# TeleBank

TeleBank

**Micha Moffie**

**Technical Informatics**
**Faculty of Information Technology and Systems**
**Delft university of Technology**
**TU Delft**

Delft, November 1999

1. Acknowledgments .......... 3
2. Introduction .......... 3
3. Dialogs .......... 4
4. Problem Definition .......... 4
    TeleBank Requirements .......... 4
    Goal .......... 4
5. SpeechMania .......... 5
6. The Application .......... 7
7. Design .......... 8
    Dialog Design .......... 8
    TeleBank Architecture .......... 8
    Cooperation and user friendliness .......... 10
8. Implementation .......... 10
    Goal .......... 10
    Assumptions .......... 11
    Examples .......... 11
9. Testing .......... 16
    Example .......... 16
    Results .......... 19
10. Conclusions and Further research .......... 20
    Conclusions .......... 20
    Further research .......... 20
11. References .......... 20
Appendix A – Dialog flow chart diagrams. .......... 21
Appendix B – HDDL code .......... 33

# 1. Acknowledgments

During the second half of 1999, I was an exchange student at the *Delft University of Technology*. My home university was the *Technion, Israel institute of Technology*. This work is a project work for the course 'practicum informatica'.

I would like to thank both exchange offices at the Technion and at TU delft for allowing me to participate in the program. I would also like to thank Dr. Leon Rothkrantz at the TU delft and Dina Katz at the Technion for their personal help and guidance during this period.

# 2. Introduction

Many banks offer telephone facilities to their customers, most of them operated by humans. Most of the automatic services, which are given, do not use speech recognition but instead use DTMF tones (tones created by pressing the numbers on the telephone). Thus, automatic services are mostly unfriendly, and involve instructions of the kind: 'for the first option press one, for the second option press two … ' and so on.

TeleBank is an automatic application, which is able to give some of the bank services over the telephone. It is designed to enable the user a friendlier interface, namely: speech. The TeleBank application initiates a dialog with the user in a similar way that a human operator would act. During this dialog TeleBank learns what the user wants and all information needed to fulfill the user requests.

TeleBank was coded in HDDL (Dialog Description Language), within the SpeechMania framework. Using the SpeechMania Online System, the TeleBank application could be tested as in real life, using a telephone line in real-time.

# 3. Dialogs

Dialogs or conversations are the way in which humans communicate. Humans communicate for social reasons, to exchange information and to seek for new information. At this moment, speech technology is not matured enough to emulate human-human conversations. The vocabulary has to be restricted to less than 1000 words. Therefore, speech technology can only be used in applications with limited domains.

The developer of the interactive system should decide where and how to limit the speech system. His decisions will influence the performance of the system. We can say that performance of a system is actually its cooperativity, i.e. how cooperative the system is when proceeding with the user the common goal.

In most cases, in human-human dialogs, the initiative is mixed. When one of the participants in the dialog is a computer it will generally take the initiative and ask the questions. The system can preserve the initiative by asking the user each time a closed question, i.e. a question that asks for a well-defined piece of information. This method limits the scope of the dialog and can increase the performance of the system.

Thus, a human-computer dialog can be seen as a limited subset of a human-human dialog. It is simpler, task oriented, constrained and unequal – only the computer has the initiative.

# 4. Problem Definition

Create an application that can provide, using the telephone, fully automated bank services.

## TeleBank Requirements

The following services are required:

1. *Open a new account.*
2. *Close an existing account.*
3. *Deposit money in an account.*
4. *Withdraw money from an account.*
5. *Inform how much money there is in an account.*
6. *Get help/information about the application.*

## Goal

1. Design and implement TeleBank.
2. Implement TeleBank using SpeechMania framework.
3. TeleBank should be as cooperative and user friendly as possible.

# 5. SpeechMania

As stated already, HDDL and SpeechMania (version 2.2) were used to develop the TeleBank application. SpeechMania framework and the HDDL language are both the products of Philips Speech Processing group http://www.speech.be.philips.com/. The HDDL language and SpeechMania environment are especially designed for the specification and creation of automatic inquiry systems. Within SpeechMania is a framework that integrates speech recognition, speech understanding and dialog control into a functional online system, which can be used to serve a telephone line in real-time. The following figure shows the SpeechMania Online System Architecture:

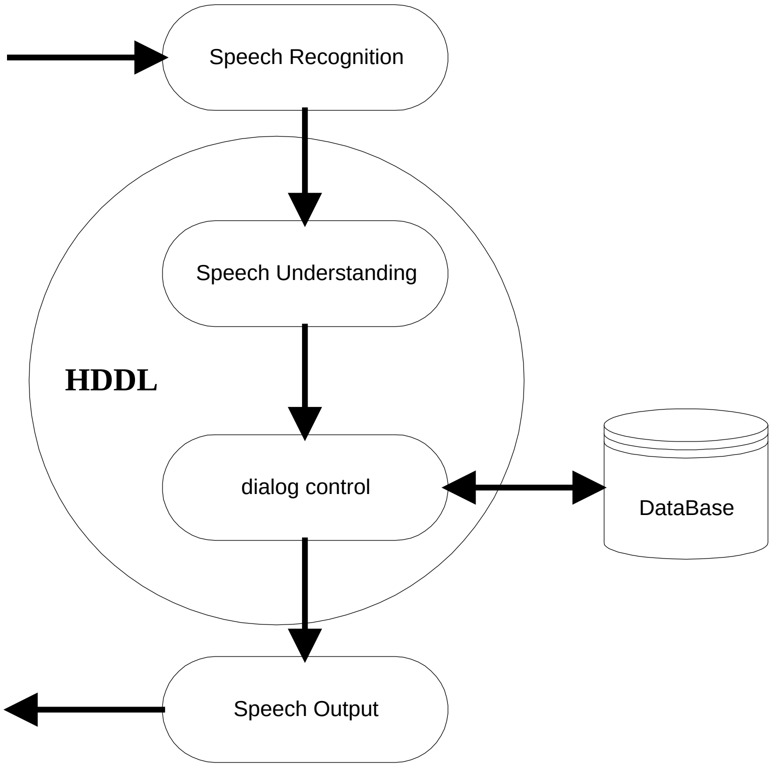


Figure 1

The main job of the developer is writing the HDDL code. This code, as can be seen in figure 1, will have to ‘understand’ the words which where recognized by the speech recognition component and control the dialog with the user. Although the speech recognition component is not a part of the HDDL language the developer can influence the speech recognition process.

The HDDL code is comprised out of dialogs – which can be seen as modules. Each dialog has a number of sections. The sections must appear in a specific order, some of the sections may be empty. The following figure shows the HDDL sections and associated components:

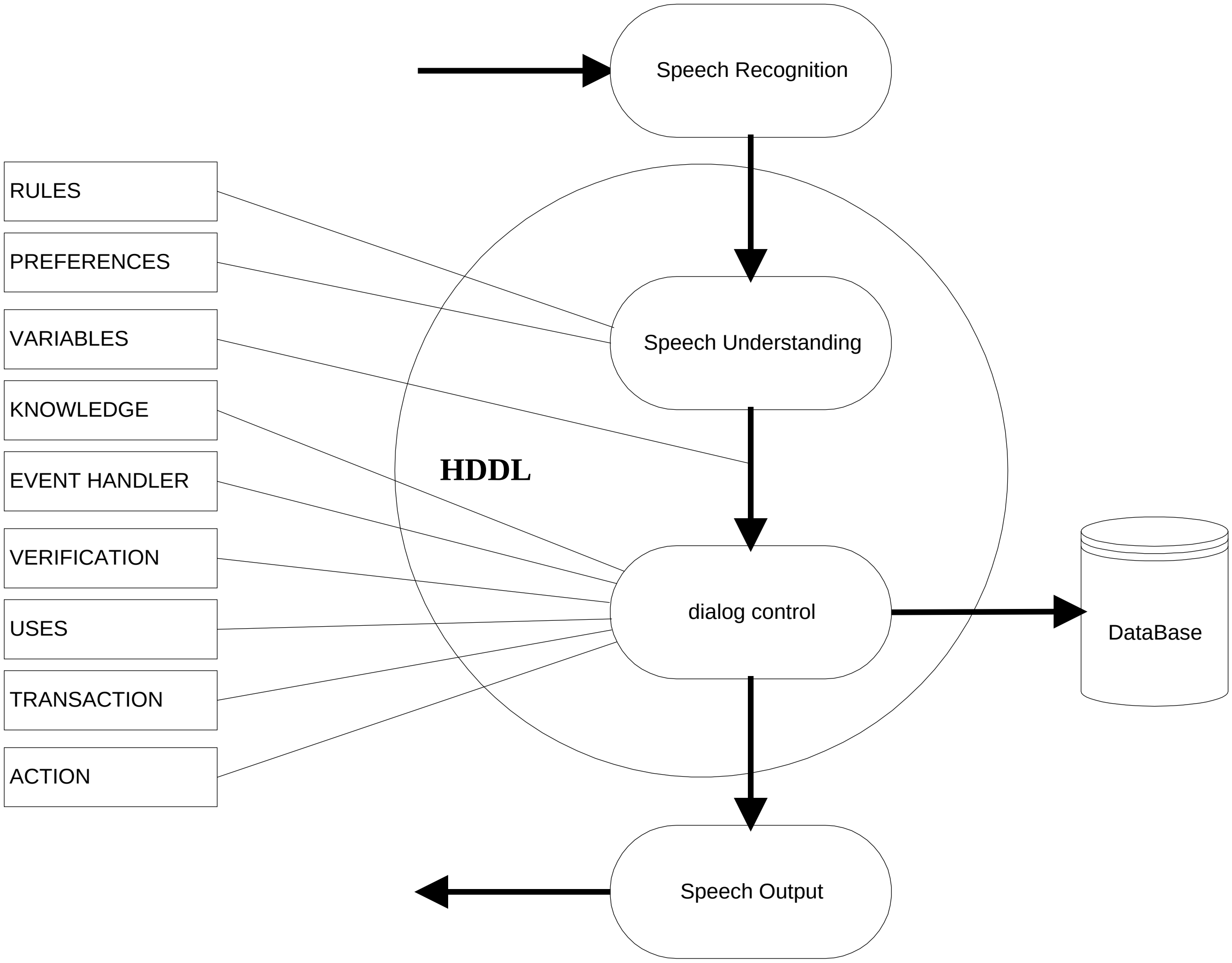


Figure 2

The RULES and PREFERENCES sections define the speech understanding part of the dialog, the VARIABLE section describes the interface between the speech understanding and dialog control. In the EVENT HANDLER and VERIFICATION sections, common components can be defined for use in the dialog. The USES section describes the interface to other dialogs while the TRANSACTION section describes the interface to external applications. The ACTION section describes the main part of the dialog; here the questions are asked, according to predefined conditions.

