

Regio: 1
Gespreksnummer: 11845602
Gespreksduur: 33.10 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachttijd: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

2: goedenavond reisinformatie
1: hallo met [voornaam+achternaam] ik wil graag weten hoe laat de trein nu gaat vanuit al met de
Driebergen Zeist naar Utrecht
2: vanaf nu
1: ja
2: ja
2: twintig uur negentwintig
1: ja
2: en anders twintig uur vijfenveertig
1: en; daarna
2: <twintig> <uur> <<nee>> eenentwintig uur negentwintig pas weer

1: ja oké bedankt #1 ja da: g
2: ja #1 tot uw dienst dag me
1: [noise] [tone] [tone]

1: ja in[uh] de
2: u kunt om zeven uur
1: ja
2: aankomst Den Haag CS acht uur op de trein naar Rotterdam nou die gaat
2: ja
2: dan stapt u daar over op de trein naar Rotterdam nou die gaat

1: ja
2: dan bent u om[uh] acht uur tweeënveertig bij het station Leidschendam Voorburg
1: ja
2: en dan bent u om[uh] acht uur tweeënveertig bij het station Zoetermeer Den Haag ik
1: nou perfect en is dat[u:h] <de> de sprinter Zoetermeer Den Haag ik
2: en dan kijken naar dat Rotterdam Zuidplein
1: nee het is niet de sprinter naar dat Rotterdam ja hij gaat dat is het
2: ik al even te kijken naar dat Rotterdam ja hij gaat dat is het
1: oh ja dus het de Oost Indië en dan station in Den Haag
2: Rotterdam Hofplein
1: Rotterdam Hofplein
2: ja en [uh] u heeft dus
1: ja en dan station du
2: ja en dan twaalf tweeënveertig uur of [u:h] ne
1: oké en [u:hm] terug naar Utrecht u om twaalf tweeënveertig
2: [uh] twaalf negentwintig kunt u ter
1: oké en [u:hm] terug naar Utrecht u om twaalf tweeënveertig
2: [uh] twaalf negentwintig kunt u ter

1: oké en [u:hm] terug naar Utrecht u om twaalf tweeënveertig
2: [uh] twaalf negentwintig kunt u ter
1: oké en [u:hm] terug naar Utrecht u om twaalf tweeënveertig
2: [uh] twaalf negentwintig kunt u ter
1: oké en [u:hm] terug naar Utrecht u om twaalf tweeënveertig
2: [uh] twaalf negentwintig kunt u ter

1: #1 waar vandaan
2: van Utrecht naar Alphen aan den Rijn ik neem aan dat de trein naar Leiden is
1: ja
2: vroeg ik me af[uh] hoe laat hij ging
1: die gaat om tien over heel en tien over half
2: tien over heel en tien over half #2 #3 #4 #5
1: oh dat is # is vijftwintig minuten
2: #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn #3 oké # en die stopt gewor
1: #4 oké # prima bedankt #5 da: g
2: #5 graag gedaan # da: g
1: [noise] [tone] [tone]

Moeilijkheid van de transcriptie: 4
Omgevingsgeluiden spreker 1: 4
Omgevingsgeluiden spreker 2: 4
Geslacht spreker 1: 4
Geslacht spreker 2: 4
Kwaliteit van telefonische verbinding: 4
Luidheid van de stem van spreker 1: 4
Luidheid van de stem van spreker 2: 4
Accent van spreker 1: 4
Accent van spreker 2: 4
Algemeen commentaar op gesprek: 4
Wachttijd: 4
Transliteratiedatum: 4
Transliteratietijd: 4

'O.V.R.'

Dialogue management

Master Thesis
Alexandra Peters
August 2000



Delft University of Technology
Faculty of Information Technology and Systems
Knowledge Based Systems Group

Graduation Committee:
Prof. Dr. H. Koppelaar
Prof. Dr. Ir. E.J.H. Kerckhoffs
Drs. Dr. L.J.M. Rothkrantz

Regio: 1
Gespreksnummer: 1789157
Gespreksduur: 96 (s)
Categorie: trein_info
Wachttijd: 17:14
Transliteratiedatum: 17-09-95
Transliteratietijd: 17:14
Algemeen commentaar op gesprek: 53
Wachttijd: 18-0-
Transliteratiedatum: 18-0-
Transliteratietijd: 17:47

Regio: 1
Gespreksnummer: 11845702
Gespreksduur: 63.99 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 11:54 - 11:57
Algemeen commentaar op gesprek: 53
Wachttijd: 18-0-
Transliteratiedatum: 18-0-
Transliteratietijd: 17:47

Abstract

Many people are using the telephone as a medium to obtain information. They call a specific information service and ask the telephone operator specific questions. The telephone operator will try to answer these questions as good as possible with the knowledge she has regarding the specific knowledge domain. A dialogue emerges between the telephone operator and the client.

Most information providers are having problems handling the increasing number of clients. To still be able to help all these clients the dialogues could be made more efficient. A way to do this is to keep the dialogue as short as possible. This can be achieved by asking only the necessary items for providing the correct information. This way the telephone operator manages the dialogue in a directive manner that makes the dialogue seem very unfriendly. However, other clients want to take the initiative and require human friendly dialogues, though this implies longer and non-directive dialogues. The telephone operator has to find the right balance between efficiency and friendliness. An example of an information-providing organisation that has to deal with this situation is OVR ('Openbaar Vervoer Reisinformatie'). The Dutch company OVR provides information concerning public transport services in the Netherlands by telephone.

To make the dialogue between the client and the telephone operator more efficient and effective while maintaining a high client appreciation the current dialogue management is analysed. One of the main disadvantages is that there is no underlying model or theory available as foundation of the dialogues. To execute the dialogue management task properly, knowledge of the dialogue model has been extracted.

A corpus-based approach is used to achieve this, and included the following steps:

- The corpus of 200 transliterated OVR-dialogues is analysed.
- Transliterated dialogues are not suitable for statistical processing; therefore, the dialogues are coded.
- An OVR-dialogue model is constructed and validated.
- The knowledge used by the operator during the dialogue is extracted and modelled.
- With the acquired knowledge a prototype of a knowledge-based training environment is designed and implemented. With this training environment telephone operators are able to train parts of and the complete dialogue.

Acknowledgements

This thesis is part of my graduation project in the Knowledge Based Systems group headed by Prof. Dr. H. Koppelaar. This group is part of the faculty of information technology and systems at Delft university of technology.

First, I would like to thank Leon Rothkrantz for his assistance, support, encouragement and patience during all the stages of my graduation.

Special thanks to Jens, thank you for everything and much more.

Thanks to my family for their ongoing support and belief in me. They told me that as long as I tried my best, everything would turn out fine. They were right.

Thank you, Caroline Scheepmaker for always believing in me.

Also, I would like to thank Leslie Moll for lending me a more powerful computer to do the things much faster than my computer ever could have done.

And last but not least I would like to thank Noppes and Ocho for being there for me when I needed company (and even when I did not).

If I forgot anyone who helped and supported me in anyway, then I would just like to say: "Thank you!"

I hope you enjoy reading this.

Alexandra Peters
Delft, August 2000.

Contents

Abstract	i
Acknowledgements	ii
Contents	iii
CHAPTER ONE Introduction	1
CHAPTER TWO Dialogue and training aspects	5
2.1 Dialogue aspects	5
2.1.1 Information seeking dialogues	5
2.1.2 Communication in a dialogue	6
2.1.3 Dialogue styles	7
2.2 Training aspects	8
2.2.1 Learning and scripts	8
2.2.2 Instruction methods	10
2.2.3 Requirements for human factors	11
2.2.4 User interfaces	12
CHAPTER THREE Dialogue coding	15
3.1 Introduction	15
3.2 Coding scheme	17
3.2.1 Coding constraints	17
3.2.2 Dialogue acts	20
3.2.2.1 Operator's dialogue acts	20
3.2.2.2 Client's dialogue acts	24
3.2.2.3 Sub-coding and sub-sub-coding	28
3.3 Coding supporting tool	29
3.3.1 Purpose and functionality of the coding tool	29
3.3.2 Technical description	33
3.4 Frequencies dialogue acts	35
3.5 Additional comments	38

CHAPTER FOUR OVR-dialogue model	41
4.1 Top-level OVR-dialogue model	41
4.2 Lower level dialogue model	42
4.2.1 Explanation of the moves in the dialogue model	44
4.2.2 Transition frequencies	50
4.2.3 Problem classes not included in this model	51
4.2.4 Dialogue model with transition frequencies	56
4.3 Dialogue management from the operator's viewpoint	58
4.3.1 Strategies	58
4.3.2 The dialogue from the operator's perspective	59
4.3.2.1 Description of the thoughts during a dialogue	59
4.3.2.2 Explanation of tasks	60
4.3.3 Computational dialogue management	63
CHAPTER FIVE Knowledge of the OVR-domain	66
5.1 The operator's knowledge and skills	66
5.1.1 Knowledge	66
5.1.2 Skills	68
5.2 Travel planner	69
5.3 Knowledge-based structures	71
5.3.1 Knowledge in a price dialogue	72
5.3.1.1 Price knowledge structure	73
5.3.1.2 Analysis of human-human price dialogues	76
5.3.1.3 Flowchart for the price dialogue training model	80
5.3.2 Knowledge in a dialogue about specific train travel routes	82
5.4 Information presentation	86
5.4.1 Question about train ticket prices	86
5.4.1.1 How is the price of a train ticket determined?	86
5.4.1.2 In what way is the price information presented to the client?	86
5.4.2 Question about a specific train travel route	88
5.4.2.1 Which travel route is chosen by the operator?	88
5.4.2.2 In what way is the travel route presented to the client?	89
CHAPTER SIX Dialogue training environment	91
6.1 Functional Design	91
6.1.1 Education	92
6.1.2 Trainee	100
6.1.3 Trainer/Supervisor	100
6.2 Technical design	101
6.2.1 Overview modules	101

6.2.2 <i>Module communication</i>	102
6.3 <i>Implementation Prototype</i>	103
CHAPTER SEVEN Conclusions and recommendations	109
7.1 <i>Conclusions</i>	109
7.2 <i>Recommendations</i>	110
References	109
List of figures	114
List of tables	116
APPENDIX A Paper ‘Dialogue control in the ALPARON system’	
APPENDIX B The coded corpus of OVR-dialogues	

Regio: 1
Gespreksnummer: 11845602
Gespreksduur: 33.10 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachttijd: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

1: ja in[uh] de
2: u kunt om zeven uur
2: #1 ja
Omgevingsgel1: ja
Omgevingsgel2: dan stapt u daar over op de trein naar Rotterdam nou die gaat om acht uur vijftentwintig
Geslacht spre1: ja
Geslacht spre2: en dan bent u om[uh] acht uur tweeënveertig bij het station Leidschendam Voorburg
Kwaliteit van 1: nou perfect en is dat[u:h] <de> de sprinter Zoetermeer Den Haag ik
Vloeiendheid v2: nee het is niet de sprinter naar dat Rotterdam Zuidplein
Luidheid van de z2: ik al even te kijken naar dat is het
Helderheid van de c2: Hofplein Bergweg Rotterdam ja hij gaat dat is het
Accent van spreker 1: Rotterdam Hofplein
Luidheid van de s2: ja en [uh] u heeft dus eerst als u in Den Haag
Helderheid van de s2: Oost Indië en dan station Voorburg het Loo en dat.
Algemeen commentaar 1: oh ja dus het derde station dus
2: [mmm] ja
1: oké en [u:hm] terug naar Utrecht rond een uur of [u:h] nu
2: [uh] twaalf negentwintig kunt u terug bijvoorbeeld dat is
is dan wel de sprinter dan bent u om twaalf tweeënveertig bij halt
loopbrug over naar station Zoetermeer en dan neemt u daar om twaa
en dan bent u in Utrecht om dertien achtentwintig
oké en eentje eerder of later
ik alleen maar bussen vanaf daar maar dat bedoelt u waarschijnlijk
e eerder <of> eentje eerder is elf negenvijftig met de[uh] t

2: #1 waar vandaan #
1: van Utrecht naar Alphen aan den Rijn ik neem aan dat de trein naar Leiden is
2: ja
1: vroeg ik me af[uh] hoe laat hij ging
2: die gaat om tien over heel en tien over half
1: tien over heel en tien over half #2 (()) #
2: #2 reistijd # is vijftwintig minuten
1: oh dat is (()) (()) tien over heel en tien over half
2: #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn #3 oké # en die stopt gewo
1: #4 oké # prima bedankt #5 da:g #
2: #5 graag gedaan # da:g
1: [noise] [tone] [tone]

Moeilijkheid van de transcriptie:
Omgevingsgeluiden spreker 1:
Omgevingsgeluiden spreker 2:
Geslacht spreker 1:
Geslacht spreker 2:
Kwaliteit van telefonische verbinding:
Vloeiendheid van gesprek:
Luidheid van de stem van spreker 1:
Helderheid van de stem van spreker 1:
Accent van spreker 1:
Luidheid van de stem van spreker 1:
Helderheid van de stem van spreker 1:
Accent van spreker 2:
Algemeen commentaar op gesprek:
11848102
176.28
trein_info
96 (s)
18-09-95
17:14 - 17:27
Wachttijd:
Transliteratiedatum:
Transliteratietijd:
18-09-95
17:47

Regio: 1
Gespreksnummer: 11848103
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 11:54 - 11:57

Regio: 1
Gespreksnummer: 11845601
Gespreksduur: 50.06 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 -

CHAPTER ONE

Introduction

Regio: 1
Gespreksnummer: 11845601
Gespreksduur: 50.06 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 -

1: ja in[uh] de
2: u kunt om zeven uur
2: #1 ja
Omgevingsgel1: ja
Omgevingsgel2: dan stapt u daar over op de trein naar Rotterdam nou die gaat om acht uur vijftentwintig
Geslacht spre1: ja
Geslacht spre2: en dan bent u om[uh] acht uur tweeënveertig bij het station Leidschendam Voorburg
Kwaliteit van 1: nou perfect en is dat[u:h] <de> de sprinter Zoetermeer Den Haag ik
Vloeiendheid v2: nee het is niet de sprinter naar dat Rotterdam Zuidplein
Luidheid van de z2: ik al even te kijken naar dat is het
Helderheid van de c2: Hofplein Bergweg Rotterdam ja hij gaat dat is het
Accent van spreker 1: Rotterdam Hofplein
Luidheid van de s2: ja en [uh] u heeft dus eerst als u in Den Haag
Helderheid van de s2: Oost Indië en dan station Voorburg het Loo en dat.
Algemeen commentaar 1: oh ja dus het derde station dus
2: [mmm] ja
1: oké en [u:hm] terug naar Utrecht rond een uur of [u:h] nu
2: [uh] twaalf negentwintig kunt u terug bijvoorbeeld dat is
is dan wel de sprinter dan bent u om twaalf tweeënveertig bij halt
loopbrug over naar station Zoetermeer en dan neemt u daar om twaa
en dan bent u in Utrecht om dertien achtentwintig
oké en eentje eerder of later
ik alleen maar bussen vanaf daar maar dat bedoelt u waarschijnlijk
e eerder <of> eentje eerder is elf negenvijftig met de[uh] t

2: #1 waar vandaan #
1: van Utrecht naar Alphen aan den Rijn ik neem aan dat de trein naar Leiden is
2: ja
1: vroeg ik me af[uh] hoe laat hij ging
2: die gaat om tien over heel en tien over half
1: tien over heel en tien over half #2 (()) #
2: #2 reistijd # is vijftwintig minuten
1: oh dat is (()) (()) tien over heel en tien over half
2: #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn #3 oké # en die stopt gewo
1: #4 oké # prima bedankt #5 da:g #
2: #5 graag gedaan # da:g
1: [noise] [tone] [tone]

Moeilijkheid van de transcriptie:
Omgevingsgeluiden spreker 1:
Omgevingsgeluiden spreker 2:
Geslacht spreker 1:
Geslacht spreker 2:
Kwaliteit van telefonische verbinding:
Vloeiendheid van gesprek:
Luidheid van de stem van spreker 1:
Helderheid van de stem van spreker 1:
Accent van spreker 1:
Luidheid van de stem van spreker 1:
Helderheid van de stem van spreker 1:
Accent van spreker 2:
Algemeen commentaar op gesprek:
11848102
176.28
trein_info
96 (s)
18-09-95
17:14 - 17:27
Wachttijd:
Transliteratiedatum:
Transliteratietijd:
18-09-95
17:47

Regio: 1
Gespreksnummer: 11845601
Gespreksduur: 50.06 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 -

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachttijd: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

Regio: 1
Gespreksnummer: 11845602
Gespreksduur: 33.10 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95

CHAPTER ONE

Introduction

More and more people are using the telephone as a medium to obtain information. They call a specific information service and ask the telephone operator specific questions. The telephone operator will try to answer these questions as good as possible with the knowledge she has regarding the specific knowledge domain. A dialogue emerges between the telephone operator and the client. This dialogue is called an *information seeking dialogue*.

Most information providers are having problems handling the increasing number of clients. To still be able to help all these clients the dialogues could be made more efficient. A way to do this is to keep the dialogue as short as possible. This can be achieved by asking only the necessary items for providing the correct information. This way the telephone operator manages the dialogue in a directive manner that makes the dialogue seem very unfriendly. However, other clients want to take the initiative and require human friendly dialogues, though this implies longer and non-directive dialogues. Therefore, the telephone operators have to ask the essential items without losing track of the client's appreciation of the dialogue. The telephone operator has to find the right balance between efficiency and friendliness.

An example of an information-providing organisation that has to deal with the situation described above is OVR ('Openbaar Vervoer Reisinformatie'). The Dutch company OVR provides information concerning public transport services in the Netherlands by telephone. This information is provided by more than 400 telephone operators. The number of requests for information has increased dramatically in the last few years reaching its full capacity. Consequently, many clients are waiting quite long before they are being helped. A long waiting period costs clients a lot of time and money. Because of this, clients can get easily annoyed and will be reluctant to call again. Therefore, one of the objectives of OVR is to have efficient and effective dialogues. However, an improvement of the efficiency and effectiveness might result in a lower client's appreciation, because another objective of OVR is a high client appreciation. One of the reasons is that a satisfied client will be a regular client.

There are several possibilities to keep queuing to a minimum and to keep a high client appreciation. One possibility is to increase the number of telephone operators. However, as said before the maximum capacity has been reached and therefore expanding is not a solution. Another option is to use an automatic information system based on speech recognition. Such a system already exists within the public domain. At this moment this system is still not good enough to be used by everyone because it is incapable of reaching the same flexibility a telephone operator has. Just the people who are very familiar with travelling by public transport find the system easy to work with. Another disadvantage is that this version only contains information about travelling by train. Therefore, until this system is useful for everyone another possibility is to focus on how to get the dialogue between the client and the telephone operator more efficient and effective while maintaining a high client appreciation. Because telephone operators will always be necessary, optimising dialogue management is useful either way.

A possible option to improve dialogue management, is to develop a different dialogue model. In the last few years, the OVR dialogue model has evolved into a matured product with a high client appreciation. The development and introduction of a new dialogue model that improves efficiency and effectiveness but also maintains a high client appreciation and the current OVR principles is not easy, if not impossible. Studies of the current dialogues show that they are effective and efficient on the average but there are still dialogues far beyond the average. Therefore, optimising the current dialogue management is an option to get more efficient and effective dialogues. A way to do this is to train the telephone operators, which score regularly below the average with certain aspects, on their shortcomings.

In the current training situation, a starting telephone operator is given an initial training. During this training, the telephone operators learn how to use the travel planner and learn how to manage a dialogue in an effective and efficient way. The travel planner is a computer program that calculates an optimal travel scheme based on several criteria given by a user. Some of these criteria are destination place and travel date. More information about the travel planner is described later. The operators are given an overview of all sorts of travel information queries, a basic instruction in dialogue management and they become accustomed to the principles of OVR. In role playing games, dialogues are being practised and if necessary, a trainer will provide feedback.

After the initial phase, OVR has chosen a *training-on-the-job* approach. The telephone operators are regularly trained by means of group meetings and monitored during regular work. Nevertheless, there are some shortcomings. On the job monitoring of telephone operators provides little or no room for giving feedback. Furthermore, when a telephone operator has to have training on a certain aspect of the dialogue, clients who address this aspect are not available by the push of a button. In addition, the combination of work, training and concentration reduces the quality of the on-line work. The creation of a realistic off-line surrounding in which the computer simulates clients is preferable. This way it is possible to have an additional individual training next to the group meetings.

Objective and approach

There are many ways to ask questions and many ways to answer a question. Several dialogue structures have assumedly developed within OVR. These possible underlying structures will be referred to as 'scripts'. The telephone operator and the client have much individual freedom within the script from which they can deviate. To manage the dialogues efficient and effective while maintaining a high client appreciation, the most adequate solution is not restricting the freedom. Therefore, the operator has to manage the dialogue while making maximum use of the freedom.

To execute this dialogue management task properly knowledge of the dialogue scripts is necessary. One of the main disadvantages is that there is no underlying model or theory available as foundation of the dialogues. Even the goals are not stated in an explicit way. The OVR-approach is very pragmatic. Making these scripts explicit and training the telephone operators in using these scripts is the most important objective of this thesis.

A corpus-based approach is used to achieve this, and includes the following steps:

- The corpus of 200 transliterated OVR-dialogues will be analysed to reveal possible underlying structures.
- The OVR-dialogues will be coded. Transliterated dialogues are not suitable for statistical processing; therefore, further coding is necessary.
- A possible OVR-dialogue model will be constructed and validated
- The knowledge used by the operator during the dialogue will be extracted and modelled
- With the acquired knowledge a prototype of a knowledge-based training environment will be designed and implemented. With this training environment telephone operators are able to train parts of and the complete dialogue.

The remainder of this thesis is structured as described below.

Chapter 2 describes several aspects of dialogues and training.

Some of these aspects are the description of information-seeking dialogues, communication types in a dialogue and user interface aspects.

Chapter 3 describes the coding of the corpus.

The dialogue acts, used in the coding, and the coding tool are described. The coding process is evaluated.

Chapter 4 describes the resulting OVR-dialogue model.

The model is constructed and validated and the dialogue is described from the operator's perspective.

Chapter 5 describes the knowledge of the OVR-domain.

The operator's knowledge that she needs and uses during the dialogue is extracted for two types of questions. These question types are a price question and a specific train travel route question.

Chapter 6 describes the design and implementation of the dialogue training environment prototype. The design is divided in a functional and a technical design. The implementation of a part of the design is described.

Chapter 7 describes the conclusions and recommendations.

Appendix A contains the article 'Dialogue control in the ALPARON system' with the main results of the current research project.

Appendix B contains the database with the transliterated dialogues and their corresponding coding.

For the remainder of the document, 'telephone operator' will be referred to as 'operator' and for reasons of practicality the operator will be indicated as 'she' and the client as 'he'. I am not implying that all operators are female but it is a liberty I permit myself for this document.

regio: 1
gespreksnummer: 11845602
gespreksduur: 33.10 (s)
categorie: trein_info
wachtwoord: 32 (s)
transliteratiedatum: 18-09-95

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachtwoord: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

CHAPTER TWO

Dialogue and training aspects

1: ja in[uh] de
2: u kunt om zeven uur
***** 1: ja
Moeilijkheid 2: aankomst Den Haag CS acht uur achtentwintig
Omgevingsgeluid 1: ja
Omgevingsgeluid 2: dan stapt u daar over op de trein naar Rotterdam nou die gaat om achtentwintig
Geslacht spreker 1: ja
Geslacht spreker 2: en dan bent u om[uh] acht uur tweeënveertig bij het station Leidschendam Voorburg
Kwaliteit van 1: nou perfect en is dat[u:h] <de> de sprinter Zoetermeer Den Haag ik
Vloeiendheid van 2: nee het is niet de sprinter naar dat Rotterdam Zuidplein
Luidheid van de stem 2: ja en [uh] u heeft dus eerst als u in Den Haag
Helderheid van de stem 2: Oost Indië en dan station Voorburg het Loo en dan.
Accent van spreker 1: oh ja dus het derde station dus
Algemeen commentaar 1: oké en [u:hm] terug naar Utrecht rond een uur of [u:h] nu
2: [mm] ja ik alleen maar bussonderzoek en dan neemt u daar om twaalf tweeënveertig bij halte
1: oké en twaalf negenentwintig dan bent u om twaalf tweeënveertig bij halte
2: [uh] twaalf negenentwintig dan bent u om twaalf tweeënveertig bij halte
1: is dan wel de sprinter dan bent u om twaalf tweeënveertig bij halte
2: loopbrug over naar station Zoetermeer en dan neemt u daar om twaalf tweeënveertig bij halte
1: ik heb geen idee [/laughter]
2: #1 waar vandaan
1: van Utrecht naar Alphen aan den Rijn ik neem aan dat de trein naar Leiden is
2: ja
1: vroeg ik me af[uh] hoe laat hij ging
2: die gaat om tien over heel en tien over half
1: #2 reistijd # is vijftientwintig minuten
2: #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn [laughter] #4 ja hoor
1: #4 oké # prima bedankt #5 da: g
2: #5 graag gedaan # da: g
1: [noise] [tone] [tone]

Moeilijkheid van de transcriptie: 4
Omgevingsgeluiden spreker 1: 4
Omgevingsgeluiden spreker 2: 4
Geslacht spreker 1: 4
Geslacht spreker 2: 4
Kwaliteit van telefonische verbinding: 4
Vloeiendheid van gesprek: 4
Luidheid van de stem van spreker 1: 4
Helderheid van de stem van spreker 2: 4
Accent van spreker 1: 4
Luidheid van de stem van spreker 1: 4
Helderheid van de stem van spreker 1: 4
Accent van spreker 2: 4
Algemeen commentaar op gesprek: 4
1: 11848102
2: 176.28
3: trein_info
4: 96 (s)
5: reksnummer: 18-09-95
6: reksduur: 17:14 - 17:27
7: categorie: Utrecht - Alphen aan den Rijn
8: wachtwoord: 53
9: wachtwoord: ACC
10: wachtwoord: Helderheid
11: wachtwoord: Algemeen commentaar op gesprek: 17:47
12: reksnummer: 11848102
13: reksduur: 27.04 (s)
14: categorie: trein_info
15: wachtwoord: 231 (s)
16: transliteratiedatum: 19-09-95
17: transliteratietijd: 11:54 - 11:57
18: regio: 1
19: gespreksnummer: 11848102
20: gespreksduur: 27.04 (s)
21: categorie: trein_info
22: wachtwoord: 231 (s)
23: transliteratiedatum: 19-09-95
24: transliteratietijd: 11:54 - 11:57

CHAPTER TWO

Dialogue and training aspects

In this chapter, several aspects are described to better understand the words 'dialogue' and 'training'. The first section deals with the dialogue and the second with the training.

2.1 Dialogue aspects

In this section, more information is given about the dialogue. First, a description of information seeking dialogues is given, the second section outlines the communication in a dialogue and the last section deals with dialogue styles.

2.1.1 Information seeking dialogues

An information-seeking dialogue contains two participants, one seeking information and the other attempting to provide that information. Analysis of naturally occurring dialogues suggests that when a human being plays the role of information provider in such a dialogue, that person generally attempts to infer the underlying task-related plan motivating the information seeker's queries and to use this plan to understand subsequent utterances and provide co-operative, helpful responses.

When two individuals participate in an information-seeking dialogue, the information provider uses the context within which each query occurs to interpret the query, determine the desired information, and formulate an appropriate response. This context consists of more than mere knowledge of the previous questions and answers. A co-operative participant uses the information exchanged during the dialogue and his knowledge of the domain to hypothesise a model of the information seeker and dynamically adjusts and expands the model as the dialogue progresses.

Information-seeking dialogues may be classified as either top-down or bottom-up. In a top-down dialogue, the information seeker directly communicates his underlying task at the outset of the dialogue and then proceeds to formulate queries relevant to constructing a plan for accomplishing the task. The following is an example of a naturally occurring top-down dialogue:

- 1 *Client* : I am trying to get to Delft from Cuijk.
- 2 *Client* : Is the best way going via Tilburg?

In the first utterance, the speaker conveys his underlying task, and in the second utterance, he formulates an information-seeking query in order to construct a plan for accomplishing this task of getting to Delft. If the information provider believes that via Tilburg is a good way of getting to Delft, her response is likely to consist of directions for reaching Delft via Tilburg. On the other hand, if the information provider does not believe that via Tilburg is a step in a good plan for travelling to Delft, then her response is likely to suggest alternative routes. This is the case in the following response to the above query:

Operator : If you want to go to Delft, the best way is going via Utrecht.

In both cases, the human information provider appears to direct her response to what she believes is the task underlying the information seeker's queries.

In a bottom-up dialogue the information seeker does not directly communicate his overall task but instead formulates queries relevant to subtasks within the overall task, constructs plans for accomplishing these subtasks, and builds his plan from the bottom up. The following is an example of a naturally occurring bottom-up dialogue:

<i>Client</i>	: Can you tell me at what time the trains from Nijmegen to Tilburg leave?
<i>Operator</i>	: The trains from Nijmegen to Tilburg leave five minutes before the hour every hour.
<i>Client</i>	: And the trains from Tilburg to Delft?
<i>Operator</i>	: They leave twice every half-hour. However, if you are travelling from Nijmegen to Delft, it is also possible to travel via Utrecht.

Although the information seeker did not directly communicate her overall task to get to Delft from Nijmegen, this was inferred by the information provider from the dialogue and knowledge about the information seeker. The information provider then responded by addressing not only the information seeker's query but also the information seeker's inferred task.

Top-down dialogues seem to occur when the information seeker has little knowledge about how to proceed in the domain and is willing to relinquish control of the dialogue to the information provider. Bottom-up dialogues seem to occur when the information seeker believes he is already knowledgeable about the domain and wants to maintain control of the dialogue and fill in those parts of his partially constructed plan that are not yet complete. In both cases, human beings seem to use the information seeker's inferred task-related plan in the subsequent dialogue.

Thus one way which human beings appear to assimilate dialogue is by inferring the underlying task-related plan motivating an information seeker's queries. The resulting model of what the speaker wants to accomplish seems to play a major role in providing co-operative responses, as illustrated in the above dialogues.

Information-seeking dialogues exhibit structure and this appears to be caused by two factors. The first is the organised nature of naturally occurring information-seeking dialogues. Analysis of naturally occurring dialogues indicates that once the information seeker's attention is focused on achieving a particular sub-goal, he will generally finish investigating a partial plan for it before starting to develop plans for achieving other sub-goals. One possible explanation for this observed behaviour is that it may require less mental effort than switching back and forth among partially constructed plans for different sub-goals. The second factor producing structure in these dialogues is their co-operative nature. Since the information provider is helping the information seeker construct a plan for the information seeker's underlying task, both participants are expected to maintain a meaningful exchange during which the relationship of individual utterances is explicitly or implicitly conveyed. Thus, one expects the participants to maintain approximately the same focus of attention in consecutive utterances ([Carberry]).

2.1.2 Communication in a dialogue

One meaning of 'to communicate' is 'to make something common', i.e., to convey information or knowledge from one person to another in as accurate a way as possible. Nevertheless, this meaning does not cover all that is signified by communication. For example, consider a dialogue. In such a dialogue, when one person says something, the other person does not in general respond with exactly the same meaning as that seen by the first person. Rather, the meanings are only *similar* and not identical. Thus, when the second person replies, the first person sees a *difference* between what he meant to say and what the other person understood. On considering this difference, he may be able to see something new, which is relevant both to his own views and to those of the other person. And so it can go back and forth, with the continual emergence of a new content that is common to both participants. Thus, in a dialogue, each person does not attempt to *make common* certain ideas or items of information that are already known to him. Rather, it may be said that the two people are making something *in common*, i.e., creating something new together [Bohm].

Normal face to face communication is multimodal employing several modalities of production and perception in order to share information. The two primary modes of production are speech and various types of bodily gestures, perhaps primarily facial gestures, head movements and manual gestures.

The two primary modes of perception are, accordingly, hearing and vision. Normally, the spoken message will in this case be the dominating one with additions given by bodily gestures. The gestures are often, in turn, reinforced by prosody, resulting in a situation where utterances through words and grammatical constructions are given supplementary support by gestures and prosody.

2.1.3 Dialogue styles

In a dialogue between two persons, both persons can have the initiative of the dialogue. The person, who has the initiative, controls the direction of the dialogue. It is also possible that the initiative during the dialogue moves between the two persons in the dialogue. This dialogue style is called 'mixed initiative'. It is a combination of the two dialogue styles that are described below.

- *Directive*

The operator has complete dialogue control and therefore has the initiative. She recommends or suggests a sub-goal to perform next and will use whatever dialogue is necessary to obtain the needed item of knowledge related to the sub-goal. If necessary the operator is also willing to change the direction of the dialogue according to stated user preferences. The dialogue is managed through directed, preferably closed, questions. The client is supposed to provide the matching answers.

<i>Operator</i>	: Good afternoon, from which station to which station do you want to travel?
<i>Client</i>	: Good afternoon, I want to go from Delft to Utrecht.
<i>Operator</i>	: At what time do you want to leave or arrive?
<i>Client</i>	: I want to leave this evening at eight o'clock.
<i>Operator</i>	: Do you have a moment, please?
<i>Client</i>	: Yes.
<i>Operator</i>	: The expected arrival at Utrecht central station is eight fifty five. You have to change trains at Rotterdam central station.
<i>Client</i>	: ...

Figure 2-1 Directive dialogue example

In Figure 2-1, a directive dialogue example is shown. In stead of waiting for the client to provide the information, the operator immediately takes the initiative and asks specific questions to get the information she needs. The needed information in this example is the two stations (Delft and Utrecht, assumedly central station), the travel date (today) and the travel time (eight o'clock).

- *Non-directive*

The user has complete dialogue control and therefore has the initiative. The operator is very awaiting and leaves the initiative to the client. The operator asks open questions and uses alignments, such as 'ok', 'yes', and 'mm' to let the client know that she is still listening. This way she stimulates the client to provide information. She encourages the client to go on with what he wants to say and does not ask for information she needs to provide the client with the information. For this information, she relies on the client. The operator responds directly to user questions and passively acknowledges user statements without recommending a sub-goal as the next course of action.

<i>Operator</i>	: Good afternoon, travel information.
<i>Client</i>	: Good afternoon, I want to go to Utrecht [ehm].
<i>Operator</i>	: Yes... [noise]
<i>Client</i>	: And I leave from Delft, no from Rotterdam.
<i>Operator</i>	: From Rotterdam to Utrecht, yes...
<i>Client</i>	: I want to be in Utrecht before lunchtime.
<i>Operator</i>	: There is a connection every hour. The first train from Utrecht to Rotterdam leaves at six ten.
<i>Client</i>	: ...

Figure 2-2 Non-directive dialogue example

As you can see in Figure 2-2 the information needed to provide travel information to the client are given by the client one by one during the dialogue. Rotterdam and Utrecht are the two cities and

the operator assumes that the client wants to leave and arrive at central station. The operator also assumes that the client wants to travel today and lunchtime is interpreted as 12 o'clock.

2.2 Training aspects

In this section more information about training a human being is given. The first section explains in what way a person learns. The next section describes several instruction methods, the third section elucidates on several requirements concerning human factors and in the last section information about the user interface is given.

2.2.1 Learning and scripts

In order to train, it is important to know how concepts are structured in the human mind, how such concepts develop and how they are used in understanding and behaviour. Roger Schank ([Schank77], [Schank84]) developed a human memory organisation model in which some basic theories regarding scripts, plans, goals and understanding are defined.

Episodic memory

The form of memory organisation upon which Schank's arguments are based is the notion of episodic memory. An episodic view of memory claims that memory is organised around personal experiences or episodes rather than around abstract semantic categories. Briefly, semantic memory is a memory for words that is organised in a hierarchical fashion using class membership as the basic link. An episodic memory is organised around propositions linked together by their occurrence in the same event or time span. Objects are most commonly defined by their place in a sequence of propositions describing the events associated with an object for an individual. A trip is stored in memory as a sequence of the conceptualisations that describe what happened on the trip. Some of the conceptualisations will be marked as salient and some will have been forgotten altogether. If memory is organised around personal experiences then one of the principal components of memory must be a procedure for recognising repeated or similar sequences. When a standard repeated sequence is recognised, it is helpful in 'filling in the blanks' in understanding.

The over-all organisation of memory is a sequence of episodes organised roughly along the time line of one's life. If a man is asked, "Who was your girlfriend in 1968?" and is asked to report his strategy for the answer, his reply is roughly: "First, I thought about where I was and what I was doing in 1968. Then I remembered who I used to go out with then." In other words, it really is not possible to answer such a question by a direct look-up. Lists of 'past girlfriends' do not exist in memory. Such a list must be constructed. The process by which that list is constructed is a search through episodes organised around times and locations in memory.

Some episodes are reminiscent of others. As an economy measure in the storage of episodes, when enough of them are alike they are remembered in terms of a standardised generalised episode that Schank calls a script. Thus, rather than list the details of what happened in a restaurant for each visit to a restaurant, memory simply lists a pointer (link) to what we call the restaurant script and stores the items in this particular episode that were significantly different from the standard script as the only items specifically in the description of that episode. This economy of storage has a side effect of poor memory for detail, because a person forgets the words and remembers the concepts that underlie them. Therefore, according to Schank people store information in knowledge structures, relevant cognitive schedules in their head. These allow us to make sense of certain situations or events. Humans are equipped with and able to use those knowledge structures. Schank calls these knowledge structures scripts.

Scripts

Scripts are pre-packaged sets of expectations, inferences, and knowledge that are applied in common situations, like a blueprint for action without the details filled in. A script is a structure that describes appropriate sequences of events in a particular context. Scripts represent the knowledge that people

use for daily activities. There are scripts for large and complex activities, such as going to a restaurant, and for small and simple activities, such as parking a car in a garage. Scripts involve routine activities that many people perform, and they are fairly specific. People might have a script for looking for a job or for throwing a party. The scripts for going to the bathroom and sharpening a pencil are among the simplest scripts people employ.

A script tells what is likely to come next in a chain of events that are stereotypes. They are not subject to much change, nor do they provide the apparatus for handling totally novel situations. Thus, a script is a predetermined, stereotyped sequence of actions that defines a well-known situation. Scripts tell us what might happen next and enable us to understand the relevance of what actually does happen next; they provide connectivity between events.

The value of scripts is not only to tell what to expect, but to know that something odd was not expected. They help understand what is special and merits closer attention. The assumption that people share our scripts allows them to be briefer in what they say. Not all the details need to be spelled out to an informed listener. Without a script, no great connectivity between events can be seen.

Most people have been to a restaurant many times in their lives, and have developed a set of rules and expectations for what goes on in the restaurants with which they are familiar. The restaurant script contains all the information necessary to understand the enormous variability of what can occur in a restaurant. Not all restaurants are created equal and the restaurant script has particular tracks for different kinds of restaurants, a fast-food track, a coffee-shop track, a fancy French restaurant track etceteras. These tracks are used to understand the various entering, ordering and paying scenes that different restaurants have. A script must be written for one particular actor's point of view. A customer sees a restaurant one way, and a cook sees it another. When 'the' restaurant script is referred to, Schank is relying on those stereotyped details, which are culturally consensual (Figure 2-3).

Roles		customer (c), waiter (w), owner (o)
Props		tables, chairs, menus, food, bills, money, tip
Preconditions		c has money, c is hungry
Result		c has less money, o has more money, c is less hungry
Scene 1	ENTER	c enters the restaurant
Scene 2	SEATING	c is waiting to be seated, c gets a table, c sits down
Scene 3	ORDERING	c asks w for menu, c studies menu, c orders the meal from w
Scene 4	EATING	w brings food, c eats the food
Scene 5	PAYING	c asks w for check, c pays check, c leaves tip
Scene 6	LEAVING	c leaves restaurant

Figure 2-3 The restaurant script

Every act in the restaurant (or any other) script is potentially subject to obstacles and errors, each of which suggests its own appropriate prescriptions or loops. A few of these will occur with sufficient frequency that a person repeatedly exposed to the script situation will learn them along with the rest of the script. This is the major way in which scripts grow.

But where do scripts come from? Schank assumes that middle-class adults have a very detailed restaurant script, which they use for a variety of things. How did this script get there? Not only is information stored by humans in episode form; it is also acquired that way. Scripts are acquired (by people) by repeated experience with an event. Schank assumes that the process of script acquisition begins with the child assuming that everything she encounters will happen that way again next time. Scripts are constantly elaborated on with each successive experience. Scripts are built up, over time, as a result of experience.

Understanding

Schank views human understanding as heavily script-based. The scripts Schank has developed for understanding systems are made up of the universal rules people all use when dealing with stereotypical situations. These rules are in the form of expectations of what will happen next and why.

A script, when used by an understander, helps the understander know what to expect. A human understander comes equipped with thousand of scripts. He uses these scripts almost without thinking. In order to understand the actions that are going on in a given situation, a person must have been in that situation before. That is, understanding is knowledge-based. The actions of others make sense only insofar as they are part of a stored pattern of actions that have been previously experienced. Deviations from the standard pattern are handled with some difficulty. Of course, people can adapt to situations with which they do not have a previous experience. This adaptability comes from knowledge of plans and goals.

Understanding is a function of the place of a piece of information in context. A script is understandable as a particular realisation of a plan. A plan is sensible only if it leads to some desired goal. In addition, a goal is understandable if it is part of a larger theme. Real understanding requires the ability to establish a connection between pieces of information for which no prescribed set of rules or scripts exist.

People need a great deal of knowledge in order to understand. That knowledge can be of two kinds: specific and general. Scripts are intended to account for the specific knowledge that people have. Most of understanding is script-based. Understanding then, is a process by which people match what they see and hear to pre-stored grouping of actions that they have already experienced. New information is understood in terms of old information. By this view, man is seen as a processor that only understands what it has previously understood.

