie maal	
McDrv	Dat was uw bestelling?	
Klant	Ja [ah] heeft u nog van de McNuggets met die stukjes kip?	
McDrv	Ja die heb ik.	
Klant	Prima	
McDrv	Wou u nog saus bij de McNuggets hebben?	
Klant	Alleen maar van die zoetzuur.	
McDrv	Ja helaas dan wordt het 28 gulden en 30 cent en dan mag u doorrijden naar het eerst raam alstublieft.	 
Klant	Goed.	

Appendix IV Wizard actions

Animation	Assign Prompts	Description	Supports Speaking	Sound Effects
Acknowledge	Give confirmation	Nods head	Yes	No
Alert	Give hints or warning	Raises eyebrows, open eyes bigger	Yes	Yes
Announce	Announce a price reduction action etc	Put both hands around the mouth	Yes	Yes
Ask	Ask for additional order/info	Raise left eyebrow and open the mouth partially	Yes	No
Attentive	Pays attentions to the customer	Raise eyebrows, open the	No	No
Blink		Blinks eyes	Yes	No
Confused	Cannot understand the customer	Confused Expression	Yes	No
Congratulate	Congratulate the customer	Smiles and applauds	Yes	Yes
DontRecognize_1	Cannot understand the customer	Holds right hand to ear, a question mark appears	Yes	No
DontRecognize_2	Cannot understand the customer	Holds left hand to ear, a question mark appears	Yes	No
Hearing_1	The customer speaks	Turns head left and holds right hand to right ear	No	No
Hearing_2	The customer speaks	Turns head right and holds right hand to right ear	No	No
Hearing_3	The customer speaks	Tiles head left	No	No
Hearing_4	The customer speaks	Tiles head right	No	No
Hide	When the talk finishes	Sink until disappears	No	No
Idle1_1	Idle	Blinks	No	No
Idle1_2	Idle	Glances left and blinks	No	No
Idle1_3	Idle	Glances up and blinks	No	No
Idle2_1	Idle	Twist head right	No	No
Idle2_2	Idle	Breaths	No	No
Idle3_1	Idle	Glances right and blinks	No	No
Idle3_2	Idle	Look down and see the hands	No	No
LookDown	To see things downside	Looks down	No	No
LookDownReturn	Go back	Returns to neutral position	No	No
LookLeft	To see things at left side	Looks left	No	No
LookLeftReturn	Go back	Returns to neutral position	No	No
LookRight	To see things at left side	Looks right	No	No
LookRightReturn	Go Back	Returns to neutral position	No	No
LookUp	To see things upside	Looks up	No	No
LookUpReturn	Go back	Returns to neutral position	No	No
MoveDown	To move downward	Moves down	No	No
MoveLeft	To move towards the left	Moves left	No	No
MoveRight	To move towards the right	Moves right	No	No
MoveUp	To move upward	Moves up	No	No
Neer	Give a negative answer.	Shake head	Yes	No
Process	System processes		No	No
Processing	System continue processing		No	No

RestPose	None	Neutral position	No	No
Sad	Yes, using Exit branches	Sad expression	Yes	No
Search	Search	Looks around	Yes	No
Show	When new customer comes	Appears from upside	No	Yes
Smile	Greet the customer	Smiles	Yes	No
Sorry	Cannot satisfy customer's demands	Sorry expression	Yes	No
StartListening	The customer begins to speak	Turn head left and puts right hand to right ear	No	No
StopListening	The customer stops speaking	Puts hands over ears	No	No
Suggest	Give customer suggestions.	Smiles and displays lightbulb	Yes	No
Sure	Satisfy the customer	Raise eyebrows and smiles	Yes	No
Surprised	Surprise at what the customer says	Looks surprised	Yes	No
Suspicious	Do not believe	Suspicious expression	Yes	No
Think_1	Think	Roll the eyeballs over	No	No
Think_2	Think	Raise right eyebrow, narrow eyes and close mouth firmly	No	No
Tired	Feel tired	Close eyes partially	No	No
Uh_Oh	Made a mistake	Raise eyebrow and open mouth partially at half width	Yes	No
Uncertain	Not sure	Sink eyebrows	Yes	No
Yes	Give customer a positive answer	Raise eyebrows and smiles	Yes	No

Appendix V Database

Menu

<i>Field</i>	<i>Type</i>	<i>Description</i>
Code	Text	Menu item code
Name	Text	Menu item name
Category	Text	McDrive menu category
Icon	LongBinary	Icon symbol of menu item
Picture	LongBinary	Picture of menu item
Attr_type	Text	Classify menu items according their attributes

Attribute

<i>Field</i>	<i>Type</i>	<i>Description</i>
Code	Text	Attribute code
Name	Text	Attribute value
Type	Text	Attribute type. For example attributes <i>small</i> and <i>large</i> belong to type <i>Size</i> .
Icon	LongBinary	Icon symbol of attribute
Picture	LongBinary	Picture of attribute. Only some attributes can have a picture.

Alias

<i>Field</i>	<i>Type</i>	<i>Description</i>
Code	Text	Alias code
Name	Text	Alias text
Alias	Text	Code of alias owner

Expression

<i>Field</i>	<i>Type</i>	<i>Description</i>
Code	Text	Expression code.
Name	Text	Expression name. Some expressions have same meaning, so they have same name but different code.
Action	Text	Corresponding wizard action of this expression
Smiley	LongBinary	Smiley Icon of expression
Show	Text	Whether added to keyboard
Type	Text	Expression type. For example type <i>Illustrators</i> .
Level	Text	Expression of same name can have different level. Expression of Level 1 will be our first choice.

Commando

<i>Field</i>	<i>Type</i>	<i>Description</i>
Code	Text	Control code
Name	Text	Control name
Icon	LongBinary	Icon symbol of control
Expression	Text	Some control item can have default expression.

Customer

<i>Field</i>	<i>Type</i>	<i>Description</i>
Plate	Text	The license plate of the customer
Order 1	Text	The most recent order of the customer.
Order 2	Text	The last most recent order of the customer.
Date	Date	The last visited datum of the customer