# 6. The Application

The TeleBank application offers a few basic telephone bank services, which are fully automated. The application does not include all the functionality needed in a real system, for example: there is no identification of the user towards the system. Therefore TeleBank system can not be used as is in real life. In spite of this fact the TeleBank application shows what can be achieved using SpeechMania and can be used as a basis for further development.

Following are the options implemented in TeleBank:

**Open a new account.**

This is a basic option in any bank. It is simplified in TeleBank since there is no user identification. Every user can open one or more bank accounts. After the system opens a bank account it will return to the user and inform the user of its new bank account number.

**Close an existing account.**

This option allows any user to close any bank account. After TeleBank verifies which bank account number the user wishes to close, it will check whether this bank account exists. Then the user will be notified of the operation results.

**Deposit money in an account.**

The Deposit money option allows any user to deposit any sum of money in any bank account. This option can also be seen as the last part of a transfer of money between two accounts. In order to execute this command, TeleBank will have to confirm the following things:

1. The user wishes to deposit money.
2. The account number where the money will be deposited.
3. The amount of money that will be deposited.

**Withdraw money from an account.**

Since the implementation of this option is **very** similar to the *Deposit money in an account* option it was not implemented.

**Inform how much money there is in an account.**

Asking for an account balance is one of the most common actions. This option is also implemented in TeleBank. A user can ask for a balance of every bank account. Once TeleBank established that this bank account exists it will inform the user of the balance.

**Get help/information about TeleBank**

Every application should have a help option. In TeleBank a user can choose the help option and receive information about the application. TeleBank will also suggest using the help option when it detects that the dialog with the user does not progress.

# 7. Design

Designing TeleBank was done incrementally. The design process was guided by two main considerations: The Dialog flow and the System capabilities. The design process began by sketching dialog flows. For each of the dialog flows the system capabilities were assessed and tested. If the capabilities satisfied the requirements the dialog flow was implemented as is, otherwise the dialog flow was rearranged according to the existing system capabilities.

## Dialog Design

Most applications are based on a corpus of recorded human-human dialogs; to set up a corpus is very time consuming. Such a corpus was not available, so we chose another approach based on our own experience and ideas.

The Dialog begins with a greeting, then it continues with a question, whether the user knows the system or not. If the user does not know the system, a help message is produced. In the next stage TeleBank asks which of the options the user would like. At this point the dialog is directed according to the option chosen by the user. For each of the options a different flow exists, this flow will retrieve the necessary information from the user. After all information is gathered the request is executed and the user is notified of the results.

After the request is handled TeleBank asks whether the user wants to continue. If the user does not want to continue the system will exit, otherwise the system will ask again which of the options the user wishes and will continue from the same point in the dialog as explained above.

For more detail, please refer to Appendix A – Dialog flow chart diagrams.

## TeleBank Architecture

The TeleBank architecture corresponds to the dialog flow; each sub flow is implemented in a separate module – a HDDL dialog. TeleBank starts the execution in the main module. This module instantiates sub modules and activates these modules, which in turn can activate other modules. For example: the main module activates the 'applicationOption' module, which is responsible to ask the user which option he would like. This module in turn activates a specific sub module that will manage the dialog with the user according to his request.

As a regular bank, the TeleBank application must have a database in which all information of the bank accounts are stored. This database is accessed through a special interface in the HDDL language.

In the following figure the TeleBank architecture is represented in a flow chart:

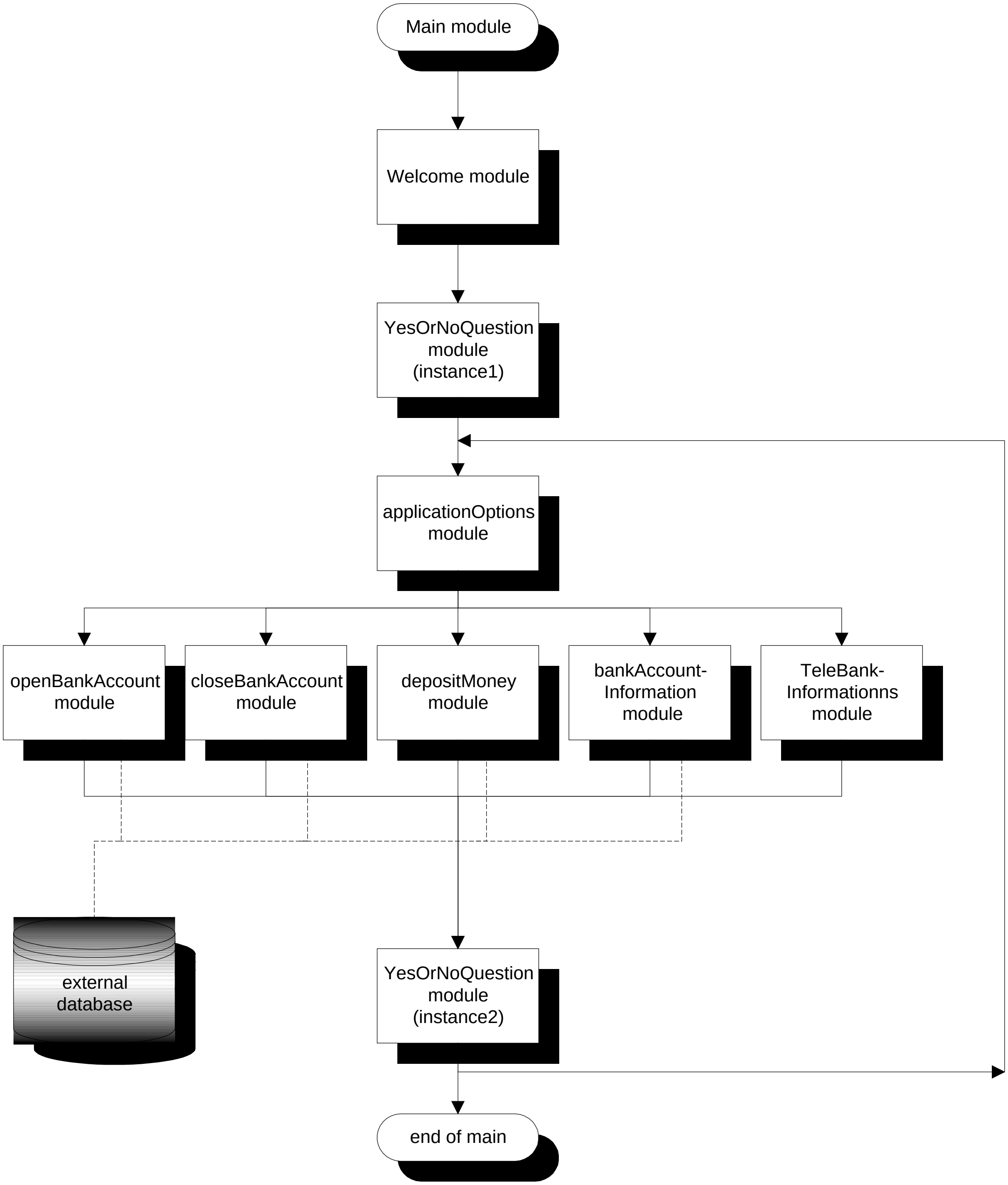

Main module
Welcome module
YesOrNoQuestion module (instance1)
applicationOptions module
openBankAccount module
closeBankAccount module
depositMoney module
bankAccount-Information module
TeleBank-Informationns module
external database
YesOrNoQuestion module (instance2)
end of main

Figure 3

The following modules were implemented in HDDL:

- Main dialog: the first dialog that is executed, here the applicationOption dialog is instantiated and is activated again and again until the user does not want to continue.
- Welcome dialog: this dialog greets the user according to the daytime.
- YesOrNoQuestion dialog: this dialog is a template to all questions, which can be answered with only yes or no. The dialog input is a question and output is the answer to that question.
- ApplicationOption dialog: this dialog asks which option the user wants and activates the corresponding dialog.
- OpenBankAccount dialog: this dialog handles the opening of a new bank account.
- CloseBankAccount dialog: this dialog implements the closing of an existing account.
- DepositMoney dialog: this dialog is activated when the user wishes to deposit money in an account.
- BankAccountInformation: this dialog is called when the user wants to know the balance of an account.
- TeleBankInformation dialog: this dialog will explain to the user how to use TeleBank.

## Cooperation and user friendliness

In order to create a prototype, which is cooperative and user friendly as possible, the following decisions where made:

- Reduce to minimum the number of closed questions.
- Ask ‘semi’ closed questions.
- Allow mixed initiative.

Asking an ‘semi’ closed question is much more user friendly in comparison to closed questions. An example of a ‘semi’ closed question in TeleBank is 'which option do you want?' this question is asked in the applicationOption module. While designing TeleBank an effort was made to reduce to minimum confirmation questions and replace them with verification questions. For example, if the user says he would like to close a bank account, the system does not ask if the user wishes to close a bank account (confirmation. A closed question which the answer to it can be yes or no). But instead asks for the user’s bank account number (verification). Thus giving the user the feeling that he was understood perfectly the first time and can continue, while at the same time allowing the user to correct TeleBank by saying: 'I don't want to close an account'.

Allowing some user initiative is done by allowing the user to add more information to that which was asked for. For example, in response to the 'which option do you want' question the user may answer: 'I would like to close bank account number 495'. In this case it is obvious that the user took the initiative and added the account number although he was not asked for it. There are numerous questions in TeleBank which 'listen' to more information than what was asked for.

# 8. Implementation

TeleBank was implemented in the HDDL language. HDDL is somewhat different from other high level languages, in that it has been designed to solve a specific set of problems. These problems are associated with dialog control, speech understanding and database queries. For this reason the language has its own paradigm, including a set of terms and structure.

## Goal

Implement TeleBank in a way that is readable and easy to maintain. Explore the different capabilities of HDDL and employ them, but also, at the same time, to find the language limitations and drawbacks.