Implicit real-world knowledge is very often applied by the understander and this knowledge can be very highly structured. The appropriate ingredients for extracting the meaning of a sentence, therefore, are often nowhere to be found within a sentence. Scripts are responsible for filling in the obvious information that has been left out of a story. Of course, it is obvious only to those understanders who actually know and can use the script.

2.2.2 Instruction methods

Successful instruction should include the following four activities ([Alessi]):

- 1 Information is presented or skills are modelled.
An important method of presenting information is through example.
- 2 The student is guided through initial use of the information or skills.
- 3 The student practices for retention and fluency.
- 4 Student learning is assessed.

The computer does not have to fulfil all the phases of instruction. Computers are but one element in an instructional environment along with teachers and other media. The computer may serve one or a combination of the four phases.

Four major types of computer-based instruction programs are:

- *Tutorials*
Tutorial lessons aim to satisfy the first two components of instruction. It begins with an introductory section, which informs the student of the purpose and nature of the lesson. After that, a cycle begins. Information is presented and elaborated. A question is asked which the student must answer. The program judges the response to assess student comprehension, and the student is given feedback to improve comprehension and future performance. At the end of each iteration, the program makes a sequencing decision to determine what information should be treated during the next iteration. The cycle continues until the lesson is terminated by either the student or the program.
- *Drills*
This methodology is primarily used for the third aspect of the instructional process, providing practice. Like a tutorial there is an introductory section, followed by a cycle which is repeated many times. Each time the cycle is repeated, an item is selected and displayed, the student responds, the program judges the response and the student receives feedback about the response. After a number of items the program terminates.

- *Simulations*

Simulations may be used for any of the four phases. As with both previous methodologies, there is an introduction, followed by a cycle that is repeated. For each cycle a scene is presented, the student is required to react and the system changes in response to this action. Depending on the nature of the simulation, the cycle may repeat very frequently, as in a flight simulation, or infrequently, as in a process simulation where the cycle may occur only once after the student chooses the initial parameters.

In a simulation, the student learns by actually performing the activities to be learned in a context that is similar to the real world. It teaches about some aspect of the world by imitating or replicating it.

- *Tests*

Tests are the primary means we have for assessment, the fourth phase. Tests can be used to determine what knowledge people possess. By using the results of these tests, decisions can be made if further training is necessary or not.

2.2.3 Requirements for human factors

A technology solution is really made up of two Systems: the technology system (for example the computer and software) and the human system (the people who use it). When you apply software engineering to designing software, you are designing for the technology system. But when do you design for the human half? Making sure your technology solutions take both technology and the human element into account creates a superior system overall.

Humans have strengths and weaknesses. In designing an interface, you want to play up the human's strengths, and design to accommodate or minimise human limits. One way that human factors specialists design for people is to consider cognitive, visual and physical considerations ([Weinschenk]).

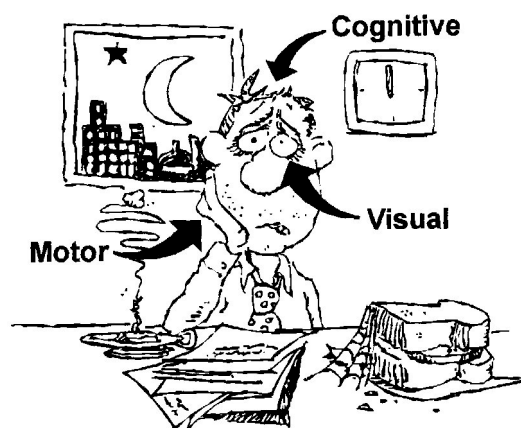


Figure 2-4 A human has cognitive, visual and motor limits

Cognitive considerations

Some of the most important considerations when designing interfaces are the ones involving how people think and learn (cognition). Below are some guiding principles for cognitive processing.

- Limit memory loads
- Break down decision making
- Provide context
- Be aware of user's mental model
- Be consistent

- Be forgiving

Visual considerations

There has been a lot of research on how people scan, read, and find information on screens. If you can reduce the amount of work it takes to visually use a screen, you can save users time and ensure that they will see critical data. Some overall principles that are used in developing guidelines in this area are described below.

- Minimise eye movement
- Adhere to principles of good format and layout
- Use colour and highlighting judiciously
- Use visual coding
- Don't assume people will read everything on a screen

Physical considerations

Sometimes we forget that we require users to interact physically when we design software, for example, manipulating the keyboard, a mouse, touch screen, and so on. Below are some of the critical physical demands to watch out for.

- Limit key combinations
- Avoid difficult combinations
- Pay attention to touch-typing skills
- Avoid a 50/50 Split
- Watch out for repetitive motion syndrome
- Make sure users get training on devices

2.2.4 User interfaces

The interface is the part of the system that the user sees and interacts with. It is the part that is being judged as usable or not usable. It is related to, but not the same as, the underlying structure, architecture and code that makes the software work. The interface includes the screens, windows, controls, menus, metaphors, online help, documentation and training. Anything the user sees and interacts with is part of the interface ([Weinschenk]).

From the user's point of view, the interface *is* the software. There is a body of knowledge about how people think, learn and work. Intelligent interface design taps into that body of knowledge and applies it to the design of a software interface. An intelligent interface is easy to learn and use. It allows users to do their work or perform a task in the way that makes the most sense to them, rather than having to adjust to the software. An intelligent interface is specifically designed for the people who will be using it. It maximises what we know about human strengths, for example, memory and complex computations. It takes the environment, tasks and experience of the people using the product into account in its design. Well-designed interfaces reduce errors, training time and costs, make people more productive, result in superior customer service and get used.

A common reason users find an interface unfriendly or not intuitive is because it does not support how they really do their work and think about their work. Too often, the software architecture design dictates the interface design, for example, logical system function partitioning and data flow dictate screen or window partitioning and dialog flow. Guidelines ensure you are following the basic principles of usable interface design, creating interfaces that are consistent and intuitive.

Learning is a different experience than either tool, browsing, searching or buying. If users are using the computer as a learning tool, for instance taking a computer-based tutorial, they will need to have their attention caught and retained. This can be difficult if there are other distractions in the environment, such as telephones or other interruptions. If the user's purpose is to learn:

- *Keep their interest*

Use graphics, colour and animation to make the information interesting and keep them from getting distracted by their environment.

- *Use instructional design principles*
Learn about and use principles of instructional design to chunk information into meaningful bits. Don't overwhelm them with too much at once.
- *Minimise teacher talk*
Use boxes with text sparingly. People do not read large blocks of text on computer screens.
- *Build with easy in's and out's.*
Make it easy for users to stop where they are and come back in later at the same place. Make it easy for them to start a small section over again without going back to the beginning.

CHAPTER THREE

Dialogue coding

Dialogue analysis provides insight in the dialogues on certain aspects. This means for example getting to know specific characteristics of the dialogue such as the context, the participants in the dialogue, which participant has the initiative and when, the way the dialogue is managed etceteras. By analysing the dialogues the existing knowledge within the dialogues will become apparent, can be extracted and used in the training. Another reason is explicating the possible underlying model of the dialogue. A way to do this is coding the dialogues. Coding labels the dialogues so that the structure of the dialogues will be transparent. In addition, coding results in data reduction and this enables statistical processing.

In this chapter the coding that is used to annotate the corpus is described. In the first section an introduction of the used coding and corpus is given. Then the dialogue constraints and the dialogue acts that are used to code the corpus are explained. The tool that is developed to code the dialogues is described in the third section. In the fourth section, the frequencies of the dialogue acts are shown and the last section concludes with additional comments about the coding.

3.1 Introduction

Currently, there is a wide variety of annotation schemes for dialogues, but no standards nor general methodological guidelines for the annotation and analysis of dialogue corpora. At this moment the MATE (Multi level Annotation Tools Engineering) project, an EU sponsored project, aims to develop a preliminary form of standard and a workbench for the annotation of spoken dialogue corpora ([Klein]) A part of this project is the review of coding schemes such as Alparon, Chat, Chiba, Coconut, Janus, Linlin, Verbmobil. The observed coding schemes are used for two agent, task-oriented dialogues, in which the participants collaborate to solve a problem. The observed schemes in the MATE project are all designed for a certain task and/ or used in a specific domain. The comparison of the coding schemes was performed differently with regard to higher and lower order categories. The definition of higher order categories was mainly driven by linguistic and/or philosophical theories the schemes were based on. Whereas the underlying task the scheme was designed for and the domain of the corpus the scheme was applied to, influenced definitions and descriptions of lower order categories. Therefore the last group can be interpreted as highly task or domain dependent. For reusability reasons, language, task and domain independence is required. This is one of the main reasons that only the Alparon scheme could be useful for the current project because this project uses the same domain.

Although the coding scheme that is used in the Alparon project uses the same domain, only certain parts are used as a point of reference for developing a coding scheme for this project. This is done because a coding scheme is strongly related to the objectives of the coding and the coding scheme used in the Alparon project is designed for an automatic speech processing system, whereas the coding scheme that is used in this project aids in constructing a dialogue model for a dialogue simulation system and therefore has to be based mainly on linguistics. The Alparon coding scheme is not based on linguistics but mainly focuses on the information exchange and dialogue semantics.

In illustration of the use of the Alparon coding scheme only as a point of reference, the parts of this scheme that are not usable in the coding scheme of the current project are explained. The Alparon coding scheme consists of several coding levels ([vanVark]). In the lower coding levels every single part of an utterance is coded and the coding is based on the semantics of the utterance. This is not useful in this project because it is not necessary to code every single part of an utterance to get a

global model on meta-level of the dialogue. One coding per utterance is enough to be able to design a dialogue structure if one exists. The higher coding levels in the Alparon coding scheme classify an utterance as a certain phase and split an utterance in linguistic elements that are called moves. Therefore, also on a higher level Alparon uses several codes per utterance. Without the context of a complete coded utterance, a move does not specify to which interlocutor or phase it belongs. These higher codes are not completely usable. To get a general overview of a dialogue without details and to see how the operator manages the dialogue it is convenient to have one code per utterance in which the phase, the interlocutor and the linguistic type (move) the utterance belongs to are combined. The advantage of using one code per utterance is that one can see very easily when (phase) what (linguistic type) is said by whom (interlocutor) in a dialogue. It also elucidates the dependencies between the interlocutors.

The Alparon coding scheme shows that a dialogue can be divided in several phases. During an initial analysis, it became evident that a dialogue can be divided in four phases. These phases will be explained in chapter four in which the dialogue model is described. These four phases were taken into account during the coding of the corpus.

Dialogue acts and dialogue moves are used to annotate corpora. Searle, Austin and others had each their own interpretations and definitions of the terms dialogue act and dialogue move ([Brady], [Burkhardt], [Ferber], [Lepore]). At this moment, there exist no common definitions to describe these kinds of events in dialogue coding. 'A dialogue act represents a speaker's intention in an utterance' is the most commonly used notion for a dialogue act. Everyone uses the original idea that Searle introduced and interprets it to accommodate it to his or her own objectives. In the current project, the original terms are also adapted to the objectives. Here dialogue acts are the several terms used to code the dialogues. The transitions between these terms are called dialogue moves. The used dialogue acts are described in this chapter. The dialogue moves that arise from these coded dialogues are analysed in the next chapter where a dialogue model is developed.

The corpus that is used as a basis for the coding consists of two hundred transliterated dialogues from OVR. These are transliterations of actual recorded conversations between OVR-operators and clients. The topic of these dialogues is restricted to travelling by train and its complementary information such as timetables, transitions, ticket information and station information. An example of a dialogue is shown in Figure 3-1. The complete corpus with the corresponding coding can be found in appendix B.

```

2: [noise/] goedemorgen reisinformatie [/noise]
1: ja goedemorgen ik wilde graag weten [u:h] hoe vaak de trein van Utrecht naar Naarden
   Bussum gaat en: welke richting dat is
2: Utrecht Naarden Bussum gewoon door de week
1: ja
2: richting Amsterdam #1 stoptrein # Amsterdam vanaf Utrecht en die gaat twee maal per
   uur #2 om: # zes over heel en #3 zes # over half
1: #1 dus[uh] ja # #2 twee maal per uur # #3 zes over heel # oké en hoe lang doet hij er
   over <v-> tot Naarden Bussum
2: een half uur
1: een half uur
2: ja
1: oké dat was het #4 bedankt #
2: #4 ja # tot uw dienst
1: da:g
2: da:g
1: [noise] [tone]

```

1 = client
2 = operator

Dialogue 71713806

Figure 3-1 Dialogue example from the corpus

3.2 Coding scheme

In the first section, the constraints that are necessary to be able to code the dialogues are introduced and illustrated with examples. These constraints were derived during the study of the corpus. There is a minimal set of codes to describe a dialogue. These codes are called dialogue acts and are used to code the dialogues. These dialogue acts are described in 3.2.2.

3.2.1 Coding constraints

- I. The dialogues of the corpus were only available in printed form and therefore coded without being able to interpret and make use of prosody. For a human it is not always possible to correctly interpret the utterances without prosody. Moreover, if an utterance is interpreted based on an assumption of the prosody it will still be a subjective coding. The following examples illustrate that the decision if the utterance must be coded as a *verification question* or a *paraphrase* can not be made because the difference between these two can only be heard. In these cases, the client's utterance is coded as *verification question* when the operator's reaction is interpretable as an answer (Figure 3-2) and as *paraphrase* otherwise (Figure 3-3).

```

2: goedemiddag reisinformatie
1: ja goedemiddag u spreekt met [achternaam] ik had graag van u vernomen hoe laat de
   trein uit Den Haag vertrekt naar Heerlen die vertrekt dacht ik elk uur op een zelfde
   tijd [typewriter]
2: ja dat klopt die gaat zevenenveertig minuten over het hele uur
1: zevenenveertig minuten over het hele uur
2: ja
1: dank u wel
2: alsjeblieft #1 dag #
1: #1 dank u wel #

```

1 = client
2 = operator

Dialogue 91536703

Figure 3-2 Example where client's utterance is coded as *verification question* because of the operator's reaction

```

2: [noise/] goedemiddag [/noise] reisinformatie
1: dag u spreekt met [achternaam] uit Hengelo kunt u mij zeggen hoe laat [u:h] vanavond
   de laatste trein rechtstreeks naar Schiphol gaat vanuit Hengelo
2: vanuit [uh] Hengelo (()) Schiphol vanavond de laatste trein [uh]even kijken
2: [talking]

2: Hengelo Schiphol de laatste trein is om tweeën- zesendertig
1: tweeëntwintig uur zesendertig
2: komt op Schiphol aan om nul uur zesendertig
1: oké ik dank u wel
2: tot uw dienst
1: dag
2: da:g
1: [noise] [tone]

```

1 = client
2 = operator

Dialogue 71715301

Figure 3-3 Example where client's utterance is coded as *paraphrase*

- II. In a dialogue, it is possible that an operator and a client talk simultaneously. Whenever this occurs this is indicated in the corpus with a hash (#). This is taken into account during the coding by marking the coding with an asterisk (*). The following constraints are applied.

- a) The utterance is not split in two parts and therefore coded with one dialogue act in the following cases:
- Whenever simultaneous speech occurs at the beginning or end of an utterance.

```

2: goedemiddag reisinformatie
1: hallo u spreekt met [voornaam+achternaam] ik zou graag komende vrijdag met de
  trein van Zwolle naar Harderwijk gaan <naar> aankomsttijd rond twaalf uur in
  Harderwijk (()) (()) hoe laat ik in Zwolle moet vertrekken
2: u kunt vertrekken om elf uur eenentwintig vanuit Zwolle
1: ja
2: bent u elf uur zesenvieftig in Harderwijk
1: en de daaropvolgende #1 mogelijkheid #
2: #1 wordt het # elf uur eenenvijftig vertrek bent u om twaalf uur zestien in
  Harderwijk
1: en dat is een stoptrein of sneltrein
2: dat is een [u:h] stoptrein
1: oké dan weet ik voldoende
2: oké #2 graag gedaan #
1: #2 bedankt da:g #
2: dag meneer
1: [noise] [tone] [tone]

```

1 = client
2 = operator

Dialogue 11845602

Figure 3-4 Example where utterance is not split in two parts because of simultaneous speech at the end

- Whenever a split utterance results in the same number and the same types of dialogue acts. This is the case when there is only one utterance of one of the two persons between two consecutive utterances of the other person. In the example in Figure 3-5 the coding of the three utterances is RC, GA and GC. If the first utterance (with hash) would have been split the utterances would be coded the same.

```

.....
1: negen uur negenentwintig #2 # Utrecht Centraal Utrecht Centraal
2: #2 in Utrecht #
1: goed dan weet ik voldoende
.....

```

1 = client
2 = operator

Dialogue 91536503

Figure 3-5 Example where utterance is not split in two parts because of only one operator's reaction

- Whenever what is said simultaneously by one of the persons is non-verbal.

```

.....
1: [uh] op welk #1 perron # vertrekt die trein naar Zwolle
2: #1 [cough] #
2: ik ga even voor u kijken
.....

```

1 = client
2 = operator

Dialogue 81494802

Figure 3-6 Example where utterance is not split in two parts because of non-verbal utterance

- b) In all the other occurrences of a hash in an utterance, the utterance is split in two parts. In Figure 3-7 an example is shown in what way the coding is stored when the utterance is split in two parts.

Utterance	Coding
..... 2: negen uur vertrekken vanuit Leiden naar Rotterdam #2 # komt u negen negenentwintig aan in Rotterdam	* GA; #2 GA
1: #2 ja #	* #2 RC2
1: en kan ik ook over Den Haag want er gaat ook iemand uit Den Haag mee of kan dat niet	QC
.....	1 = client 2 = operator
Dialogue 91537302	

Figure 3-7 Example of the way of coding when simultaneous speech occurs

- III. In some utterances, it is possible that two dialogue acts occur. Whenever one of the two dialogue acts is not an information dialogue act (a dialogue act containing essential information for the dialogue), the information dialogue act takes precedence and the utterance is coded with this information dialogue act because this dialogue act is reacted upon. For example when the client is confirming the operator's answer and asking a question in the same utterance, the utterance is coded as a question because the operator is reacting on the question by giving an answer and is not reacting to the remark. The remark is discarded because it contains no essential information for the dialogue. In the case of two information dialogue acts in the same utterance the utterance is coded with two dialogue acts. This is marked in the code with a '+'- character between the two dialogue acts as shown in Figure 3-8.

Utterance	Coding
2: onder voorbehoud van perron negen a b wilt u graag weten welk perron u aankomt dan pak ik er even een map bij	GA + NI
1: ja 's morgens dus hoe laat moet ik dan uit Venlo vertrekken	ACLI + OC
2: enkele reis tweede klas is drie%vijftig gulden vijftig binnen welke termijn komt u terug meneer	GA + NI
	1 = client 2 = operator

Figure 3-8 Example of the way of coding when two information dialogue acts occur in one utterance

- IV. When two utterances of the same person are consecutive sentences in the corpus they are regarded as one utterance and are therefore coded the same. Often two dialogue acts occur in these consecutive sentences. In that case, the utterance is coded the same way as described at criterion III.

2: goedenavond reisinformatie	
1: hallo met [voornaam+achternaam] ik wil graag weten hoe laat de: eerste trein nu gaat vanuit Driebergen Zeist naar Utrecht	
2: vanaf nu	
1: ja	
2: ja	
2: twintig uur negenentwintig	
1: ja	
2: en anders twintig uur vijfenveertig	
1: en: daarna	
2: <twintig> <uur> <<nee>> eenentwintig uur negenentwintig pas weer	
1: ja oke bedankt #1 ja da:g #	
2: ja #1 tot uw dienst dag mevrouw #	
1: [noise] [tone] [tone]	
	1 = client 2 = operator
Dialogue 11848102	

Figure 3-9 Example of constraint IV: Two consecutive utterances of the same person become one

- V. The non-verbal utterances are also transliterated. These utterances are less important for this analysis because there is no need to simulate these and they have little or no influence on the dialogue structure. Therefore, they are coded as *OUT OF DIALOGUE*. A few examples are shown in Figure 3-10.

```

1: [noise] [tone] [tone]

1: [laughter]

1: [cough]

1: [loud_breath] (% zo lacht dit meisje )

1: [talking]

```

Figure 3-10 Examples of non-verbal utterances

- VI. Sometimes there are blank sentences in between the dialogue utterances. It is possible that these blank sentences are silences that are used as alignment in the dialogue. This is an assumption because to correctly interpret such an utterance it is necessary to actually hear this utterance to know if it has some meaning. Because prosodic information is not available, these sentences are coded as *EMPTY*.

3.2.2 Dialogue acts

First, the dialogue acts used to code the operator's utterances are described. Then the client's side is illustrated. Finally, the sub-coding is explained.

3.2.2.1 Operator's dialogue acts

Table 3-1 gives an overview of the dialogue acts used to code the operator's utterances. The abbreviation and meaning, as shown in Table 3-1, are used in the headings of the remainder of this section where the dialogue acts are described one by one.

Table 3-1 Dialogue acts operator

<i>Abbreviation</i>	<i>Meaning</i>
OO	Opening operator
V	Verification
CL	Clarification
NI	New information
R	Repetition
RO	Remark operator
GA	Given answer
GO	Goodbye operator

OO – opening operator

All the opening utterances of the operator are coded as *OO*.

```

2: goedemiddag reisinformatie

2: reisinformatie goedenavond

2: reisinformatie goedemorgen

```

Figure 3-11 Examples of coding 'opening operator' (OO)

V – Verification

When the operator verifies the information given by the client then this question is coded as V.

```

1: goedemorgen [voornaam+achternaam] [bedrijfsnaam] [uh] [cough] sorry vraagje welke
   treinen van Amsterdam Schiphol naar Delft Centraal rijden er op dertig mei tussen half
   negen 's ochtends en tien uur 's ochtends
2: tussen half negen tot tien uur 's morgens en vanaf Schiphol naar Delft
1: ja
.....

=====

1: goedenavond [uh] ik wou graag morgenochtend om kwart voor negen in Bussum Zuid zijn en
   vertrekken vanuit Centraal Station Amsterdam
2: kwart over negen zegt #1 u he #
1: #1 kwart voor # negen
2: kwart voor negen
1: ja
.....

```

1 = client
2 = operator

Figure 3-12 Examples of coding 'verification question' (V)

Sometimes the operator asks a question which does not seem to be a verification question because of the way in which the question is asked. Usually the utterance can then be interpreted as a new information question. In these cases, the utterance is coded as *verification* when the operator is asking for information the client already provided in the question (Figure 3-13). *New information* is used when the operator is asking for information the client did not provide in the question.

```

2: goedenavond reisinformatie
1: goedenavond [voornaam+achternaam] ik wilde[uh] vrijdag avond [uh] zo rond kwart over
   zeven of [uh] half acht in Utrecht zijn en ik vertrek vanuit Delft
2: en waar wilt u heen
1: naar Utrecht
2: naar Utrecht en daar zijn om
1: zo rond[uh] kwart over zeven of half acht half acht mag ook
2: de trein van achttien uur dertien naar Rotterdam aankomst achttien vijftientig
   overstappen vertrek achttien zevenendertig die is negentien uur dertien in Utrecht
1: negentien uur dertien
2: ja
1: en als ik een trein heb via Den Haag
2: achttien uur zeventien uit Den Haag aankomst achttien tweeëndertig overstappen vertrekt
   achttien negenendertig die is negentien achtentwintig in Utrecht
1: en even kijken moet ik [uh] achttien uur tweeëndertig ben ik in Den Haag
2: Den Haag
2: en achttien negenendertig gaat die naar Utrecht
1: en dat is Centraal he
2: ja
1: ja oké bedankt
2: tot uw dienst
1: #1 dag #
2: #1 dag #

```

1 = client
2 = operator

Dialogue 51797501

Figure 3-13 Example where utterance is coded as *verification* because information is already provided by client

CL - Clarification

When the operator needs more information on a certain information element, she asks the client for an elaboration. This question is coded as *CL*. The operator reduces ambiguity by requesting additional information.

```

1: goedemiddag u spreekt met [voornaam] ik wou even vragen voor donderdag van Leiden naar
   Beilen wat[uh] dat kost en wat de tijden zijn dat de treinen gaan
2: Beilen en donderdag dus [uh] Hemelvaartsdag bedoelt u
1: ja
.....

=====

1: goedenavond u spreekt met [achternaam] [uh] zou ik van u [u:h] tijden mogen weten van
   Hilversum Noord naar Amsterdam de treinen tot hoe laat ze vanavond nog gaan
2: [uh] Amsterdam Centraal meneer
1: ja
.....

```

1 = client
2 = operator

Figure 3-14 Examples of coding 'clarification question' (CL)

NI – New information

The information that the client did not provide in the question can be asked for by the operator. When the operator asks for this information the question is coded as *NI*.

```

1: ja goedemorgen ik wilde graag [uh] weten hoe laat de treinen van Amsterdam naar
   Castricum
2: en hoe laat wilt u vertrekken
1: ja binnen nu en een uurtje
.....

=====

1: met [achternaam] Enschede goedemorgen [uh] van Enschede naar Lelylaan Amsterdam en dan
   moet ik er om een uur zijn kunt u mij vertellen hoe ik moet reizen
2: vandaag
1: vandaag ja
.....

```

1 = client
2 = operator

Figure 3-15 Examples of coding 'new information question' (NI)

R - Repetition

The questions of the operator in which she asks to repeat the last utterance if it was not clear or understandable are coded as *R*.

```

.....
1: zonder overstappen he
2: wat zegt u
1: dat is zonder #2 overstappen #
.....

```

1 = client
2 = operator

Figure 3-16 Example of coding 'repetition question' (R)

GA – Given answer

All the operator's answers on the client's questions are coded as *GA*. When the operator is presenting the travel scheme she is answering a question and it is therefore coded as *GA*.

```
2: en bent u in Maastricht om tien uur eenenveertig
2: nee op[uh] dit moment kunt u het beste de trein nemen om kwart voor vanuit Maarssen om
zevenenveertig dan heeft u aansluiting in Utrecht op de rechtstreekse trein Maastricht
2: dat is onder voorbehoud op negen A B
2: om vier over het hele uur
```

Figure 3-17 Examples of coding 'given answer' (*GA*)

The operator sometimes takes the initiative to give advice that is not explicitly asked for by the client. The advice concerns the last question asked by the client. This information is given because it can be useful for the client. This advice is regarded as a part of the answer of the operator and is coded the same, because an advice also provides the client with information and the difference between an advice or an answer is not always clear.

```
2: en dan wil ik even kijken of er nog werkzaamheden zijn en nacht zaterdag op zondag trein
heeft bij aankomst te Dordrecht een vertraging van vijf minuten
2: dan zit u op een gegeven moment wel in een stoptrein hoor
```

Figure 3-18 Examples of operator's advice, which are coded as *GA*

RO – Remark operator

All the utterances in which the operator is not giving an answer or is asking a question are coded as *RO*. These utterances do not contain essential information for the dialogue because most of these utterances can be classified as not information dialogue acts. Usually these are confirmations to previously made utterances (alignments). The remarks that are made because the operator is excusing herself for making a mistake are also coded as *RO*. (The last two examples in Figure 3-19)

```
2: ja
2: goed zo
2: ok,
2: goed
2: oh sorry dat is hetzelfde
2: dat wordt moeilijk meneer omdat ik eerst niet goed gekeken had
```

Figure 3-19 Examples of coding 'remark operator' (*RO*)

The operator sometimes asks the client to wait for a moment so that she can look up some information or process the given information. These utterances are also coded as *RO*, because this information is regarded as less important for the dialogue.

```
2: ik kijk even een moment
2: ik ga even kijken
```

Figure 3-20 Examples of utterances in which operator asks the client to wait for a moment

As indicated in the introduction (section 3.1) the four phases in which a dialogue can be divided are taken into account during the coding. This dialogue act is divided into *RO1*, *RO2*, *RO3* and *RO4*. The

first one is used during the first phase, the second one is used during the second phase and the last two are used during the third phase.

GO – Goodbye operator

The utterances in which the operator is saying goodbye and is ending the dialogue are coded as GO.

```
2: tot uw dienst
2: dag [tone]
2: graag gedaan
2: dag mevrouw
```

Figure 3-21 Examples of coding 'goodbye operator' (GO)

3.2.2.2 Client's dialogue acts

Table 3-2 gives an overview of the dialogue acts used to code the client's utterances. The abbreviation and meaning, as shown in Table 3-2, are used in the headings of the remainder of this section where the dialogue acts are described one by one.

Table 3-2 Dialogue acts client

<i>Abbreviation</i>	<i>Meaning</i>
OC	Opening client
AV	Answer verification
ACL	Answer clarification
ANI	Answer new information
AR	Answer repetition
RC	Remark client
QC	Question client
GC	Goodbye client

OC – Opening client

The opening questions of the client are coded as OC.

```
1: hallo u spreekt met [voornaam+achternaam] ik zou graag komende vrijdag met de trein van
Zwolle naar Harderwijk gaan <naar> aankomsttijd rond twaalf uur in Harderwijk (()) (())
hoe laat ik in Zwolle moet vertrekken

1: ja goedemiddag [u:h] ik wou graag om half drie van Barneveld Noord naar Den Helder naar
de boot ik wou graag weten [laughter/] waar ik in vredesnaam allemaal langs moet
[/laughter]

1: goedemiddag u spreekt met [voornaam+achternaam] ik wil graag morgen vanaf Utrecht
Centraal met de trein naar Leidschendam Voorburg

1: [uh] ja goedemiddag ik heb een vraagje hoeveel kost een enkeltje Barneveld Tilburg
```

Figure 3-22 Examples of coding 'opening question client' (OC)

AV – Answer verification

The client's answer on the operator's verification question is coded as *AV*.

```

1: goedenavond [voornaam+achternaam] ik had een vraagje morgen de: morgenochtend om half
   zes de trein vanuit Leiden naar Schiphol rijdt die op tijd weet u dat of[uh] rijdt die
   uberhaupt wel
2: ik zal even kijken hoor half zes zei u toch he
1: ja
.....

=====

1: [uh] vanuit Nijmegen [uh] hoe laat vertrekken de treinen ik moet ongeveer kwart voor
   twee in Arnhem zijn
2: kwart voor twee wilt u daar zijn #1 (()) #
1: #1 ja zeg maar # half twee
.....

1 = client
2 = operator

```

Figure 3-23 Examples of coding 'answer verification' (AV)

ACL – Answer clarification

The client's answer on the operator's clarification question is coded as *ACL*.

```

1: ja goedenavond met [voornaam+achternaam] ik wilde graag weten als ik om half twaalf in
   Boxtel wil zijn hoe laat moet ik dan uit Bilthoven weg
2: 's morgens
1: [u:h] 's morgens ja
.....

=====

1: ja goedenavond [achternaam] ik wilde graag weten van morgenochtend om zeven uur moet ik
   (()) op Schiphol zijn [u:h] hoe laat gaat de trein van Rotterdam rijden
2: vanaf Rotterdam Centraal mevrouw
1: ja of Alexander kan ook
.....

1 = client
2 = operator

```

Figure 3-24 Examples of coding 'answer clarification' (ACL)

ANI- Answer new information

The client's answer on the operator's new information question is coded as *ANI*.

```

1: je goedemorgen met mevrouw [achternaam] ik wou vragen hoe laat de treinen vertrekken
   naar Hoogeveen van Zwolle naar Hoogeveen
2: gewoon door de week gewoon
1: ja gewoon door de week
.....

=====

1: goedemiddag [u:hm] ik zou graag inlichtingen hoe laat de trein loopt naar Schiphol is
   die in enen door #1 gaat Schiphol #
2: #1 ja # en waar vandaan komt u dan
1: van Enschede
2: ja Enschede Schiphol hoe laat wilt u graag op Schiphol zijn
1: tegen half twaalf twaalf uur
.....

1 = client
2 = operator

```

Figure 3-25 Examples of coding 'answer new information' (ANI)

AR – Answer repetition

The client's answer on the operator's repetition question is coded as *AR*.

```

.....
1: en daarna
2: wat zegt u
1: en daarna wat hoe laat ben ik in Utrecht
.....

```

1 = client
2 = operator

Figure 3-26 Example of coding 'answer repetition' (AR)

QC – Question client

All the client's questions besides the opening question are coded as *QC*.

```

1: ja en dan moet ik overstappen in Den Bosch he
1: hoe kom ik via Amsterdam [uh]
1: ok, en de volgende trein is
1: [mm] en hoe laat zou ik dan moeten vertrekken vanuit Arnhem

```

Figure 3-27 Examples of coding 'question client' (QC)

Some remarks are grammatically not a question but are still regarded as one because the operator's response is an answer. Therefore, these remarks are also coded as *QC*.

```

1: zes uur vijftie- nee ik wou om negen uur weggaan
1: ja maar ik wil niet over moeten stappen
1: #3 o:h oh want ik # dacht dat alle tijden net veranderd waren namelijk
1: [uhm] dat is wel heel vroeg
1: (( )) die haal ik nu niet

```

Figure 3-28 Examples of remarks that are coded as question

RC – Remark client

All the utterances in which the client is not giving an answer or is asking a question are coded as *RC*. They can be classified as the client's utterances in which no information dialogue act is present (alignments).

```

1: twaalf (( )) dit is een heel stom potlood [laughter] tweeëndertig Echt O K
1: ja O K
1: ja
1: ja vanaf Zoetermeer
1: ja dat weet ik ja

```

Figure 3-29 Examples of coding 'remark client' (RC)

As indicated in the introduction (section 3.1) the four phases in which a dialogue can be divided are taken into account during the coding. This dialogue act is divided into RC1 and RC2. The first one is used during the second phase and the second one is used during the third phase.

GC – Goodbye client

The utterances in which the client is saying goodbye and is ending the dialogue are coded as GC.

```
1: goed bedankt
1: da:g
1: ok, dan weet ik voldoende
1: #2 bedankt da:g #
```

Figure 3-30 Examples of coding 'goodbye client' (GC)

Sometimes the client first thanks the operator for her services and says then goodbye (Figure 3-31). These 'thank you's' are regarded as part of client's goodbye and coded the same (GC), because these can be seen as an introduction of the actual goodbye. The operator's response on this is regarded as part of the operator's goodbye and coded as GO. The same applies when the client is saying that he does not need more information as shown in Figure 3-32. These are coded the same

```
2: [noise] reisinformatie [tone] goedemorgen
1: hallo u spreekt met [voornaam+achternaam] ik heb een vraagje: [uh] hoe laat vertrekt de
  trein vanuit Dalfsen naar Zwolle
2: van Dalfsen naar Zwolle
1: ja
2: om vierenzeventig ieder uur dus de eerstvolgende om acht uur vierenzeventig
1: oké bedankt
2: graag gedaan
1: doeg
2: da:g
1: [tone] [noise] [tone]
```

1 = client
2 = operator

Dialogue 61573306

Figure 3-31 Example dialogue in which the client the operator first thanks before saying goodbye

```
2: goedenavond reisinformatie
1: ja goedenavond u spreekt met [voornaam+achternaam] ik wil graag weten hoe laat vanuit
  Amsterdam de treinen naar Schiphol gaan vanaf nu ongeveer
2: vanaf Amsterdam Centraal
1: ja
2: [uh] negentien vijfenvijftig is de eerstkomende
1: ja
2: dan twintig uur acht
1: ja
2: dan twintig vijftwintig en twintig achtendertig
1: oké dan weet ik genoeg
2: ok
1: bedankt
2: tot #1 uw dienst #
1: #1 dag #
2: dag mevrouw
```

1 = client
2 = operator

Dialogue 51797901

Figure 3-32 Example dialogue in which the client indicates he has sufficient information before saying goodbye

3.2.2.3 Sub-coding and sub-sub-coding

The utterances coded as GA, QC, RC2 and RO4 have a sub-coding that clarifies the main coding and some have a sub-sub-coding that clarifies the sub-coding. None of the other dialogue acts has an additional coding because they need no further clarification. In the following, the sub-coding and sub-sub-coding are described and examples are shown in Figure 3-33.

Every operator's answer (GA) has a two-digit number. The first digit determines which answer belongs to which client's question. The second digit determines which part of the answer the utterance belongs to, because usually an answer is split into more than one utterance.

The sub-coding for QC is *question about answer (vraag over antwoord)* or *new question (nieuwe vraag)*. The utterance is coded as *question about answer* when the utterance concerns a verification, a clarification or a repetition question on the preceding answer given by the operator. In the sub-sub-coding the question type (verification, clarification or repetition) is represented. All the other QC-utterances are coded as *new question* when the question concerns a new topic. To determine if an utterance has to be sub-coded as *new question* or as *question about answer* with sub-sub-coding *verification* the explanation described earlier in section 3.2.2.1 at the 'verification' dialogue act is applied. In Table 3-3 an overview of the sub-coding and sub-sub-coding for QC is given.

Table 3-3 Sub-coding and sub-sub-coding QC

<i>Sub-coding QC</i>	<i>Sub-sub-coding</i>
new question	
Question about answer	Verification, clarification, repetition

As explained earlier in section 3.2.2.2 at the 'remark client' dialogue act, the client's utterances said during the third phase contain remarks that are coded as RC2. The sub-coding distributes these remarks among three types. These three types are *alignment word/acknowledgement (stopwoord/bevestiging)*, *paraphrase (parafrasering)* and *rest*. The utterance is coded as *alignment word/acknowledgement* when the client acknowledges the operator's answer. Usually these utterances consist of only one or two words. These utterances also have a sub-sub-coding. This sub-sub-coding contains the actual word that is said. The sub-coding *paraphrase* is used when the client repeats a part of the operator's answer. All the other RC2-utterances are coded as *rest*. Usually the utterances coded as *rest* do not influence the dialogue in a significant way because these remarks are only useful to the client and therefore do not contain any information that the operator needs to be able to present the travel scheme. In Table 3-4 an overview of the sub-coding and sub-sub-coding for RC2 is given.

Table 3-4 Sub-coding and sub-sub-coding RC2 and RO4

<i>Sub-coding</i>	<i>Sub-sub-coding</i>
alignment word/acknowledgement	The actual alignment word
paraphrase	
rest	

As explained earlier in section 3.2.2.1 at the 'remark operator' dialogue act, the operator's utterances said during the third phase contain remarks that are coded as RO4. The sub-coding and sub-sub-coding for RC2 also apply to RO4. Therefore, the sub-coding for RO4 is also divided into the types *alignment word/acknowledgement (stopwoord/bevestiging)*, *paraphrase (parafrasering)* and *rest* as shown in Table 3-4.

<i>Utterance</i>	<i>Coding</i>	<i>Sub-coding</i>	<i>Sub-Sub-coding</i>
2: goedemorgen reisinformatie	OO		
1: goedemorgen mevrouw ik wou[uh] graag weten ik wou volgende week donderdag wou ik met de trein van Amsterdam Centraal Station naar Alkmaar	OC		
2: ja	RO1		
1: en: daar wou ik om twaalf uur zijn	OC		
2: [uh] dan kunt u de trein nemen van elf uur twee%ntwintig	GA	00	
1: elf uur twee%ntwintig	RC2	parafrasering	
2: en dan bent u elf uur drie%nvijftig in Alkmaar	GA	01	
1: en van welk perron vertrekt die	QC	nieuwe vraag	
2: van spoor tien	GA	10	
1: spoor tien	QC	vraag over antwoord	verificatie
2: ja	GA	20	
1: ok, en [uh] om de hoeveel tijd gaan die treinen	QC	nieuwe vraag	
2: [uh] die gaan om het half uur twee%ntwintig en twee%nvijftig over het uur	GA	30	
1: oh ja ok, bedankt	GC		
2: tot uw dienst	GO		
1: dag	GC		
2: dag mevrouw	GO		

1 = client
2 = operator

Dialogue 51804302

Figure 3-33 Example dialogue in which examples of the sub-coding and sub-sub-coding

3.3 Coding supporting tool

In this section, the coding tool that is developed and used for coding the dialogues of the corpus is described. First the purpose and the functionality of the tool are explained. Then a technical description of the tool is given.

3.3.1 Purpose and functionality of the coding tool

The coding tool is developed to be able to save the coding information in a database and to use this information for analysis. With this tool, it is possible to read existing dialogues into a database and to code these dialogues. Within each dialogue the coding per utterance can be entered and is immediately stored in the database. The tool has two windows, a main window ('Dialogue coding supporting tool') and a sub-window ('Inhoud database'). The sub-window pops up when the user presses the button on the main window. This window gives the user an overview of the current dialogue and the possibility to see the contents of the database. The main window displays the current dialogue number and body. In addition, per utterance the coding, possible sub-coding and sub-sub-coding can be indicated. The dialogue acts described in section 3.2.2 are grouped per dialogue phase at the left side of the main window. In the centre, the sub-coding and sub-sub-coding are displayed. If a user wants to code a certain dialogue, he can select a dialogue number from a list. The selected dialogue will then be displayed on the screen.

Because the main purpose for this program is entering data, it is useful to be able to develop a program quickly and easy. Therefore, the coding tool is developed with Borland Delphi, which is very easy for rapid prototyping and rapid application development.

A screen shot of the main window of the coding tool can be seen in Figure 3-34. The different parts of this window are numbered and will be explained in the following.