## Assumptions

In order to concentrate on the more important issues some assumptions where made. These assumptions are easy to change if needed.

1. All numbers in TeleBank range from 0 – 999 including the bank account numbers and amounts.
2. Each number must be spelled digit by digit, i.e. if the user wants to deposit money in account number 23 he should say: ‘I want to deposit an amount at bank account two three’.
3. The TeleBank database is implemented using a DLL. It is very simplified and can hold up to 1000 bank accounts.

## Examples

In this section a few examples of HDDL code will be presented, and a few issues will be discussed. All the example code is taken from TeleBank code that is presented in full in Appendix B.

- The first example will be of a question. A question is a basic part in each dialog.

```
QUESTION(option,accountNumber)
{
        INIT
        {
                "wat is de bank rekening nummer waarvan u de";
                VERIFICATION(option); // "saldo";
                "wilt weten";
        }
        MyEvents( NUstatements;  ,
                  NUstatements;  ,
                  NRstatements;  ,
                  NUstatements;  ,
                  "wat is de bank rekening nummer waarvan u de";
                  VERIFICATION(option); // "saldo";
                  "wilt weten";
        );
}
```

Figure 4

The example in figure 4 shows a question that is asked in the bankAccountInformation dialog (module), and can be translated as ‘what is the bank account number of which you want to know the balance?’ The question is asked immediately after the user says he would like to know the balance of a bank account.

After a the keyword QUESTION there is a list of variables, these variables are the ones expected to be understood in the user response to the question. In the body of the question different events are specified: INIT, NOTHING UNDERSTOOD and so on. (MyEvents… is a macro, which can be found in declarations.hdl file; some events are specified there).

Each of these events will be executed under a specific condition. For example the INIT event will be executed immediately when the QUESTION is entered. The NOTHING UNDERSTOOD event will be executed if none of the variables stated after the QUESTION keyword has been found in the user response. When an event is generated the event handler defined in the question will be executed, if there is no event handler defined in the question, the system will look for an event handler defined in the EVENT HANDLER section of that dialog. If no event handler is specified in the sub dialog EVENT HANDLER section the system will execute the event handler in the EVENT HANDLER section in the main dialog (module).

This mechanism allows handling of unexpected user responses. Using the MyEvents macro, these event handlers can easily be defined for each specific question. There is one event, which is defined globally in main; this event is the RECURRING_PROBLEMS event. This event is executed when events are activated repetitively. In this case the system asks the user to call the usual bank and exists.

- As stated above a variable describes the interface between the speech understanding and dialog control. This interface will be shown in this and the next example. First let us see how a variable is defined, linked and how it gets its value.

```
…
ENUMERATION teleBankOptions
{
    OPEN_BANK_ACCOUNT,
    CLOSE_BANK_ACCOUNT,
    DEPOSIT_MONEY,
    WITHDRAW_MONEY,
    BANK_ACCOUNT_INFORMATION,
    TELEBANK_INFORMATION
}
…
RULES
    …
    <bankAccount>
            : 'rekening'
            | 'bank' 'rekening'
    ….
    <teleBankOption> teleBankOptions option,
                     INTEGER accountNumber,
                     INTEGER amount
    …
    | 'geld' 'storten'                          { option := DEPOSIT_MONEY;  }
    | 'storten' 'op' <bankAccount> <number>     { option := DEPOSIT_MONEY;
                                                  accountNumber := #4.value; }
    | <number> <gulden> 'storten' 'op'
                     <bankAccount> <number>{ option := DEPOSIT_MONEY;
                                                  amount := #1.value;
                                                  accountNumber := #6.value;  }
    …
END RULES
…
VARIABLES
    ….
    teleBankOptions option
    {
            <- <teleBankOption>.option
    };
    …
END VARIABLES
    …
```

Figure 5

The code in the example in figure 5 is actually taken from three different files. The partitioning is done mainly to be able to reuse the code.

Figure 5 contains three different parts, lets explain each of the parts. The first part defines a new type (enumeration). In the second part syntactic and semantic rules for speech understanding are defined. We can see here two different definitions the first is a terminal rule and the second is non terminal. The second definition defines three attributes: option, accountNumber and amount, for each of the attributes a semantic rule is also defined. In the last part, the variable *option* is defined, this definition creates a link between the teleBankOption rule (its option attribute) and the variable.

If a variable is linked to a rule and the rule is in the currently active scope, the user’s answer to the questions will be ‘understood’ using (also) this rule. The attributes of this rule will be ‘filled’ according to the semantics and the variable will get a value.

- The example in figure 6 will show how a dialog is organized using condition actions. The different states of variables will also be explained. Thus the interface between speech understanding and dialog control will be clear.

```
…
ACTIONS
    ….
    STOP_COND("closeBankAccount.hdl"); // PRIORITY > MULTIPLE
    ….
    COND ( UNDEFINED(option) ) // default priotiry.
    {
            …
    }
    ….
    COND ( MyVerifiable(option); &&
             MyUndefined(accountNumber);  )   PRIORITY < UNDEFINED
            {
                    ….
                    QUESTION(option,accountNumber)
                    {
                            …
                    }
            }
    …
    COND ( MyVerifiable(option); &&
             MyVerifiable(accountNumber);  )  PRIORITY < UNDEFINED
            {
                    ….
                    QUESTION(YES,NO,option,accountNumber)
                    {
                            ….
                    }
            }
    …
END ACTIONS;
```

Figure 6

The dialog flow is implemented using conditional actions; i.e. the COND statements in the ACTIONS section. After every stage in the dialog, for example: after the user responds, the system traverses over the cond-actions until one of the conditions evaluate to TRUE. When the condition evaluates to TRUE the cond-action body is executed. In the cond-action a new question may be asked or some operation on a database might be executed.

Every cond-action has a priority attached to it; the designer can change this priority. This allows the designer to specify an order in which the cond-actions will be traversed. After the cond-action is executed the system traverses again over the cond-actions from the cond-action with the highest priority too the lowest. The designer must make sure that a cond-action will not be executed twice in a row. (Otherwise an endless loop will be created)

A question presented to the user might be in order to verify information known or in order to obtain new information from the user. As explained above this information is stored in HDDL variables. To allow the designer to distinguish between verified information and information yet to be verified each HDDL variable has status flags. Following are the status flags:

1. Undefined – the variable has no value.
2. Defined – the variable has one value (SINGLE), or the variable has more than one value (MULTIPLE).
3. Verified – the variable value is verified.

The status of a variable is set automatically by the system when VERIFICATION and CONFIRMATION macros are used. (The designer is also able to set the status manually).

Since TeleBank does not confirm explicitly each variable, but continues directly to the next question (verification), a variable might be verified although the user intentions where different. When this occurs the only cond-actions that will be activated are the cond-actions which conditions satisfy the fact that the variable is verified. This means that the dialog continues with this variable and its information as verified. This might lead to a situation in which the user will complete a transaction that he does not desire. Therefore a 'reset' option is implemented in TeleBank. This allows the user to reset the system in any phase of any transaction by just saying 'stop'.

The STOP_COND statement in figure 6 implements part of the 'reset' capability. It is actually a macro that defines two conned-actions. These actions have a very high priority and will be executed immediately when the stop variable is defined or verified i.e. the user had said 'stop'.

- The last example will show how a sub dialog (module) is instantiated and activated. We will also explain why some variables were passed by reference, and how it was used in TeleBank

```
IMPORT DIALOG closeBankAccount(teleBankOptions &option,
                               INTEGER &accountNumber,
                               status  &dialogStatus,
                               BOOLEAN &stop);
….
DIALOG…
….
    USES
            closeBankAccount            closeBankAccountDialog;
            ….
    END USES;
    …
    COND ( SINGLE(option,option == CLOSE_BANK_ACCOUNT) )
            {
            ….
            ACTIVATE      closeBankAccountDialog
                                  (option,accountNumber,subDialogStatus,stop);
            CALL          closeBankAccountDialog;
            DEACTIVATE    closeBankAccountDialog;
            ….
            }
    ….
END DIALOG….
```

Figure 7

As explained before, the HDDL code is compromised out of dialogs (modules). Each module contains a part of a dialog flow. Partitioning the code into modules has numerous advantages among them is the ability to reuse code. The application starts executing in the main module, this module as any other can instantiate other modules. To instantiate a module it must declare it. The IMPORT statement in figure 7 shows how to declare a module that will be used. In the same figure the USES section shows how instances of modules are defined. This allows a module to be instantiated numerous times.

Each module can be in one of the following modes: ACTIVE, LISTEN and PASSIVE. These modes effect the speech understanding component. In TeleBank only the ACTIVE mode was used. When a dialog is activated and called, as is done in figure 7, the control is transferred to that dialog and it is then executed.

In figure 7, at the declaration of the closeBankAccount dialog (module), one can see that all parameters are passed by reference. When a variable is passed by reference to a sub dialog, the actual variable is accessed from the sub dialog. Thus, when the value or status of such variable is changed in the sub dialog it will also change in the calling dialog.

When considering the design of TeleBank we can notice the following things:

1. Questions asked in one dialog might be verified in others.
2. Answers given to a question in a dialog can be used in different dialogs.

This means that information obtained from the user is shared between TeleBank dialogs (modules).

The *option* variable, for example, is defined in applicationOption dialog (module). In that dialog the user is asked which option he would like. When the user answers, the *option* variable will be defined. At this point the applicationOption dialog transfers the control to a specific sub dialog, let us assume it is transferred to closeBankAccount dialog. Notice that at this stage the *option* variable is **not** verified.

When closeBankAccount dialog is executed it should verify the *option* and find out the account number that the user wants to close. The dialog asks the user for the account number he would like to close. The user is expected to answer in one of the two following ways: 1. If the user intended to close an account he will give the account number 2. If the user did not intend to close an account he might say ‘I don’t want to close an account I want to open an account’. In this case the control transfers back again to applicationOptions dialog and from there to another sub dialog. (The reader is referred to the code to see how the variable was linked in the sub dialogs Appendix B).

# 9. Testing

Since TeleBank application was not trained the test results can only give an idea about the performance. The training of the application can boost up the speech recognition. This in turn, will increase the performance of the system i.e. cooperativity and user friendliness of TeleBank.

## Example

Follow is a log of a few test cases. The log files were produced by the SpeechMania online system. The user original responses where added to this log (User: ).

<u>Test 1:</u>

```
INFO: *** START DIALOG ***
INFO: Caller id: unknown
INFO: Called id: unknown
INFO: goede middag
INFO: dit is telebank een telefonische dienstverlening van armo bank
INFO: kent u de systeem
User: ja
INFO: ja
INFO: welke optie wenst u
User: ik wil een bank rekening openen.
INFO: ik wilde bank rekening openen
INFO: wilt u een nieuwe bank rekening|openen
User: ja
INFO: HDDL transaction call: value of passed parameter accountNumber is undefined. It will be used as OUT parameter
INFO: ja
INFO: HDDL transaction call: value of passed parameter res is undefined. It will be used as OUT parameter
INFO: uw bank rekening is nu geopend.|u nieuwe bank rekening nummer is|een|nul|zes
INFO: wilt u nog iets doen
User: nee
INFO: nee
INFO: goeie dag
INFO: DISCONNECT
INFO: *** END DIALOG ***
```

### Test 2:

```
INFO: *** START DIALOG ***
INFO: Caller id: unknown
INFO: Called id: unknown
INFO: goede middag
INFO: dit is telebank een telefonische dienstverlening van armo bank
INFO: kent u de systeem
User: ja
INFO: NOTHING RECORDED
INFO:
INFO: het systeem heeft uw antwoord niet gehoord|probeer alstublieft nogmaals duidelijk antwoord te geven op de volgende vraag|kent u de systeem
User: ja
INFO: ja
INFO: welke optie wenst u
User: ik wil drie gulden storten op bank rekening een nul zes
INFO: ik wilde goede storten op bank rekening negen willen deze
INFO: welk bedrag wilt u|storten|op bank rekening|negen
User: nee rekening een nul zes en ik wil drie gulden storten
INFO: nee rekening een nul zes ik wil drie goede storten
INFO: wilt u het bedrag van|drie|gulden|storten|op bank rekening|een|nul|zes
User: ja
INFO: ja
INFO: u heeft|drie|guldens gestort|op bank rekening nummer|een|nul|zes
INFO: wilt u nog iets doen
User: nee
INFO: nee
INFO: goeie dag
INFO: DISCONNECT
INFO: *** END DIALOG ***
```

### Test 3

```
INFO: *** START DIALOG ***
INFO: Caller id: unknown
INFO: Called id: unknown
INFO: goede middag
INFO: dit is telebank een telefonische dienstverlening van armo bank
INFO: kent u de systeem
User: nee
INFO: NOTHING RECORDED
INFO:
INFO: het systeem heeft uw antwoord niet gehoord|probeer alstublieft nogmaals duidelijk antwoord te geven op de volgende vraag|kent u de systeem
User: nee
INFO: nee
INFO: dit is een automatische telefonische bank|u kunt de volgende transacties verrichten|u kunt een nieuwe prive-rekening openen|u kunt een prive-rekening sluiten|u kunt geld storten op een prive-rekening|u kunt geld afhalen van een prive-rekening|u kunt het saldo van een prive-rekening opvragen|u kunt deze informatie nogmals afluisteren|als u een transactie wilt stoppen kunt u stop zeggen
INFO: welke optie wenst u
User: wat is de saldo van bank rekening een nul zes
INFO: met willen hallo stand bank rekening een willen
```

```
INFO: wat is de bank rekening nummer waarvan u de|saldo|wilt weten
User: een nul zes
INFO: negen nul zes
INFO: wilt u de saldo van de bank rekening|negen|nul|zes|weten
User: nee een nul zes
INFO: nee negen nul zes
INFO: uw antwoord op de vraag is onduidelijk|wilt u nogmaals antwoord geven op de volgende vraag|wilt u de saldo van de bank rekening|negen|nul|zes|weten
User:nee
INFO: nee
INFO: over welke bank rekening wilt u informatie
User:een nul zes
INFO: negen nul zes
INFO: wilt u de saldo van de bank rekening|negen|nul|zes|weten
User:nee een nul zes
INFO: nee negen nul zes
INFO: uw antwoord op de vraag is onduidelijk|wilt u nogmaals antwoord geven op de volgende vraag|wilt u de saldo van de bank rekening|negen|nul|zes|weten
User: nee
INFO: nee
INFO: over welke bank rekening wilt u informatie
User:bank rekening een nul zes
INFO: drie drie twee negen nul zes
INFO: wilt u de saldo van de bank rekening|negen|nul|zes|weten
User:nee
INFO: nee
INFO: wilt u de saldo van de bank rekening|drie|drie|twee|weten
User: nee
INFO: nee
INFO: over welke bank rekening wilt u informatie
User: een nul zes
INFO: een nul zes
INFO: wilt u de saldo van de bank rekening|een|nul|zes|weten
User: ja
INFO: ja
INFO: uw bank rekening saldo is|drie|gulden
INFO: wilt u nog iets doen
User: ja
INFO: ja
INFO: welke optie wenst u
User: rekening nummer een nul zes sluiten alstublieft
INFO: rekening nummer een nul zes sluiten alstublieft
INFO: wilt u de bank rekening met nummer|een|nul|zes|sluiten
User: ja
INFO: ja
INFO: uw bank rekening|een|nul|zes|is gesloten
INFO: wilt u nog iets doen
User: nee
INFO: nee
INFO: goeie dag
INFO: DISCONNECT
INFO: *** END DIALOG ***
```

# Results

TeleBank was tested using the SpeechMania online system. The Tests were aimed to find out the following things:

1. Does the system work. I.e. did the user accomplish his intended transaction? (yes/no)
2. Did the system understand the user? ( 5 yes – 1 no)
3. Are the dialogs conducted with TeleBank user friendly? (5 friendly – 1 unfriendly)

The last question is of course hard to evaluate and is individual to each user. Its aim was to try to evaluate the Dialog structure implemented in the HDDL code.

The users were asked to test TeleBank. They where informed the implementation assumptions and were asked to call TeleBank and request the following in three separate sessions:

1. Open a bank account.
2. Deposit a small amount in that bank account.
3. Ask for the balance of the bank account and after that to close it.

After the test the user answered the above questions.

| Testers name | First Transaction | Second Transaction | Third Transaction | System understands? | User friendly dialogs? | Comments |
|---|---|---|---|---|---|---|
| Arnon | Yes | Second try | No | 3.5 | 3 | While working on the third transaction Arnon repeatedly tried to ask for balance using words which weren't in the lexicon. After the test, these words were added. |
| Dan | Yes | No | Yes | 3 | 5 | TeleBank did not understand the bank account number in the second transaction. |
| Vanessa | Yes | No | Yes | 3.5 | 4 | Vanessa first tried to close an account using words that weren't in my lexicon she then asked for help and used the words in the help message. |
| Amir | Yes | Yes | Yes | 5 | 5 | The system recognized everything Amir said |
| Nitzan | Yes | Yes | Yes | 5 | 4 | The system recognized most of what Nitzan said |
| Daphna | Second try | Yes | Yes | 4 | 4 | |
| Freddy | Yes | Yes | Yes | 4 | 3 | Freddy did not try to close the account. |

Most users used simple and short sentences, this occurred especially towards the last sessions. Many of the users stated in the beginning of the test that they knew the system thus, they were not informed of the 'stop' option.

# 10. Conclusions and Further research

## Conclusions

The TeleBank application is a prototype that can give using the telephone, fully automated bank services. The prototype was designed and implemented using the SpeechMania System. It was tested using the online system.

The approach used in TeleBank proved to be successful. Semi ‘closed’ questions enabled the users to complete their transaction fast. The users did rarely take any initiative since they believed TeleBank would not understand them. A better understanding would no doubt cause the users to take more initiative.

SpeechMania is a suitable framework for a designer to create automatic inquiry systems. During the design of TeleBank effort was made to reuse code. This was done using the HDDL macros. Real functions and maybe dialog templates could increase the reuse of code and make it easier.

## Further research

The functionality that is implemented in TeleBank is a ‘simplified’ version of the true situation. Much of the functionality needed by a real bank including all the security issues is missing.

The application should be trained and should be able to recognize different accents. The vocabulary should be enlarged and enriched using a corpus of human-human dialogs. This will improve the understanding of TeleBank.

Additional ‘parallel’ flows should be added. These should deal with different problems that may occur in the main flow (SIBLING dialogs).

# 11. References

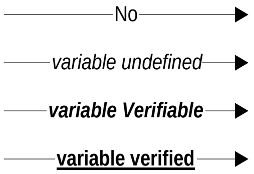
- Philips Speech Processing Aachen 1996-1999: *SpeechMania Application Creation Environment, Version 2.2, Developer’s Guide*. Aachen, Germany, January 1999.
- Philips Speech Processing Aachen 1996-1999: *HDDL Dialog Description Language, Version 2.2, User’s Guide*. Aachen, Germany, February 1999.
- Niels Ole Bernsen, Hans Dybkjaer and Laila Dybkjaer: *Designing Interactive Speech Systems: From First Ideas To User Testing.* Springer-Verlag London Limited 1998.

# Appendix A – Dialog flow chart diagrams.

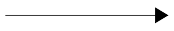
Following are the flow chats of the dialogs in TeleBank.

# The following conventions where used in the flow charts.

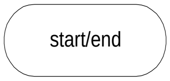
Edge with attribute,
The following state will be entered if the attribute is true



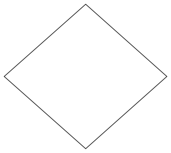
Edge without attribute,
The following state will be entered if all other
edges (if exist) are false



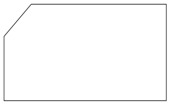
First/last state in the automate



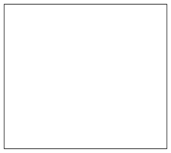
Yes or No question.