- 1 This button and edit box are only enabled when the database is empty and has to be filled with dialogues. This is most often the case when the tool is started for the first time. When the database is empty and the 'Inlezen dialogen' button is enabled, all the other parts of the main window ('Dialogue coding supporting tool') are disabled. These are available when the database is filled with dialogues.
- 2 In this label, the number of the current dialogue is presented.
- 3 This edit box displays the current utterance that can be coded with the tool. This edit box is read-only.
- 4 By selecting one of the options of this radio group, the user defines that the current utterance belongs to the first or fourth phase. The available dialogue acts for these phases are OO, GO, OC, GC or RO1. These are explained in section 3.2.2.
- 5 By selecting one of the options of this radio group, the user defines that the current utterance belongs to the second phase. The available dialogue acts for this phase are V, CL1, NI, R, RO2, AV, ACL1, ANI, AR and RC1. These are explained in section 3.2.2.
- 6 By selecting one of the options of this radio group, the user defines that the current utterance belongs to the third phase. The available dialogue acts for this phase are GA, CL2, ACL2, RC2, RO3, QC and RO4. These are explained in section 3.2.2.
- 7 The user can use these edit boxes to code an utterance that contains more than one coding. This is the case when an utterance has to be split in two parts. This was explained earlier in section 3.2.1. In the case of the 'Extra invoer subcodering' edit box the required number can be entered in this edit box, whenever the utterances that require a 'subcodering antwoord telefoniste' exceed the '55'.
- 8 This label contains the number of the last 'subcodering antwoord telefoniste'. This is useful as a reference if the user has to determine the number of the 'subcodering antwoord telefoniste' for the current active utterance.
- 9 By selecting one of the options of this radio group, the user defines that the current utterance is 'empty' or 'out of dialogue'. These are explained in section 3.2.1.
- 10 One of the options in this radio group is required when 'GA' is selected in the 'Derde fase' radio group (6). This radio group contains the sub-coding for 'GA'.
- 11 One of the options in this radio group is required when 'QC' or 'RC2' is selected in the 'Derde fase' radio group (6). This radio group contains the possible client's reactions.
- 12 One of the options in this radio group is required when the user selects the sub-coding 'stopwoord/bevestiging' in the 'subcodering reactie client' radio group (11). This radio group contains the possible alignment words.
- 13 One of the options in this radio group is required when the user selects the sub-coding 'vraag over antwoord' in the 'subcodering reactie client' radio group (11). This radio group contains the question type of the current utterance.
- 14 If the user presses this button, the previous utterance will be displayed and will become the active utterance. If the active utterance before pressing this button is the first utterance of the dialogue, the tool will go to the previous dialogue and select the last utterance.
- 15 If the user presses this button, the next utterance will be displayed and will become the active utterance. If the active utterance before pressing this button is the last utterance of the dialogue, the tool will go to the next dialogue and select the first utterance.
- 16 On the right hand side of the main window, the complete contents of the current active dialogue are displayed. This list is read-only.
- 17 The dropdown list at the top of the window contains all the dialogue numbers of all the dialogues that are in the database. By selecting a number from this dropdown list the tool will retrieve this dialogue from the database, present the contents of this dialogue in the main window, and make the first utterance of the selected dialogue the current utterance. A screen shot of the main window with the list dropped down is presented in Figure 3-35.
- 18 The user can press this button ('Stoppen') if he wants to quit the tool. The main window will be closed and the tool will be stopped.

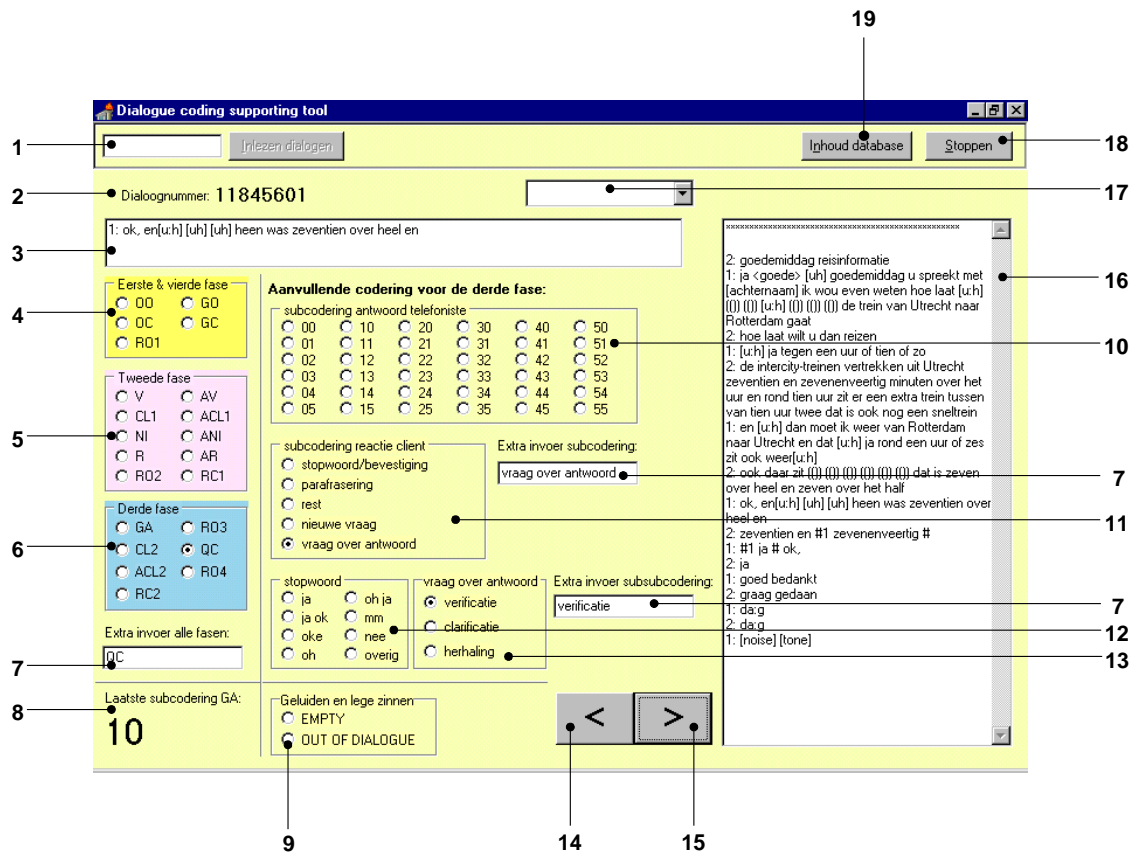


Figure 3-34 Main window of coding tool

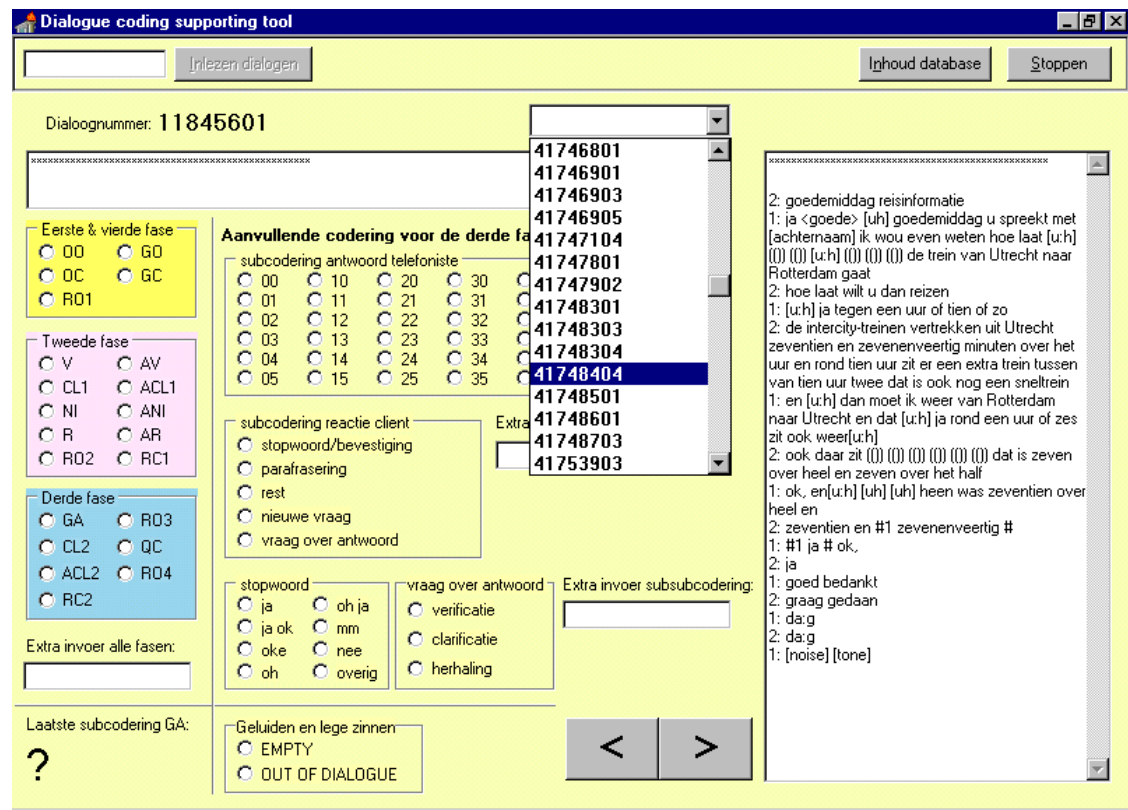


Figure 3-35 Screen shot of main window with list dropped down

- 19 When this button ('Inhoud database') is pressed, the 'Inhoud database' window (Figure 3-36) pops up and gives the user an overview of the current dialogue. This is particularly useful if the user wants to look at a similar dialogue to see how this is coded. In the top part of the window, the details of the current utterance in the selected dialogue are presented (A). In the bottom part the complete dialogue is listed (B) and the user can select an utterance from this list which then will be displayed in the top part of the window. In the 'Inhoud database' window, the user is not allowed to change the content of the dialogue. The user can only navigate through the dialogues that are in the database. The user can view a particular dialogue by selecting the dialogue number from the dropdown list (C) that contains the dialogue numbers that are in the database. Furthermore, the user can navigate through the database by using the navigation buttons (D). With these buttons the user can select the first, previous, next and last dialogue. If the user wants to switch back to the main window ('Dialogue coding supporting tool') he can press the 'Terug naar invoer' button (E) and the 'Inhoud database' window will be closed.

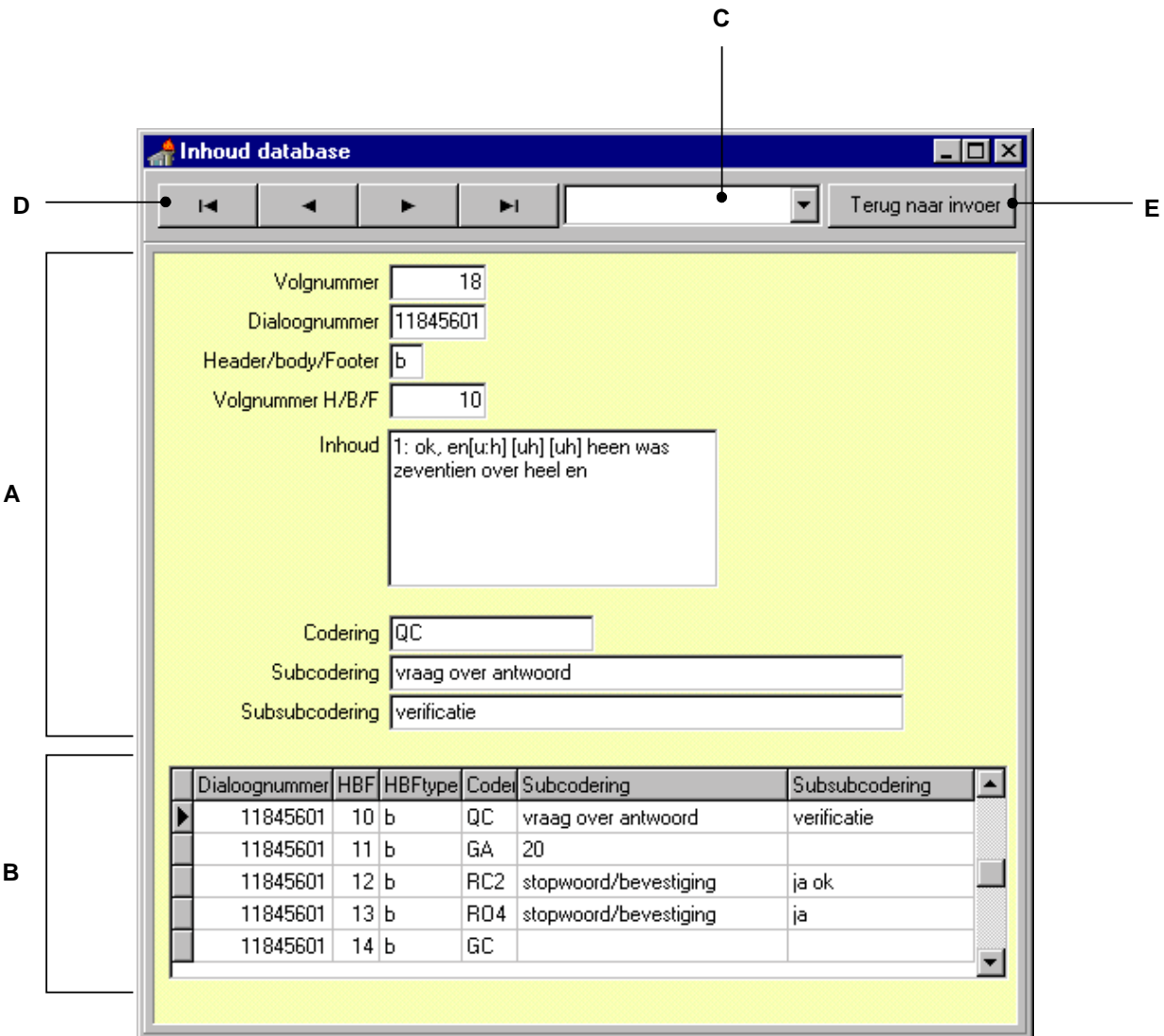


Figure 3-36 Window of coding tool in which database can be viewed

3.3.2 Technical description

The database, that is created to store the dialogue information and coding in, is described. Then the message windows and the program flowchart are explained. Finally the connections of several controls to certain database fields are given.

The database structure

The database is created as a Paradox database by using the database tools provided with Borland Delphi. In Table 3-5 the fields, their types and lengths of every record in the database are shown.

Table 3-5 Structure of database record

<i>Fieldname</i>	<i>Type</i>	<i>Length</i>
Volgnummer	Integer	
Dialoognummer	Integer	
HBFType	Alphanumeric	1
HBFSVolgnummer	Integer	
Inhoud	Memofield	64 Kb (240 in db, remains in mb-file)
Codering	Alphanumeric	20
Subcodering	Alphanumeric	100
Stopwoordje	Alphanumeric	100

Volgnummer contains a unique number for every record in the database. Every record contains a sentence from a dialogue.

Dialoognummer contains the number of the dialogue the sentence comes from.

HBFType contains a type description of what part of the dialogue a sentence comes from. H means that the sentence belongs to the header, B means body and F means that the sentence belongs to the footer of a dialogue.

HBFSVolgnummer contains a unique number for every sentence within a dialogue part.

Inhoud contains the actual sentence.

Codering contains the two or three character description for describing a sentence. These dialogue acts are described in section 3.2.2.

Subcodering contains an elaborated code on *Codering*. *Subcodering* refers to the content of a sentence.

Stopwoordje contains an elaborated code on *Subcodering*.

Message windows

There are certain dependencies between radio groups because of the dependencies between the dialogue acts and their sub-coding and sub-sub-coding. If the user forgets to select a necessary sub-coding or sub-sub-coding a message window pops up stating which coding is not selected. There are no message windows when the user enters incorrect text in the edit boxes because of the many input possibilities. It will take a disproportionate amount of time to build a check mechanism.

Program flowchart of the coding tool

The program flowchart of Figure 3-37 visualises the user options and the actions of the program as a reaction to the user's selection. The program actions are displayed in a rectangle and the user options are displayed in a trapezium.

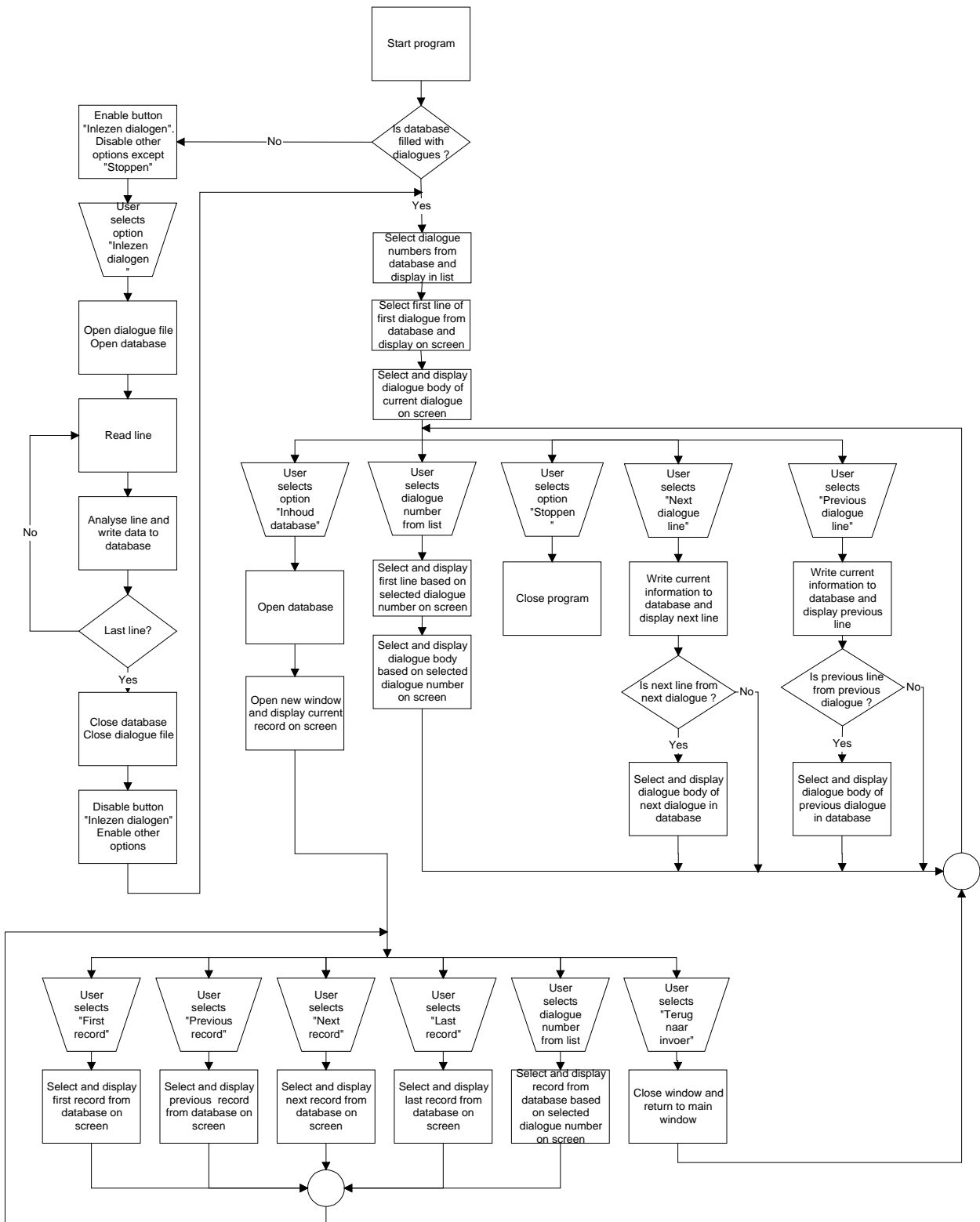


Figure 3-37 Program flowchart of coding tool

Controls and Connections

All the radio groups and edit boxes on the main window are connected to certain database fields. When one of the radio buttons is selected or text is entered, the corresponding dialogue act is stored in the database. In the first column of Table 3-6 the radio groups and edit boxes are shown with their corresponding number as used in Figure 3-34. In the second column, the database fields that are connected to the radio groups and edit boxes are listed.

Table 3-6 Main window controls with their database fields

<i>Radio groups and edit boxes</i>	<i>Fieldname</i>
Eerste & vierde fase (4)	Codering
Tweede fase (5)	Codering
Derde fase (6)	Codering
Subcodering antwoord telefoniste (10)	Subcodering
Subcodering reactie client (11)	Subcodering
Stopwoord (12)	Stopwoordje
Vraag over antwoord (13)	Stopwoordje
Geluiden en lege zinnen (9)	Codering
Extra invoer alle fasen (7)	Codering
Extra invoer subcodering (7)	Subcodering
Extra invoer subsubcodering (7)	Stopwoordje

The label showing the dialogue number (2) and the edit box showing the current utterance (3) are also connected to database fields. The label is connected to the field 'Dialognummer' and the edit box is connected to the field 'Inhoud'. The edit boxes in the sub-window (Figure 3-36) are connected to their corresponding label name. In Table 3-7 an overview of the edit boxes in the sub-window and the fields they are connected to are listed

Table 3-7 Sub-window controls with their database fields

<i>Edit boxes</i>	<i>Fieldname</i>
Volgnummer	Volgnummer
Dialognummer	Dialognummer
Header/Body/Footer	HBFType
Volgnummer H/B/F	HBFVolgnummer
Inhoud	Inhoud
Codering	Codering
Subcodering	Subcodering
Subsubcodering	Stopwoordje

3.4 Frequencies dialogue acts

After the coding of the two hundred dialogues of the corpus, a statistical analysis can be performed to gain insight in a possible underlying structure. This structure could be used as a training model. The statistical analysis comprises the determination of the frequencies of the used dialogue acts in the corpus. The frequencies are obtained by importing the database with the coded dialogues (Appendix B) into 'Microsoft Access' (a database management program) and using a query language (SQL) to count the records per dialogue act.

In Table 3-8 and Table 3-9, the frequencies of the dialogue acts, explained in section 3.2.2, are shown. In Table 3-10, Table 3-11 and Table 3-12, the frequencies of the sub-coding and the sub-sub-coding of the dialogue acts *QC*, *RC2* and *RO4* are displayed. The numbers shown in the second columns are the total frequencies of each dialogue act in all the two hundred dialogues of the corpus.

Table 3-8 Frequencies per dialogue act operator

<i>Dialogue acts operator</i>	<i>Frequency</i>
OO	202
V	125
CL	101
NI	99
R	2
RO	234
GA	1383
GO	331

Table 3-9 Frequencies per dialogue act client

<i>Dialogue acts client</i>	<i>Frequency</i>
OC	264
AV	119
ACL	95
ANI	96
AR	2
RC	738
QC	575
GC	406

There are several frequencies that are noteworthy. As shown in Table 3-8 the dialogue act *GA* occurs much more often than the other dialogue acts. This number is so high because most of the time during a dialogue, the operator is giving an answer and the answer is often distributed among several utterances alternating with remarks of the client (see Figure 3-38). Therefore, the number of the dialogue act *RC* is also high.

'Saying goodbye' (*GO* and *GC*) is also distributed among several utterances because the number is almost twice the number of the dialogues. The operator does not often use the repetition question (*R*). Assumedly she uses a verification question to ask the client for a repetition of the provided information.

Table 3-10 Frequencies per additional coding QC

<i>Sub-coding QC</i>	<i>Frequency</i>
New question	284
Question about answer	291
Verification	192
Clarification	98
Repetition	1

Table 3-11 Frequencies per additional coding RC2

<i>Sub-coding RC2</i>	<i>Frequency</i>
Rest	102
Paraphrase	124
Alignment word/Acknowledgement	466
Ja	348
Ja ok	7
Nee	6
Prima	8
Mm	20
Oke	23
Oh ja	10
Oh	9
Overig	35*

Table 3-12 Frequencies per additional coding RO4

<i>Sub-coding RO4</i>	<i>Frequency</i>
Rest	30
Paraphrase	0
Alignment word/Acknowledgement	97
Ja	69
Goed	5
Prima	8
Oke	6
Overige	9*

* Alignment words are coded as *Overige* when their frequency is less than five.

The number of additional questions that are asked by the client (*RC*) is higher than the number of opening questions (*OC*). As shown in Table 3-10 this number is distributed evenly among a *new question* and a *question about an answer*. In this last sub-coding category, it is noticeable that most of these questions are verifications. The most occurring sub-coding with dialogue acts *RC2* and *RO4* is *alignment word*. Moreover, the most actual used alignment word by both the operator and the client is 'yes'. So as mentioned above the client often acknowledges the operator's answer utterances with the alignment word 'yes'.

The only sub-coding of which no frequency is determined is the sub-coding of GA, because if the numbering of the operator's answer would be counted it would not result in a clear overview. Nevertheless, there is a thing noteworthy of the sub-coding of GA. Most of the time, whenever the operator is giving an answer, this is distributed among several utterances (as described above). In these cases, the sub-coding of GA increases regularly. In other words, the right digit is raised until the operator is finished with the answer. However, sometimes during the giving of the answer the operator is interrupted by a client's question to which the operator reacts. Then the left digit of the sub-coding of GA is raised to indicate that it is an answer to another question. The left digit decreases again and continues with the previous sub-coding when the operator returns to give the answer of the previous question (of which she was interrupted). So the travel scheme is not listened to entirely before asking a question. Probably clients do this because they immediately want to know the information they ask for without listening to remainder of the operator's answer first. In Figure 3-38, this is illustrated in a coded dialogue example.

Utterance	Coding	Sub-coding	Sub-Sub-coding
2: goedemiddag reisinformatie	OO		
1: goedemiddag met [voornaam+achternaam] ik wilde graag weten hoe laat ik uit Rotterdam C S moet vertrekken om om kwart voor twaalf in Den Bosch Oost aan te komen met de trein	OC		
	EMPTY		
2: als u mee gaat om tien uur twaalf met de trein richting Heerlen	GA	00	
1: Heerlen	QC	vraag over antwoord	verificatie
2: ja	GA	10	
1: tien uur twaalf richting Heerlen	RC2	parafrasering	
2: dan stapt u in Breda over om tien uur vijftig op de trein naar Zwolle	GA	01	
1: Breda overstappen naar Zwolle	RC2	parafrasering	
2: [cough] pardon	RO4	rest	
1: [uh] op welk #1 perron # vertrekt die trein naar Zwolle	QC	nieuwe vraag	
2: #1 [cough] #	OUT OF DIALOGUE		
2: ik ga even voor u kijken	GA	20	
2: [uh] in: Rotterdam vertrekt die om zes a b	GA	20	
1: ja	RC2	stopwoord/bevestiging	ja
2: en in Breda vertrekt die van spoor drie en dan stapt u in Den Bosch over op spoor vier a maar dan moet u eerst nog in Den Bosch aankomen he	GA	21	
1: ja #2 [laughter] #	RC2	stopwoord/bevestiging	ja
2: #2 tien uur vijftig vertrok u # richting Zwolle en u komt daar elf uur vierentwintig aan	GA	02	
1: elf uur vierentwintig	RC2	parafrasering	
2: komt u in Den Bosch aan	GA	03	
1: ja	RC2	stopwoord/bevestiging	ja
2: en om elf zesendertig stapt u daar over op de trein naar Nijmegen en dan bent u elf negenendertig in Den Bosch Oost	GA	04	
1:...			
			1 = client 2 = operator

Dialogue 81494802

Figure 3-38 Dialogue fragment in which sub-coding of GA is not increased regularly

3.5 Additional comments

During the coding of the corpus, several distinct issues were encountered that influence the coding. These issues are described below.

- *Prosody*

As explained in section 3.2.1 the dialogues of the corpus were only available in printed form and therefore coded without being able to interpret and make use of prosody. Without prosody the meaning of a sentence is not always clear and can be interpreted in more than one way. It is not always possible to correctly interpret the utterances without prosody. Sometimes it can not be solved with the help of the previous or next utterance and if an utterance is interpreted based on an assumption of the prosody, then this leads to subjective coding. It is also difficult to determine how much influence non-verbal utterances, pauses or silences have on the dialogue. These utterances can have a meaning but without prosody this meaning is not clear.

The following client utterance is an example where it is not clear if it could be coded as a remark or a verification question, because the difference can only be heard.

Client : nine fifty eight

- *History and future*

To interpret the client utterance given in the above example the history and future are needed. As a stand-alone utterance, it could be coded as an answer, a verification question or a paraphrase remark. It depends on the location in the dialogue and the previous and next utterance how this utterance is coded, because an utterance has no stand-alone interpretation. So the history and the future of the utterance are needed in coding the utterances because there are relations between utterances. For example a question is followed by an answer most of the time. These relations determine what is meant by the utterances. The only exception to this is the first utterance of the dialogue. To illustrate this the same example as above is used.

Operator : you can leave at nine fifty eight
Client : nine fifty eight
Operator : then you arrive at Amsterdam central station at ten fifty five

The client utterance is coded as paraphrase remark because the operator did not ask a question, therefore it can not be an answer and it is not a verification question because the operator does not respond with an answer.

- *Context*

To correctly interpret and code a dialogue elementary knowledge about the context is necessary. Therefore, knowledge is required about the domain, how an information seeking dialogue is conducted, the dialogue topic and the dialogue scripts. In the case of OVR, this includes knowledge about OVR, public transport and the OVR-dialogue. Without this knowledge, the dialogues can still be coded but the results will differ.

- *Classification*

Classifying dialogue acts or to determine the characteristics of a dialogue act are reflected in the coding. Even if a dialogue act is described very well, there will always be an unclear part. It is almost impossible to describe a dialogue act unambiguously. When the utterance does not fit the description exactly, it is difficult to determine how to code the utterance. In these cases, a subjective interpretation and coding is used.

- *Coding level*

Coding every single part of an utterance is not useful because this will make the resulting code and model very complex. Therefore, it is necessary to give one part of an utterance precedence over another. Deciding what is important in a dialogue and thus establishing a coding level influences the coding.

- *Content*

When coding the dialogues, besides the linguistics, the content of an utterance is used in determining the dialogue act of that utterance. For example, if the given answer to a question is not the corresponding response then the utterance is not coded as an answer. This can only be done by looking at the contents of the utterance and not just looking at the mere linguistics.

The client utterance is not coded as an answer in the following example:

Operator : what time would you like to travel
Client : yes, please

This leads to subjective coding because of the interpretation of the content.

All these issues not only influence the coding but also make coding the dialogues very complex and time consuming. Since a large part of the solutions for the factors described above rely on interpretation, automation of the coding process will be very difficult, if not impossible.

Regio: 1
Gespreksnummer: 11845602
Gespreksduur: 33.10 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 11:54 - 11:57

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachttijd: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

1: ja in[uh] de
2: u kunt om zeven uur
2: ja
2: ja
2: twintig uur nege
dag 1: ja
en anders twintig
en: daarna
<twintig> <uur> <<nee>> eenentwintig uur negentwintig pas weer
1: ja oké bedankt #1 ja da:g #
2: ja #1 tot uw dienst dag mevrouw #
1: [noise] [tone] [tone]

2: #1 waar vandaan
1: van Utrecht naar #
2: ja
1: vroeg ik me af[uh] hoe laat hij ging
2: die gaat om tien over heel en tien over half
2: #2 reistijd # is vijftwintig minuten
1: oh dat is (()) (()) tien over heel en tien over half
2: #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn #3 oké # en die stopt gewor
1: #4 oké # prima bedankt #5 da:g #
2: #5 graag gedaan # da:g
1: [noise] [tone] [tone]

Regio: 1
Gespreksnummer: 11845702
Gespreksduur: 63.99 (s)
Categorie: trein_info
Wachttijd: 191 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 14:02

2: goedemiddag reisinformatie
1: ja <goede> [uh] goedemiddag
[u:h] (()) (()) (()) dan re
2: hoe laat wilt u dan re
1: [u:h] ja tegen een ur
2: de intercitty-trein
1: de intercitty-trein
1: en [u:h] dan
2: ook daar
1: oké er
2: zee

2: ja Oké
2: <ik> <w
1: ja
2: n
1: ja
2: n
1: ja
2: n
1: ja
2: n

2: #1 waar vandaan
1: van Utrecht naar #
2: ja
1: vroeg ik me af[uh] hoe laat hij ging
2: die gaat om tien over heel en tien over half
2: #2 reistijd # is vijftwintig minuten
1: oh dat is (()) (()) tien over heel en tien over half
2: #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn #3 oké # en die stopt gewor
1: #4 oké # prima bedankt #5 da:g #
2: #5 graag gedaan # da:g
1: [noise] [tone] [tone]

2: #1 waar vandaan
1: van Utrecht naar #
2: ja
1: vroeg ik me af[uh] hoe laat hij ging
2: die gaat om tien over heel en tien over half
2: #2 reistijd # is vijftwintig minuten
1: oh dat is (()) (()) tien over heel en tien over half
2: #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn #3 oké # en die stopt gewor
1: #4 oké # prima bedankt #5 da:g #
2: #5 graag gedaan # da:g
1: [noise] [tone] [tone]

CHAPTER FOUR

OVR-dialogue model

CHAPTER FOUR

OVR-dialogue model

When a random sample of dialogues of the corpus is compared then these OVR dialogues look very similar. If OVR-operators are asked to describe the flow of these dialogues, they will also show many similarities. This is because the calling person, the client, calls with a specific purpose, which is to get certain information about travelling with the public transport. It is possible that the dialogue has one or more specific basic structures. In that case, it is possible to develop a script for these kind of dialogues. This script is useful in learning and training because according to Schank (chapter two) people store and process information with the help of scripts. Because of this script, it is possible to learn to manage the dialogue in a structured way and this knowledge can be used in the training environment. Based on the coding information, described in chapter three, an OVR-dialogue script can be constructed. This script is graphically presented in a dialogue model and is described in this chapter. First, a top-level model in section 4.1 and then a lower level model in section 4.2. The last section describes the dialogue from the operator's perspective.

4.1 Top-level OVR-dialogue model

During the coding process, it became apparent, that all OVR-dialogues can be split into four phases on the highest level. This hypothesis was verified based on the coded dialogues. These four phases and their mutual relationships are represented in the 'top-level' model, presented in Figure 4-1 and described below.

1. *Greeting or Opening*. In the first phase, the acquaintance between the operator and the client takes place by means of a greeting. The operator and the client greet each other and often introduce themselves (the operator as 'reisinformatie' and the client by his or her name). This is followed by the client's opening question that includes information about the trip to be made. If the client has provided sufficient information during the opening phase for the operator to determine an answer then the operator supplies the answer immediately skipping the second phase. This is represented in Figure 4-1 with the line that connects the first phase to the third phase.
2. *Query or Filling of the 'slots'*. In this phase the operator tries to retrieve the client's plan and collect the necessary information to be able to answer the client's question when the information provided by the client in the opening question is insufficient. The operator asks for further information if there are still slots to be filled. In order to create a query for the travel planner, the operator must get as much information from the client as possible.
3. *Presenting information or Give answer*. During this phase the operator retrieves information from the travel planner based on the client's information and provides this generated travel information to the client. After the travel scheme is presented, it is possible that the client has another question, either, a new one or a question concerning the given information. This is represented in Figure 4-1 with the line connecting the third phase to the second phase.
4. *Goodbye or End of dialogue*. In this phase, the participants say goodbye. When the question(s) of the client has (have) been answered, the two parties thank each other and send each respective goodbyes. The dialogue is ended this way.

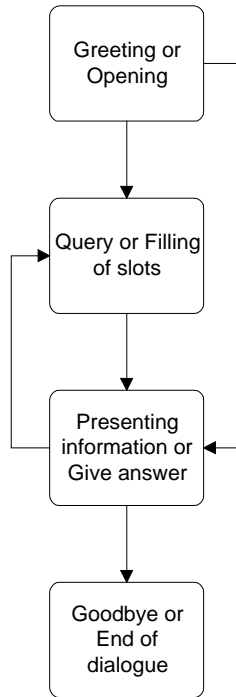


Figure 4-1 Top-level dialogue model

In Figure 4-2, an OVR-dialogue example shows each of the four phases. Each phase is indicated with a bracket and a number on the left side of the dialogue.

```

1  2: [noise/] informatie [/noise] goedemiddag
   1: goedemiddag [u:hm] ik zou graag inlichtingen hoe laat de trein loopt naar
   Schiphol is die in enen door #1 gaat Schiphol #
2  2: #1 ja # en waar vandaan komt u dan
   1: van Enschede
   2: ja Enschede Schiphol hoe laat wilt u graag op Schiphol zijn
   1: tegen half twaalf twaalf uur
   2: ja dat kan rechtstreeks dan moet u om negen uur achtentwintig vertrekken in
3  Enschede
   1: ja
   2: en die trein gaat in een keer door die is elf uur zesendertig op Schiphol
   1: fijn dank u wel
   2: tot uw dienst
4  1: ja
   2: dag
   1: da:g [noise] [tone]
                                     1 = client
                                     2 = operator
dialogue 71714707
  
```

Figure 4-2 The four phases indicated in a dialogue example

4.2 Lower level dialogue model

The model of section 4.1 is the top-level OVR-dialogue model. In this section, a lower level version of this model is constructed and described. The lower level dialogue model is displayed in Figure 4-3. Each phase is split into more details based on the coded dialogues. The dialogue acts described in chapter three are used in this model. In section 4.2.1, the moves between the dialogue acts in the model are described. The validation of the model is described in section 4.2.2, by using transition frequencies. In section 4.2.3, the problem dialogues that are not included in the model are explained. The last section describes the dialogue model with transition frequencies.

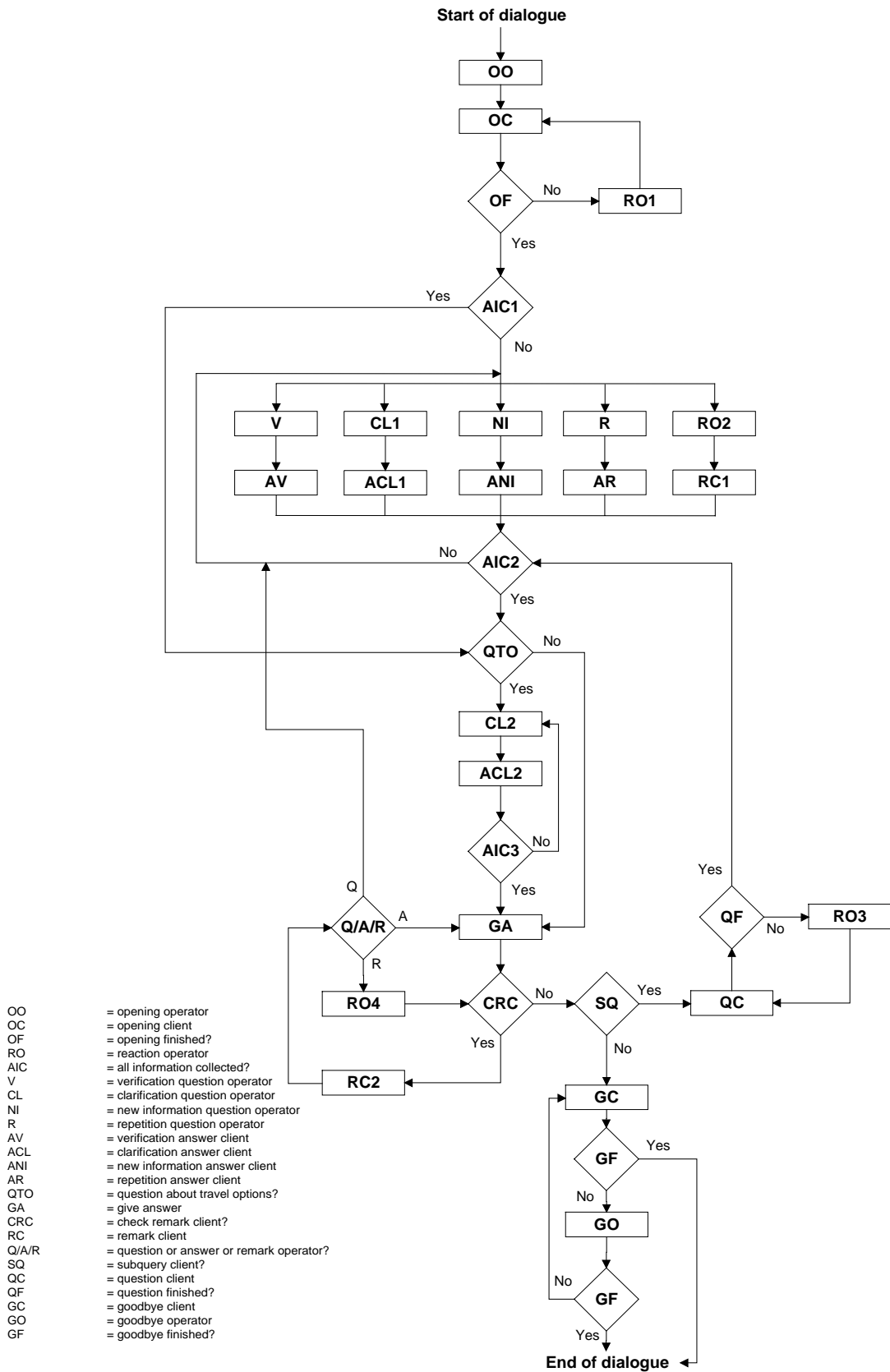


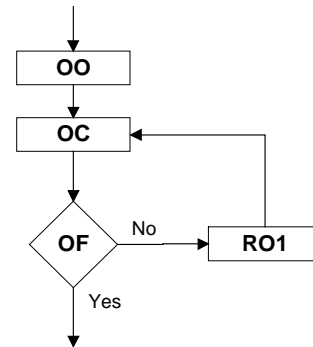
Figure 4-3 OVR-Dialogue model

4.2.1 Explanation of the moves in the dialogue model

The dialogue acts used to code the dialogues are used for designing the dialogue model. They are not explained again in this section because their meaning is the same as described in chapter three. Here the moves between the dialogue acts are described.