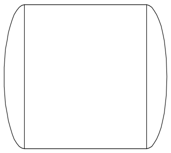
all other questions



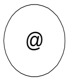
system output

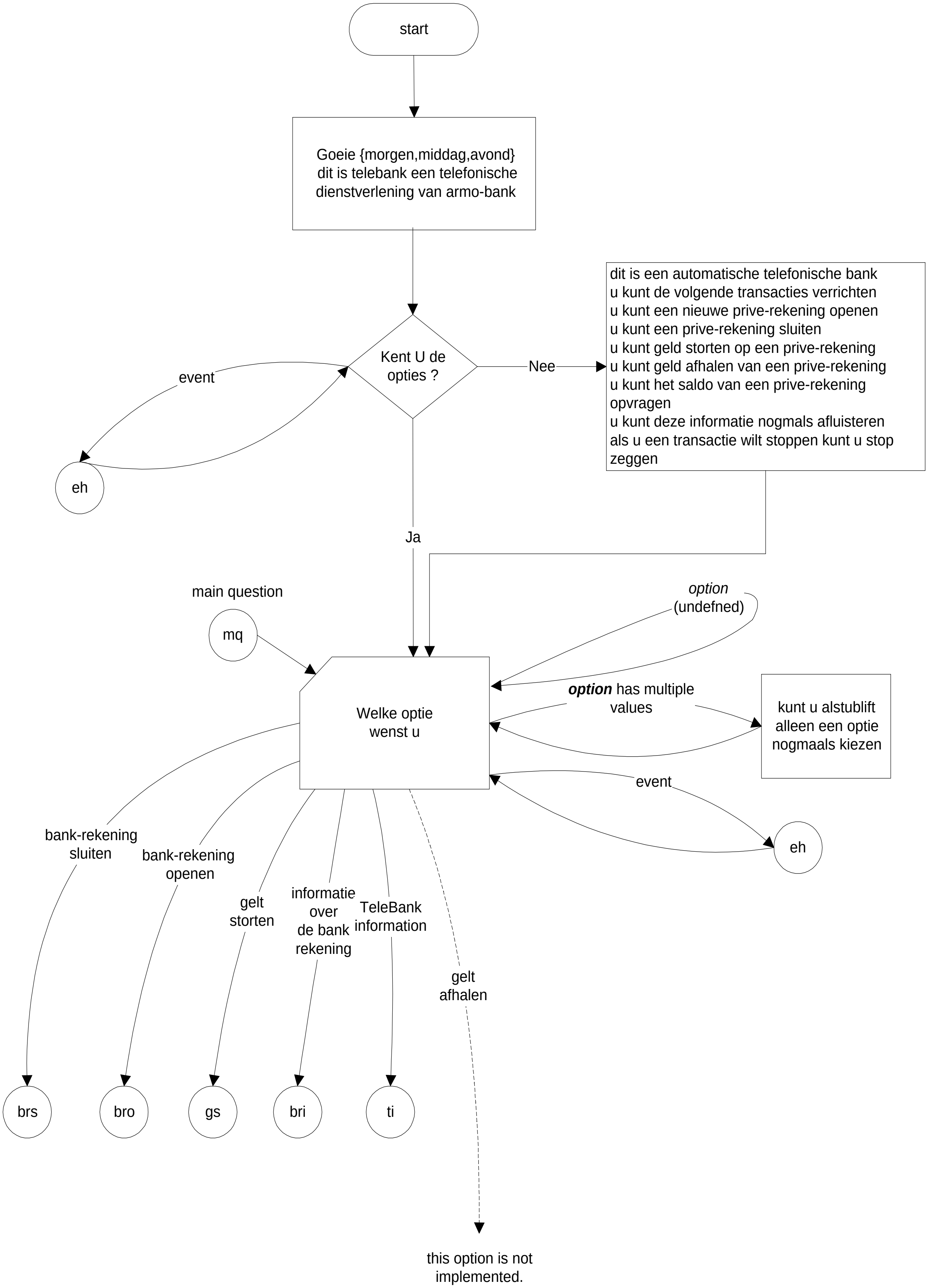


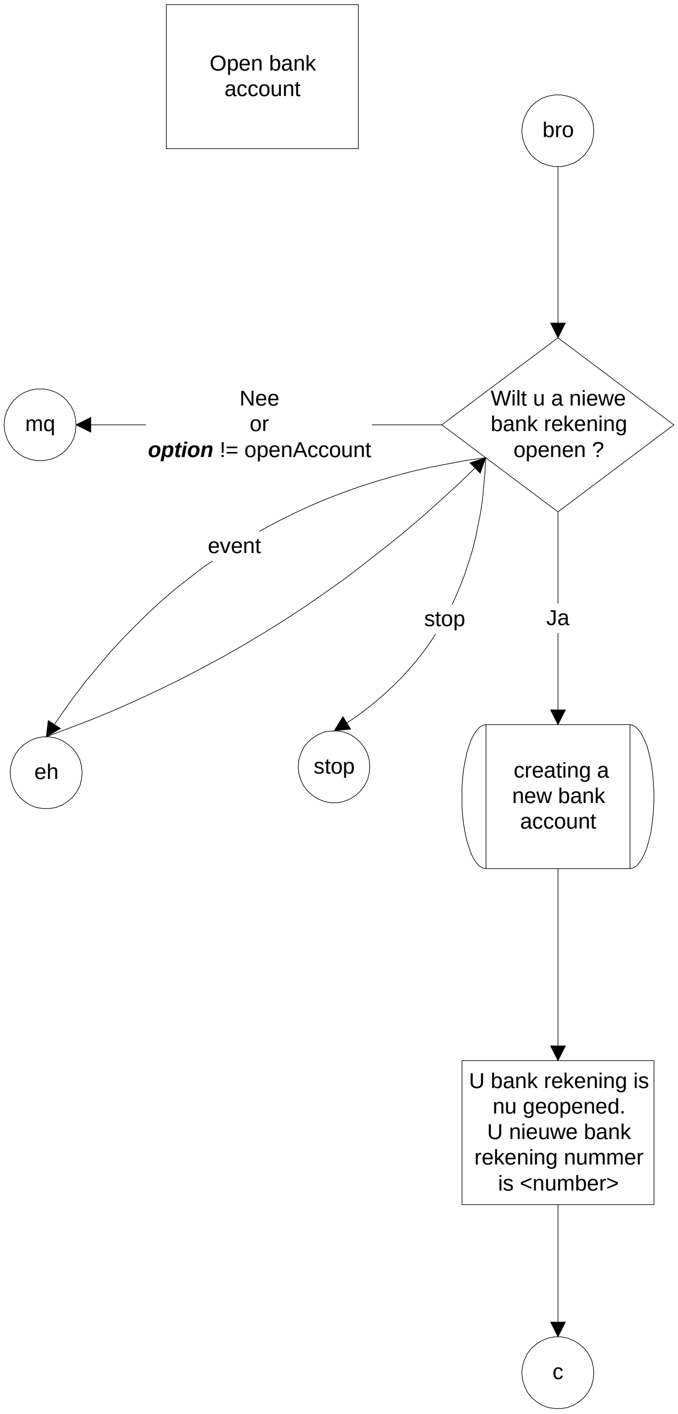
system internal action
using data base.

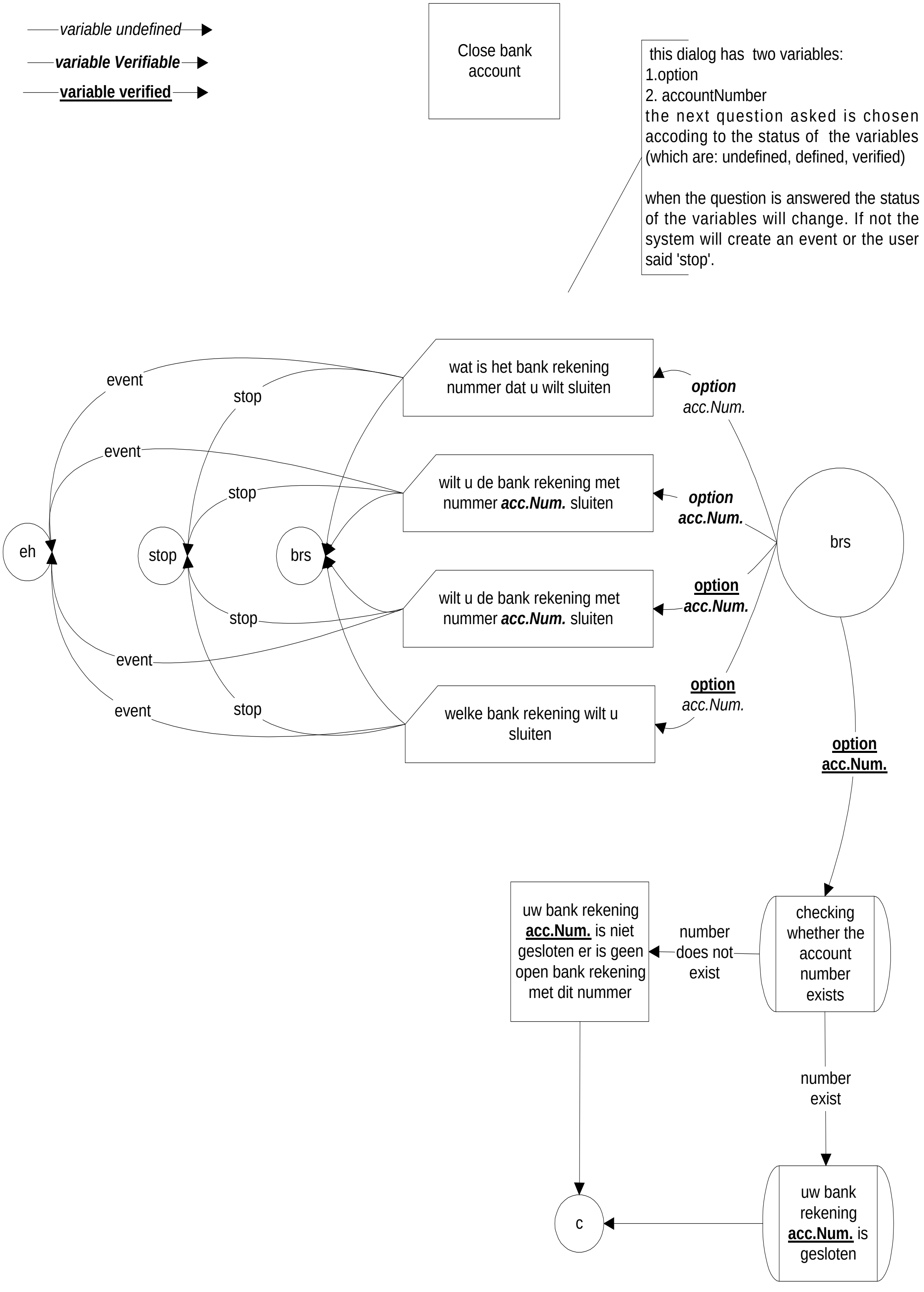


connector



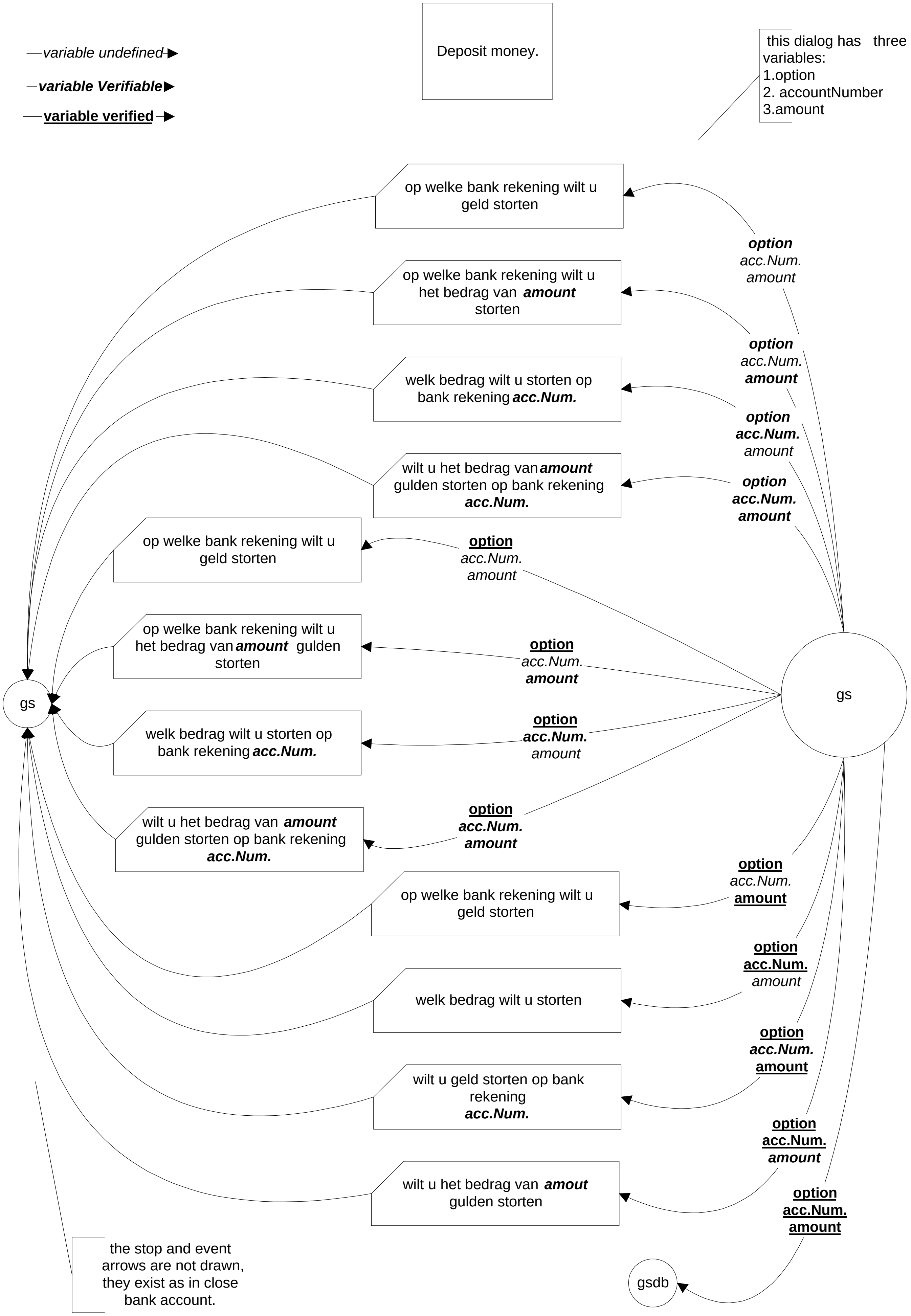

start
Goeie {morgen,middag,avond}
dit is telebank een telefonische
dienstverlening van armo-bank
Kent U de
opties ?
Nee
dit is een automatische telefonische bank
u kunt de volgende transacties verrichten
u kunt een nieuwe prive-rekening openen
u kunt een prive-rekening sluiten
u kunt geld storten op een prive-rekening
u kunt geld afhalen van een prive-rekening
u kunt het saldo van een prive-rekening
opvragen
u kunt deze informatie nogmaals afluisteren
als u een transactie wilt stoppen kunt u stop
zeggen
event
eh
Ja
main question
mq
*option*
(undefned)
Welke optie
wenst u
***option*** has multiple
values
kunt u alstublift
alleen een optie
nogmaals kiezen
event
eh
bank-rekening
sluiten
bank-rekening
openen
gelt
storten
informatie
over
de bank
rekening
TeleBank
information
gelt
afhalen
brs
bro
gs
bri
ti
this option is not
implemented.


Open bank account
bro
Wilt u a niewe bank rekening openen ?
Nee
or
**option** != openAccount
mq
event
stop
Ja
eh
stop
creating a new bank account
U bank rekening is nu geopened.
U nieuwe bank rekening nummer is <number>
c

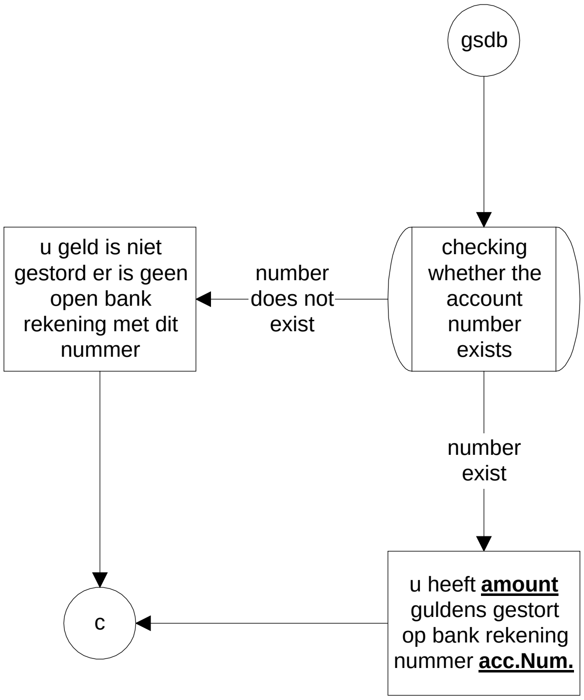

variable undefined
**variable Verifiable**
**variable verified**
Close bank account
event
stop
wat is het bank rekening nummer dat u wilt sluiten
option
acc.Num.
event
wilt u de bank rekening met nummer acc.Num. sluiten
stop
option
acc.Num.
eh
stop
brs
brs
stop
wilt u de bank rekening met nummer acc.Num. sluiten
option
acc.Num.
event
event
stop
welke bank rekening wilt u sluiten
option
acc.Num.
option
acc.Num.
uw bank rekening acc.Num. is niet gesloten er is geen open bank rekening met dit nummer
number does not exist
checking whether the account number exists
number exist
c
uw bank rekening acc.Num. is gesloten

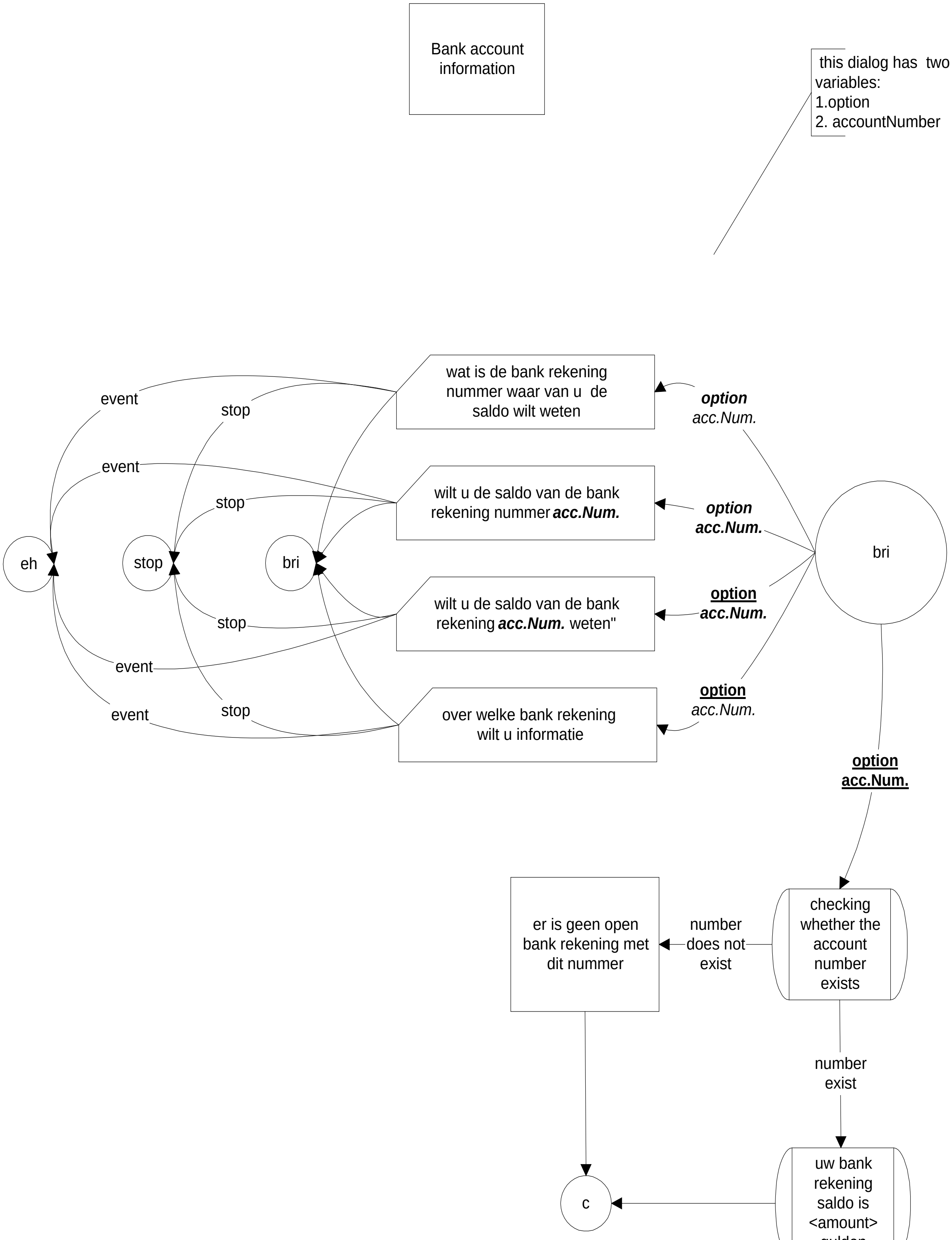
this dialog has two variables:
1.option
2. accountNumber
the next question asked is chosen accoding to the status of the variables (which are: undefined, defined, verified)