Opening finished (OF)

The operator starts the dialogue with a greeting and waits for the client's opening utterance (OO). The client responds by stating his opening utterance (OC). The first phase (greeting/opening) ends when the client has finished his opening question. In several dialogues the operator confirms the opening utterances of the client with a simple 'yes' ('ja') (RO1) and leaves the initiative to the client to finish his opening question. The operator is confirming the client's question because she perceives that the client is not finished with his question and she wants to make the client aware that she is still listening. The decision if the operator is confirming the client's opening question is made in decision box 'OF'. An example dialogue is shown in Figure 4-4.



```

2: goedenavond reisinformatie
1: ja goedenavond [uhm:] ik wil morgen om: kwart over tien 's ochtends in Hoofddorp zijn vanuit Leiden
2: ja
1: met de trein hoe laat moet ik dan weg
2: ik ga even kijken
1: ja
2: u kunt vertrekken om negen negenenveertig dan bent u om tien uur vier in Hoofddorp
1: negen uur negenenveertig en dan ben ik om tien uur vier
2: ja
1: en dat is gewoon doordeweeks
2: op een doordeweekse #1 dag # ja
1: #1 ja #
1: goed
2: oke
1: bedankt
2: ja hoor
1: da:g
2: da:g
  
```

1 = client
2 = operator

Dialogue 21437503

Figure 4-4 Example of 'Opening finished': Operator confirms client's question with 'yes'

All Information Collected 1 (AIC1)

When the client has provided the operator in the opening question with all the necessary information needed to present a travel scheme, the second phase is omitted. The decision to proceed to the third phase and skip the second phase is made in the decision box 'AIC1' (Figure 4-5).

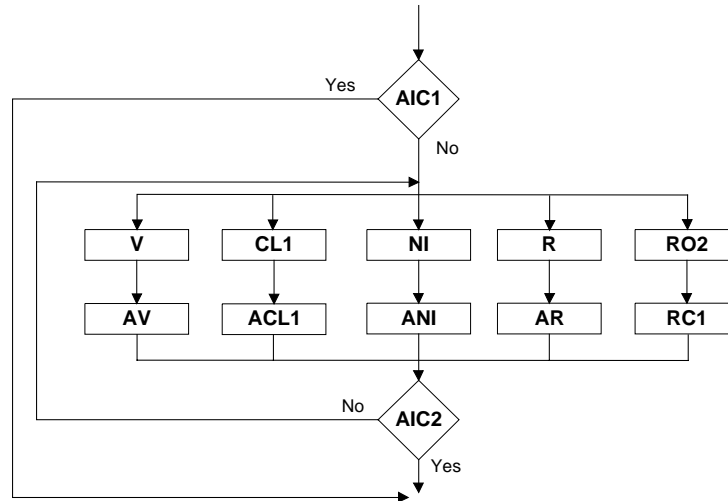
```

2: goedenavond reisinformatie
1: ja hallo u spreekt met [voornaam+achternaam] en ik wou graag weten hoe laat de eerstvolgende trein en die daarna vanuit [uh] Almere stad naar Weesp gaat
2: de eerstvolgende is drieëntwintig eenentwintig
1: ja
2: en dan drieëntwintig zevenendertig
1: ja
2: en dan drieëntwintig eenenvijftig
1: oke bedankt
2: tot uw dienst #1 da:g #
1: #1 da:g # [noise] [tone]
  
```

1 = client
2 = operator

Dialogue 11849205

Figure 4-5 Example of 'All information collected 1': The second phase is skipped



All Information Collected 2 (AIC2)

If the operator requires more information to present a travel scheme she asks a question and enters the query. The questions of the operator within the query phase can be classified into five different categories. These are verification (V), clarification (CL1), new information (NI), repetition (R) and remark operator 2 (RO2). The interpretation of these terms is explained in section 3.2.2. The client answers the question (AV, ACL1, ANI, AR, RC1) and the operator decides if she requires more information. The decision if all necessary information is collected is made in the decision box 'AIC2'. An example dialogue illustrating this is shown in Figure 4-6.

```

2: goedemiddag reisinformatie
1: ja goedemiddag ik moet morgen vanuit[uh] Breukelen naar Amsterdam Centraal [uhm] hoe
  laat moet ik dan vertrekken vanuit Breukelen met de trein
2: ik zal even voor u kijken
2: sorry en hoe laat wilt u daar zijn
1: kwart over een
2: kwart #1 over een #
1: #1 ja dertien # uur #2 vijftien #
2: #2 voor # morgen #3 he was # het
1: #3 ja #
2: [uh] twaalf uur tweeënveertig kunt u dan uit Breukelen vertrekken
1: twaalf uur tweeënveertig
2: ja en dan bent u dertien uur elf in Amsterdam
1: dertien uur elf oké prima
2: ja hoor
1: da:g
2: da:g
1: [noise] [tone] [noise] [tone]
    
```

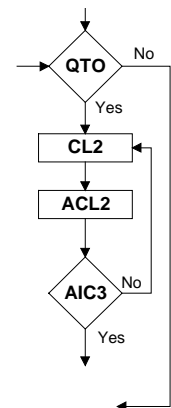
1 = client
2 = operator

Dialogue 31395601

Figure 4-6 Example of 'All information collected 2'

Question about Travel Options (QTO) and All Information Collected 3 (AIC3)

Sometimes the operator has more than one acceptable travel option to present to the client. Some operators ask the client to clarify (CL2) which option they prefer before they present the final answer (GA). The decision if the operator asks a question about the travel options is made in decision box 'QTO'. In most cases, the question concerns a clarification about the travel time. The client has expressed his preferred travel time but the acceptable travel options do not meet this time exactly and the operator wants to know what time is preferred by the client. If more information is necessary to choose a travel option, the operator asks another



question. The decision if all information about the travel options is collected is made in decision box 'AIC3'. In Figure 4-7 and Figure 4-8 examples are shown of two different clarification questions in the third phase.

```

2: goedemiddag reisinformatie
1: goedemiddag ik wil [uhm] morgen vanuit Arnhem
2: ja
1: naar Amersfoort met de trein
2: Arnhem Amersfoort met de trein en om hoe wilt u reizen
1: [uh] even kijken ik moet om half twaalf in Amersfoort zijn
2: en dat was voor morgen he
1: ja
2: mag het elf uur zevenendertig worden
1: [uh] ja dat mag ook [laughter]
2: dan kunt u vertrekken vanuit Arnhem om tien uur zevenendertig richting Den Helder
1: tien uur zevenendertig Den Helder
2: overstappen in Utrecht
1: overstappen in Utrecht
2: u komt in Utrecht aan om elf uur dertien
1: ja
2: en u vertrekt om elf uur tweeëntwintig <i-> [uh] vanuit Utrecht richting Leeuwarden
1: Leeuwarden
2: ja bent u om elf uur zevendertig in Amersfoort
1: oke dank u wel
2: graag gedaan
1: da:g
2: da:g
1: [noise] [noise] [tone] [tone]

```

1 = client
2 = operator

Dialogue 31396102

Figure 4-7 Example of clarification question in the third phase (QTO)

```

2: #1 reisinformatie # goedenavond
1: #1 [noise] [noise] # hallo met [achternaam] Groningen ik wou graag even weten ik moet
morgenochtend [uh] of morgen moet ik om twaalf uur in Dieren zijn #2 en #
2: #2 waar # vandaan #3 uit Groningen #
1: #3 <van-> # vanuit Groningen ja welke trein kan ik het beste nemen of treinen
2: u kunt aankomen om zes over half twaalf of anders om zes over twaalf
1: doe maar zes over half twaalf dan
2: negen uur zevenendertig uit Groningen
1: negen uur zevenendertig
2: ja
1: [mm]
2: dan stapt u over in Zwolle
1: ja
2: u komt aan in Zwolle om tien drieënveertig en u stapt over o:m tien uur vijftig meestal
vanaf perron vijf A elf zesendertig in Dieren
1: #4 uitstekend #
2: #4 <dat> <is> # dat is de trein richting Roosendaal
1: prima dank u wel
2: graag gedaan
1: oké #5 dag # [noise] [tone] [noise] [tone]
2: #5 da:g #

```

1 = client
2 = operator

Dialogue 41748303

Figure 4-8 Second example of clarification question in the third phase (QTO)

If the operator does not ask a question (CL2) about the travel options, it seems like she decides what option is the best or she provides the client (if the client wants to know) with several travel options that closely approach the preferred time (Figure 4-9).

```

2: goede:middag reisinformatie
1: goedemiddag met[uh] [voornaam+achternaam] [uhm] de trein naar Schiphol #1 # [uh] op
  maandagochtend <als> <ik> ik moet om zeven uur 's ochtends <op> <Sch-> op Schiphol zijn
  vanuit Utrecht
2: #1 ja: #
2: vanuit Utrecht
2: [uhm:] ja u kunt dan om[uh] zeven uur twee aankomen
1: ja
2: die vertrekt om zes uur vijf vanuit Utrecht
1: zes uur vijf
2: ja en die gaat rechtstreeks
1: die gaat rechtstreeks
2: en er is wel een trein die komt om tien voor zeven aan
1: ja
2: <maar> <die[uh]> heeft u een trein met een overstap in Duivendrecht
1: o:h ja
2: en dan zou u zes uur zestien uit Utrecht kunnen vertrekken
1: nee oké dus vijf over zes rechtstreeks naar Schiphol
2: ja hoor
1: oké en[uh] kunt u[uh] ook zien wel spoor dat toevallig is #2 of[uh] #
2: #2 ja # dat is[uh] vanaf spoor twee
1: spoor twee
2: ja
1: oké bedankt
2: tot uw dienst
1: hoi
2: da:g
1: [noise] [tone]

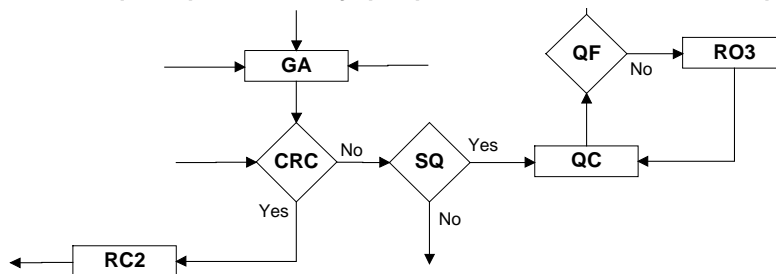
```

1 = client
2 = operator

Dialogue 31395301

Figure 4-9 Example where the operator provides two options to the client without clarifying first

Check Remark Client (CRC), SubQuery (SQ) and Question Finished (QF)



The operator gives the answer of the client's question (GA) and the client responds with a remark (RC2) or asks another question (QC). The decision if the client responds with a remark is made in decision box 'CRC'. An example dialogue with several remarks is shown in Figure 4-10.

```

2: goedemiddag reisinformatie
1: ja goedemiddag kunt u mij vertellen hoe ik van[uh] Eindhoven naar Keulen kom <en[uh]> en
  zodat ik om acht uur in Keulen ben #1 vanavond #
2: #1 momentje #
2: u kunt weg vanaf Eindhoven om z:eventien uur drieëndertig #2 pakt # u de trein richting
  Keulen
1: #2 ja #
1: ja
2: komt u in de Keulen Hauptbahnhof om negentien na negentien rechtstreeks
1: negentien uur negentien dank u wel
2: graag gedaan #3 dag: #
1: #3 dag # meneer [noise] [tone] [tone]

```

1 = client
2 = operator

Dialogue 91536603

Figure 4-10 Example of several clients' remarks on the operator's answer

If the client does not respond with a remark, the decision if the client asks another question is made in decision box 'SQ'. In several dialogues the operator confirms the client's question with a simple 'yes' (RO3) and leaves the initiative to the client to finish his question. The decision if the operator is confirming the client's question is made in decision box 'QF'. The client's question (QC) can be a question about the preceding utterance of the operator or a question about a new topic. For the latter, the query can be entered again if the operator needs more information (AIC2). For the former, this is mostly not the case. In Figure 4-11 an example dialogue is shown.

```

2: goedemiddag reisinformatie
1: ja goedemiddag met [achternaam] ik wou graag weten hoe laat[uh] de trein [uh] vanuit
  Breda naar [uh] Tilburg vertrekt[uh] de intercity die tijden zijn als het goed veranderd
  dat ik[uh] #1 gehoord #
2: #1 twee keer # per uur zeventien minuten over het uur en zevenenveertig minuten over het
  uur
1: dat is gewoon hetzelfde gebleven
2: dat kan ja #2 het is # niet alles veranderd he
1: #2 ah [uh] #
1: weet u of er momenteel[uh] op dit traject ook vertragingen zijn opgetreden
2: nee niks bekend
1: niks bekend
2: #3 nee #
1: #3 oké # hartelijk dank
2: alsjeblieft
1: he #4 dag #
2: #4 dag: #
1: [noise] [tone] [tone]

```

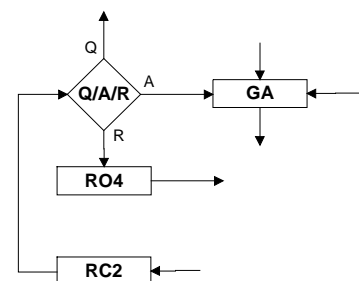
1 = client
2 = operator

Dialogue 91536602

Figure 4-11 Example of 'Sub-query' – Clients asks several questions

Question, Answer or remark operator (Q/A/R)

After a client's remark (RC2) the operator mostly continues with answering the question (GA) but sometimes she utters a remark (RO4) or asks a question and enters the query again. The decision if the operator utters a question, an answer or a remark is made in decision box 'Q/A/R'. An example of an operator's remark (RO4) is shown in Figure 4-12 and an operator's question (Q) is shown in Figure 4-13.



```

2: goedemiddag reisinformatie
1: ja goedemiddag [uhm] ik wil vanavond omstreeks zeven uur in Nijmegen zijn vertrek vanuit
  Roermond [uh] hoe laat kom ik precies aan in Nijmegen [typewriter] en[uh] hoe laat moet
  ik in Roermond vertrekken
2: dus u wilt er rond zeven uur zijn mevrouw
1: ja ietsje eerder eventueel tussen half zeven en zeven uur
2: ja dan vertrekt u om zeventien uur elf uit Roermond
1: ja
2: bent u om achttien achtendertig in Nijmegen
1: oké prima
2: ja
1: en op welk perron[uh] vertrekt die van zeventien uur elf
2: drie A mevrouw
1: drie A #1 # oké #2 da:g #
2: #1 ja #
2: #2 oké da:g #

```

1 = client
2 = operator

Dialogue 91536303

Figure 4-12 Example of 'Remark operator 4'

```

2: reisinformatie goedemiddag
1: goedemiddag u spreekt met [achternaam] ik moet [u:h] [uh] vanuit Apeldoorn naar Nijmegen
  Dukenburg en in: Nijmegen Dukenburg [uh] wil ik nou zo rond vijf uur iets voor vijven
  zijn
2: vandaag
1: vandaag met de trein hoe laat #1 moet #
2: #1 vijftien # negenendertig kunt u vertrekken uit Apeldoorn
1: vijftien negenendertig ja
2: overstap in Zutphen en in Nijmegen Centraal
1: [u:h] overstap in: Zutphen hoor ik [u:hm]
2: wilt u de aankomst tijden erbij
1: ja
2: u komt aan in Zutphen vijftien zesenvijftig
1: ja
2: overstappen in de trein naar Roosendaal die vertrekt om vijftien achtenvijftig
1: Roosendaal vijftien <achten> <<oe>> oh die staat aan de andere #2 kant van het perron
  waarschijnlijk #
2: #2 is een directe ja is of het zelfde of aan de andere kant #
1: ja
2: en dan bent u in Nijmegen zestien vijfendertig
1: zestien vijfendertig
2: hier: is het dan overstappen in de trein richting Den Bosch
1: Den Bosch
2: die vertrekt om zestien eenenvijftig
1: zestien eenenvijftig
2: en is om zestien zesenvijftig in Dukenburg
1: fijn bedankt
2: graag gedaan #3 da:g #
1: #3 dag # [noise] [tone] [tone]

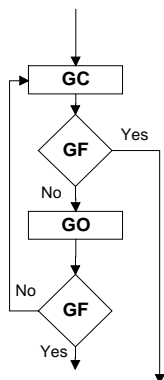
```

1 = client
2 = operator

Dialogue 71715802

Figure 4-13 Example of operator's question after client's remark

Goodbye Finished (GF)



The client ends the dialogue with a goodbye (GC) and the operator responds in the same way (GO). Sometimes the client responds with another goodbye. The decision if the dialogue has ended is made in decision box 'GF'. An example dialogue is shown in Figure 4-14.

```

2: goedemorgen reisinformatie
1: hallo [u:h] ik moet de trein van[u:h] Zaandam naar Amsterdam Centraal hebben en ik moet
  om half acht op Centraal zijn weet u hoe laat die gaat
2: ik kijk even moment zeven uur acht uit Zaandam vertrekken
1: oké bedankt
2: graag gedaan
1: da:g
2: da:g
1: [noise] [tone]

```

1 = client
2 = operator

Dialogue 11852602

Figure 4-14 Example of goodbyes

4.2.2 Transition frequencies

To be able to validate the dialogue model the transitions between the dialogue acts are used. To determine all possible transitions that occur in the coded corpus and to count the frequency per transition, a program was written in Microsoft Access. These are shown in Table 4-1. The dialogue acts are ordered in time, corresponding to the natural dialogue. The numbers in the cells represent the total number of transitions between the dialogue acts in the corresponding row and column header of the table in the two hundred coded corpus dialogues. The table shows that most of the transitions are distributed around the main table diagonal except for the transitions concerning the dialogue act GA. The dialogue act GA has transitions with several other dialogue acts, because the operator can give an answer at any time in the dialogue whenever the provided information is sufficient. This can also be seen in Figure 4-3 where, in comparison with other dialogue acts, more arrows are entering and exiting the dialogue act GA. The distribution of almost all transitions around the main table diagonal indicates that the dialogue acts have a transition with primarily the next dialogue act, which is the next step in time in the dialogue. This is also reflected in the model.

The table shows that there are many transitions that occur frequently and a few less frequently. All except the least frequent transitions are represented in the dialogue model of Figure 4-3, because the least frequent transitions are not significant enough.

Table 4-1 Transitions between the used dialogue acts (read from left to right)

	OO	OC	RO1	V	AV	CL1	ACL1	NI	ANI	R	AR	RO2	RC1	CL2	ACL2	GA	QC	RO3	RC2	RO4	GC	GO	GA+NI	AV+OC	ANI+QC	GA+CL1	ANI+OC	ACL1+OC
OO	202																											
OC	2	48	55			26		45				10		4		63							1					
RO1		51																										
V				113																				5				
AV				23	13		14					9		7		48												
CL1						70																						
ACL1			2	7		5	8					6		4		38					2							1
NI								86					1															
ANI			1	14		7		10	86			7	1			49								2			1	
R										2																		
AR											2					2												
RO2					1		2		1																			
RC1													40															
CL2	1			2		1						9		2		27												
ACL2															17		6				1							
GA				3											1	16						1						
QC					8		13		12		2	2		2		4	498				653	145						
RO3																	12	12										
RC2				5		4		1						2		549												
RO4									1																			
GC																												
GO																	6											
GA+NI									1				1															
AV+OC				1								1		1		2												
ANI+QC						1										2												
GA+CL1							1																					
ANI+OC																1												
ACL1+OC																1												

4.2.3 Problem classes not included in this model

The model covers about 80 percent of the dialogues from the corpus. The remaining 20 percent of the dialogues can be classified into a number of problem classes. These problem classes consist of the least frequent transitions in Table 4-1. The problems are mainly caused by initiative switches that differ from the ones in the model. The operator or the client is interrupting the other one or is not responding to a remark or question from the other one. The problem classes are explained.

- 1 The starting utterance of the operator is repeated later in the dialogue, because of connecting through to another person. To illustrate this problem a fragment of an example dialogue is shown in Figure 4-15.

```

2: goedemiddag reisinformatie
1: ja goedemiddag [bedrijfsnaam] ogenblikje ik verbind u door
2: ja hoor
1: [tone]
1: goedemiddag met [voornaam+achternaam]
2: reisinformatie goedemiddag
1: hallo goedemiddag (()) (()) graag weten voor komende dinsdag zes juni
2: ja
1: [uhm] de aankomsttijd van Utrecht Heerlen Heerlen Utrecht retour ik moet rond kwart
   voor twaalf twaalf uur in Utrecht zijn
.....

```

1 = client
2 = operator

Dialogue 91536203

Figure 4-15 Example of problem 1: Starting utterance of operator is repeated later in the dialogue

- 2 The operator interrupts the client during his opening utterance. After this interruption, the operator leaves the initiative to the client so that he can finish the utterance. In some dialogues, the operator interrupts the client because the client is drifting away from the main subject. In that case, it can be useful to interrupt the client because then the operator saves some time by taking over the initiative. A fragment of an example dialogue is shown in Figure 4-16.

```

2: reisinformatie
1: goedemiddag u spreekt met[uh] [voornaam+achternaam] ik wilde graag even weten morgen
   rond een uur of half zeven
2: 's morgens
1: ja
2: ja
1: [uh] <naar[uh]> van Rotterdam Centraal station naar Groningen hoe laat dan de trein
   gaat
2: zes uur zevenendertig
1: zes zevenendertig
2: van perron twaalf a
1: ja
2: bent u in Utrecht om zeven uur dertien
1: ja
2: neemt u daar op perron negen a om zeven uur tweeëntwintig de trein naar Groningen
1: ja
2: bent u om negen uur veertien in Groningen
.....

```

1 = client
2 = operator

Dialogue 81496303

Figure 4-16 Example of problem 2: Operator interrupts client during his opening utterance

- 3 The client answers a question of the operator but at the same time asks a question. In some of the dialogues, this last question is the remainder of the opening question. This combination of answer and opening question is possibly caused by the operator's interruption. In Figure 4-17 an example dialogue is shown in which the client gives an answer and finishes his opening question.

```

2: [noise] reisinformatie
1: ja goeie<morgen> [uh] middag [laughter] [uh] morgen moet ik zo rond half: elf vanuit
  Deventer naar [u:h] <Hooge-> Hoogeveen ja Hoogeveen zijn dus [u:h] #1 wat is dan #
2: #1 half elf daar aankomen zegt u #
1: ja zo rond half elf wat is dan de beste tijd om hier vanuit Deventer te vertrekken en
  een goeie aansluiting te hebben in Zwolle
2: negen uur negentien mevrouw
1: negen uur negentien
2: bent u negen drieënveertig in Zwolle
1: ja
2: en negen negenenveertig kunt u verder reizen van spoor drie
1: spoor drie
2: en dan bent u tien uur zestien in Hoogeveen
1: oh dat is een prachtige tijd en daar is treintaxi he
2: daar is treintaxi
1: ja fijn hoor vriendelijk bedankt
2: tot uw dienst #2 da:g #
1: #2 dag mevrouw # [noise] [tone]

```

1 = client
2 = operator

Dialogue 71714904

Figure 4-17 Example of problem 3: Client answers question and at the same time asks a question

- 4 The operator answers a question but at the same time asks a new question. This problem probably occurs because the operator is able to quickly give an answer but wants to have more information to provide the client with possibly a more complete answer. Figure 4-18 shows a fragment of an example dialogue.

```

.....
2: dan bent u in Middelburg om veertien uur vijf
1: veertien uur vijf en dat is gewoon de trein[uh] richting Vlissingen Middelburg of
  Middelburg Vlissingen
2: ja de eindbestemming op die trein staat Vlissingen en #6 stopt in Middelburg #
1: #6 ja ja #
1: oké zet ik dat even achter [uh] wat is een enkele reis[uh] Zwolle Middelburg
2: enkele reis tweede klas is drieënvijftig gulden vijftig binnen welke termijn komt u
  terug meneer
1: nou <ik[uh]> het is gewoon enkele reis want ik ga[uh] later met de auto weer terug
2: oh op die manier
1: dan rij ik met iemand anders mee
2: drieënvijftig vijftig
1: goed en[uh] even denken moet ik nog meer vragen nee ik denk dat ik weet goed bedankt
  hoor
2: ja hoor
1: #7 da:g #
2: #7 dag # meneer
1: [noise] [tone]

```

1 = client
2 = operator

Dialogue 61573307

Figure 4-18 Example of problem 4: Operator answers question and at the same time asks a new question

- 5 The client's answer, on a query question of the operator, is split in two by a remark of the operator. This problem is illustrated in Figure 4-19.

```

2: reisinformatie goedemiddag
1: ja goedemiddag u spreekt met mevrouw [achternaam] uit Eindhoven ik wou weten een
   trein die morgenvroeg rond negen uur uit Eindhoven vertrekt richting Amsterdam hoe
   laat is die in Utrecht
2: ja dus van Eindhoven naar Utrecht eigenlijk
1: ja
2: en dan negen uur vertrekken vanuit #1 Eindhoven #
1: #1 ja ik weet niet # ongeveer ik weet niet precies hoe
2: ja
1: laat die gaat
2: ja er vertrekt een trein in Eindhoven mevrouw om negen uur zeven
1: negen uur zeven
2: ja en die komt aan in Utrecht om negen uur negenenvijftig
1: negen negenenvijftig
2: ja
.....

```

1 = client
2 = operator

Dialogue 81496106

Figure 4-19 Example of problem 5: Client's answer is split in two by operator's remark

- 6 This problem is an elaboration on the clarification question of the operator during the third phase (CL2). After this question is asked by the operator, the client does not respond with an answer but with another question. Another problem resulting from this is that in some dialogues the client's answer on the clarification question of the operator follows the operator's answer on the aforementioned unexpected question of the client. Figure 4-20 shows an example dialogue in which these problems occur. A probable cause for this problem is that the operator does not provide the client with all the travel alternatives. Therefore, the client has to ask a question to collect enough information to be able to give an answer. This problem probably does not occur when the information provided by the operator in the question is more explicit.

```

2: [tone] goedemiddag reisinformatie
1: goedemiddag kunt u mij vertellen [uhm] vanaf Almelo naar Schiphol aankomst op
   Schiphol rond half drie welke trein moet worden genomen op een vrijdagmiddag vanuit
   Almelo
2: vrijdag is het he
1: ja
2: [uh] veertien uur zesendertig aankomen mag dat ook nog of wordt dat te laat
1: [uh] ja en een trein eerder
2: nou als u een trein eerder neemt heeft u de trein van twaalf uur zeventien uit Almelo
   en dan moet u in Amersfoort overstappen
1: nee dat is niet de bedoeling dan is aankomst veertien uur zesendertig prima
2: ja twaalf uur zevenveertig uit Almelo is om veertien uur zesendertig op Schiphol
1: oké prima
2: ja
1: weet ik voldoende #1 (( )) (( )) # zonder overstappen he
2: #1 tot uw dienst da:g #
2: wat zegt u
1: dat is zonder #2 overstappen #
2: #2 dat is # zonder overstappen
1: oké
2: ja
1: dank u da:g
2: tot uw dienst da:g
1: [noise] [tone]

```

1 = client
2 = operator

Dialogue 61574906

Figure 4-20 Example of problem 6: Client responds with question instead of answer

- 7 The operator verifies the client's answer (ACL2) on the second clarification question of the operator. An example dialogue is shown in Figure 4-21.

```

2: goedenavond reisinformatie
1: goedenavond [achternaam] spreekt u [u:hm] ik wilde graag weten als ik morgen om:
  morgenochtend om negen uur [u:h] op de Lelylaan in Amsterdam moet zijn hoe laat moet
  ik vanuit Hilversum vertrekken dan
2: ik heb een aankomsttijd van acht uur drieënveertig
1: [u:h] acht uur drieënveertig
2: of negen nul drie
1: oh die vind ik helemaal mooi
2: ja
1: ja
2: dat wordt[u:h] acht uur negentien vertrekken uit Hilversum
1: acht uur ja
2: en dan overstappen Amsterdam
1: [u:h] ja
2: vertrek Amsterdam acht vijfenvijftig
1: acht vijfenvijftig
2: komt u negen nul drie aan Amsterdam Lelylaan
1: oké dus acht uur negentien vertrekken oké hartstikke bedankt
2: tot uw dienst #1 da:g #
1: #1 da:g # [noise] [tone] [tone]

```

1 = client
2 = operator

Dialogue 11848103

Figure 4-21 Example of problem 7: Operator verifies the answer on the second clarification question

- 8 The operator takes the initiative to start with the goodbyes. Generally, the client takes the initiative to start with the goodbyes. This problem possibly occurs because the operator senses that the client is provided with all the information he wants to have. In the example dialogue in Figure 4-22 the operator is taking the initiative twice to start with saying goodbye.

```

.....
1: en daarna wat hoe laat ben ik in Utrecht
2: zeventien uur elf
1: aha en daarna
2: u stapt daar over vertrek om zeventien tweeëntwintig
1: tweeëntwintig
2: en dan bent u in Groningen om negentien uur veertien
1: ne- hoeveel
2: negentien uur veertien (% telefoniste begint nu harder te praten )
1: veertien oké
2: ja
1: ja
2: tot uw dienst
1: welke spoor moet ik hebben nee welke spoor
2: spoor twaalf a b
1: spoor twaalf oké
2: in Rotterdam #3 ja #
1: #3 ja #
2: da:g
1: da:g [noise] [tone]

```

1 = client
2 = operator

Dialogue 81495801

Figure 4-22 Example of problem 8: Operator takes initiative to start with saying goodbye

- 9 The client starts saying goodbye and then asks another question. This problem is illustrated in Figure 4-23.

```

.....
2: die om veertien zeven in Rotterdam is en daar moet u overstappen op de trein van
   veertien uur twaalf van spoor zes
1: ja
2: en vijftien vierentwintig Eindhoven
1: vijf- oh die is te laat laat ik dan die maar niet opschrijven
1: dertien uur dertien zie u de eerste he
2: ja
1: dertien uur dertien en veertien uur vierenvijftig in Eindhoven
2: ja
1: vriendelijk bedankt
2: tot uw dienst
1: en die gaat direct
2: ja
1: zonder over te stappen
2: ja
1: bedankt
2: dag
1: dag

```

1 = client
2 = operator

Dialogue 51804304

Figure 4-23 Example of problem 9: Client says goodbye and then asks another question

- 10 The client is asking questions about other topics apart from travelling with the public transport. In this thesis, public transport is associated with travelling by train and its complementary information such as timetables, transitions, ticket information and station information.

```

2: reisinformatie goedenavond
1: ja goedenavond [achternaam] ik wilde graag weten van morgenochtend
2: ja
1: om zeven uur moet ik (()) op Schiphol zijn
2: ja
1: [u:h] hoe laat gaat de trein van Rotterdam rijden
2: vanaf Rotterdam Centraal mevrouw
1: ja of Alexander kan ook
2: of Alexander maar dan heeft u geen rechtstreekse trein he van <Alex-> Alexander want
   dan gaat u eerst naar Rotterdam Centraal en dan neemt u de trein naar Schiphol dus #1
   ik weet niet #
1: #1 oh nee # dan ga ik in ieder geval (()) direct #2 van Centraal #
2: #2 direct van # Centraal en dan wilt u om zeven uur op Schiphol zijn #3 (()) (()) #
1: #3 het laatst ja # liever ietsje vroeger
2: ja precies dan kunt u weg om precies zes uur vanaf Rotterdam
1: ja
2: en dan bent u om zes uur drieënveertig op Schiphol en dat is dan gewoon rechtstreeks
1: ja en dat[u:h] haal ik nog wel mijn laatste incheck is[uh] tien over zeven dus
2: [u:h] dat ja <u> u bent dan wel ja u bent dan vrij laat denk ik he om daar als u[uh]
   hoe laat gaat het vliegtuig
1: even kijken (()) (())
2: ja
2: ja (()) laatste incheck staan dan zou u zeggen dat dat gewoon op tijd is maar u weet
   niet hoe laat het vliegtuig gaat
1: [mm] wacht even
2: ja
1: ik dacht tien over acht
2: tien over is het een lijnvliegtuig staat er lijnvliegtuig dan is het geen charter
   denk ik #4 nee # dan is het voldoende tijd hoor dan is het geen probleem
1: #4 nee het een lijn # ja oké nou goed dan[uh] ga ik om zes uur [laughter]
2: uitstekend
1: bedankt
2: graag gedaan
1: da:g
2: goedenavond mevrouw
1: [noise] [tone]

```

1 = client
2 = operator

Dialogue 11849402

Figure 4-24 Example of problem 10: Client asks questions about Schiphol

4.2.4 Dialogue model with transition frequencies

In Table 4-2, the frequencies of the transitions that are represented in the dialogue model are shown. So the transitions that occur in the problem dialogues, described in section 4.2.3, are not included in the model.

Table 4-2 Transitions between dialogue acts without problem dialogues

	OO	OC	RO1	V	AV	CL1	ACL1	NI	ANI	R	AR	RO2	RC1	CL2	ACL2	GA	QC	RO3	RC2	RO4	GC	GO
OO	160																					
OC		36	43			20		32				7				58						
RO1		36																				
V				82																		
AV				16	9			8				7		4		38						
CL1						50																
ACL1				4		2	7					4		2		31						
NI								61														
ANI				9	7		5					5		1		34						
R																						
AR																						
RO2													33									
RC1				1								8		1		23						
CL2															11							
ACL2														1		10						
GA																	395		481		118	
QC				6	10		8					2		2		386		9				
RO3																	9					
RC2				3	2		1									414				78		
RO4																						42
GC																	19		17			257
GO																					158	

In Figure 4-25, these frequencies are shown in the dialogue model. The boxes next to the model display the individual values from which the total number per transition, placed next to the transition line, is constructed. For example the number 522, for the transition 'AIC2' to 'QTO', consists of 10 transitions going to 'CL2' and 512 going to 'GA'.

Since all the transitions that occur the most in the corpus also occur in the dialogue model, it can be assumed that the model in Figure 4-25 is an underlying structure of the OVR-dialogue. In addition, some transitions included in the model do not occur often in the corpus. The reason for this is that these transitions make the dialogue model more flexible.

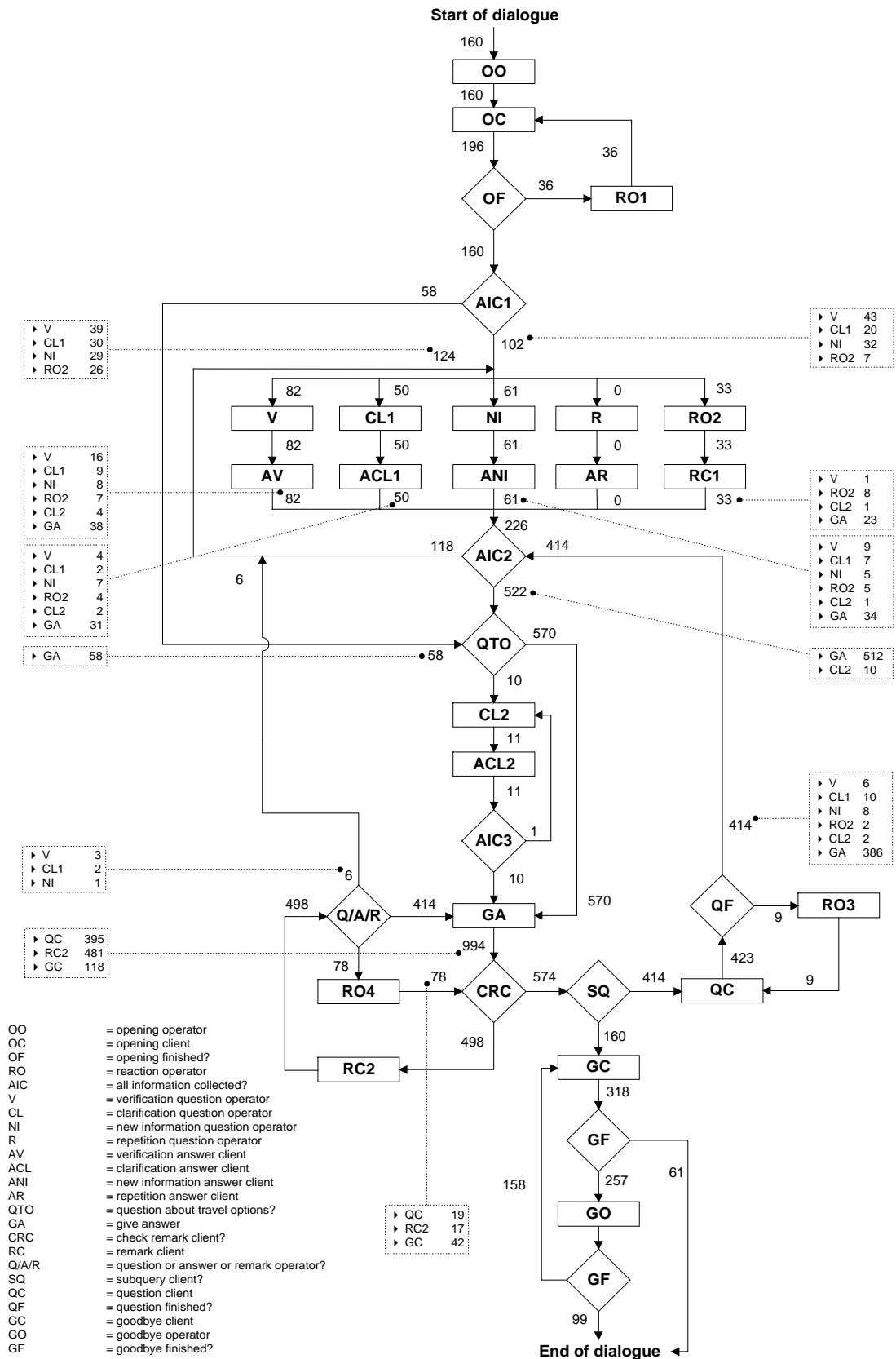


Figure 4-25 Dialogue model with transition frequencies

4.3 Dialogue management from the operator's viewpoint

Information seeking dialogues, such as OVR-dialogues, are all about the exchange of information between the client and the operator. This is however not a symmetric relationship.

In what way does the operator manage the dialogue? Why are most of the corpus dialogues structured in the manner of the dialogue model? In this section, the possible strategies during the management of a dialogue and the operator's perspective on the dialogue are described. These two are combined into a computational model that is described in the last section.

4.3.1 Strategies

Dialogue management can occur in different ways. The simplest way is that the operator asks several successive closed questions. The client has to answer these questions. If the client does not answer, the operator repeats the questions a few times. And if these questions are still not answered the operator will eventually end the dialogue. A more adaptive variant is that the operator can ask open questions and tries to understand and comprehend the client. This way many things can go wrong. Many clients are flexible and tolerant and are willing to accept that the operator takes over the initiative because maybe the question was not formulated properly or was too difficult and so on. The most comprehensible operator tries to consider everything and to understand the client as good as possible. An example of superficial dialogue management is just to analyse if the client provides concrete slot-filling information or clarifies insecurities or obscurities. If not the operator negates the client's utterance and takes over the initiative. These two ways are represented in Figure 4-26.

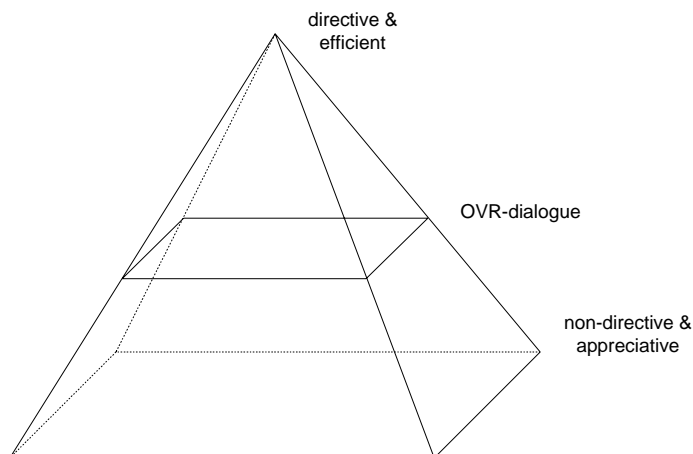


Figure 4-26 Pyramid representing dialogue strategies

The top in the pyramid represents directive, rigid and efficient dialogues. There are not many possible ways to conduct such a dialogue. The bottom represents non-directive and appreciative dialogues. There are many possibilities within the dialogue and various ways to conduct such a dialogue. The OVR-dialogues and the dialogue model are a combination of efficiency and appreciation and therefore these are moving between top and bottom of the pyramid. The position in the pyramid depends on the goals of the dialogue. The consideration between efficiency and appreciation is what the operator is trying to accomplish in the best possible way.

4.3.2 The dialogue from the operator's perspective

This section describes the possible train of thoughts of an operator that she wants to solve during the dialogue. The first section gives an overview of the thoughts. The second section gives a description of the tasks that are involved with these thoughts.

4.3.2.1 Description of the thoughts during a dialogue

At the start of a dialogue, the client formulates his question for information. This request can be put implicitly or explicitly, complete or incomplete, understandable or not understandable, comprehensible or incomprehensible to name just a few variables. A first thought could be if the client is understandable. If this is not the case, the operator can ask for a repetition of the utterance. If the client is still not understandable after these repetitions, the operator ends the dialogue. The same procedures apply when a client keeps mentioning topics that do not concern OVR. Secondly, the operator can ask herself if the utterance of the client is comprehensible. If she didn't understand what the client was saying, she can ask the client to clarify his request.