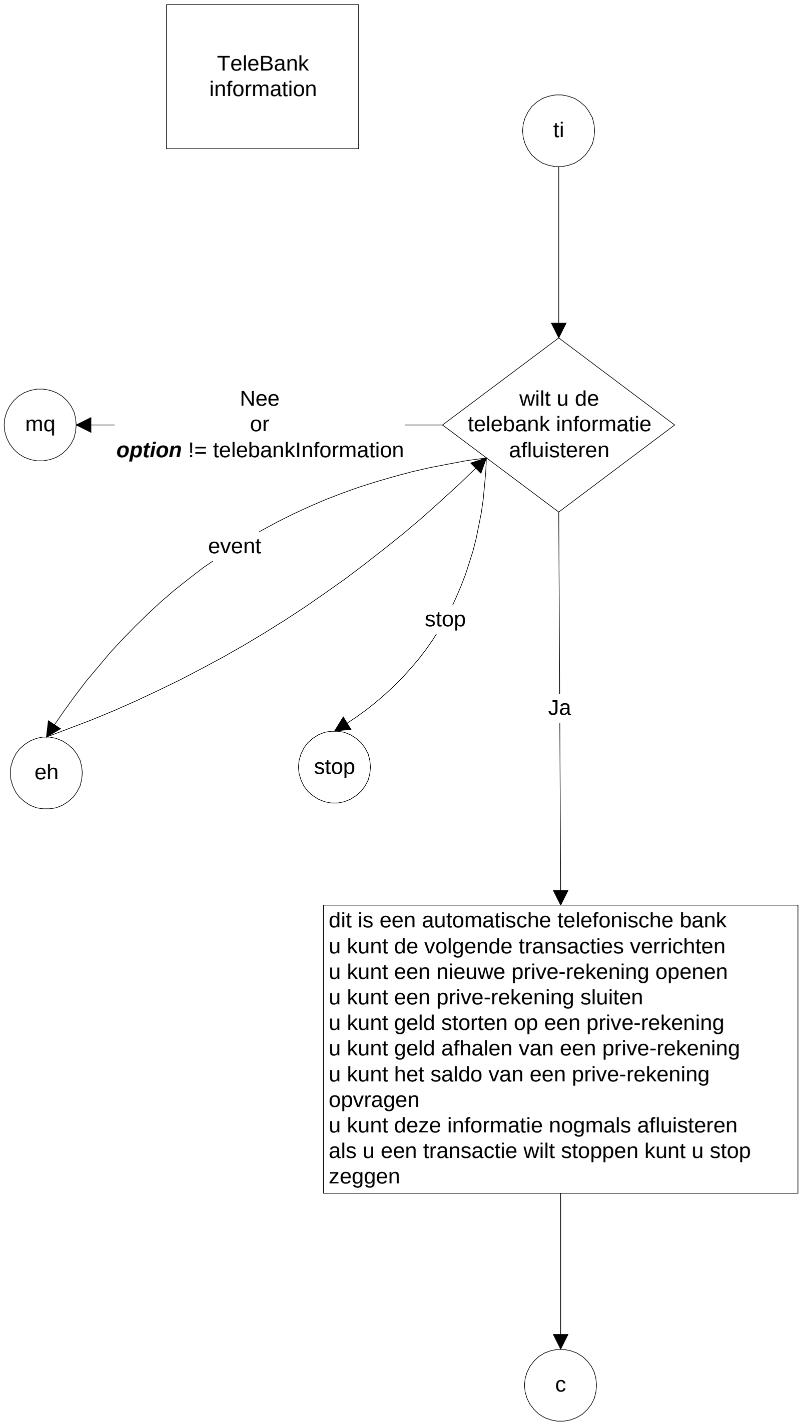
when the question is answered the status of the variables will change. If not the system will create an event or the user said 'stop'.


variable undefined
variable Verifiable
variable verified
Deposit money.
this dialog has three variables:
1.option
2. accountNumber
3.amount
op welke bank rekening wilt u geld storten
option
acc.Num.
amount
op welke bank rekening wilt u het bedrag van amount storten
option
acc.Num.
amount
welk bedrag wilt u storten op bank rekening acc.Num.
option
acc.Num.
amount
wilt u het bedrag van amount gulden storten op bank rekening acc.Num.
option
acc.Num.
amount
op welke bank rekening wilt u geld storten
option
acc.Num.
amount
op welke bank rekening wilt u het bedrag van amount gulden storten
option
acc.Num.
amount
gs
gs
welk bedrag wilt u storten op bank rekening acc.Num.
option
acc.Num.
amount
wilt u het bedrag van amount gulden storten op bank rekening acc.Num.
option
acc.Num.
amount
op welke bank rekening wilt u geld storten
option
acc.Num.
amount
welk bedrag wilt u storten
option
acc.Num.
amount
wilt u geld storten op bank rekening acc.Num.
option
acc.Num.
amount
wilt u het bedrag van amout gulden storten
option
acc.Num.
amount
option
acc.Num.
amount
the stop and event arrows are not drawn, they exist as in close bank account.
gsdb

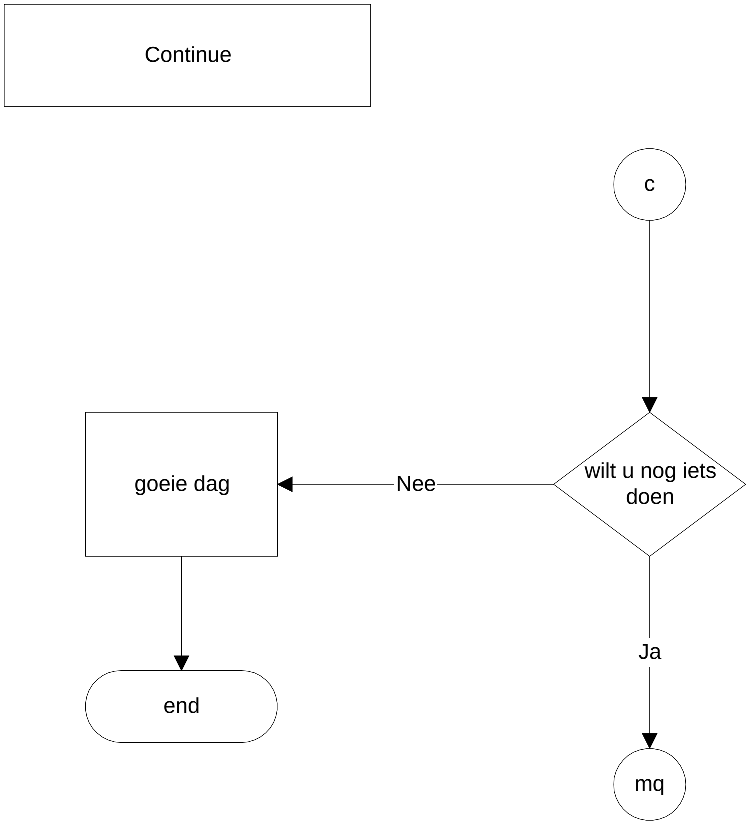
Deposit money - continue.

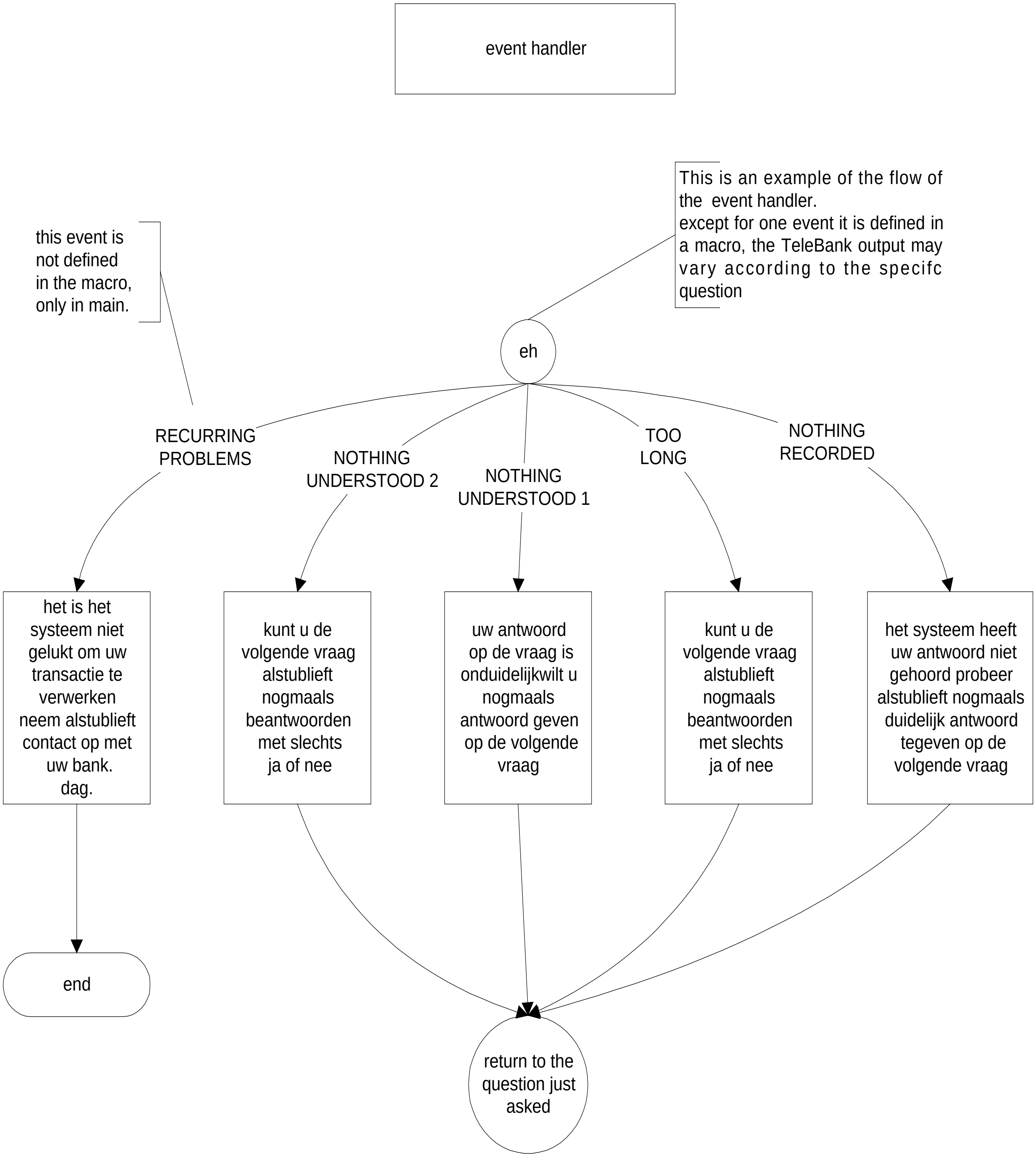

gsdb
checking whether the account number exists
number does not exist
u geld is niet gestord er is geen open bank rekening met dit nummer
number exist
u heeft **amount** guldens gestort op bank rekening nummer **acc.Num.**
c