The next task or thought for the operator is to find out what the client's intentions are. The operator filters the opening question for meaningful elements and certain key words or notions. These words or notions trigger a certain script of successive actions with the operator. With questions concerning arrival and departure times, the operator knows that she needs to have information about four slots to be able to determine the content of the fifth slot by selecting the relevant information in the database. Through the operator's knowledge and experience, she has a variety of words and notions available that she can interpret and place in a certain context. Sometimes key words have to be altered or combined to be interpretable. The result is that the operator has one or more ideas or formulated one or more hypotheses on the client's intentions. These hypotheses will be further validated or tested. If there is still some sort of misunderstanding with the client, the operator understands that her assumption was wrong, otherwise the operator knows she is on the right track with her hypothesis. Within each hypothesis, there is a client problem that needs a solution. The operator then tries to solve this problem in one or more steps. With each step, the operator tries to reach her goal (providing the client with the information) by choosing the best reaction to a client's prompt. This is the fourth thought of the operator. The operator has several thoughts during this dialogue, as shown in Figure 4-27.



4.3.2.2 Explanation of tasks

To solve the thoughts during a dialogue the operator has to perform certain tasks. These will be discussed in more detail. It outlines in what way the client's intentions are met, which prompts are possible and which prompts are selected.

An OVR-operator expects OVR-context dependent information. The operator expects that clients want information about travelling by public transport. The OVR-domain is extensive and includes several topics. A client can ask for information about train departures and arrivals or other ways of public transport. The client can also ask questions about prices or accommodations and even ask for information outside the OVR-domain. It is important that the operator determines the client's intentions from the given information as soon as possible. Sometimes the client introduces his intentions himself with questions such as:

"I would like to have information about the prices of train tickets."

However in most cases, the client opens the dialogue without any introduction to what his intentions are, but his request contains one or more meaningful elements. For example:

"I have to be at Amsterdam Central station at two o'clock today."

At this point, it is not clear what kind of public transport the client wants to use. Probably the client wants to get some information about departures and it is very likely that the client wants to travel by train to arrive at Amsterdam Central station. It is less likely that the client wants to arrive by bus or tram and then transfer at Amsterdam Central station. An experienced operator can estimate the probabilities. In general, the operator filters all relevant information from the client's prompts, relates this information to hypotheses, and determines the most relevant one. In Figure 4-28 possible hypotheses as a reaction on a client's utterance are shown.

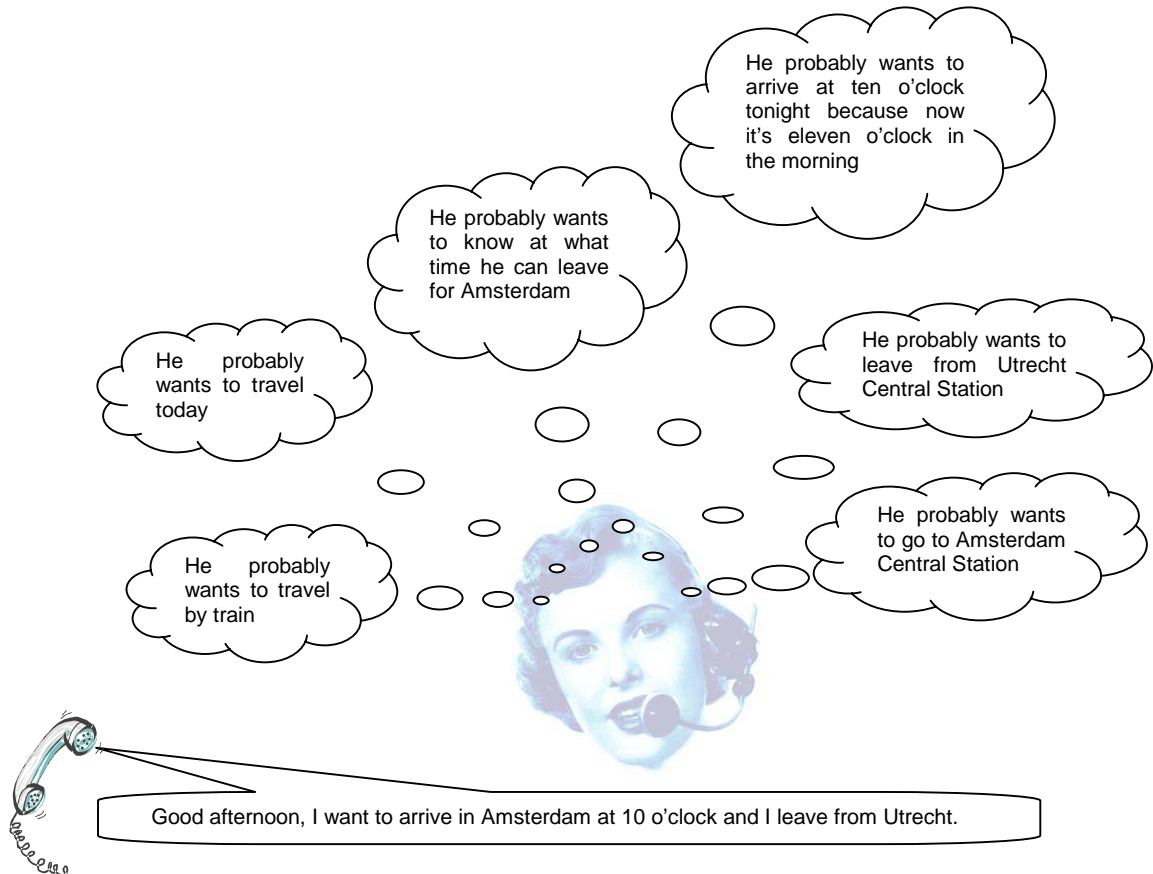


Figure 4-28 Possible hypotheses as a reaction on a client's prompt

To reduce the factor of uncertainty or to determine the most likely hypotheses the operator can react in several ways:

- *With an alignment statement*
The operator can stimulate the client to continue talking so that the client will provide more information for the operator.
- *Directive*
The operator can ask the client how he wants to travel with a direct question.
- *Implicit assumption*
For example the operator assumes the client wants to arrive at Central station and asks where the client wants to depart from. In case the operator's assumption proves to be wrong the client applies a correction.

An operator will try to avoid mistakes and try to reach her target as quick as possible and as customer friendly as possible. In fact the operator calculates the risks and the different uncertainties and calculates implicitly an optimal prompt. Every step in a dialogue is a step closer to the target. As a reaction to a client's prompt, there are several prompts possible, as shown in Figure 4-29.



Figure 4-29 Possible operator's prompts as a reaction on a client's utterance

These prompts can be divided into different categories, as shown in the dialogue model in Figure 4-3. The question is, based on what does the operator select a certain category and based on what does the operator select a prompt from this category. Several factors influence this selection process, these are:

- *Prosody*
If a client takes a moment to take a breath, the operator responds in an understanding and encouraging way by saying 'yes' which means 'please continue your request'. In addition, if the client asks a question it is characterised by its melody and it asks for an answer (not for a counter question).
- *Content*
The contents of the prompt or the contents of the dialogue up until now determine the reaction of an operator. Client's utterances are processed by the operator and may lead to hypotheses that generate certain prompts such as verifications or additional information questions.
- *Certainty*
If an operator is uncertain about the provided information, she can ask questions about it. However, if she is more certain about the information she can make assumptions about it and ask no further questions.
- *Order*
The order in which the information has to be entered in the travel planner influences the prompt choice.
- *Experience*
The experience, routine and the individual style that is acquired during other dialogues influence the prompt selection.
- *Common sense*
For example if it is three o'clock and the client want to leave at six, he probably means six p.m.
- *Context*
The knowledge the operator has about context dependent information such as the topics and the scripts is important in deciding which prompt to use.

The operator can continue with the dialogue in several ways. In most cases, there is a next action or step with a high priority. A certain prompt on a certain moment can be useful and on another moment this same prompt can be inappropriate. A useful prompt causes new information, reduces the uncertainty or provides clearness, whereas an inappropriate prompt takes extra time or causes confusion. In Table 4-3 a number of examples of a client's prompt, an appropriate and inappropriate operator's response are given.

Table 4-3 Several examples of operator's responses on client's prompts

<i>Client prompt</i>	<i>Appropriate operator's response</i>	<i>Inappropriate operator's response</i>
er wordt vandaag gestaakt rijden er treinen	ja	van welk station naar welk station wilt u reizen
wat kost een kaartje van delft naar amsterdam	een enkeltje of een retourtje	hoe laat wilt u reizen
ik moet vanavond om 12 uur in maastricht zijn wat kost dat	waarvandaan vertrekt u?	een enkeltje of een retourtje
hoe laat gaan vandaag de laatste treinen van nijmegen naar amsterdam	wilt u naar amsterdam centraal	hoe laat wilt u vertrekken
ik moet naar amsterdam en ...	ja	van waar vertrekt u
ik moet morgen om 9 uur in delft zijn hoe laat kan ik uit utrecht vertrekken	vertrekt u van utrecht centraal	ja
kunt u mij helpen	ja	nee


4.3.3 Computational dialogue management

In this section, a computational model of the operator is explained which indicates how a dialogue can be managed from the operator's point of view. It combines thoughts about the hypotheses, the client's intentions and the prompt choices. The model is constructed using the goal-directed principle. The operator bases her behaviour on a comparison of a representation of the goal-state and the current state. Based on this comparison, a plan is constructed to move from the current state to the goal-state. The operator can use the planning strategy means-end analysis. Means-end analysis requires a measure of distance between the current and the goal-state. The next step is then chosen based on an evaluation of how much this step will reduce the distance to the goal-state.

Therefore the quality of an operator's prompt is defined by, to which extent the prompt is getting the dialogue nearer to the goal, that is providing the information the client wants to have. One way to assess the quality of the prompts is with the help of an evaluation function with which the prompts are weighed. At every step, the prompt with the highest weight is chosen. It is possible to think several steps ahead and combine the weights that belong to these steps. This way all the possible paths through the dialogue are weighted and the path with the highest weight can be chosen. Thinking several steps ahead is effective in finding the best path through a dialogue. The shortest path without detours produces the most efficient dialogue but it is questionable if the client appreciates such a dialogue. The factors that influence the operator in deciding which prompt to choose (described in the last section), have to be combined in these weights. Because the operator uses her own interpretation and common sense in deciding which prompts to use, it is difficult to calculate exact numbers for these weights. These numbers could possibly be estimated through an advance corpus based research but this is not within the scope of this thesis. However, an explicit evaluation function can be approached by using deduction and reasoning.

The client starts with a question. The operator can react with several prompts to answer this question. Every reaction can accomplish to get nearer or to get farther away from the goal. It depends on the way the operator reacts on a certain moment. A certain reaction on a certain moment can be useful and some other moment this same reaction can be inappropriate. A useful reaction causes new information, reduces the uncertainty or provides clearness, whereas an inappropriate reaction takes extra time or causes confusion. It is also possible to get farther away from the goal because the client gives certain answers that do not contribute to the goal or mentions additional problems or refuses to give answers. One way the operator can resolve this last situation is to repeat the question until she has a satisfying answer. Of course, it is better to ask questions through which the client is inclined to answer the questions satisfactory.

As explained before there are several strategies to manage a dialogue. To visualise an OVR-dialogue from the operator's point of view a certain level somewhere in-between the top and the bottom of the pyramid is taken. To get a clear and not too complicated and elaborated drawing (because then too many reactions would be possible for both operator and client) several variables are neglected while constructing the drawing such as sub-dialogues, inappropriate client reactions and comprehensibility. If these variables were taken into account, the dialogue would be placed lower in the pyramid.

In Figure 4-30 the operator's mental state during a dialogue is displayed. The operator's thoughts are illustrated in the drawing by using slots, represented by five circles. The goal is to fill the slots with the information needed to give an answer. These information elements are described in the legend and indicated at the top of drawing above the first five slots. In the beginning of the dialogue all the slots are empty, because the operator does not have any thoughts about the slots at that moment. The operator's next thoughts (next state) are displayed after an operator's prompt and the client's reaction on it. The next state indicates if there are differences in the operator's thoughts about the slots as compared to the previous state. At every state, there are different ways to continue the dialogue. This is indicated in the figure by multiple arrows on a level. Using an optimal dialogue strategy implies that in the next state, a maximal filling of the slots is created. This filling of the slots is done by reducing uncertainty about the slots or asking for new information. The dialogue acts at the left side of the prompts show how the dialogue proceeds through the dialogue model. In the legend, the possible filling types of the slots are described. Sometimes a slot is filled with two types. This means that the operator combines the two filling types. For example  means that the operator has part of the information and has a hypothesis about the other part of the information.

Dialogue models, as shown in Figure 4-30, were constructed for 200 dialogues of the corpus. In constructing these models, possible hypotheses of the client's intentions were not taken into account, contrary to the model that is described in this section. In a one-step approach, those prompts were selected providing a maximal filling of the slots. In about half of the cases it was possible to generate knowledge rules and heuristics how to select the appropriate prompt. In the other cases, the history has to be taken into account or a multiple step approach, or the client took the freedom to take the initiative or could not provide the information. Some examples of heuristics derived using this strategy are:

- if there is a (non-empty) subset of open slots and a (non-empty) subset of filled slots, ask for information covering as much of the open slots as possible
- if one slot in the current state is not completely filled, immediately ask for the missing information to solve the ambiguity concerning the slot
- if the subset of not complete filled slots contains more than one slot, handle the individual slots one after the other
- as long as new information can be provided, assumptions of the operator are not verified

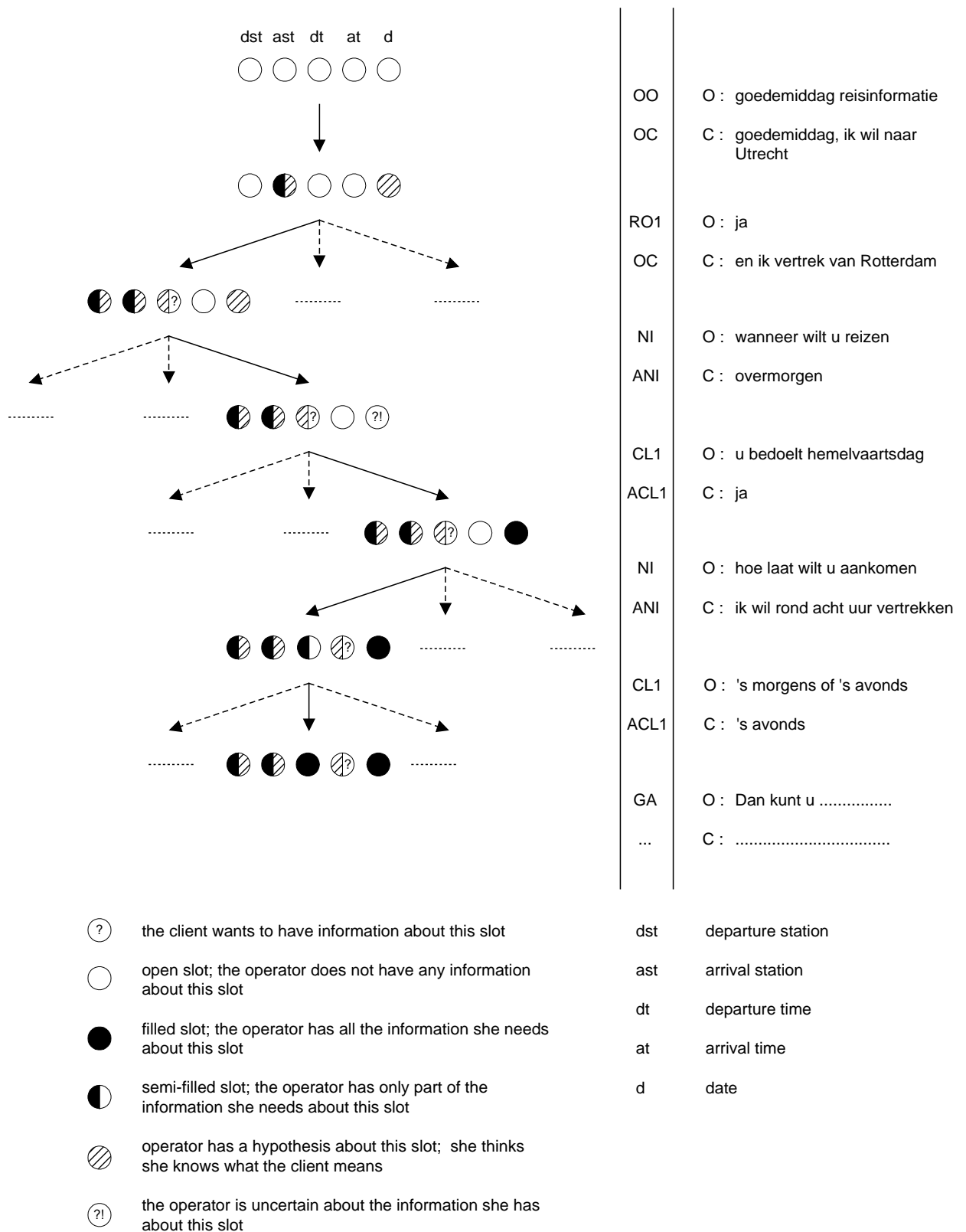


Figure 4-30 Dialogue with the operator's train of thoughts visualised in slots

Regio: 1
Gespreksnummer: 11845602
Gespreksduur: 33.10 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd:

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachttijd: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

Regio: 2
Gespreksnummer: 11845702
Gespreksduur: 63.99 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 14:02

Regio: 1
Gespreksnummer: 11848103
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd:
Transliteratiedatum:
Transliteratietijd:

Regio: 1
Gespreksnummer: 11848103
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd:
Transliteratiedatum:
Transliteratietijd:

Regio: 1
Gespreksnummer: 176.28
Gespreksduur: 96 (s)
Categorie: trein_info
Wachttijd: 18-09-95
Transliteratiedatum: 17:14 - 17:27
Transliteratietijd:
Algemeen commentaar op gesprek:

Regio: 2
Gespreksnummer: 11845702
Gespreksduur: 63.99 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 14:02

Regio: 1
Gespreksnummer: 11848103
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd:
Transliteratiedatum:
Transliteratietijd:

Regio: 2
Gespreksnummer: 11845702
Gespreksduur: 63.99 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 14:02

Regio: 1
Gespreksnummer: 176.28
Gespreksduur: 96 (s)
Categorie: trein_info
Wachttijd: 18-09-95
Transliteratiedatum: 17:14 - 17:27
Transliteratietijd:
Algemeen commentaar op gesprek:

CHAPTER FIVE

Knowledge of the OVR-domain

1: ja
2: twintig uur negenentwintig
1: en anders twintig uur negenentwintig
2: en: daarna
2: <twintig> <uur> <<nee>> eenentwintig uur negenentwintig pas weer
1: ja oké bedankt #1 ja da:g #
2: ja #1 tot uw dienst dag mevrouw #
1: [noise] [tone] [tone]

2: #1
1: van Utrecht naar Alphen aan den Rijn ik neem aan dat de trein naar Leiden is
2: #2
1: oh dat is [laughte/] stopt gewoon in Alphen aan den Rijn [laughte] #4 ja hoor
2: #5
1: [noise] [tone] [tone]

Moeilijkheid van de transcriptie: 4
Omgevingsgeluiden spreker 1: 4
Omgevingsgeluiden spreker 2: 4
Geslacht spreker 1: vrouw
Geslacht spreker 2: vrouw
Kwaliteit van telefonische verbinding: 4
Vloeiendheid van gesprek: 4
Luidheid van de stem van spreker 1: vrouw
Luidheid van de stem van spreker 2: vrouw
Helderheid van de stem van spreker 1: 4
Helderheid van de stem van spreker 2: 4
Accent van spreker 1: 4
Accent van spreker 2: 4
Algemeen commentaar op gesprek: 4

Regio: 1
Gespreksnummer: 176.28
Gespreksduur: 96 (s)
Categorie: trein_info
Wachttijd: 18-09-95
Transliteratiedatum: 17:14 - 17:27
Transliteratietijd:
Algemeen commentaar op gesprek:

CHAPTER FIVE

Knowledge of the OVR-domain

In this chapter, the focus lies on the knowledge or information that is used during the dialogue. This knowledge could be used in the training. In the first section, the knowledge and skills that the operators need to have are described. In section 5.2, the travel planner, that the operators use, is explained. In section 5.3, the information elements and the matching actions that are needed to solve a client's problem are represented in structures. The last section describes in what way the travel information is presented to the client.

5.1 The operator's knowledge and skills

In this section, the knowledge and skills an operator has at her disposal or needs to have at her disposal are described. These knowledge and skills are needed to conduct the dialogues as efficient and effective as possible. In addition, to provide the clients with the most optimal travel information.

5.1.1 Knowledge

An OVR-operator uses a lot of knowledge to be able to answer a client's question and to conduct a dialogue as efficient as possible. Operators use much more knowledge in their daily work than the factual knowledge that can be looked up in the computer (travel planner). This knowledge is often applied naturally and unconsciously. To engage in a dialogue the operator uses the following knowledge:

- **Domain knowledge**
This concerns knowledge about public transport in general.
 - *Terminology*
The operator knows what the several specific public transport terms mean, such as Thalys, intercity, transfer, platform, track, connection etc.
 - *Infrastructure of the public transport net*
The operator knows all the station names so that she knows if certain stations exist. For example, see Figure 5-1.

```
2: goedenavond reisinformatie
1: dag met [achternaam] spreekt u ik zou morgenochtend van Eindhoven Centraal naar
  Amsterdam moeten
2: ja
1: als het goed is is dat Amsterdam Burgemeester de Vlugtlaan kan dat
2: ja dat is een station
.....
```

Dialogue 21429403

1 = client
2 = operator

Figure 5-1 Example of the operator's knowledge about the infrastructure

- *Regulations*

- The operator is familiar with the regulations of the public transport. These can be for example:
- The regulations of season tickets and discounts (NS-jaarkaart, OV-jaarkaart, jaartrajectkaart, maandtjactkaart, maandnetkaart, dagkaart, meerman's kaart, railrunner, voordeel-urenkaart)
 - Taking a pet or a bike on your trip
 - How to use the public transport, the station facilities
 - What to do with lost luggage
 - What to do if you are handicapped
 - What the differences are between the several train types
 - How to use the ticket machine
 - The latest changes in the public transport
 - International train travelling
- **Common sense**
The operator uses her common sense. For example, if a client calls at two o'clock and wants to travel at eight the same day, it is clear that the client means eight o'clock in the evening.

➤ *Time interpretation*

The operator knows how to interpret the time. If the client is saying around two o'clock, she can give a travel option that is a quarter to or maybe only five minutes to two or five past. The operator has to define the window size. In the example shown in Figure 5-2, the operator chooses to give the option that is a quarter to the preferred arrival time. Later in the dialogue can be seen that the operator also could have chosen a quarter past the preferred arrival time.

```

2: goedemiddag reisinformatie
1: hallo u spreekt met [voornaam+achternaam] ik zou graag komende vrijdag met de
  trein van Zwolle naar Harderwijk gaan <naar> aankomsttijd rond twaalf uur in
  Harderwijk (( )) (( )) hoe laat ik in Zwolle moet vertrekken
2: u kunt vertrekken om elf uur eenentwintig vanuit Zwolle
1: ja
2: bent u elf uur zesenvieertig in Harderwijk
1: en de daaropvolgende #1 mogelijkheid #
2: #1 wordt het # elf uur eenenvijftig vertrek bent u om twaalf uur zestien in
  Harderwijk
1: en dat is een stoptrein of sneltrein
2: dat is een [u:h] stoptrein
1: oké dan weet ik voldoende
2: oké #2 graag gedaan #
1: #2 bedankt da:g #
2: dag meneer
1: [noise] [tone] [tone]

```

1 = client
2 = operator

Dialogue 11845602

Figure 5-2 Example of the operator's knowledge about time interpretation

➤ *Knowledge about people*

The operator knows how to react to certain people and certain circumstances. They must be able to place themselves in the client's shoes to get the most satisfactory solution. This way they can anticipate to the client's needs.

➤ *Aliases*

For example, the operator knows that 'Den Bosch' is the same as 's-Hertogenbosch'.

➤ *Clarifying confusing words or clarifying the client's own wording/terms*

The client can make mistakes in the provided information. For example the client says: "I want to go to Artis in Rotterdam". Then the operator has to clarify if the client means Artis in Amsterdam or Blijdorp in Rotterdam. In addition, the client can use foreign or dialect wording.

➤ *Events and theme parks, like the 'Efteling' and zoological gardens*

For example, see Figure 5-3.

```

2: reisinformatie goedemiddag
1: [uh] goedemiddag u spreekt met [voornaam+achternaam] uit Sint Maartensdijk [uhm]
   ik wil zaterdag wil ik met de trein van Roosendaal naar Landgraaf een enkeltje
   [uhm] en hoe duur is dat
[typewriter]
2: Roosendaal naar Landgraaf
1: ja
2: [typewriter]
2: Pinkpop #1 # neem ik aan [laughter]
1: #1 ja #
2: enkeltje achtendertig gulden vijftig
.....

```

1 = client
2 = operator

Dialogue 91535804

Figure 5-3 Example of the operator's knowledge about the event 'Pinkpop'

- **Use of the travel planner**

The operator knows how to use the travel planner efficiently so that she can provide the information to the client. The travel planner is described in section 5.2. Examples of the information stored in the computer are:

- Train/bus/tram/metro/boat/ connections
- Train stations and their facilities
- Prices/rates of trips, season-tickets etc.
- Train types used in trips
- Transfer times
- Temporary railway work
- Excess fares
- Platforms
- Events and theme parks, like the 'Efteling' and zoological gardens

- **Geographical knowledge**

To provide the client with the best information the operator has to know the cities geographically so she can visualise what travel route the client is taking. An experienced operator has in her memory a cognitive map of the railroad structure according to the geographical location of these cities. She uses this knowledge at different moments:

- Some clients do not know which station is the nearest to a certain city. To look up this information in the travel planner can be very cumbersome.
- With clients that do or do not want to transfer, want to know the shortest travel route in time or distance, geographical knowledge is necessary to provide the client with the best solution.
- If the city names are barely understandable, the most plausible solution can be verified with the help of geographical and logical considerations.

- **Dialogue management**

The operator knows how she has to participate in a dialogue. That not only means knowing how to manage a dialogue but also what is appropriate to say.

- **Knowledge gained by experience**

The operator gains knowledge by engaging in many dialogues and by experiencing things firsthand. For example, knowledge about travelling by plane, certain station characteristics. The more experienced an operator is, the more she knows and doesn't have to look up information.

5.1.2 Skills

Because OVR provides information the clients have to pay for, an operator should be able to deliver a high service level. To be able to do this the operator has to have several client-friendly behaviour skills. Most of these skills can be learned or improved through experience and/or training. Amongst others these skills are:

- *Client-focused*
The client is the focal point of the operator's attention.
- *Emotions*
Dialogues are interactions between people and emotions play an important role in this process. The operator should be able to control her emotions. This means that an operator never gets angry, never throws the telephone on the hook and never bursts out in tears.
- *Stress*
Even when it is very busy, the operator should be able to control her feelings so that her behaviour does not imply that she is rushed and that there is a long waiting queue.
- *Mood*
An operator should always sound cheerful and do not show any signs of boredom or irritation.
- *Prejudice*
An operator should not judge a client by his sex, background, stereotype etc.
- *Voice characteristics*
An operator should talk clear and understandable Dutch, in a normal pace and without a regional accent.
- *Fluency*
An operator should be able to formulate proper sentences clearly and concisely.
- *Listening*
It is important that an operator listens carefully, to understand quickly and correctly what the client wants to know.
- *Comprehension*
From the information provided by the client, the operator should be able to put things together and quickly comprehend the client's problem.

5.2 Travel planner

The travel planner is a software tool developed to easily look up travel information. This tool calculates several optimal travel routes with the help of information entered by the operator. The travel planner considers several public transport types and their respective timetables, while calculating the travel route. Some of these types are train, bus, subway, tram and boat. Therefore, with the help of the travel planner the operator is able to provide the client quickly with a travel advice.

Input window (Figure 5-4)

The operator has to enter the correct information systematically. When the travel planner does not recognise the entered information, the travel planner shows several alternatives from which the operator can make a choice. Therefore, when the operator is not sure of a certain name the travel planner gives assistance. The information that the travel planner needs to be able to calculate a travel route (and the operator has to enter), is:

- departure city
- departure address
- destination city
- destination address
- time - departure/destination
- date

Instead of a certain address, it is also possible to enter a generally well-known

Figure 5-4 Travel planner input window

point such as a hospital, a theme park or a zoological garden. This way it is not necessary to know the exact address.

Whenever the client prefers to travel via a certain travel route the operator can use the 'via'-option in the travel planner to indicate via which city the client wants to travel. Because normally the travel planner calculates the travel route with the shortest travel time and does not take into account a certain preferred travel route. This function is optional.

Output window (Figure 5-5)

When all the information is entered and the operator has given the command to start searching, the travel planner calculates several different travel routes. The number of travel routes depends on the number of possibilities. About six travel-routes are displayed concisely in the top of the window. The travel route, that corresponds to the entered departure or destination time the best, is indicated with a dark rectangle and displayed in detail at the bottom of the window. This detailed information shows the different stages of the trip and it includes the departure, transfer and destination places and times. In the case of travelling by train, it also includes the train type, the direction of the train and platform numbers. The detailed information of another travel route can be displayed by clicking on the preferred route with the mouse.

The travel planner also has the possibility to look up supplementary information. This information can be about train stations facilities, ticket rates, frequencies and connections to the requested route during the day, temporary railway work, topical traffic information and route information (Figure 5-6).

The operator can request the travel planner to calculate the same trip at an earlier or a later time without having to enter the information again. The same applies for the return trip.

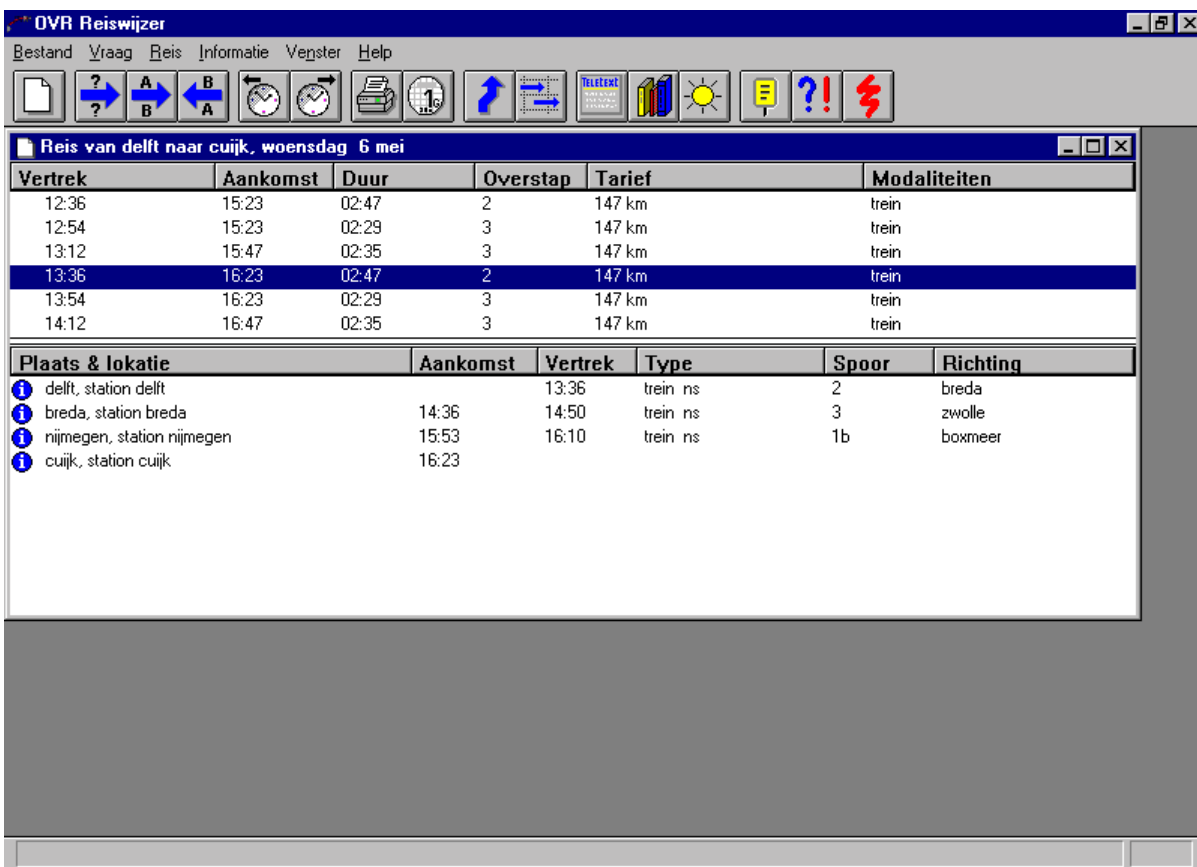


Figure 5-5 Travel planner output window

5.3 Knowledge-based structures

The operator has to solve a problem with every client's question. It depends on the given information if

Aankomst	Vertrek	Plaats	Opmerkingen
		delft	station delft (0:01)
		delft	station delft
08:13	08:06	delft	station delft
	08:13	schiedam	station schiedam centrum
08:18	08:20	rotterdam	station rotterdam cs
08:25	08:26	rotterdam	station rotterdam lombardijen
08:35	08:36	zwijndrecht	station zwijndrecht
08:40	08:42	dordrecht	station dordrecht
08:46	08:46	dordrecht	station dordrecht zuid
08:54	08:54	lage zwaluwe	station lage zwaluwe
09:02	09:02	prinsenbeek	station breda-prinsenbeek
09:07		breda	station breda
	09:22	breda	station breda
09:29	09:29	gilze	station gilze rijen
09:35	09:35	tilburg	station tilburg west
09:38	09:39	tilburg	station tilburg
09:55	09:56	s hertogenbosch	station s hertogenbosch
10:08	10:08	oss	station oss
10:23		nijmegen	station nijmegen
	10:34	nijmegen	station nijmegen
10:37	10:37	nijmegen	station nijmegen heyendaal
10:45		cuijk	station cuijk

the problem can be solved easily. When the client does not give much information, the operator will ask for information that is more specific or use her own knowledge to solve the problem. For example if the client gives as a departure place 'Delft, at the old church'. The operator will encourage the client to give more specific location information or if she is familiar with Delft she could know the address of the church herself. For this thesis, the focus lies mainly on dialogues where the client asks questions about travelling by train only. Therefore, a problem like in the aforementioned example will not occur that frequently in these dialogues because the client does not need to know the exact address but just the train station.

The operator has to react according to the client's question. It is dependent of this question what knowledge is used in the dialogue. As mentioned in section 5.2 several information elements or slots are needed to enter in the travel planner and to be able to solve the client's questions. For every type of client question there is a specific set of slots that the operator needs to fill. For questions about specific train travel routes, the needed information or set of slots is displayed in Figure 5-7.

The operator has to know what information to ask for to be able to solve the client's question. The structures in this section represent the knowledge or information that the operator uses or can use to answer a client's opening question. To develop these knowledge structures information is needed. The information is derived from the following sources:

- Interview with experts
- Information in flyers
- Information on the NS-internet site
- A book with general information about public transport

The information is rather distributed. All these information resources are combined into the knowledge structures. The structures show what slots are necessary to answer a certain client's question and their mutual dependencies. In theory the knowledge structure can be seen as a dialogue structure but practically it would not suffice as a dialogue structure because it is knowledge-driven and not dialogue-driven. The knowledge that is used depends on the information that is asked for by the client. In section 5.3.1, the client's question concerns train fares. The operator has to determine the exact and most advantageous price. In section 5.3.2, the client's question concerns a specific train travel route. The operator has to determine the fastest route from A to B. The structures will show that in

comparison with the question about a specific train travel route, the question about prices involves more slots and the dialogue is therefore more complex.

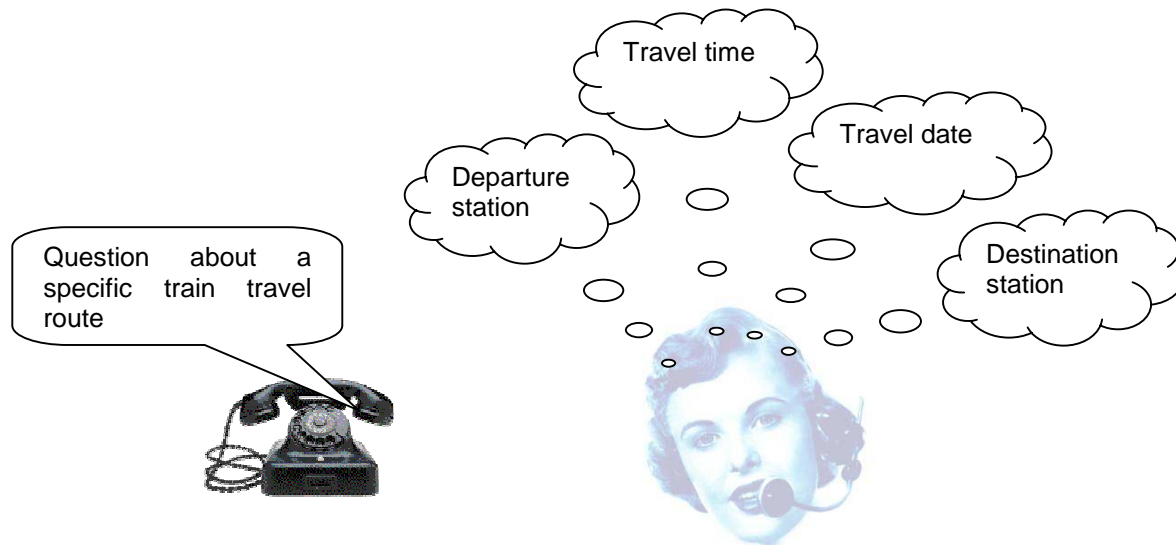


Figure 5-7 Slots the operator has to think about while solving the client's question about a specific travel route

5.3.1 Knowledge in a price dialogue

A client can ask a human operator for the price of a train ticket. The client can ask for the price in many ways. There are also many ways to respond to this request. To determine the price of a train ticket the client needs to provide information.

The price of a train ticket depends on several variables, such as

- The traveller's age
- Travelling distance
- One-way or round trip
- First or second class
- Number of persons travelling

Furthermore, the public train transport organisation (NS) has several reduction possibilities in combination with season tickets. In this case, the price is also dependent of the travel date and the travel time.

The purpose of this section is to design a knowledge structure for the price dialogue. In section 5.3.1.1, the price knowledge structure is described, in which all variables a train ticket depends on are included. To develop a knowledge structure for prices of the current situation a corpus-based approach is chosen. Therefore, in section 5.3.1.2 some examples of human-human dialogues are analysed and a general price dialogue structure of the human operator is created. Finally, these two are combined in a new price dialogue training model.

5.3.1.1 Price knowledge structure

In this section a knowledge structure is described that contains all the knowledge that is needed in a price question. The variables mentioned in the introduction, and the matching actions that are necessary to determine the price of a train ticket are represented in the structure. One of the variables, number of persons travelling, is not included in the structure, because the structure is designed for the price of a train ticket for only one person. This structure is used to compare the knowledge used by the operators to the overall knowledge that can be used to answer a price question.

The knowledge structure also shows the most efficient order of questions to determine a ticket price by using all the variables on which that price depends. For example, asking for the travel time before the travel date. When starting with the travel date instead of the travel time the chance to have to ask for the travel time also is bigger than the other way around. This is because there are time restrictions on working days and not on holidays and the number of working days is greater than the number of holidays (210 working days as opposed to approximately 160 holidays). In addition, if the travel time is in the afternoon the travel date is not necessary because there is always a reduction applicable on that part of the day.

When answering the client's question the operator needs unambiguous values of the variables. The client has to provide these values. If a client does not provide all the necessary variables the operator can ask for the values of the missing variables or can use default values. So the client can provide the unambiguous values or the operator chooses them. However, in the knowledge structure no default values are being used therefore the client must provide all values. The default values that are used by operators in the current situation for variables that are not given by the client are included in the knowledge structure. Only three variables depend on the client's input and therefore do not have default values. These variables are the departure and destination place and the travel time.

To understand some design decisions in the price knowledge structure certain regulations of NS-prices in combination with a season ticket have to be explained first.

A season ticket is a special ticket that gives a client opportunity to travel with certain discounts. These discounts are travelling with a 40% reduction, an evening round trip and a weekend round trip but they are limited to several regulations. Regulations for a client with a season ticket are:

- Travelling with reduction is only possible on working days after nine o'clock. There is no time restriction, during the weekends, in July, August and on some holidays.
- An evening round trip ticket has the same price as a one-way ticket with reduction and is only valid after six o'clock p.m. On Friday this reduction is not possible.
- A weekend round trip ticket has the same price as a normal round trip ticket with reduction and is valid from Friday seven o'clock p.m. up to and including Sunday.
- When a person with a season ticket is travelling together with one, two or three other people, these persons also get a reduction on their ticket price.

Also children younger than eleven that travel alone are always eligible for reduction and a weekend round trip is possible for clients without a season ticket, only without the reduction.

Now, the knowledge structure is discussed in more detail and displayed in Figure 5-8.

The knowledge structure in detail

The client provides information in the starting sentence. If the client gives the value of a variable, the next step in the structure can be carried out, otherwise the value of the variable has to be asked for. Based on these values decisions can be made and these are shown as decision-symbols in the structure. The structure is divided into ten stages and they are marked with a number. Below, each stage is explained.

- I. *Start.* The structure starts with a price question of the client. The questions are restricted to prices of train tickets, not other means of public transportation.
- II. *Traveller's age.* The first variable in the structure is the traveller's age. It starts with this one because if the traveller is younger than 12 and does not travel alone the price of a train ticket is fixed and independent of the travelling distance. After that no more variables are needed and the price information can be given. If the traveller is older than 11 or travels without escort, the following variables (III, IV, V and VII) are needed.
- III. *Departure place and destination place.* The price depends on the travelling distance therefore the departure and destination stations are necessary. If the client provides the place and not the station and if there are several stations in the place then the precise station has to be asked for.
- IV. *One-way or round trip.* The information about travelling one-way or taking a round trip is necessary to determine the price.
- V. *First or second class.* The information about travelling first or second class is necessary to determine the price.
- VI. *Traveller's age.* If traveller's age is less than 12 (is already known at this stage), the price of a train ticket is reduced. Therefore, it is not necessary to ask if the client has a season ticket to determine if he is eligible for reduction.
- VII. *Season ticket.* It is only possible to use an evening round trip ticket in combination with a season ticket. Therefore, when the client asks for the price of an evening round trip, it is obvious that he has a season ticket and it is not necessary to ask for this anymore. The question if the client wants a one-way or round trip ticket is therefore also superfluous. The question about an evening round trip is not explicitly mentioned in the structure because it is a part of the one-way/round trip stage in the structure. The same applies to a weekend round trip, except for the question about using a season ticket. This question still has to be asked because a weekend round trip can be used in combination with and without a season ticket.
- VIII. *With or without reduction.* When the client does not travel with a season ticket, the price information can be given. However, if the client has a season ticket, more information is needed. It depends on the travel time and date if the price is with or without reduction. The price of a train ticket is different per part of the day and for certain days. Therefore, more questions have to be asked about when (time and day) the client wants to travel. The questions and conditions/knowledge-based decisions differ when the client wants to know the price of a one-way or round trip ticket. Therefore, the structure is split in two at this point. At these stages, it is determined if a reduction is applicable. This is done by evaluating the starting and ending period of the trip because the reduction is restricted to certain times.
- IX. *Reduction when round trip.* This stage applies to the price of a round trip. When the client starts the trip somewhere in the afternoon (after nine a.m. and before six p.m.) all the necessary information is available and the price information can be given, because in this case the reduction always applies. If the departure time is after six p.m. and the travel date is a Saturday, the departure time of the return trip is needed to determine the price. Because if the client returns on Saturday, an evening round trip is applicable, otherwise a weekend round trip is applicable. The departure time of the return trip is also needed when the departure time is before nine a.m. and the travel date is not a holiday because if the return time is after nine a.m. a combined (with and without reduction) price can be determined. There are no questions to determine if the client wants to know the price of a weekend round trip because the price is the same as that of a normal round trip.
- X. *Reduction when one-way trip.* This stage applies to the price of a one-way trip. If the travel time is before nine a.m. the travel date is needed to determine if a reduction is applicable.

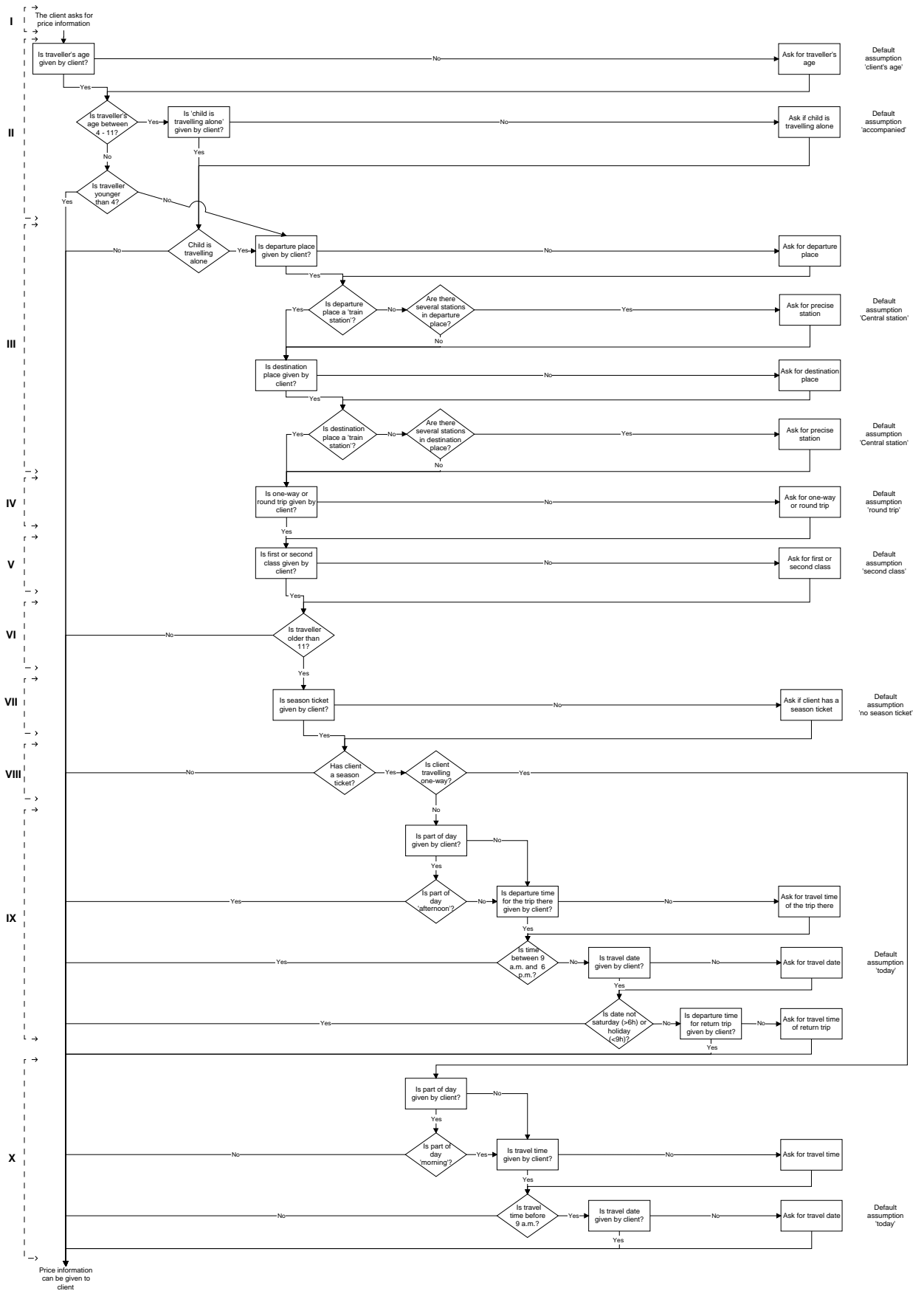


Figure 5-8 Price knowledge structure

5.3.1.2 Analysis of human-human price dialogues

This section describes how operators in the current situation deal with a question about the price of a train ticket. From the corpus of 2000 recorded train information dialogues, dialogues about prices are selected. Based on various examples a general underlying model is developed.

At what moment is the price question asked?

In the current situation the price question is asked at the following moments:

- *At the beginning of the dialogue*
The client starts by asking about the price of a particular trip. In Figure 5-9, the client starts with the price question.

```

2: goedemiddag reisinformatie
1: ja goedemiddag met [achternaam] uit Maastricht ik had graag [uh] geweten hoeveel een
   treinkaartje enkel Venlo kost
2: momentje vanaf welke plaats
1: van Maastricht
2: Maastricht naar Venlo enkele reis
2: zeventien vijftig
1: zeventien vijftig en [uh] enkeltje Utrecht hoeveel kost me dat
2: ook vanaf Maastricht weer
1: ook van Maastricht ja
2: achtendertig vijftig
1: ok bedankt
2: graag gedaan dag
1: dag
2: [noise] [tone]

```

1 = client
2 = operator

Dialogue 91565401

Figure 5-9 Example of a dialogue where the price question is asked at the beginning of the dialogue

- *During a travel advice dialogue*
The dialogue starts with travel advice questions. Therefore, when other questions have been asked about several travel topics the price question is asked. Figure 5-10 shows an example where the price question is asked at the end of the dialogue.

```

.....
2: aankomst Hengelo zes achteventig #2 # hier zou u dan over kunnen stappen in de
   trein naar Rotterdam die vertrekt om zeven uur zes
1: #2 ja # [talking]
2: aankomst Amersfoort acht uur vierentwintig
1: (())
2: hier overstappen in de trein naar Hoofddorp die vertrekt om acht uur zevenentwintig
1: zevenentwintig Hoofddorp #3 ja heeft # u de sporen er ook bij #4 en [u:h] # in
   Hengelo is dat geen probleem maar in Amersfoort
2: #3 u komt # #4 dan druk ik even op een ander knopje #
2: in Amersfoort is het aan de overkant en dat is perron vijf
1: vijf <in> <Hoofddo-> [u:h] ja
2: en u komt dan aan in Amsterdam om acht zeven vijftig
1: Amsterdam acht zevenvijftig
2: hier overstappen in de trein naar Den Helder die vertrekt om negen tweeëntwintig
1: #5 negen tweeëntwintig #
2: #5 van het perron # tien A
1: tien A
2: en u bent dan om tien uur een in Alkmaar Noord
1: om tien uur in Alkmaar Noord goed en wat kost dan een retourtje tweede klas
2: dat kost u vijftig gulden dagkaart
1: gulden prima dan weet ik voldoende
2: goed
1: bedankt #6 da:g #
2: #6 graag gedaan # dag
1: [noise] [tone] [tone]

```

1 = client
2 = operator

Dialogue 71715804

Figure 5-10 Example of a dialogue where the price question is asked at the end of the dialogue

What information does the client provide?

The client can provide all the necessary information in the question so that the operator is able to answer the question immediately. It is also possible that a part of the information is provided and some information is not provided. In that case, the operator has to ask for the missing information. The client can provide several of the following variables on which the price depends:

- departure place or station
- destination place or station
- information on the price of an one-way or round trip ticket
- information on the price of a first or second class ticket
- information on the price in combination with a season ticket
- the number of persons travelling
- the age of the traveller

In Figure 5-11, several examples of client questions concerning prices are shown. The examples are taken from the corpus. In all the questions, the client provides different information.

```

1: goedemiddag ik wil u wat vragen ik wil een trein pakken van Heerlen naar Valkenburg
hoeveel kost me dat?

1: ja goedemiddag met [achternaam] uit Maastricht ik had graag [uh] geweten hoeveel een
treinkaartje enkel Venlo kost.

1: goedemiddag [voornaam + achternaam] ik wilde graag wat prijzen weten van retourtjes

2: goedemiddag reisinformatie.
1: [uh] ja goedemiddag, ik heb een vraagje, hoeveel kost een enkeltje Barneveld Tilburg?
2: Barneveld Centrum?
1: [u:h] ja, Tilburg West.
2: Tilburg West.
2: enkele reis is zesentwintig vijftig tweede klas.
1: ja en [u:h] ik (( )) (( )) hoe laat gaan de treinen weg vanuit Barneveld?
.....

1 = client
2 = operator

Dialogue 11846903

```

Figure 5-13 Example where the operator asks for the exact stations

Figure 5-11 Several examples of a client's question concerning prices

How does the operator respond to the client's question?

The operator leaves the initiative mainly to the client. The operator will ask certain questions dependent on what information is provided by the client. When the price question is asked later in the

```

2: goedemiddag reisinformatie
1: ja goedemiddag [uh] ik wilde eens vragen de: trein morgen vanaf Vlissingen naar
Rotterdam
2: <hoe> <laat> hoe laat wilt u reizen
1: ja [uh] rond een uur of tien
2: momentje mevrouw
1: ja:
2: rechtstreekse verbinding een minuut voor tien
1: oh dat is mooi #1 # dat is de rechtstreekse verbinding
2: #1 ja #
2: ja
1: en kunt u mij ook zeggen wat dat dan kost die reis
2: een enkeltje dertig gulden vijftentwintig een retour eenenvijftig vijftig
1: dank u wel
2: alstublieft #2 dag #
1: #2 da:g #

1 = client
2 = operator

Dialogue 91535801

```

Figure 5-14 Example where the default station values are used and both prices are given

dialogue, the operator uses the information that was given earlier in the dialogue to answer the question. If the information isn't sufficient to answer the question immediately, the operator will ask for the missing information. Figure 5-12 shows an example where the operator can give an answer immediately because the client has provided all the necessary information in the question.

The corpus shows that, in order to determine the ticket price, the operator doesn't ask for all the variables on which the price depends. These variables are mentioned in the introduction. The operator asks for a minimum of information and uses default values for several variables. The corpus shows that these default values apply to the majority of the callers. The operator waits for the client to ask for the variables for which default values are assumed. When the client provides these variables, the operator uses them to determine the price of the train ticket.

The first part of the dialogue involves the query for departure and destination. When the client doesn't provide this information, the operator always asks for it. There are no default values for the departure and destination place. The departure and destination information the client provides can be stations or

places. In this last case, the operator sometimes asks for the exact station (if there are at least two stations in a place) but mostly they use the default value (central station). In the example in Figure 5-13, the operator asks for the exact stations and in Figure 5-14, the default values are used.

Most of the time, when the client doesn't mention if he wants to know the price for a one-way or a round trip, the operator asks for this information. Sometimes the default value 'round trip' is used (Figure 5-15) or both prices are given (Figure 5-14).

The corpus shows that the operator usually uses the default value ('second class') when the client does not mention if he wants to travel first or second class. They assume that the client wants to travel 'second class' instead of asking for it. Figure 5-16 shows this default assumption of the operator.

The same applies to the variable 'season ticket'. When the client does not mention if he wants to know the price of a train ticket with reduction, the operator uses the default value 'no reduction'. In the example in Figure 5-17, the client wants to know the price with reduction. Here the operator takes the price with a season ticket into account, but in the other examples (Figure 5-13, Figure 5-14, Figure 5-15, Figure 5-16) she does not ask the client if he wants to know the reduced price. The operator does not ask for the two last variables 'traveller's age' and 'number of travelling persons' either. Therefore, when the client does not mention them, the operator gives the price for one adult person. In Figure 5-15 the price of a train ticket for a minor is asked.

```

2: goedemiddag reisinformatie
1: goedemiddag ik wil u wat vragen ik wil een trein pakken van Heerlen naar Valkenburg

2: goedemorgen reisinformatie
1: [uh] goedemorgen mevrouw ik had een vraagje kunt u (()) wat een retourtje kost [uh] (())
  Hilversum
2: [uh] Geldrop Hilversum
1: Helmond
2: Helmond Hilversum
1: ja
2: momentje
2: retour tweede klas is vijfenveertig gulden
1: tweede klas vijfenveertig gulden
2: jawel
1: [uh] even kijken ja dan weet ik voldoende
2: ja
1: ok
2: tot uw dienst #1 dag meneer #
1: #1 tot ziens dag #
2: [noise] [tone]

```

1 = client
2 = operator

Dialogue 91563708

Figure 5-16 Example where default value 'second class' is used

Summarising all these examples, a knowledge structure of the human-human dialogues is developed. Figure 5-18 shows how the operator responds to a client's price question. Of course not all possible dialogues suit this model but it is generally applicable. It covers about 70% of the dialogues about prices. It is driven by the utterance of the client or client-centred. The operator listens if the client provides the value of a variable and uses it or asks for it or uses the default value. The operator uses many default values.

```

2: goedemiddag reisinformatie
1: goedemiddag u spreekt met mevrouw [achternaam] uit Breda ik wilde graag weten wat een
  enkele reis Breda Weert met een jongerenkaart kost
2: Weert
2: enkele reis tweede klas met jongerenkaart
1: #1 ja #
2: #1 twaalf # gulden vijfenzeventig
1: twaalf vijfenzeventig
2: ja
1: fijn dank u wel hoor
2: graag gedaan
1: goedemiddag
2: da:g [noise] [tone]

```

1 = client
2 = operator

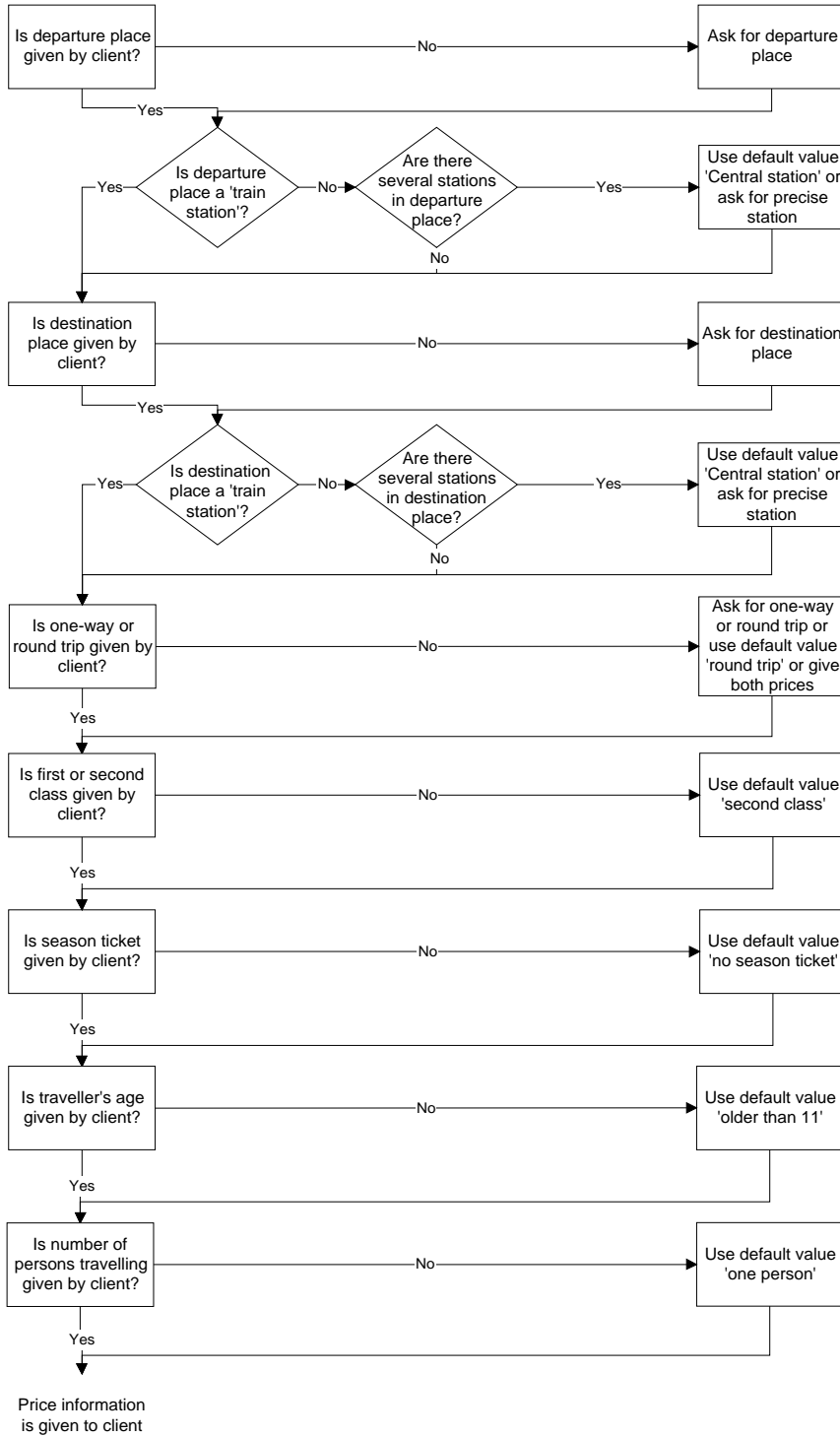
Dialogue 81516902

Figure 5-17 Example of asking a reduced price

Figure 5-18 Model of the operator's response to a price question

5.3.1.3 F

lowchart for the price dialogue



training model

The last two models are used as starting-points for the training model that is described in this section. These models can be improved. The total knowledge structure contains too much information to take

into consideration in a dialogue and it is questionable if there are enough questions asked in the information model currently in use by the operator.

The human operator dialogue only consists of necessary questions which benefits the efficiency but if a customer oriented approach is being chosen, it is better to elaborate the dialogue with additional questions and therefore supply more exact answers. This will result in a more satisfied client. Therefore in the new training model, displayed in Figure 5-19, the question about season tickets is asked but not if a client does explicitly state that he is a non-frequent traveller.

The questions about the travel time and date that follow the question about the 'season ticket' in the knowledge structure of Figure 5-8 are not included in the new model, because these tend to make a dialogue unnecessary long. This problem can be dealt with by warning the client that the price with a discount is not valid before nine o' clock on Monday to Friday. The assumption is made that if a client

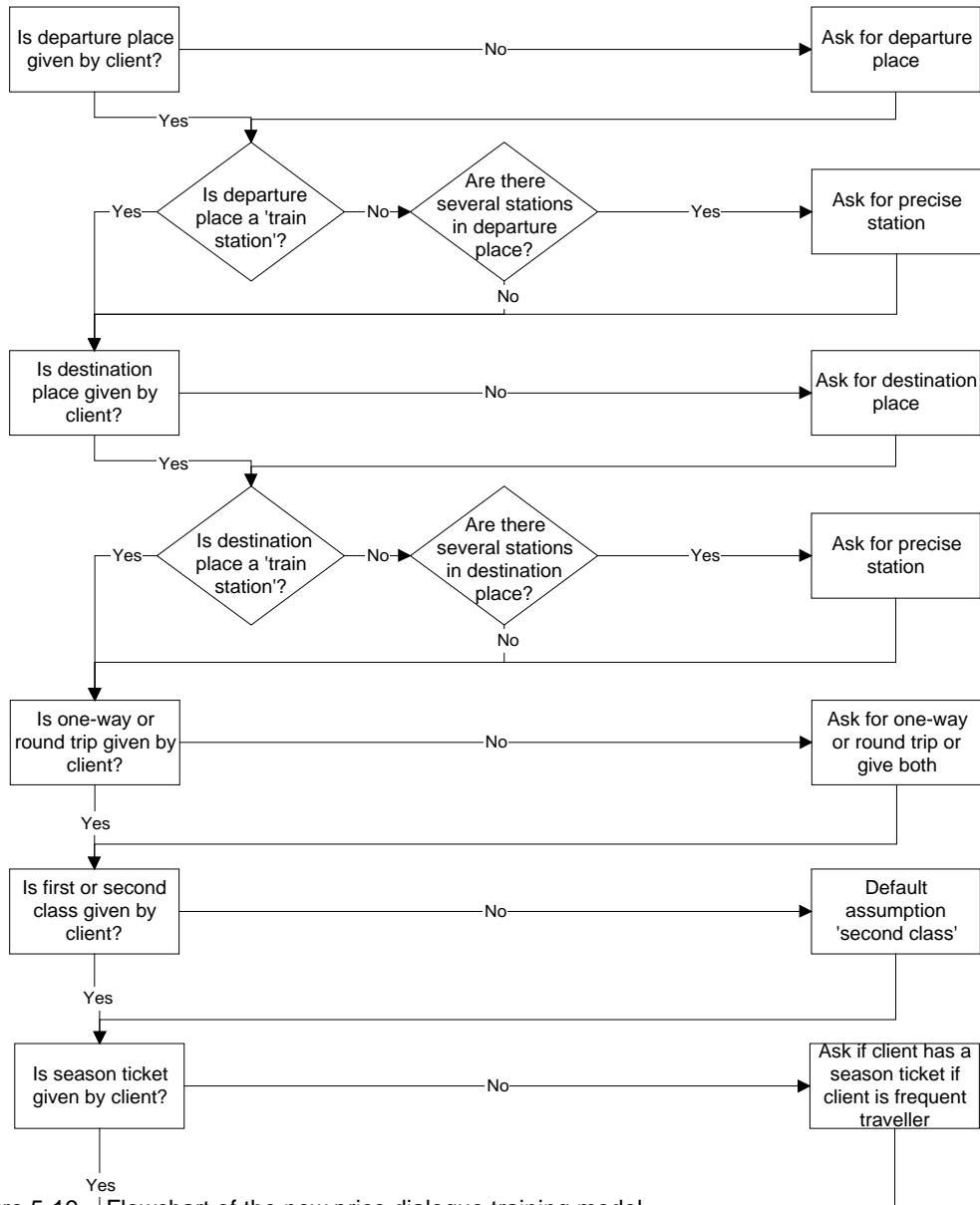


Figure 5-19 Flowchart of the new price dialogue training model

Price information can be given to client

wants to know the exact price the client will explicitly ask for it.

There are two possibilities to deal with the variable 'one-way or round trip'. The operator could ask the client if he wants to know the price of a one-way or round trip ticket or give both prices as an answer. Providing both prices creates no confusion with the client and this way the operator does not have to ask an extra question. In the current situation the operator sometimes uses one of the two as a default value but this does not improve the efficiency because then the client can explicitly ask for the other one.

The majority of the travellers travels second class therefore the operator can use the default value 'second class' and does not have to ask which class the client wants to travel. The client will ask for the price of the first class when he wants to know it. However, it is better if the operator mentions that a default value is used by providing it in the answer. In Figure 5-20, the operator provides 'second class' with the answer.

```

2: goedenavond reisinformatie
1: ja goedenavond u spreekt met [voornaam+achternaam] ik wilde graag de prijs weten van een
   dagretour met een jongerenkaart
2: ja
1: van Maastricht naar Wijchen
2: naar Wijchen
1: ja
2: retour met jongerenkaart tweede klas is drieëndertig vijfentwintig
1: ok dank u wel
2: graag gedaan #1 dag #
1: #1 dag #
2: [noise] [tone]

```

1 = client
2 = operator

Dialogue 91568206

Figure 5-20 A dialogue example where 'second class' is provided with the answer

The traveller's age, which is included in the knowledge structure of Figure 5-8, is not included in this model, because starting a dialogue by asking for the age is not polite and creates confusion. Starting a dialogue by asking for the traveller's age has as advantage that a number of questions does not have to be asked. To ask for the age later in the dialogue is not useful, because then most of the questions are dealt with. Moreover, most clients will probably mention themselves the age of a child when they want to know the train ticket price for a child. The variable 'number of travelling persons' is not included either, because most people want to know the price of a train-ticket for *one* person. In addition, most of the time if there are more persons travelling the client provides this information himself.

Asking for the exact departure and destination stations and not using the default value 'central station' is also better because by asking two simple questions a more exact price can be provided to the client.

5.3.2 Knowledge in a dialogue about specific train travel routes

In this section the knowledge and the matching actions that are needed to answer a question about a specific train travel route are described and combined in a structure. This structure will show in what way and with what elements the question can be solved. In section 5.3.1 three structures were described, in this section only one structure is described because during the analysis it became apparent that the three structures did not have many differences. One of the reasons is probably that a specific train travel route depends on fewer variables than the price of a train ticket. Therefore, for this problem, only a few slots are necessary and for most of these slots, a default value is not possible. The difference between the total knowledge structure and the human operator's structure is the possible use of three default values by the operator. These default values are included in the structure. Because the first two structures are almost the same, this structure is also a reasonably optimal training model for this problem. A possible improvement could be that the operator asks for the slots and does not assume the default values.

Explanation structure (Figure 5-23)

The slots needed in this dialogue type are already mentioned in the beginning of section 5.3. These slots and the matching actions that are necessary to determine the travel route are represented in the structure. The client provides information in the starting sentence. If the client gives the value of a variable, the next step in the structure can be carried out, otherwise the value of the variable has to be asked for. Based on these values, decisions can be made and these are shown as decision-symbols in the structure.

- I. **Start.** The structure starts with a question of the client. The questions are restricted to specific train travel routes. In Figure 5-21, several examples are shown.

```

1: ja goedmorgen met [voornaam+achternaam] kunt u mij vertellen hoe laat de treinen
naar Amsterdam gaan vanuit Hoorn

1: ja goedmorgen u spreekt met [voornaam] uit Haaksbergen aanstaande vrijdag de
negende [uh] wil ik graag de verbinding weten van Leiden naar Roosendaal met een
aankomst of tenminste [uh] ja een aankomst in Roosendaal in ieder geval voor acht
uur drieënvijftig want dan vertrekt mijn volgende trein

1: ja goedmorgen met mevrouw [achternaam] uit Vollenhove ik wou graag even: weten
wanneer dat hoe laat de trein vanaf Steenwijk naar Leeuwarden ging

1: goedemiddag met [voornaam+achternaam] mevrouw [uh] als ik <om> vanuit Zwolle naar
Eindhoven wil en ik moet om kwart voor tien in Eindhoven zijn

```

Figure 5-21 Examples of questions about a specific travel route

- II. **Departure place and destination place.** This is the same as part III in the price knowledge structure, displayed in Figure 5-8. Because the client wants have information about a certain route, he has to provide the operator with a departure and a destination place. If the client provides the place and not the station and if there are several stations in the place then the precise station has to be asked for. The default assumption of the operator when the precise station is not provided by the client, is 'Central station'.
- III. **Travel time.** The client has to provide the departure or arrival time. If the time is ambiguous because the client did not provide the part of day with the time, the operator has to ask the client for the part of the day he wants to travel (Figure 5-22). If the client wants to know the first, the last or the next possible travel route or a daily schedule than this is also regarded as a travel time.

```
2: reisinformatie goedemiddag
1: goedemiddag met [bedrijfsnaam] Eindhoven ik wil graag weten [uh] van Venlo naar
   Delft en in Delft moet ik dan om half negen uiterlijk zijn
2: #1 's ochtends #
1: ja 's morgens dus hoe laat moet ik dan uit Venlo vertrekken
.....
```

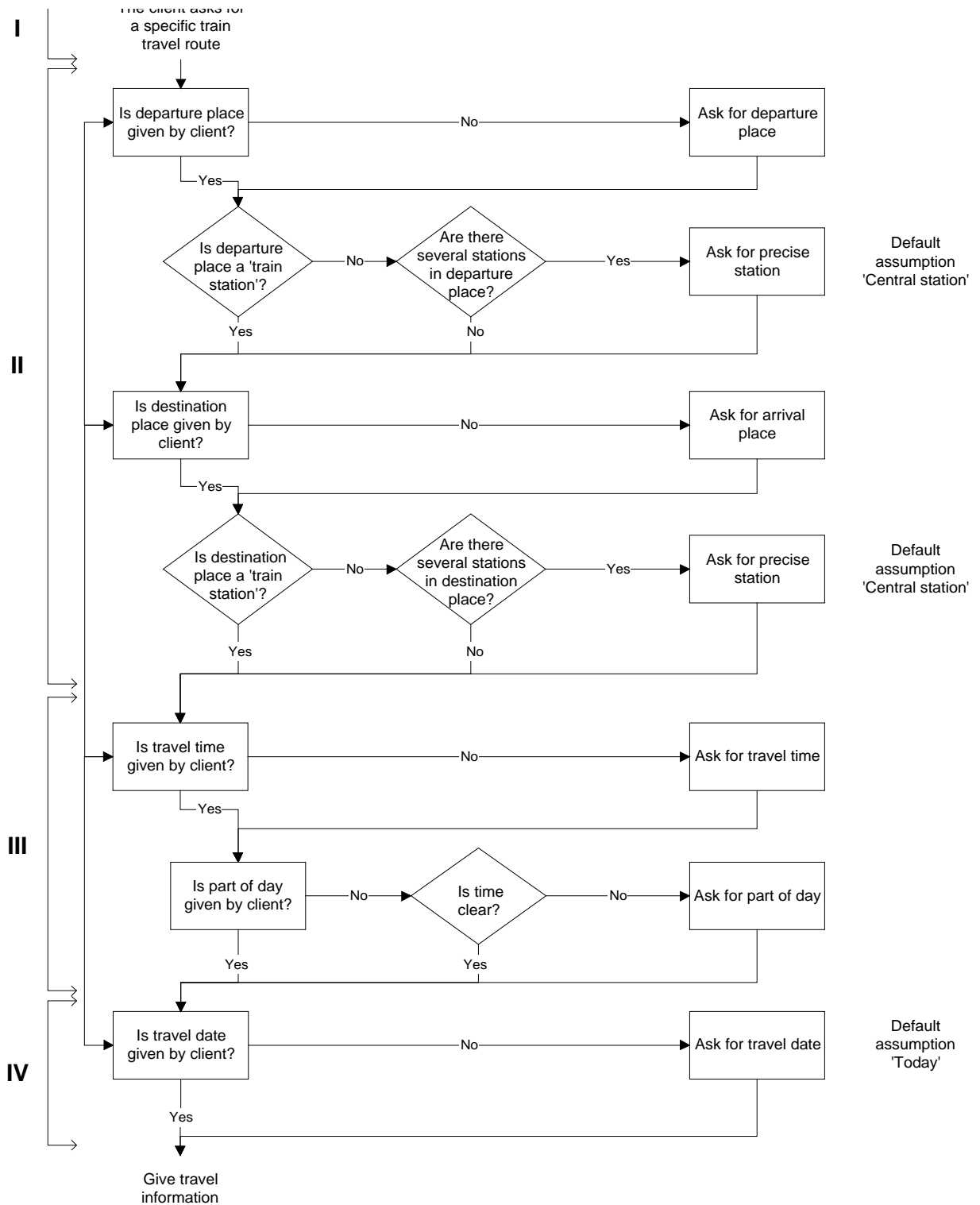
1 = client
2 = operator

Dialogue 81495104

Figure 5-22 Example dialogue where operator asks for part of day

- IV. *Travel date.* The last slot the operator needs is the travel date. In Figure 5-24, the operator asks for the travel date. Most of the time the operator uses the default value 'today', as shown in Figure 5-25. The travel time and travel date do not have to be asked in this order. They can be reversed.

Figure 5-23 Knowledge structure of question about a specific train travel route



```

2: [tone] goedemorgen reisinformatie
1: met [achternaam] Enschede goedemorgen [uh] van Enschede naar Lelylaan Amsterdam en
   dan moet ik er om een uur zijn kunt u mij vertellen hoe ik moet reizen
2: vandaag
1: vandaag ja
2: tien uur achtentwintig uit Enschede vertrekken rechtstreeks twaalf uur vijftig
   Amsterdam Lelylaan
2: da:g
.....

```

1 = client
2 = operator

Dialogue 61573703

Figure 5-24 Example where the operator asks for the travel date

```

2: goedemiddag reisinformatie
1: goedemiddag hoe laat gaat er[uh] rond vijf uur een trein van Heerlen richting
   Tilburg
2: een minuut voor
1: een minuut voor vijf
2: jawel
1: en: is dat die[uh] nieuwe trein
2: dat is die intercity wat rechtstreeks doorgaat naar Tilburg
1: oké dank je wel
2: graag gedaan
1: dag:
2: dag:
1: [noise] [tone] [tone]

```

1 = client
2 = operator

Dialogue 91535906

Figure 5-25 Example where the operator assumes that the client wants to travel 'today'

5.4 Information presentation

In this section, the presentation of the information/answer is described. As in section 5.3, the two question types are taken as a starting point. These two questions are used to illustrate which information is chosen and in what way the information is presented to the client by the operator.

5.4.1 Question about train ticket prices

During the presentation process two steps have to be carried out. The first step is deciding which price to choose to provide to the client and the second step is presenting the price information to the client.

5.4.1.1 How is the price of a train ticket determined?

The operator only needs the values of the departure and the destination station to enter in the travel planner to get a summary of all the prices regarding the entered route. An example price summary is shown in Figure 5-26. The operators have to choose a price from this list to present to the client. By deduction and asking additional information, as described in section 5.3.1, the exact and most advantageous price is chosen from the summary.

Van station delft in delft naar station cuijk in cuijk 147 km

Kaartsoort	2 ^e klas	2 ^e klas reductie	1 ^e klas	1 ^e klas reductie
Enkele reis	36,00	21,50	56,75	33,25
Retour	61,00	36,50	94,50	56,50
5-Retourkaart	289,00	173,00	448,00	268,00
Maandtrajectkaart**	654,00		1014,00	
Jaartrajectkaart**	5045,00		7765,00	
Jeugdmaandkaart**	556,00			
Avondretour[†]		21,50		33,25
Weekendretour	61,00	36,50	94,50	56,50
Fiets enkele reis	15,00			
Fiets retour	25,00			

[†]Alleen in combinatie met NS reductiekaart
^{**}Maximum tarief

5.4.1.2 In what way is the price information presented to the client?

There are many possibilities for the operator to present the price information to the client. In Figure 5-27, several examples of presenting the train-ticket price are shown. These examples are taken from the corpus.

Information elements that can be provided to the client by the operator, besides the price of the train ticket, are:

- First or second class
- One-way or round trip
- With or without reduction, if with then also which reduction
- Departure and destination stations
- Number of persons

When the client did not provide the values of these information elements and the operator did not ask for the values of these information elements, the operator assumes default values. When an operator assumes a default value, then this is not always clear to the client. In most cases, the operator includes the default values with the answer but sometimes she only provides the price. And in this last case the client can verify the answer. An example of this is shown in Figure 5-15. In most of the other examples, the operator verifies the variables before she gives the answer to the client or includes them with the answer.

```

2: twaalf gulden vijfenzeventig
2: even kijken mevrouw dat kost zevenenveertig vijftig
2: eenenzestig vijftig per persoon
2: vanuit Wolvega kost dat vijfenvijftig gulden vijftig
2: een enkeltje is achtendertig vijftig
2: enkele reis zesendertig gulden tweede klas
2: dan komt een dagretour op vijfenveertig gulden
2: een retour tweede klas is elf gulden vijfenzeventig mevrouw
2: een retourtje dertig gulden vijfenzeventig
2: een retour <is> u zit aan het maximale tarief van vijfenzestig gulden
2: een enkeltje dertig gulden vijfentwintig een retour eenenvijftig vijftig
2: retour kost negenendertig gulden vijftig van Apeldoorn naar Gouda dat kost u
  vijfenzestig gulden dagkaart
2: Venlo Eindhoven vierentwintig gulden vijftig
2: ja dat is enkele reis met reductie is vijf gulden vijfentwintig
2: retour met jongerenkaart tweede klas is drieëndertig vijfentwintig
2: een weekendretour kost drieëntwintig vijftig
2: dan is een avondretour [uh] tweede klas negen gulden vijfentwintig meneer
2: enkele reis met [uh] zestig plus seniorenkaart negenentwintig vijfentwintig

```

Figure 5-27 Several examples of price presentation

5.4.2 Question about a specific train travel route

During the presentation process two steps have to be carried out. The first step is deciding which travel route to choose to provide to the client and the second step is presenting the travel route to the client.

5.4.2.1 Which travel route is chosen by the operator?

After entering all the needed slots in the travel planner, the travel planner generates several travel routes from which the operator can choose to present to the client. As described in section 5.2, the travel planner indicates the travel route that corresponds to the entered departure or destination time the best. If there are more travel routes with the corresponding time than the travel planner will indicate the travel route that is the fastest. By default, the number of transfers, the transfer times and the train types are not taken into account by the travel planner. Presumably most of the time the operator follows the advice of the travel planner. Sometimes it would be better if the operator would choose the travel route she thinks is the best or provides other travel routes also. Because she could consider several other variables that improve the convenience of the client. Some of these variables are:

- The number of transfers. Sometimes it is better to have as little as possible transfers, even if the total travel time is longer.
- Train type. Some clients prefer to travel with a certain train type, if they would have the choice.

- Transfer time. If the client has to wait for half an hour in between transfers instead of travelling in a slower train, maybe the client would choose the last option.
- Luggage. When the client has luggage, it would be better to have a travel route with the least transfers and with a reasonable transfer time.
- Delay. The operator can anticipate on delays that can occur on a certain travel route.

5.4.2.2 In what way is the travel route presented to the client?

There are many possibilities for the operator to present the travel information to the client as shown in Figure 5-28. The corpus shows that the following information elements can be provided to the client while presenting the travel route are:

- Departure station
- Departure time
- Transfer station
- Transfer arrival time
- Transfer departure time
- Arrival station
- Arrival time
- Platform
- Transfer time
- Train type
- Total travel time
- Direct connection or not
- Train destination of the transfer train
- Hourly schedule
- Supplementary station information

```

2: de eerstvolgende is drieëntwintig eenentwintig

2: ik kijk even moment zeven uur acht uit Zaandam vertrekken

2: dat u daar echt om negen uur bent betekent vertrekken vanaf de Lelylaan acht uur vijf op
Den haag Hollands Spoor overstappen en daar vertrekken acht vierenvestig en dan bent u
in Delft acht tweeënvijftig

2: twintig uur een naar Den Haag Hollands Spoor #1 vanaf # spoor vijf uitstappen bij
Hollands Spoor om twintig negentien dan even op spoor zes de trein naar Schiphol pakken
om twintig tweeëntwintig

2: de intercity-treinen vertrekken uit Utrecht zeventien en zevenenvestig minuten over het
uur en rond tien uur zit er een extra trein tussen van tien uur twee dat is ook nog een
sneltrain

2: #1 wordt het # elf uur eenenvijftig vertrek bent u om twaalf uur zestien in Harderwijk

2: u kunt om zeven uur negenenvestig uit Utrecht weg met de intercity naar Den Haag

2: dat is om zeven nul zeven vanuit Arnhem spoor zes #3 richting # Den Helder

2: die doet er [uh] zeg maar een half uur over

2: rechtstreekse verbinding een minuut voor tien

```

Figure 5-28 Several examples of travel route presentation

Which information elements are given to the client by the operator depends on what information the client asked for. If the client asks for specific information such as the total travel time, this information is given. However, if the client does not ask for this specific information the operator is not providing it with the answer. Whenever a client does ask for some specific information, such as platforms, after the operator has given her answer, often the operator is providing it with the answer on a following

question. The operators do not have a general approach. In some cases only the essential elements are provided and then in other cases detailed information is given aside from the essential information. For the information elements 'transfer stations', 'transfer arrival and departure times' and 'platforms' the operator is more likely to provide these with the answer although the client only asked for the departure time. Nevertheless, in some cases the operator just provides the departure time. Most of the time the operator provides the following information elements, while presenting a travel plan: departure station, departure time, arrival station and arrival time. If the client has to transfer, often the transfer stations and its corresponding transfer times are given also. For every dialogue and every operator, the given information is very different. The same question can be and is answered in several ways.

The presentation of a travel plan involves more than just a monologue that presents the entire plan at once. The information presentation has a more interactive form. The information is presented in a stepwise way, generally giving at least one piece of new information with each turn alternated with an acknowledgement of the client. The presentation follows the temporal order of the different stages in the travel plan. If a client is asking a question during the presentation of the travel plan, the operator will try to answer this other question first and then continues with presenting the interrupted travel plan. If the answer to the client's question is affecting the interrupted travel plan then the travel plan is adapted and the operator resumes giving the adapted travel plan.

Sometimes it would be useful if the operator would take further initiative to provide the client with additional information that could be useful. From an efficiency point of view this is not preferable, but it is if client appreciation is considered.

CHAPTER SIX

Dialogue training environment

Regio: 1
Gespreksnummer: 11845602
Gespreksduur: 33.10 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: *****

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachttijd: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

en 2: goedenavond reisinformatie
#1 1: hallo met [voornaam+achtere
en Driebergen Zeist naar Utrecht
dat 2: vanaf nu
oké 1: ja
oké 2: ja
#2 2: twintig uur negene
dag 1: ja
[no: 2: en anders twintig
1: en: daarna
2: <twintig> <uur> <<nee>> eenentwintig uur negenentwintig pas weer
***** 1: ja oké bedankt #1 ja da:g #
0eili: 2: ja #1 tot uw dienst dag mevrouw
1: [noise] [tone] [tone]

en dan in Utrecht neem je een trein naar Eindhoven stap je in
Amersfoort
2: #1 waar vandaan
1: van Utrecht naar #
2: ja
1: vroeg ik me af[uh] hoe laat hij ging
2: die gaat om tien over heel en tien over half
1: tien over heel en tien over half #2 (()) #
2: #2 reistijd # is vijftwintig minuten
1: oh dat is (()) (()) tien over heel en tien over half
2: #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn #3 oké # en die stopt gewo
1: #4 oké # prima bedankt #5 da:g #
2: #5 graag gedaan # da:g
1: [noise] [tone] [tone]

Moeilijkheid van de transcriptie: *****
Omgevingsgeluiden spreker 1: 4
Omgevingsgeluiden spreker 2: 4
Geslacht spreker 1: 4
Geslacht spreker 2: 4
Kwaliteit van telefonische verbinding: 4
Vloeiendheid van gesprek: 4
Luidheid van de stem van spreker 1: vrouw
Helderheid van de stem van spreker 2: vrouw
Accent van spreker 1: 4
Luidheid van de stem van spreker 1: 4
Helderheid van de stem van spreker 1: 4
Accent van spreker 2: 4
Algemeen commentaar op gesprek: 4

Regio: Utrecht - Alphen aan den Rijn
Gespreksnummer: 176.28
Gespreksduur: 96 (s)
Categorie: reksnummer:
Wachttijd: 18-09-95 reksduur:
Transliteratiedatum: 17:14 - 17:27 reksduur:
Transliteratietijd: 17:14 - 17:27 reksduur:

Algemeen commentaar op gesprek: *****
Wachttijd: 53
Transliteratiedatum: 18-0- Helder
Transliteratietijd: 17:47 Algemeen commentaar op gesprek: *****

*****%*****

CHAPTER SIX

Dialogue training environment

This chapter focuses on the corpus-based design of a dialogue training environment (DTE) in which information seeking dialogues are simulated to optimise the efficiency and effectiveness of these dialogues while maintaining client appreciation. The knowledge in the corpus and therefore the knowledge described in the preceding chapters is used as the basis for the design of the DTE. The main purpose of the DTE is to train operators in effectively conversing with clients. Another important objective of the DTE is that the operators will become more familiar on how to manage dialogues. To do this, operators can train on complete dialogues and several aspects of the dialogues. At the same time, the trainer can monitor the training. In addition, the trainer can enter new data for exercises. This environment is developed as a stand-alone application and has no links with current existing systems. A prototype of a knowledge-based training tool that simulates dialogue elements and OVR-dialogues between the client and the operator is implemented. The development environment is a rapid application development tool, called *Delphi*. In the first section the functional design will be explained, the technical design is described in section 6.2 and the implementation of the prototype is described in the last section.

6.1 Functional Design

Analysis of the current training situation and analysis of the corpus has resulted in several requirements. The requirements the dialogue training environment (DTE) has to meet are:

- *Off-line, individual training in a realistic simulated surrounding.*
On the job monitoring of trainees provides little or no room for giving feedback. Furthermore, when a trainee has to have training on a certain aspect of the dialogue, clients who address this aspect are not available by the push of a button. In addition, the combination of work, training and concentration reduces the quality of the on-line work. The creation of a realistic individual off-line surrounding in which the computer simulates clients is preferable.
- *Corpus based*
The dialogues used by the simulator are based on real dialogues. This way the simulation resembles the real situation.
- *Prototype approach*
A prototype is the best approach to develop a system that will give the users an idea of a possible training system because no off-line training system exists at this moment.
- *Easy modification and/or expansion*
Because the DTE will be developed as a prototype and thus will not be complete, it has to be implemented in such a way that it can be easily modified and/or expanded. In addition, this way adding lessons is easier.
- *Educational*
The trainee learns and practices skills with the help of the training environment.
- *Scoring and feedback*
To see if a trainee improves her skills or to judge/review trainees a scoring and feedback mechanism is necessary.
- *External trainer*
Trainer is needed to monitor the training, to enter information and to judge/review the trainees in general.
- *Easy to use*
Because the trainees already have to learn managing the dialogues, it would only be difficult to also have to learn a difficult computer tool.

- *Training according to scripts*
According to Schank (see 2.2.1) people learn by using scripts. Because underlying scripts were found during the analysis of the OVR-dialogues these scripts are used in the training.
- *Identification*
To monitor and judge trainees individually it is necessary that the trainees can identify themselves.
- *Train on certain dialogue aspects*
It is useful to train the trainee on certain dialogue aspects that the trainee does not master enough, instead of the complete dialogue.

The functional design consists of the following three distinguishable components, shown in Figure 6-1:

- **Trainee**
The trainee is in this case the operator who uses the DTE to train on managing and conducting dialogues.
- **Trainer/supervisor**
The trainer/supervisor is someone who has knowledge of the training aspects of the dialogues. Preferably someone who has experience with managing and conducting dialogues.
- **Education**
This component represents the training model. This includes the training structure, the scoring and the feedback mechanism.

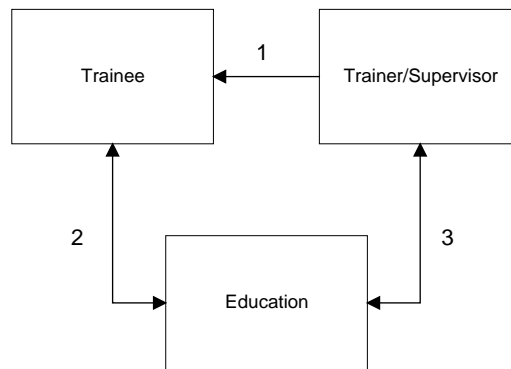


Figure 6-1 Overview of functional design of the DTE

The three different components interact with each other. The arrow numbers in Figure 6-1 are mentioned between brackets in the following sentences. Trainees can train on complete dialogues and several aspects of the dialogues [2]. They also can request statistical information on their training sessions [2]. At the same time, the trainer can monitor the training sessions [1]. In addition, the trainer can enter new data and request statistical information on trainees [3].

The tasks of these three components are described in detail in the next sections.

6.1.1 Education

In this section, the various educational aspects of the training model are described. These aspects are the educational factors, the training modules, the lesson structure, the training control, the exercises, the scoring and feedback.

Educational factors

The training model is designed while taking into account the following educational factors:

- *Corpus based approach*
The training material is not only based on realistic dialogues but is representative for the corpus of registered dialogues.
- *Dialogue model/script*
The dialogue model or script, described in chapter four, is used as a basis for the several exercises. Because according to Schank people remember things via scripts. When the structure

behind the dialogue is learned, the dialogue can be more efficient than when the structure is not followed. Therefore, the operator must have knowledge of the structure.

- *Slots*
The slots or information elements that are needed to provide the information to the client are used as a basis and point of reference in the exercises. These slots are necessary to retrieve travel information from the travel planner. In addition, the knowledge models, which are based on slots as described in chapter five, are used as training material.
- *Quality of the dialogues*
There are a few ways to improve the quality of the dialogues:
 1. Train the trainees in the content of a dialogue
 2. Train the trainees in the structure of a dialogue
 The focus of the training lies more on the transitions in a dialogue and less on the content of the dialogue, because training on the content (wording, the words used to compose an utterance) is too complex to develop judgement rules for. Not all possible expressions for the same content can be stored in a computer system and be used to judge a dialogue. For example, a simple acknowledgement can be expressed in numerous ways, such as 'yes', 'great', 'ok', 'that would be fine', 'sure, no problem', etceteras. This kind of training is more suitable for a person to person training environment.
- *Training principles*
The aim of the DTE is to enhance the trainee's knowledge and skills by training. The following different principles were used for the design of the DTE and combined into an eclectic approach:
 - *Learning by doing*
An important aspect of the DTE is the trainee's self-activity in the exercises. In most cases, the trainee has to contribute an active part to the dialogue. This has been a successful approach in skills training.
 - *Mastery learning*
The intention is that the trainee uses the exercises and their many variants until she has mastered them completely. The exercises generate feedback on the level of controlling the exercise.
 - *Self discovery*
Dialogues can be conducted in innumerable ways. By using the exercises, the trainee will learn which dialogue in which situation will be most efficient for her.
 - *Imitation*
The current OVR-style is a product of many years of experience. It is not necessary to rediscover all possible dialogue types but it is necessary to learn the dialogue types that match the already existing knowledge and skills and then to enhance them.

Training modules

The training is divided into several modules. Every module deals with a different subject. The idea behind the order of the modules is first to learn the different components a dialogue consists of and the specific skills that are needed during a dialogue and then training the total dialogue. The several modules or parts are:

Part One: Learn terminology

- Lesson one 'The script of the dialogue'
- Lesson two 'The categories'
- Lesson three 'Dialogue styles – Directive and non-directive'

Part Two: Learn the different phases of the script or dialogue model

- Lesson four 'Phase one: Starting the dialogue'
- Lesson five 'Phase two: Extracting information'
- Lesson six 'Phase three: Presenting the information'
- Lesson seven 'Phase four: Ending the dialogue'

Part Three: Training on several skills

- Lesson eight 'Specific skills'

Part Four: Training a total dialogue

- Lesson nine 'Total dialogue'

It is possible for the trainer/supervisor to create training programs that consist of the existing lessons. This way the training can be adapted to individual needs.

Lesson structure design

A lesson in the DTE is constructed out of three parts. The first part covers the theoretical part of a lesson. In this part the material of the lesson is explained. There are some examples included in this explanation. In the second part the trainee has to execute exercises based on the explained material. The third part of a lesson is the summary of the lesson. A small summary of the theory and the exercises is presented. This lesson overview can be seen in Figure 6-2 where the design of the lesson screen is displayed.

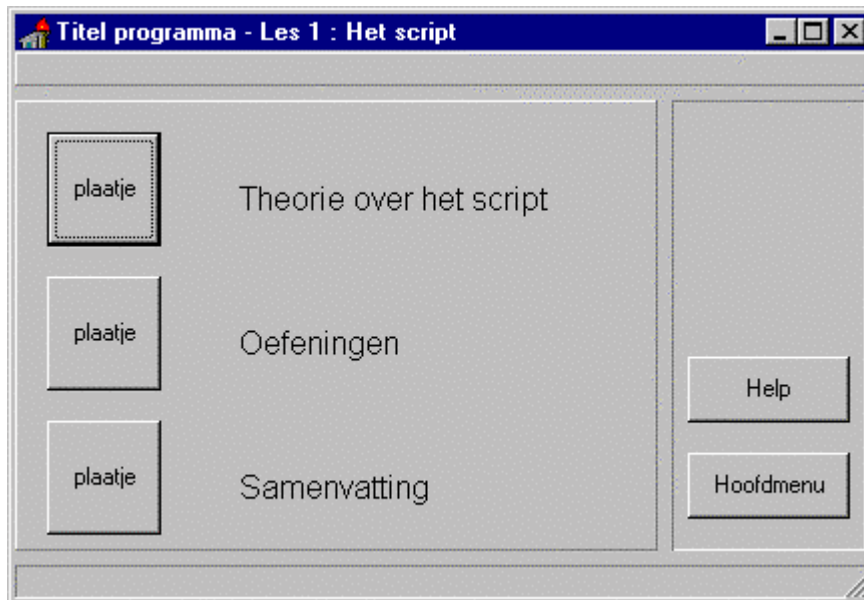


Figure 6-2 Lesson screen design

Training control

There are several levels of control possible for the trainee. This level of control is set by the trainer/supervisor. If no level of control is set by the trainer/supervisor, then default value will be no control. These will be described below.

- *No control*
The trainee is guided through the whole training from the start of DTE. She can not choose anything. A log is being kept on where the trainee was the last training session. This is done to determine where to continue the training the next time the trainee uses the DTE.
- *Partial control*
The trainee can only choose which lesson she wants to perform. If she has selected a lesson she will be guided through all parts of the lesson. She has to study the theory, perform all exercises and read the summary of that lesson in this order.
- *Total control*
The trainee is free to choose which lesson or what part of a lesson she wants to execute. She can choose her own order for the training.

Description exercises

All exercises per lesson are described below. These are the lessons that are mentioned in this section under the heading 'Training modules'. The design of the exercise screen is shown in Figure 6-3.

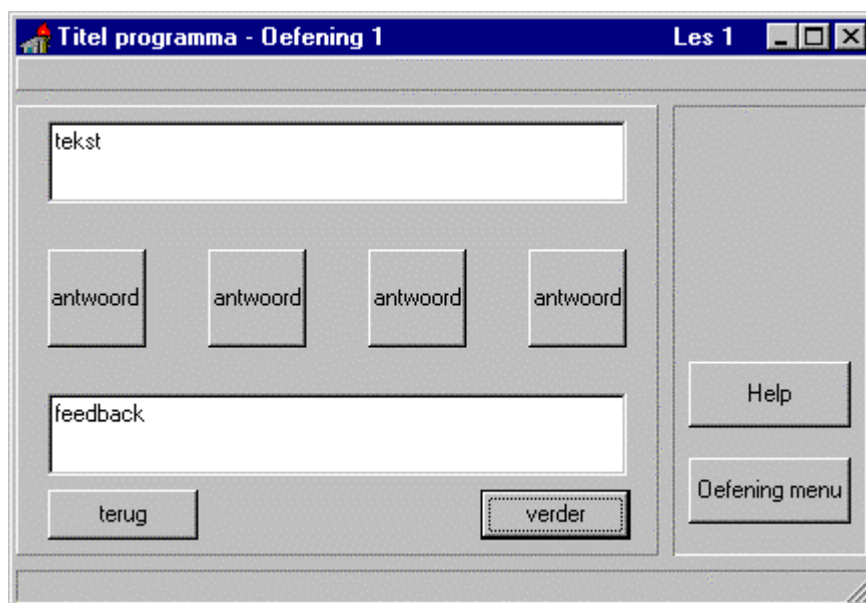


Figure 6-3 Exercise screen design

- **Lesson one 'The script of the dialogue'**

- Exercise one
 - *Title*
Identifying the phases
 - *Objectives*
Learn to combine phases with fragments
Learn the names of the phases
Learn to recognise a phase from multiple fragments
Learn to recognise/identify the phase in a fragment
Learn to recognise a phase
Learn the differences between the phases
 - *Description*
The trainee is given the four phases and four fragments of these phases and has to decide which fragment belongs to what phase.
- Exercise two
 - *Title*
Select the correct phase
 - *Objectives*
To learn to recognise a phase
To learn what the typical utterances of the phases are
To learn the names of the phases
Learn to combine phases with fragments
Learn to recognise/identify the phase in a fragment
Learn the differences between the phases
 - *Description*
The trainee not only has to learn the order of the phases but also to recognise the phases. In this exercise the trainee learns to recognise the phases by matching their names with their contents.
- Exercise three
 - *Title*
Order of the script
 - *Objective*

To learn the order of the phases of the script
 To learn to recognise the phases of a dialogue

◉ *Description*

In this exercise the trainee is given four fragments of the four script phases. The trainee has to place them in the correct order by giving them a number from one to four.

• **Lesson two ‘The categories’**

○ Exercise one

◉ *Title*

Matching fragments and categories

◉ *Objectives*

Learn to recognise the categories

Learn to match categories with fragments

Learn to recognise a category from multiple fragments

Learn to recognise/identify a category in a fragment

Learn the differences between the categories

Learn the names of the categories

◉ *Description*

The trainee is given the four categories and four fragments of these categories and has to decide which fragment belongs to which category.

○ Exercise two

◉ *Title*

Choosing the correct category

◉ *Objectives*

Learn the differences between the categories

Learn to recognise a category in a fragment

Learn the names of the categories

◉ *Description*

A fragment of one of the categories is given and the trainee has to decide to which category the fragment belongs.

• **Lesson three ‘Dialogue styles– Directive and non-directive’**

○ Exercise one

◉ *Title*

Matching fragments and dialogue styles

◉ *Objectives*

Learn the difference between the two dialogue styles

Learn to recognise the two styles

◉ *Description*

The trainee is given the two styles and two fragments of these styles and has to decide which fragment belongs to which style.

○ Exercise two

◉ *Title*

Directive or non-directive?

◉ *Objectives*

Learn the difference between the two dialogue styles

Learn to recognise the two styles

◉ *Description*

A fragment is given and the trainee has to decide if it is directive or non-directive.

• **Lesson four ‘Phase one: Starting the dialogue’**

○ Exercise one

◉ *Title*

Your own starting sentence

◉ *Objectives*

- To get comfortable with your own voice
 - To hear how your voice and intonation sounds to a client
 - To hear how one's own voice sounds
 - *Description*
The trainee can record his/her own starting sentence and listen to it
- Exercise two
 - *Title*
Starting the dialogue
 - *Objectives*
Learn to start a dialogue
See how a client reacts to different starting sentences
 - *Description*
The trainee has to start the dialogue by choosing one of the four options and can hear how the client reacts to it.
- **Lesson five 'Phase two: Extracting information'**
 - Exercise one
 - *Title*
Extracting information from the starting sentence
 - *Objective*
Learn to extract essential information from the starting sentence
Learn to listen more carefully to the client
Learn to react within *n* seconds
 - *Description*
In a starting sentence, a client can present different information in many ways. The trainee hears or sees a client starting sentence and has to extract the slot information from it. The essential information is information concerning the five slots, station/place of departure, station/place of destination, departure time, arrival time and date. The trainee has to learn to extract the information fast and efficient.
 - Exercise two
 - *Title*
Responding to the client's starting sentence
 - *Objectives*
Learn to react to the client
Learn to answer in one of the categories
Learn not to use unnecessary utterances
 - *Description*
The trainee is given a client starting sentence and has to respond to it by choosing an answer from one of the four categories (verification, clarification, new information and repetition). This is continued until the trainee chooses to present the travel scheme.
- **Lesson six 'Phase three: Presenting the information'**
 - Exercise one
 - *Title*
Present the travel scheme
 - *Objectives*
Learn to present the travel scheme in an efficient and polite way
Learn the different ways of presenting a travel scheme
 - *Description*
The trainee is given the primary information for the travel scheme and has to choose from a number of ways to present it.
 - *Exercise two*
 - *Title*
Responding to additional questions of the client
 - *Objectives*

- Learn to react to additional questions of the client
 - ⊙ *Description*
The trainee has to respond to additional questions of the client during or after the information is presented.
- Exercise three
 - ⊙ *Title*
Choosing the right information elements
 - ⊙ *Objectives*
Learn to select the necessary information elements
 - ⊙ *Description*
The trainee sees or hears a client question. She has to choose which information she wants to present to the client from a list of all possible information elements that could be presented to the client.
- **Lesson seven ‘Phase four: Ending the dialogue’**
 - Exercise one
 - ⊙ *Title*
Ending the dialogue politely
 - ⊙ *Objectives*
Learn to end the dialogue politely
Learn to end the dialogue efficiently
 - ⊙ *Description*
After exchanging the information and presenting the travel scheme a client can talk about unimportant subjects. This takes time and meanwhile the operator can’t help other clients. Then the operator has to end the dialogue, preferably in a polite way.
- **Lesson eight ‘Specific skills’**
 - Exercise one
 - ⊙ *Title*
Adapting to the client
 - ⊙ *Objective*
Learn to adapt to the client situation
Assisting the client with something he didn’t explicitly ask for
 - ⊙ *Description*
The trainee has to try to adapt to the client situation by asking for or giving additional information.
 - Exercise two
 - ⊙ *Title*
Characterising the dialogue
 - ⊙ *Objective*
Learn to hear essential elements of a dialogue
 - ⊙ *Description*
The trainee listens to some sentences of the client and has to hear the characteristics of the dialogue.
- **Lesson nine ‘The total dialogue’**
 - Exercise one
 - ⊙ *Title*
A normal conversation
 - ⊙ *Objectives*
Learn to converse efficiently with clients
Learn to talk through a dialogue with the help of categories and phases
Learn to manage a normal dialogue
 - ⊙ *Description*

A normal dialogue can be trained.

- Exercise two
 - *Title*
A difficult conversation
 - *Objectives*
Learn to converse with an angry, clumsy, or inexperienced client
Learn to assist the client as much as possible
 - *Description*
A dialogue with an angry, clumsy, or inexperienced client and therefore a difficult conversation can be trained.

- Exercise three
 - *Title*
Using the dialogue styles in a dialogue
 - *Objectives*
Learning to converse with the dialogue styles
Learn to choose the appropriate style during the dialogue
 - *Description*
A dialogue with the two styles (directive and non-directive) can be trained.

Scoring and Feedback

The most important objective is that trainees learn more about conversing with clients. Therefore, the comments they are getting are very important, otherwise their skills would not improve much. The comments consist of a score and feedback, because when giving grades and feedback the trainee learns more and the trained dialogues can be compared with each other based on the given scores. Only giving a score is not useful because the trainee does not learn from her mistakes and she feels like being in school again. She knows that it went good or bad (depending on the grade) but does not know what was bad or good. Therefore, also giving feedback on the mistakes or the things that went right is a better option.

Depending on the exercise type feedback is given during and/or after the exercise. Hints are also a useful form of feedback. The only time when these are useful is during the session. It can help the trainee when she does not know how to continue and can be a better choice than quitting with the session at that time or giving the correct answer for her. However, asking for a hint will influence the score negatively. The score is always given after the lesson.

Depending on the exercise type the feedback consists of a message about the result of the last performed action and/or a summary about the performed exercise. Based on the result of the exercise and the experience of the trainee a score will be given to the trainee for the specific exercise. If a trainee has executed the same exercise more than once than that will be reflected in the score of that exercise. The score consists of points gathered during the training. The scoring is different per the exercise type. When the lesson is finished the DTE checks in what category the score belongs and displays the matching text for that lesson and score. For example:

0 - 10 The dialogue was efficient. You made just a few errors according to the script.

11 - 20 The dialogue could have gone more efficient, too much verification.....

21 - 30 This time the dialogue wasn't efficient, it took too long. Ask questions that are more direct.

It is possible to only display the text and not the score. The score is especially important for the overall survey in which all the training sessions are compared.

The possibility for the trainer to apply scores to the different exercises makes the DTE more flexible and let's the trainer determine the weight and importance of the exercise.

6.1.2 Trainee

In this section, the trainee's tasks in the dialogue training environment (DTE) are described.

In the DTE the trainee can train dialogues. She can do this by starting the application and after she logs on, she can begin the training. The trainee first has to identify herself so the computer can keep track of the scores, the lessons and the training programs the trainee already has had. The available lessons can be executed independently from each other or a training program can be selected. If the trainer has pre-set a certain lesson or training program for the trainee she has to execute this lesson or training program. After the lesson or training session is finished, the trainee can view and print the results. At any point, the trainee can get help about the DTE or about a lesson.

The above-described functionality is outlined in the following single tasks:

- Start program
- Log in/ enter identification
- Select training program or lesson or execute lesson/training program pre-set by trainer
- Start training
- End training
- View results of the training session
- Print the results of the training session
- Get help
- End program

6.1.3 Trainer/Supervisor

In this section, the trainer's tasks in the dialogue training environment (DTE) are described.

The trainer starts the application and has to log on as a trainer to get access to the trainer component. Then he has the choice to maintain data or monitor a current training session. The trainer has the possibility to create, delete and modify user profiles for the system. For every trainee the trainer can view the results. And the trainer can create, delete and modify training programs. The trainer can enter data for the exercises. The trainer can set parameters that influence and shape an exercise. These parameters can be set for a certain trainee or in general for a certain exercise. If the parameters are not set, the system will use default values. The trainer can also enter feedback and scoring to the corresponding exercises, so the trainee gets the results that are needed to learn more. The trainer can remotely monitor trainees and intervene to give feedback during a training session. Also, the trainer can set a certain lesson or training program for a certain trainee. So when the trainee starts the DTE she has to execute the pre-set lesson or training program. At any point, the trainer can get help about the DTE.

The above-described functionality is outlined in the following single tasks:

- Start program
- Log in/enter identification
- Maintain trainee profiles
- Maintain training programs
- Generate results overview of trainees
- Maintain exercise data
 - Feedback per exercise
 - Scoring per exercise
 - Parameters per exercise
- Monitoring/supervising
 - Set training program or lesson
 - Give live feedback
- Get help
- End program

6.2 Technical design

In this section the technical design of the dialogue training environment (DTE) is described. The several modules the DTE consist of are explained. In the second section the communication between the modules is explained.

6.2.1 Overview modules

The DTE consists of several modules, operating independently from each other. The different tasks the program has to execute are distributed over the different modules. This way the different modules are easier to adjust or replace in the future. The only thing they have in common is shared data. In this section, the modules are described.

Database

The database is the storage for any data concerning the DTE. Data can be retrieved, changed, deleted and added from the database via queries.

Dialogue manager

The dialogue manager controls the DTE. All knowledge that is needed to run a training or actions related to the training is contained in the dialogue manager. The dialogue manager determines the next step, dependent of the user's actions. In addition, the dialogue manager takes care of all the communications between the modules. If the dialogue manager receives a message from a module, it knows where to send it. For instance if the dialogue manager receives a message from the trainer module to retrieve statistical data on a specific trainee, the dialogue manager sends a message to the Log trainee profiles, score and feedback module.

Trainee user interface

This module displays the objects that are necessary for the trainee to interact with the program. Examples of these objects are buttons, menus and list boxes. When the trainee starts a training session, the trainee user interface module communicates with the dialogue manager. It also sends an identification to the dialogue manager when the trainee uses an object on the user display, such as pushing a button or selecting an item from a list. It displays the data it receives from the dialogue manager. All the visual changes on the screen are handled by this module.

Trainer/supervisor user interface

This module displays the objects that are necessary for the trainer/supervisor to interact with the program. Examples of these objects are buttons, menus and list boxes. When the trainer/supervisor starts the program, the trainer/supervisor user interface module communicates with the dialogue manager. It also sends an identification to the dialogue manager when the trainer/supervisor uses an object on the trainer/supervisor display, such as pushing a button or selecting an item from a list. It displays the data it receives from the dialogue manager. All the visual changes on the screen are handled by this module.

Log trainee profiles, score and feedback

All knowledge to create statistical information on trainees and their training history, training programs, lessons and exercises is contained in this module. The knowledge to create overviews of this information is also contained in this module.

Training knowledge

This module contains knowledge where to get the data for lessons. It also provides the knowledge on the flow of a lesson and specific lesson knowledge. This knowledge includes theory, exercise, summary, training program, feedback and scoring information.

6.2.2 Module communication

The modules communicate by passing each other information through the dialogue manager. In Figure 6-4 the several modules and their interactions are displayed.

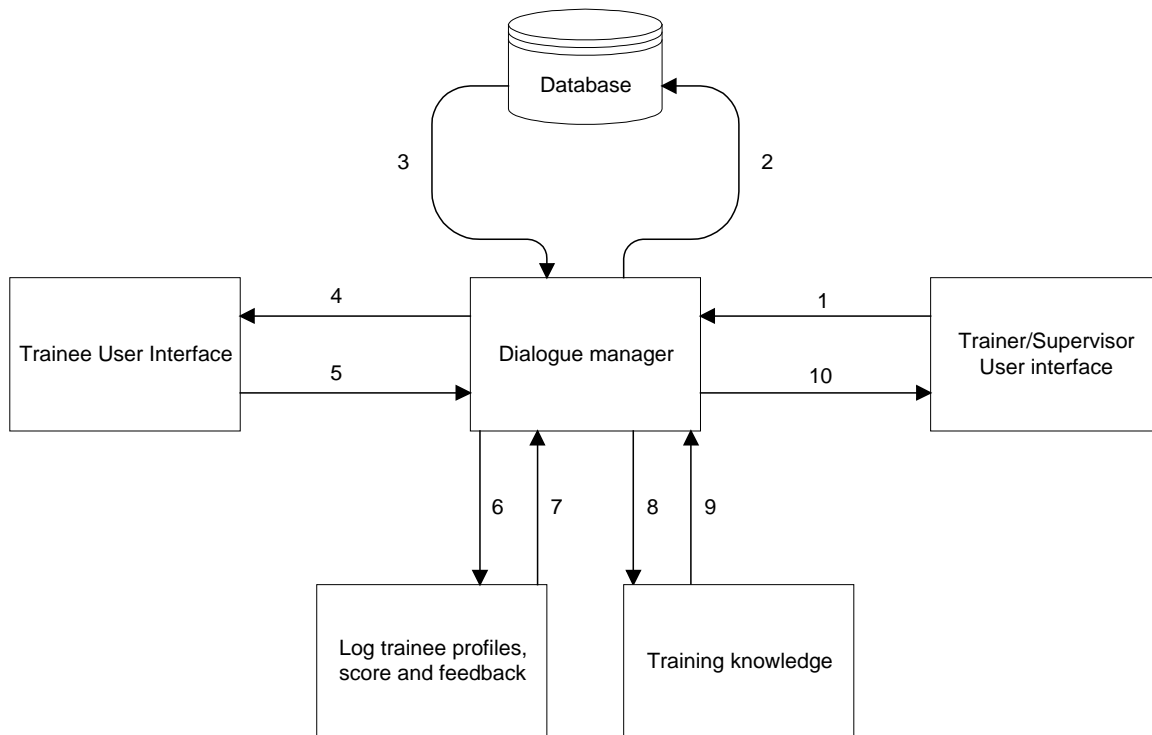


Figure 6-4 Overview technical design

Below, the communication between the modules is described. The numbers are corresponding with the numbers shown in Figure 6-4.

1. The data that is entered by the trainer/supervisor is sent to the *dialogue manager module*. The data can be trainee, training program, exercise and monitor/supervisor information.
2. The *dialogue manager module* handles any data from any module that wants to send queries to the *database*. These queries can retrieve, change, add and delete data from the *database*.
3. The results from the queries on the *database* are sent to the *dialogue manager module*.
4. The data that is necessary to be displayed to the trainee is sent to the *trainee user interface module*. This is any training data.
5. The data of the trainee is sent to the *dialogue manager module*. This data can be actions from the keyboard or mouse during the session.
6. Any data concerning the trainee profiles and training results are sent to the *log trainee profiles, score and feedback module*.
7. The results of a query from the trainer or the trainee regarding statistical information on trainees and their training history, training programs, lessons and exercises are sent to the *dialogue manager module*.
8. Requests to obtain knowledge about a training are sent to the *training knowledge module*. This knowledge includes theory, exercise, summary, training program, feedback and scoring information. In addition, data that is entered by the trainer to maintain the exercises is sent to the *training knowledge module*.
9. The results of the request for knowledge about a training (see 8) are sent to the *dialogue manager module*. In addition, the results of maintaining the exercise data are sent to the *dialogue manager module*.
10. The data that is necessary to be displayed to the trainer is sent to the *trainer/supervisor user interface module*. This is trainee, training program, exercise and monitor/supervisor information.

Any data that is received by the dialogue manager is processed and sent to the corresponding module.

The following example illustrates how the modules work and how they interact. The example is described and illustrated by using Figure 6-4. The arrow numbers in the figure are mentioned between brackets in the sentences. The example concerns the execution of an exercise without the involvement of the trainer/supervisor.

The trainee selects an exercise on her screen. The *trainee user interface module* sends a message to the *dialogue manager module* [5]. The *dialogue manager module* retrieves data about the exercise from the *trainee knowledge module* [8], [9]. The *dialogue manager module* retrieves data from the *database* based on the results from the previous request [2], [3]. The retrieved data is sent to the *trainee user interface module* by the *dialogue manager module* [4]. The *trainee user interface* displays the correct data for the selected exercise. This process will continue until the exercise is ended or aborted by the trainee. If this occurs, the *dialogue manager module* retrieves data from the *log trainee profiles, score and feedback module* [6], [7]. The *dialogue manager module* stores the result data in the *database* using the results from the previous request [2].

6.3 Implementation Prototype

In this section the implementation of the prototype of the DTE is described. First the development tool used to develop the prototype is described. Then the implementation details are explained.

Development tool

The development tool with which the prototype would be build should comply with a few criteria:

- the developed software should be fast
- the development tool should have a powerful and fast database
- the development tool should run on a MS-Windows platform
- the code should be reusable
- the development tool should be easy to learn
- it should be easy to develop a prototype

Because of these criteria, Borland Delphi has become the environment in which the prototype is developed. Delphi is a 'visual' programming tool based on object oriented Pascal. It is a RAD-tool. RAD stands for Rapid Application Development. Applications can be created very quickly in Delphi and depending on the program complexity, further programming is needed. The database is developed with the Borland database environment.

Implementation details

Parts of all modules except for the *trainer user interface* module and the *Log trainee profiles, score and feedback* module, described in section 6.2.1, are implemented. The database and some of the windows of this partial implementation are described.

Database

The database consists of three types of tables. The first table type is used in exercises where definitions are learned. These exercises make use of the multiple-choice technique. In Table 6-1 the fields and their types in the first table type are shown.

Table 6-1 Structure of first table type

<i>Fieldname</i>	<i>Type</i>
Nummer	Integer
Fragmenttekst	Alphanumeric
Antwoord	Alphanumeric
Feedback1	Alphanumeric
Feedback2	Alphanumeric
Feedback3	Alphanumeric
Feedback4	Alphanumeric
Gebruikt	Boolean
Wavfile	Alphanumeric

Nummer contains a unique number for every record in the database.

Fragmenttekst contains an utterance of the operator. This utterance has characteristics that have to be learned.

Antwoord contains the number that is the actual answer.

Feedback1 contains the feedback text that is displayed when answer 1 is chosen.

Feedback2 contains the feedback text that is displayed when answer 2 is chosen.

Feedback3 contains the feedback text that is displayed when answer 3 is chosen.

Feedback4 contains the feedback text that is displayed when answer 4 is chosen.

Gebruikt contains the value if the utterance in *fragmenttekst* is already used within the exercise.

Wavfile contains the reference to an audio file in which the utterance in *fragmenttekst* is recorded.

The second table type (Table 6-2) contains utterances from the client and the operator. There is no specific information in these utterances such as the departure or arrival time. This way the same utterance can be used repeatedly with each time different detailed information. This information will be filled in later by the dialogue manager module. The following is an example of a client utterance in the database:

I want to go to <arrival place> and want to leave at <departure time>. (1)

After parsing the above utterance might look like this (*arrival place* and *departure time* are filled in):

I want to go to *Delft* and want to leave at *eight o'clock*. (2)

Besides utterances, the tables also contain codes to determine the type of the utterance and the kind of slots that are in it. Every utterance has an accompanying code that contains information about the presence of a slot in it. By keeping track of the status of slots, the dialogue manager is able to create a dynamic query that retrieves the necessary data from the database. The status of a slot means if it is filled or not (true or false). So with these codes, the dialogue manager can decide each time which utterance to use. The sentence above (1) is a starting utterance of the client and the slots that are in it are 'arrival place' and 'departure time'. In the database this is indicated by setting the type to 'opening' and giving arrival place and departure time a 'true' status. This applies to the other types and slots also.

Table 6-2 Structure of second table type

<i>Fieldname</i>	<i>Type</i>
Nummer	Integer
Responses	Alphanumeric
Soort	Alphanumeric
Vertrekplaats	Boolean
Aankomstplaats	Boolean
Vertrektijd	Boolean
Aankomsttijd	Boolean
Reisdatum	Boolean
Gebruikt	Boolean
Wavfile	Alphanumeric

Nummer contains a unique number for every record in the table.

Responses contains utterances from the operator or the client without specific information on the slots, as described above.

Soort contains a description which type of dialogue act (as described in 3.2.2) the record is.

Vertrekplaats contains the value describing if the utterance in *Responses* contains the slot *vertrekplaats* or not.

Aankomstplaats contains the value describing if the utterance in *Responses* contains the slot *aankomstplaats* or not.

Vertrektijd contains the value describing if the utterance in *Responses* contains the slot *vertrektijd* or not.

Aankomsttijd contains the value describing if the utterance in *Responses* contains the slot *aankomsttijd* or not.

Reisdatum contains the value describing if the utterance in *Responses* contains the slot *reisdatum* or not.

Gebruikt contains the value if the utterance in *Responses* is already used within the exercise.

Wavfile contains the reference to an audio file in which the utterance in *responses* is recorded.

A third table type contains places in the Netherlands with their train stations. It is used whenever the initial values of the arrival and departure places have to be determined. In Table 6-3 the fields and their types in the third table type are shown.

Table 6-3 Structure of third table type

Fieldname	Type
Nummer	Integer
Plaats	Alphanumeric
Station	Alphanumeric
Meerstations	Boolean

Nummer contains a unique number for every record in the table.

Plaats contains the name of the place in the Netherlands where a train-station is located.

Station contains the name of the station of the place in *plaats*.

Meerstations contains the value if the place in *plaats* has more than one station.

Windows

To illustrate the implementation some examples of the windows are shown. In Figure 6-5 the main menu of the dialogue environment is shown. The parts displayed in the figure are the same parts as described earlier in section 6.1.1. Figure 6-6 shows the lesson menu where the trainee can choose between the theory, the exercises or the summary of the lesson.

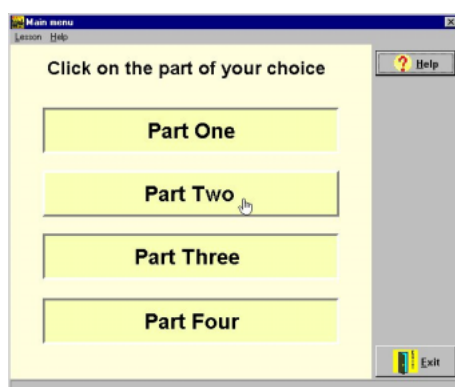


Figure 6-5 Main menu where the trainee can choose between the several training parts

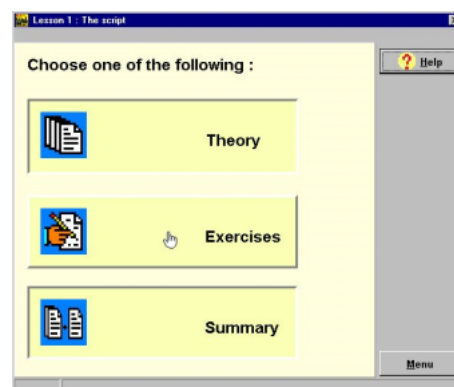


Figure 6-6 Lesson window

Lesson one – Exercise three

In Figure 6-7, the window of this exercise is shown. Every phase of the script is represented by a fragment. The fragments have to be placed in order of the script by moving the numbers to the corresponding fragments. The numbers are placed in a separate box at the bottom of the screen. These numbers can be moved to the empty places next to the dialogue fragments at the top of the screen.

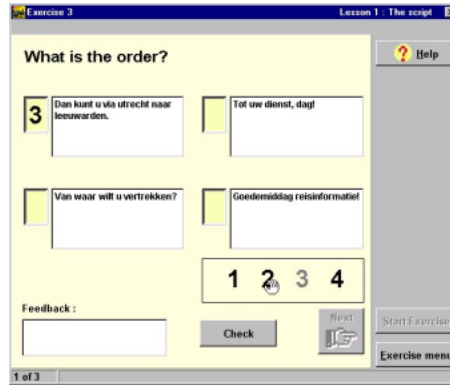


Figure 6-7 Window of exercise three of lesson one

Lesson two – Exercises one and two

In Figure 6-8 and Figure 6-9 the exercises of lesson two are shown. In the first exercise the trainee is given the four categories of the dialogue model and has to decide which fragment belongs to what category.

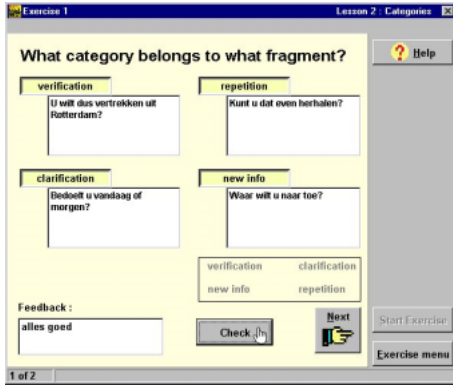


Figure 6-8 Window of exercise one of lesson two

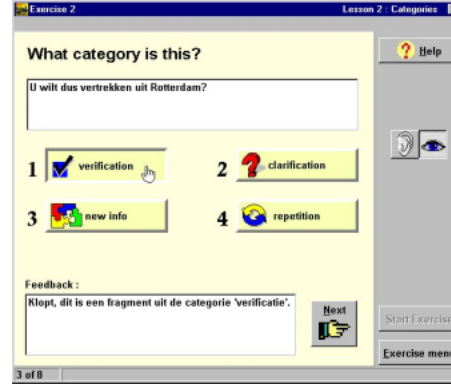


Figure 6-9 Window of exercise two of lesson two

In the second exercise a fragment of one of the categories is displayed or can be heard (indicated with a choice between an ear and an eye at the right side of the window) and the trainee has to decide to which category the fragment belongs.

Lesson five –Exercise one and exercise two

In Figure 6-10, the window of exercise one is shown. In this exercise, the trainee sees or hears (indicated with a choice between an ear and an eye at the right side of the window) a client's starting sentence and has to extract the slot information from it. The essential information is information concerning the five slots, station/place of departure, station/place of destination, departure time, arrival time and date. The trainee has to indicate what slots where present in the utterance by clicking on the correct buttons.

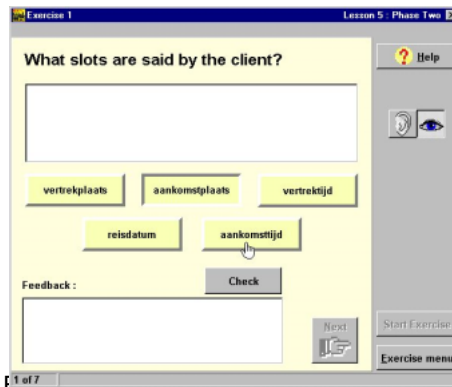


Figure 6-10 Window of exercise one of lesson five

In Figure 6-11, the window of exercise two of lesson five is shown. In this exercise, the second phase of the dialogue is simulated. The trainee has to respond to the starting sentence of the client by choosing one of the four categories. Then a list of possible utterances is displayed and the trainee has to choose which response she wants to use (Figure 6-12). This process continues until the operator chooses to present the travel scheme to the client. If not all-necessary slots are filled, a message is displayed that indicates that presenting a travel scheme is not possible at that moment due to insufficient data. If all-necessary slots are filled, the phase is ended and the next one can be trained.

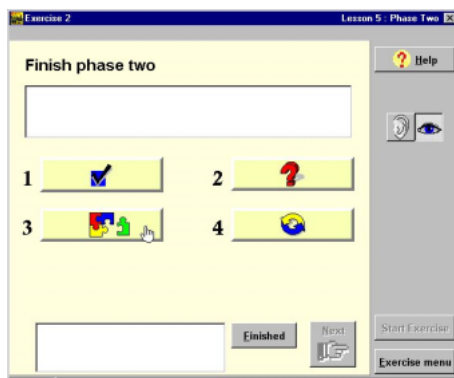


Figure 6-11 Window of exercise two of lesson five

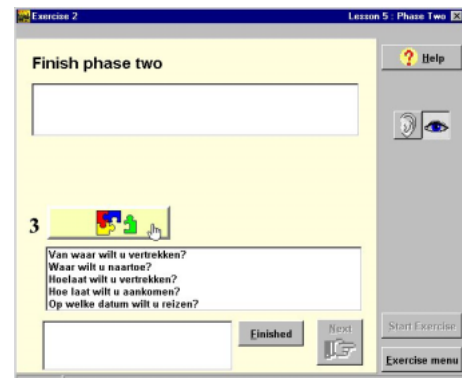


Figure 6-12 Window, where the choice is made to respond with a reaction from the 'new information' category

CHAPTER SEVEN

Conclusions and recommendations

In this chapter, conclusions will be drawn and recommendations for future research will be given.

7.1 Conclusions

In this thesis, research has been done to optimise the current O.V.R. dialogue management by analysing the O.V.R.-dialogues and developing a knowledge-based dialogue training environment prototype. Based on this research the following conclusions have been drawn.

- In the introduction, the most important objective of this thesis is stated. It comprised making explicit the possible underlying scripts of the OVR-dialogue to train the operators in using these scripts to optimise the current dialogue management. To gain knowledge about the possible underlying models a corpus-based approach is taken. A coding is developed with which the corpus is coded, an underlying dialogue model is extracted and validated, knowledge models are constructed and with the acquired knowledge a training environment prototype is designed and partly implemented. The acquired knowledge about the dialogue management has been useful in giving insight in optimising the dialogue management. Therefore, the approach turned out to be successful.
- A lot of insight in the dialogues is gained during the process of annotation, coding and analysing of the dialogues. A dialogue is more complicated than it looks at first sight. There are many ways to conduct a dialogue. While conducting a dialogue, an interlocutor has to think of what is said when and in what way. The content, the structure and the management have to be taken into account in a dialogue.
- Coding turned out to be complex, because a problem is that there is no universal accepted coding approach and/or coding tool. There have to be coding restrictions to exclude ambiguities and even then coding is complex, because several factors influence the coding and have to be taken into account. Especially the prosody, deciding which dialogue acts to use and on which coding level to code the dialogues and thus deciding what in the dialogue is important in reaching the coding goal. The research in this thesis is restricted to text recordings of the corpus dialogues and therefore prosody is excluded. The main reason is that it is still too complex to code these non-verbal features and in current automatic speech processing systems this is not taken into account because these systems are not able to handle these non-verbal features yet. The next step is to include them in ASP-systems. Despite these complexities, the coding is accomplished successfully. With the developed coding tool, 100% of the corpus dialogues is coded.
- It can be concluded that an OVR-dialogue has one underlying script or model, because one of the models is based on the coding extracted and validated from the corpus. The constructed dialogue model is an underlying script for 80% of the corpus dialogues.
- A lot of domain knowledge can be extracted from the dialogues of the corpus. This knowledge includes among other things which information is given and in what way, and what information is needed to provide the client with a fulfilling answer and in what order, depending on the type of client's question. The questions that are analysed and modelled concerned the price of a train-ticket and specific train-travel routes. Because a lot of information is involved in a dialogue, an operator has to have a lot of knowledge on a variety of subjects. Especially common sense knowledge is difficult to explicit and extract.

- The slots turned out to be an important part of the dialogue. Most of the management of the dialogue is based on these slots. The computational model that is constructed is also based on these slots.
- It turned out to be possible to develop a knowledge-based dialogue training environment with the acquired knowledge of the coding, the analysis, the model and the domain knowledge. With this training environment, the OVR-dialogues can be trained. A first prototype is implemented.

7.2 Recommendations

In this section several recommendations for future research and development are given.

On a long-term view, recommendations can be made about the new role of the operator. The research in this thesis was restricted to the OVR-domain. The research could be extended to more domains of information seeking dialogues, resulting in a possible generic coding and dialogue model for information seeking dialogues. Then the operator tasks could partly be handled by automatic speech processing systems and more modalities could be included. It would also mean that the generic models could be integrated with the internet. The new role of the operators would then be to extend their working domain with other domains and answering specific questions only. The standard questions could then be handled automatically. Training the operators on their new role would be very important then.

On a short-term, the following list of recommendations could be taken into account:

- The knowledge extracted in this thesis could also be used in improving automatic speech processing, because an improved insight in the dialogues and the dialogue management can increase the chances of recognising utterances and patterns in a dialogue and determining the appropriate reaction to them.
- Researching if the extracted dialogue model of chapter four is a general model for information seeking dialogues and therefore if other information seeking dialogues also have this model as an underlying script.
- Extending the analysis and the training environment to other OVR-dialogue types, not only the train dialogues.
- Extending the computational model through an advanced corpus based research with an evaluation function.
- Classifying the client's opening questions, so more insight can be gained in what information is asked by the client.
- Analysing the operator's interpretation of time while she decides what travel scheme she wants to provide to the client.
- Extending the implementation of the prototype with the rest of the design. That is the development of the *log trainee profiles, score and feedback* module and the *trainer/supervisor* module and the parts of the other modules that are not yet implemented.
- It is important to expand the development of the prototype so that it can be tested and therefore to get to know the influence of the dialogue training environment on the dialogue management of the operators. Thus to get to know if the dialogues will be more efficient and effective while maintaining client appreciation and so reducing the waiting queues and helping more people.
- Extending the dialogue training environment with the functionality to train different types of dialogues. Because in real life the reactions to clients are very different from each other, for

example the reaction to an angry client is different from the reaction to a kind person. An operator can find it more difficult to react to an angry client than to a friendly client.

- Adding the functionality to the dialogue training environment to train more on the content or wording in a dialogue. This way the operator can improve her expressions and therefore get a higher client appreciation.
- Developing an author-tool for the dialogue training environment. With this tool, trainers should be able to develop other lessons and exercises that can be easily incorporated into the dialogue training environment.
- Expanding the monitoring part with the possibility to set live parameters, to intervene during a training session, to simulate the client by choosing from displayed utterances or simulating the client with speech.
- Developing a grammar for the OVR-dialogues so the dialogue training environment is able to construct utterances automatically, by concatenating the words to form complete sentences. Because this way more different simulated dialogues are possible without storing them all in the database and the functionality to hear the utterances during a dialogue can be more easily developed. And after this, if possible, adding an automatic speech processing module to make the system even more flexible.

References

List of

figures

List of tables

 regio: 1
 gespreknummer: 11845602
 gespreksduur: 33.10 (s)
 categorie: trein_info
 wachttijd: 32 (s)
 transliteratiedatum: 18-09-95

 Regio: 1
 Gespreksnummer: 11848102
 Gespreksduur: 27.04 (s)
 Categorie: trein_info
 Wachttijd: 231 (s)
 Transliteratiedatum: 19-09-95
 Transliteratietijd: 11:54 - 11:57

 2: goedenavond reisinformatie
 #1 1: hallo met [voornaam+achternaam] ik wil graag morgen vanaf [tijd] naar Utrecht
 2: vanaf nu
 #1 1: ja
 #2 2: twintig uur negentwintig
 dag 1: ja
 [no: 2: en anders twintig uur vijfenveertig
 1: en: daarna
 2: <twintig> <uur> <<nee>> eenentwintig uur negentwintig pas weer
 **** 1: ja oké bedankt #1 ja da:g #
 moeilijk: 2: ja #1 tot uw dienst dag mevrouw #
 1: [noise] [tone] [tone]

 2: ja in[uh] de
 2: u kunt om zeven uur
 **** 1: ja
 2: aankomst Den Haag CS acht uur
 Moelijkheid: 2: ja
 Omgevingsgel: 1: ja
 Omgevingsgel: 2: dan stapt u daar over op de trein naar Rotterdam Voorburg
 Geslacht spre: 1: ja
 Geslacht spre: 2: en dan bent u in Den Haag
 Kwaliteit van: 1: nou het is ja hij gaat naar Rotterdam
 Vloeiendheid v: 2: nee het is al even te
 Luidheid van de z: 2: ik al even te
 Helderheid van c: 2: Hofplein Bergweg Rotterdam eerst als u in Den Haag
 Accent van spre: 1: Rotterdam Hofplein
 Luidheid van de s: 2: ja en [uh] u heeft dus eerst als u in Den Haag
 Helderheid van de Oost Indië en dan station Voorburg het Loo en daar
 Accent van spre: 1: oh ja dus het derde station dus
 Algemeen commentaa: 1: [mm] ja
 2: oké en [u:hm] terug naar Utrecht rond een uur of [u:h] nu
 **** 1: oké twaalf negentwintig kunt u terug bijvoorbeeld dat is
 2: [uh] twaalf negentwintig dan neemt u om twaalf tweeënveertig bij halt
 is dan wel de sprinter naar station Zoetermeer en dan neemt u daar om twaalf
 loopbrug over naar station Zoetermeer en dan neemt u daar om twaalf
 en dan bent u in Utrecht om dertien achtentwintig
 2: #1 waar vandaan #
 1: van Utrecht naar Alphen aan den Rijn ik neem aan dat dat de trein naar Leiden is
 2: ja
 1: vroeg ik me af[uh] hoe laat hij ging
 2: die gaat om tien over heel en tien over half
 #2 reistijd # is vijftwintig minuten
 1: oh dat is [] [] tien over heel en tien over half
 #3 ja # die [laughter/] stopt gewoon in Alphen aan den Rijn [laughter] #4 ja hoor
 #4 oké # prima bedankt #5 da:g #
 #5 graag gedaan # da:g
 1: [noise] [tone] [tone]

 Moelijkheid van de transcriptie: 4
 Omgevingsgeluiden spreker 1: 4
 Omgevingsgeluiden spreker 2: 4
 Geslacht spreker 1: 4
 Geslacht spreker 2: 4
 Kwaliteit van telefonische verbinding: 4
 Vloeiendheid van gesprek: 4
 Luidheid van de stem van spreker 1: 4
 Helderheid van de stem van spreker 1: 4
 Accent van spreker 1: 4
 Luidheid van de stem van spreker 2: 4
 Helderheid van de stem van spreker 2: 4
 Algemeen commentaar op gesprek: 4

 Regio: Utrecht - Alphen aan den Rijn
 Gespreksnummer: 176.28
 Gespreksduur: 96 (s)
 Categorie: reksnummer:
 Wachttijd: 18-09-95 reksduur:
 Transliteratiedatum: 17:14 - 17:27 reksduur:
 Transliteratietijd: 17:47 reksduur:
 Algemeen commentaar op gesprek: Wachttijd:
 Transliteratiedatum: 18-09-95 reksduur:
 Transliteratietijd: 17:47 reksduur:

Moeilijkheid van de transcriptie:	Omgevingsgeluiden spreker 1:	Omgevingsgeluiden spreker 2:	Geslacht spreker 1:	Geslacht spreker 2:	Kwaliteit van telefonische verbinding:	Vloeiendheid van gesprek:	Luidheid van de stem van spreker 1:	Helderheid van de stem van spreker 1:	Accent van spreker 1:	Luidheid van de stem van spreker 2:	Helderheid van de stem van spreker 2:	Algemeen commentaar op gesprek:
4	4	4	4	4	4	4	4	4	4	4	4	4

 Moeilijkheid van de transcriptie: 5
 Omgevingsgeluiden spreker 1: 4
 Omgevingsgeluiden spreker 2: 4
 Geslacht spreker 1: 4
 Geslacht spreker 2: 4
 Kwaliteit van telefonische verbinding: 5
 Vloeiendheid van gesprek: 4
 Luidheid van de stem van spreker 1: 4
 Helderheid van de stem van spreker 1: 4
 Accent van spreker 1: 4
 Luidheid van de stem van spreker 2: 4
 Helderheid van de stem van spreker 2: 4
 Algemeen commentaar op gesprek: 4

 Regio: Utrecht
 Gespreksnummer: 176.28
 Gespreksduur: 96 (s)
 Categorie: reksnummer:
 Wachttijd: 18-09-95 reksduur:
 Transliteratiedatum: 17:14 - 17:27 reksduur:
 Transliteratietijd: 17:47 reksduur:
 Algemeen commentaar op gesprek: Wachttijd:
 Transliteratiedatum: 18-09-95 reksduur:
 Transliteratietijd: 17:47 reksduur:

References

- [Alessi] Alessi, S., and Trollip, S. *Computer-based instruction: Methods and development 2nd ed.* Prentice Hall, Englewood Cliffs, 1991.
- [Bohm] Bohm, D. *On dialogue.* Edited by Nichol, L. Routledge, London, 1996.
- [Brady] Brady, M., and Berwick, R. *Computational models of discourse.* The Massachusetts Institute of Technology press, Massachusetts, 1983.
- [Brown] Brown, C.M. *Human-computer interface design guidelines.* Ablex Publishing corporation, Norwood, 1988.
- [Burkhardt] Burkhardt, A. *Speech acts, meaning and intentions: Critical approaches to the Philosophy of John R. Searle.* Walter de Gruyter, Berlin, 1990.
- [Carberry] Carberry, S. *Plan recognition in natural language dialogue.* The Massachusetts Institute of Technology press, Massachusetts, 1990.
- [Ferber] Ferber, J. *Multi-agent systems.* Addison-Wesley, New York, 1999.
- [Klein] Klein, M. *An overview of the state of the art of coding schemes for dialogue act annotation.* In: Text, Speech and Dialogue. Lecture Notes In Artificial intelligence 1692, Springer Verlag, Berlin, 1999.
- [Lepore] Lepore, E., and van Gulick, R. *John Searle and his critics.* Basil Blackwell, Massachusetts, 1991.
- [vdMast] van der Mast, C. *Developing educational software: Integrating disciplines and media.* Ph.D. Thesis, Delft University of Technology, Delft, 1995.
- [Pfeifer] Pfeifer, R., and Scheier, C. *Understanding intelligence.* The Massachusetts Institute of Technology press, Massachusetts, 1999.
- [Schank77] Schank, R., and Abelson, R. *Scripts plans goals and understanding: An inquiry into human knowledge structures.* Lawrence Erlbaum Associates, New Jersey, 1977.
- [Schank84] Schank, R.C., and Childers, P.G. *The cognitive computer: On language, learning and artificial intelligence.* Addison-Wesley Publishing Company, Massachusetts, 1984.
- [Smith] Smith, R.W., and Hipp, D.R. *Spoken natural language dialog systems: A practical approach.* Oxford University Press, New York, 1994.
- [vanVark] van Vark, R.J., de Vreught, J.P.M., and Rothkrantz, L.J.M. *Classification of public transport information dialogues using an information based coding scheme.* In: Dialogue Processing in spoken language systems. Lecture Notes In Artificial intelligence 1236, Springer Verlag, Berlin, 1997.
- [Weinschenk] Weinschenk, S., Jamar, P., Yeo, S. *GUI Design Essentials for Windows 95, Windows 3.1, World Wide Web.* Wiley Computer Publishing of John Wiley & Sons, 1997.

List of figures

Figure 2-1	Directive dialogue example	7
Figure 2-2	Non-directive dialogue example	7
Figure 2-3	The restaurant script	9
Figure 2-4	A human has cognitive, visual and motor limits	11
Figure 3-1	Dialogue example from the corpus	16
Figure 3-2	Example where client's utterance is coded as <i>verification question</i> because of the operator's reaction	17
Figure 3-3	Example where client's utterance is coded as <i>paraphrase</i>	17
Figure 3-4	Example where utterance is not split in two parts because of simultaneous speech at the end	18
Figure 3-5	Example where utterance is not split in two parts because of only one operator's reaction	18
Figure 3-6	Example where utterance is not split in two parts because of non-verbal utterance	18
Figure 3-7	Example of the way of coding when simultaneous speech occurs	19
Figure 3-8	Example of the way of coding when two information dialogue acts occur in one utterance	19
Figure 3-9	Example of constraint IV: Two consecutive utterances of the same person become one	19
Figure 3-10	Examples of non-verbal utterances	20
Figure 3-11	Examples of coding 'opening operator' (OO)	20
Figure 3-12	Examples of coding 'verification question' (V)	21
Figure 3-13	Example where utterance is coded as <i>verification</i> because information is already provided by client	21
Figure 3-14	Examples of coding 'clarification question' (CL)	22
Figure 3-15	Examples of coding 'new information question' (NI)	22
Figure 3-16	Example of coding 'repetition question' (R)	22
Figure 3-17	Examples of coding 'given answer' (GA)	23
Figure 3-18	Examples of operator's advice, which are coded as GA	23
Figure 3-19	Examples of coding 'remark operator' (RO)	23
Figure 3-20	Examples of utterances in which operator asks the client to wait for a moment	23
Figure 3-21	Examples of coding 'goodbye operator' (GO)	24
Figure 3-22	Examples of coding 'opening question client' (OC)	24
Figure 3-23	Examples of coding 'answer verification' (AV)	25
Figure 3-24	Examples of coding 'answer clarification' (ACL)	25
Figure 3-25	Examples of coding 'answer new information' (ANI)	25
Figure 3-26	Example of coding 'answer repetition' (AR)	26
Figure 3-27	Examples of coding 'question client' (QC)	26
Figure 3-28	Examples of remarks that are coded as question	26
Figure 3-29	Examples of coding 'remark client' (RC)	26
Figure 3-30	Examples of coding 'goodbye client' (GC)	27
Figure 3-31	Example dialogue in which the client the operator first thanks before saying goodbye	27
Figure 3-32	Example dialogue in which the client indicates he has sufficient information before saying goodbye	27
Figure 3-33	Example dialogue in which examples of the sub-coding and sub-sub-coding	29
Figure 3-34	Main window of coding tool	31
Figure 3-35	Screen shot of main window with list dropped down	31
Figure 3-36	Window of coding tool in which database can be viewed	32
Figure 3-37	Program flowchart of coding tool	34
Figure 3-38	Dialogue fragment in which sub-coding of GA is not increased regularly	37
Figure 4-1	Top-level dialogue model	42
Figure 4-2	The four phases indicated in a dialogue example	42
Figure 4-3	OVR-Dialogue model	43
Figure 4-4	Example of 'Opening finished': Operator confirms client's question with 'yes'	44
Figure 4-5	Example of 'All information collected 1': The second phase is skipped	44
Figure 4-6	Example of 'All information collected 2'	45
Figure 4-7	Example of clarification question in the third phase (QTO)	46
Figure 4-8	Second example of clarification question in the third phase (QTO)	46
Figure 4-9	Example where the operator provides two options to the client without clarifying first	47
Figure 4-10	Example of several clients' remarks on the operator's answer	47
Figure 4-11	Example of 'Sub-query' – Clients asks several questions	48
Figure 4-12	Example of 'Remark operator 4'	48
Figure 4-13	Example of operator's question after client's remark	49
Figure 4-14	Example of goodbyes	49
Figure 4-15	Example of problem 1: Starting utterance of operator is repeated later in the dialogue	51
Figure 4-16	Example of problem 2: Operator interrupts client during his opening utterance	51
Figure 4-17	Example of problem 3: Client answers question and at the same time asks a question	52
Figure 4-18	Example of problem 4: Operator answers question and at the same time asks a new question	52
Figure 4-19	Example of problem 5: Client's answer is split in two by operator's remark	53
Figure 4-20	Example of problem 6: Client responds with question instead of answer	53
Figure 4-21	Example of problem 7: Operator verifies the answer on the second clarification question	54
Figure 4-22	Example of problem 8: Operator takes initiative to start with saying goodbye	54
Figure 4-23	Example of problem 9: Client says goodbye and then asks another question	55
Figure 4-24	Example of problem 10: Client asks questions about Schiphol	55
Figure 4-25	Dialogue model with transition frequencies	57
Figure 4-26	Pyramid representing dialogue strategies	58

Figure 4-27	Possible train of thoughts of an operator	59	
Figure 4-28	Possible hypotheses as a reaction on a client's prompt	61	
Figure 4-29	Possible operator's prompts as a reaction on a client's utterance	62	
Figure 4-30	Dialogue with the operator's train of thoughts visualised in slots	65	
Figure 5-1	Example of the operator's knowledge about the infrastructure	66	
Figure 5-2	Example of the operator's knowledge about time interpretation	67	
Figure 5-3	Example of the operator's knowledge about the event 'Pinkpop'	68	
Figure 5-4	Travel planner input window	69	
Figure 5-5	Travel planner output window	70	
Figure 5-6	Supplementary route information	71	
Figure 5-7	Slots the operator has to think about while solving the client's question about a specific travel route		72
Figure 5-8	Price knowledge structure	75	
Figure 5-9	Example of a dialogue where the price question is asked at the beginning of the dialogue		76
Figure 5-10	Example of a dialogue where the price question is asked at the end of the dialogue	77	
Figure 5-11	Several examples of a client's question concerning prices	78	
Figure 5-12	Example where the operator can answer immediately	78	
Figure 5-13	Example where the operator asks for the exact stations	78	
Figure 5-14	Example where the default station values are used and both prices are given		78
Figure 5-15	Example where the default value 'round trip' is used	79	
Figure 5-16	Example where default value 'second class' is used	79	
Figure 5-17	Example of asking a reduced price	79	
Figure 5-18	Model of the operator's response to a price question	80	
Figure 5-19	Flowchart of the new price dialogue training model	81	
Figure 5-20	A dialogue example where 'second class' is provided with the answer	82	
Figure 5-21	Examples of questions about a specific travel route	83	
Figure 5-22	Example dialogue where operator asks for part of day	84	
Figure 5-23	Knowledge structure of question about a specific train travel route		85
Figure 5-24	Example where the operator asks for the travel date	86	
Figure 5-25	Example where the operator assumes that the client wants to travel 'today'		86
Figure 5-26	Train ticket price summary	86	
Figure 5-27	Several examples of price presentation	88	
Figure 5-28	Several examples of travel route presentation	89	
Figure 6-1	Overview of functional design of the DTE	92	
Figure 6-2	Lesson screen design	94	
Figure 6-3	Exercise screen design	95	
Figure 6-4	Overview technical design	102	
Figure 6-5	Main menu where the trainee can choose between the several training parts		105
Figure 6-6	Lesson window	105	
Figure 6-7	Window of exercise three of lesson one	106	
Figure 6-8	Window of exercise one of lesson two	106	
Figure 6-9	Window of exercise two of lesson two	106	
Figure 6-10	Window of exercise one of lesson five	107	
Figure 6-11	Window of exercise two of lesson five	107	
Figure 6-12	Window, where the choice is made to respond with a reaction from the 'new information' category		107

List of tables

Table 3-1	Dialogue acts operator	20
Table 3-2	Dialogue acts client	24
Table 3-3	Sub-coding and sub-sub-coding QC	28
Table 3-4	Sub-coding and sub-sub-coding RC2 and RO4	28
Table 3-5	Structure of database record	33
Table 3-6	Main window controls with their database fields	35
Table 3-7	Sub-window controls with their database fields	35
Table 3-8	Frequencies per dialogue act operator	36
Table 3-9	Frequencies per dialogue act client	36
Table 3-10	Frequencies per additional coding QC	36
Table 3-11	Frequencies per additional coding RC2	36
Table 3-12	Frequencies per additional coding RO4	36
Table 4-1	Transitions between the used dialogue acts (read from left to right)	50
Table 4-2	Transitions between dialogue acts without problem dialogues	56
Table 4-3	Several examples of operator's responses on client's prompts	63
Table 6-1	Structure of first table type	104
Table 6-2	Structure of second table type	104
Table 6-3	Structure of third table type	105

Regio: 1
Gespreksnummer: 11845602
Gespreksduur: 33.10 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachttijd: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

Regio: 1
Gespreksnummer: 11848103
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848104
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848105
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848106
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848107
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848108
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848109
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848110
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

APPENDIX A

Paper 'Dialogue control in the ALPARON system'

The third international workshop on Text, Speech and Dialogue. Brno, Czech Republic, 2000 Lecture notes in Artificial Intelligence, Springer Verlag, Berlin.

Regio: 1
Gespreksnummer: 11845602
Gespreksduur: 33.10 (s)
Categorie: trein_info
Wachttijd: 32 (s)
Transliteratiedatum: 18-09-95

Regio: 1
Gespreksnummer: 11848102
Gespreksduur: 27.04 (s)
Categorie: trein_info
Wachttijd: 231 (s)
Transliteratiedatum: 19-09-95
Transliteratietijd: 11:54 - 11:57

Regio: 1
Gespreksnummer: 11848103
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848104
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848105
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848106
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848107
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848108
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848109
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Regio: 1
Gespreksnummer: 11848110
Gespreksduur: 53.08 (s)
Categorie: trein_info
Wachttijd: 38 (s)
Transliteratiedatum: 18-09-95
Transliteratietijd: 12:56 - 13:03

Dialogue control in the ALPARON system

L.J.M.Rothkrantz, R.J. van Vark, A. Peters, A.C. Andeweg

Knowledge Based System Group
Delft University of Technology,
Zuidplantsoen 4
2628 BZ Delft, The Netherlands
{L.J.M.Rothkrantz,R.J.vanVark}@cs.tudelft.nl

Abstract. The main topic of this paper is on modelling a human operator in the dialogue manager of the Alparon system. First, a corpus of 200 human-human dialogues have been analysed applying an approach surpassing a finite state automation approach. The corpus analysis resulted in a set of common strategies applied by professional human operators in similar situations. Secondly, a prototype system has been built based on the Alparon dialogue manager. This has been done by translating the strategies into knowledge rules and heuristics as these are used by the dialogue control modules in the Alparon dialogue manager.

1 Introduction

Dialogue management is the key factor in automated speech processing systems for information retrieval and transaction services for it can overcome the shortcomings of a one-shot approach. In a one-shot approach the user specifies his requirements to the system in one turn. Then, the system tries to understand the user's requirements using the information provided in this turn and the system gives an appropriate response, also in one turn. If this approach fails, the user has to start again from scratch. In a one-shot approach one cannot speak of a dialogue-like interaction as there is simply an iteration of actions followed by reactions.

To behave in a more user-friendly fashion the system has to be able to cope a wide variety of phenomena like speech recognition errors, misunderstandings, partial specification of user requirements, conflicting requirements, etc. To cope with these phenomena, the system engages in a dialogue with the user using information collected in previous turns to improve the current interpretation of the user's requirements. Therefore dialogue management is an essential element for a user-friendly automated speech processing system.

Researchers at the Delft University of Technology have developed a natural language system aimed at information retrieval and transaction services. The focus of the Alparon system is on dialogue management, especially for speech-enabled applications. Other components of the automated speech processing system, like a speech recogniser, parser and speech generation, are only incorporated to make a complete test bed for testing dialogue strategies. The goal of

the Alparon project is to investigate the application of human-human strategies in human-computer dialogues.

This paper describes an approach to model a human operator using the dialogue manager of the Alparon system. This paper especially focusses on retrieving the user's requirements and not on presenting the information itself. Modelling the information of presentation has been presented earlier by researchers of the Delft University of Technology (see [1]). First, applicable human-human strategies have to be analysed. This was done by analysing a corpus of over 200 human-human dialogues between customers requiring travel information and professional operators. The analysis of this corpus is described in the second section. Secondly, the human-human strategies have to be translated to knowledge rules and heuristics as they are applied in the dialogue control components of the Alparon dialogue manager. A description of the Alparon prototype modelling a human operator can be found in section 3. The final section presents conclusions and future work.

2 Computational dialogue management

One way to design an effective, efficient and user-friendly dialogue manager is to apply a corpus-based approach. In such an approach, a corpus of human-human dialogues is studied to derive applicable strategies. In the Alparon project, a corpus was available consisting of 5.000 recorded and annotated human-human dialogues on timetable information in public transport. A selection of 500 was coded using the Alparon coding scheme [2]. This coding scheme applies dialogue acts to code the influence of an utterance to the dialogue context.

A dialogue can be represented as a tree. Every layer in the tree represents a turn in the dialogue and the nodes correspond to individual dialogue acts. As many dialogue acts can be found in a dialogue, the tree grows exponentially as the dialogue progresses. Applying such methods would call for a huge corpus.

Therefore, another way of representing the dialogue discourse was needed. In this paper, a dialogue is considered from the operator's point of view. This corresponds to the dialogue manager's point of view as the the operator will be modelled in the Alparon system. A computational model of the operator combines thoughts about the hypotheses, the client's intentions and the prompt choices. The model is constructed using goal-directed principles. The operator bases his/her behavior on a comparison of a representation of the goal-state and the current state. The operator can use the planning strategy means-end analysis. Means-end analysis requires a measure of distance between the current state and the goal-state. The next step is then chosen based on an evaluation of how much this step will reduce the distance to the goal state. A useful reaction of the operator provides new information, reduces the uncertainty or provides clearness, whereas an inappropriate reaction takes extra time or causes confusion.

As such, the quality of an operator's prompt is defined by the extent the prompt will get the dialogue nearer to the goal, being defined as providing the information the client desires. One way to assess the quality of the prompts is

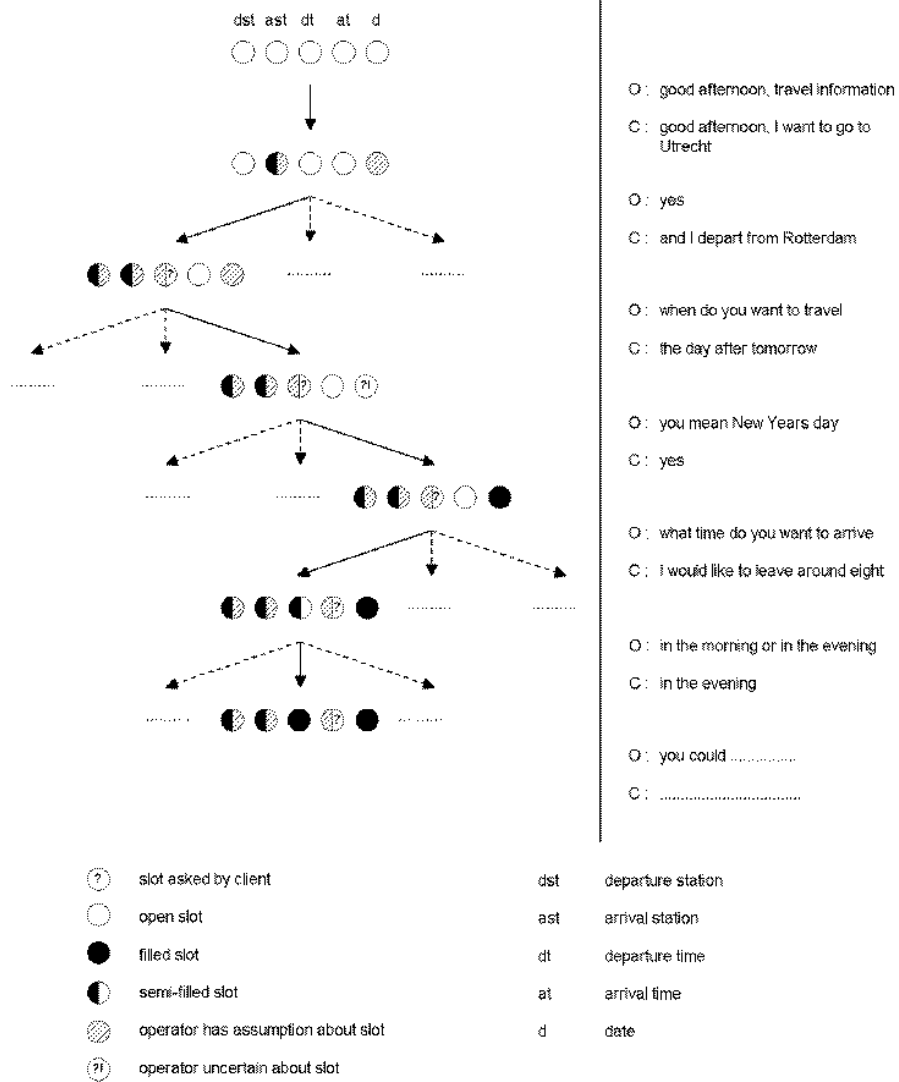


Fig. 1. Dialogue model from the operator's perspective

by applying an evaluation function that provides a score for each prompt. At each step the prompt with the highest score is chosen. It is possible to think several steps ahead and combine the weights that belong to these steps. This way all the possible paths through a dialogue are scored and the path with the highest score can be chosen. Thinking several steps ahead is effective in finding the best path through a dialogue. The shortest path without detours produces the most efficient dialogue but it is questionable whether the client appreciates such a dialogue. The factors that influence the professional operator in deciding on which prompt to choose have to be combined in these weights.

In figure 1 the operator's mental state during a dialogue is displayed. The operator's thoughts are illustrated in the picture by using slots, represented by five circles. The goal is to fill the slots. All slots are empty at the beginning of the dialogue. Every state represents the operator's thoughts at that time. A new state is displayed after an operators prompt and the client's reaction to it. The next state indicates if there are differences in the operator's thoughts about the slots as compared to the previous state. At every state there are different ways to continue the dialogue. This is indicated in the figure by multiple arrows on a level. Using an optimal strategy implies that in the next state a maximal slot-filling is achieved. This slot-filling is done by reducing uncertainty about the slots or asking for new information.

Dialogue graphs were constructed for 200 dialogues selected from the Alparon corpus. In a one-step approach those prompts were selected providing a maximal filling of the slots. In about half of the cases it was possible to generate knowledge rules and heuristics how to select the appropriate prompt. In the other cases the history has to be taken into account or a multiple step approach, or the client took the freedom to take the initiative or could not provide the information. Some examples of heuristics derived using this strategy, are:

- if there is a (non-empty) subset of open slots and a (non-empty) subset of filled slots, ask for information covering as much of the open slots as possible;
- if one slot in the current state is not completely filled, immediately ask for the missing information to solve the ambiguity concerning the slot;
- if the subset of not complete filled slots contains more than one slot, handle the individual slots one after the other;
- as long as new information can be provided, assumptions of the operator are not verified;

It should be stressed that in the current application the operator needs information from the client to define a query for a database. This might cause the operator's behaviour to be rather slot-oriented, as was found for the current domain. There is a tendency from the operator to formulate open questions. In case of automated speech-driven systems there is a tendency to generate closed questions to prevent misunderstanding. In the next section an approach is presented which compromises both extreme viewpoints.

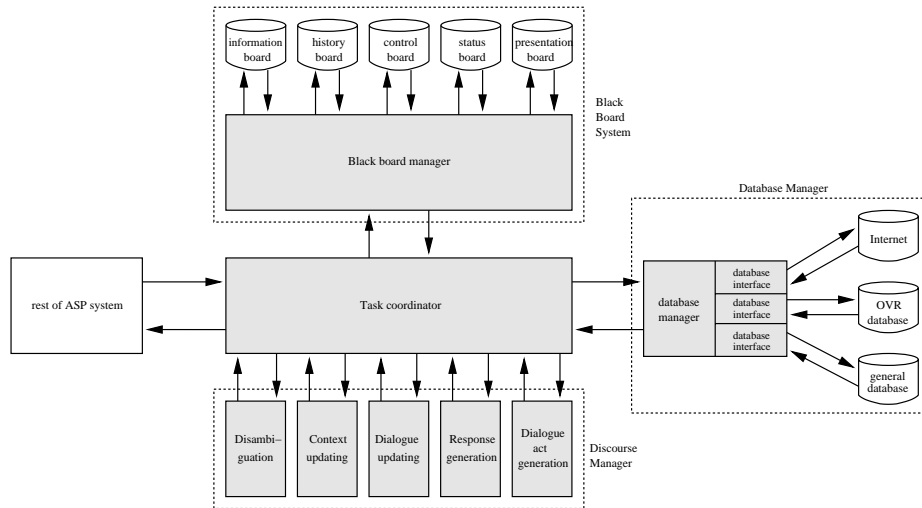


Fig. 2. Alparon dialogue manager overview

3 The Alparon natural language system

The Alparon natural language system focusses on applying human-human strategies in dialogue management for information and transaction services. The Alparon dialogue manager consists of several modules (see figure 2). This modular design was chosen to create a flexible and extensible system. When implementing information services, the dialogue manager should be able to interpret the user's utterance in the context of the ongoing dialogue. This interpretation is then used to generate a response to the user's utterance. Information on the current state of the dialogue and domain knowledge can be used to perform these tasks.

The main topic of this paper is on using human operator strategies to implement the dialogue control mechanism in the Alparon dialogue manager. This mechanism is to reason out a system response after the user's utterance is interpreted. These responses are the system's only means of influencing the user and thus controlling the dialogue. An approach surpassing a finite-state automation approach should be applied.

In the original design of the Alparon system, the task of generating system responses was placed with the response generation module and the control board. All the processing necessary to generate the system response should be done by the response generation module. This would include processing the user's turn on a dialogue level, since the context-updating module only takes care of the user's turn on an information level. In the actual design for dialogue control as described in this paper, another module was added, the dialogue updating module, to determine the consequences of the user's turn on a dialogue level.

The dialogue updating module is charged with updating of goalstack according to the state of the dialogue in each turn. The goalstack is used to structure the goals that define what must be achieved during the dialogue. In information retrieval dialogues the dialogue structure highly resembles the task structure. So a plan based approach is used in which the goals embody the plan to accomplish the task and subtasks in such a dialogue. When disambiguation and context updating modules have interpreted the user's turn and extracted the factual information from it, the dialogue updating module processes the consequences of the new information and the user's turn on a dialogue level. The goalstack is updated to reflect this new situation.

The task of the response generation module is to generate the actual system response. Given the goals that still have to be achieved, the response generation module places communicative actions linked to those goals and selects communicative actions to be combined in a system response. The communicative actions are linked to goals and they are designed to help achieve the goals.

The control board is used to store the goal structure and the communicative actions, both the possible ones and the ones selected for the system response. This goal structure and the communicative actions implement the rules and strategies extracted by our corpus analysis. Other information related to dialogue control is also placed on blackboards, e.g. conflicts in the system beliefs of the dialogue. All the information used for dialogue control is present on the blackboards.

4 Conclusions

In this paper we presented a prototype of a dialogue management system. The dialogue control strategies implement a human operator applying rules and strategies extracted from a corpus of human-human dialogues. In the research environment of the Alparon system, dialogue control is designed as a generic framework. Within this framework various rulesets can be defined to specify the behaviour of dialogue control. By changing these rulesets various approaches to dialogue control can be implemented and refined. The rulesets specify one engine, consisting of all components for dialogue control. As a result, the actual strategy for dialogue control is defined by these rulesets.

References

1. M.M.M. Rats, R.J. van Vark, and J.P.M. de Vreught. Corpus-based information presentation for a spoken public transport information system. In *ACL/EACL Workshop on Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications*, pages 106–113, 1997.
2. R.J. van Vark, J.P.M. de Vreught, and L.J.M. Rothkrantz. Classification of public transport information dialogues using an information based coding scheme. *Dialogue Processing in Spoken Language Systems, Lecture Notes in Artificial Intelligence (1236)*, pages 55–69, 1997.