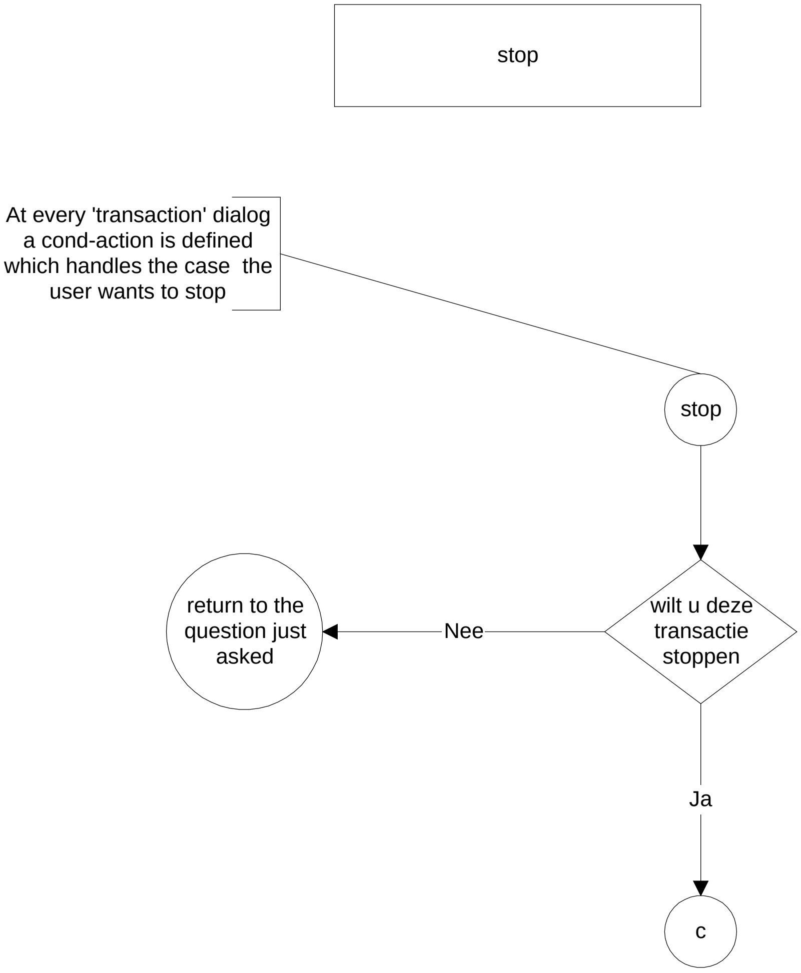

Bank account information
this dialog has two variables:
1.option
2. accountNumber
wat is de bank rekening nummer waar van u de saldo wilt weten
option
acc.Num.
wilt u de saldo van de bank rekening nummer acc.Num.
option
acc.Num.
wilt u de saldo van de bank rekening acc.Num. weten"
option
acc.Num.
over welke bank rekening wilt u informatie
option
acc.Num.
event
stop
event
stop
eh
stop
bri
stop
event
event
stop
bri
option
acc.Num.
checking whether the account number exists
number does not exist
er is geen open bank rekening met dit nummer
number exist
uw bank rekening saldo is <amount> gulden
c


TeleBank information
ti
wilt u de telebank informatie afluisteren
Nee or ***option*** != telebankInformation
mq
event
stop
Ja
eh
stop
c

dit is een automatische telefonische bank
u kunt de volgende transacties verrichten
u kunt een nieuwe prive-rekening openen
u kunt een prive-rekening sluiten
u kunt geld storten op een prive-rekening
u kunt geld afhalen van een prive-rekening
u kunt het saldo van een prive-rekening opvragen
u kunt deze informatie nogmals afluisteren
als u een transactie wilt stoppen kunt u stop zeggen


Continue
c
wilt u nog iets doen
Nee
goeie dag
Ja
end
mq


event handler
This is an example of the flow of the event handler.
except for one event it is defined in a macro, the TeleBank output may vary according to the specifc question
this event is not defined in the macro, only in main.
eh
RECURRING PROBLEMS
NOTHING UNDERSTOOD 2
NOTHING UNDERSTOOD 1
TOO LONG
NOTHING RECORDED
het is het systeem niet gelukt om uw transactie te verwerken neem alstublieft contact op met uw bank. dag.
kunt u de volgende vraag alstublieft nogmaals beantwoorden met slechts ja of nee
uw antwoord op de vraag is onduidelijkwilt u nogmaals antwoord geven op de volgende vraag
kunt u de volgende vraag alstublieft nogmaals beantwoorden met slechts ja of nee
het systeem heeft uw antwoord niet gehoord probeer alstublieft nogmaals duidelijk antwoord tegeven op de volgende vraag
end
return to the question just asked


stop
At every 'transaction' dialog a cond-action is defined which handles the case the user wants to stop
stop
wilt u deze transactie stoppen
Nee
return to the question just asked
Ja
c

# Appendix B – HDDL code

Following are the TeleBank modules as implemented in HDDL code.

```
// main.hdl


IMPORT DIALOG welcome();
IMPORT DIALOG yesOrNoQuestion(STRING question,BOOLEAN& answer);
IMPORT DIALOG applicationOptions();

INCLUDE "declarations.hdl";


DIALOG MAIN

        SETTINGS

                 /// this does not work as expected
                RECURRING_PROBLEMS PROBLEM_DETECTOR.REPETITION > 3;

        END SETTINGS;

        RULES
        END RULES;

        PREFERENCES
        END PREFERENCES;

        VARIABLES
                BOOLEAN userKnowsTheSystem  TEMPORARY;
                BOOLEAN userWantsToContinue TEMPORARY;
        END VARIABLES;

        KNOWLEDGE
        END KNOWLEDGE;

        EVENT HANDLER


                NOTHING_RECORDED
                {
                IF ( DEBUG ) {   "main.hdl: event handler NOTHING_RECORDED."; }
                }
                TOO_LONG
                {
                        IF ( DEBUG ) { "main.hdl: event handler TOO_LONG."; }
                }
                NOTHING_UNDERSTOOD
                {
                        IF ( DEBUG ) { "main.hdl: event handler NOTHING_UNDERSTOOD."; }
                }
                // all recurring problems events are delt here.
```

```
    // here the system is exiting.
    RECURRING_PROBLEMS
    {
        IF ( DEBUG ) {
            "main.hdl: event handler RECURRING_PROBLEMS.";
        }

        "het is het systeem niet gelukt om uw transactie te verwerken";
        "neem alstublieft contact op met uw bank";
        "dag";
        BYE;
    }

END EVENT HANDLER;

VERIFICATION
END VERIFICATION;

USES
    welcome          welcomeDialog;
    yesOrNoQuestion   doesTheUserKnowTheSystemQuestion;
    yesOrNoQuestion   doesTheUserWantToContinue;
    applicationOptions applicationOptionsDialog;
END USES;

TRANSACTIONS
    // one transaction instance. this instance will be used throughout
    // the application.
    teleBankDB teleBankDBinstance EXPORT;

END TRANSACTIONS;


ACTIONS

    COND(ONCE)
    {

  LANGUAGE <- "TeleBank"; // language resource name. TeleBank.lex

        ACTIVATE   welcomeDialog();  // activating the welcome module.
        CALL       welcomeDialog;
        DEACTIVATE welcomeDialog;

        IF ( DEBUG ) {   teleBankDBinstance.printAccounts(); }


        // userKnowsTheSystem variable is not defined, it is passed
        // by reference.
        ACTIVATE   doesTheUserKnowTheSystemQuestion
                            ("kent u de systeem ",userKnowsTheSystem);
        CALL       doesTheUserKnowTheSystemQuestion;
        DEACTIVATE doesTheUserKnowTheSystemQuestion;

        IF ( userKnowsTheSystem == FALSE )
        {
```

```
                    PrintTeleBankInformation
                }

                userWantsToContinue := TRUE;

                WHILE ( userWantsToContinue )
                {
                    ACTIVATE   applicationOptionsDialog();
                    CALL       applicationOptionsDialog;
                    DEACTIVATE applicationOptionsDialog;


                    ACTIVATE   doesTheUserWantToContinue
                                    ("wilt u nog iets doen",userWantsToContinue);
                    CALL       doesTheUserWantToContinue;
                    DEACTIVATE doesTheUserWantToContinue;
                }

                OUTPUT
                {
                    "goeie dag";
                }

            }

        END ACTIONS;

END DIALOG MAIN.
```

```
// applicationOptions.hdl

INCLUDE "declarations.hdl";


IMPORT DIALOG openBankAccount(teleBankOptions &option,
                                        status  &subDialogStatus,
                                                INTEGER &newAccountNumber,
                                                BOOLEAN &stop);

IMPORT DIALOG closeBankAccount(teleBankOptions &option,
                                            INTEGER &accountNumber,
                                            status  &dialogStatus,
                                                BOOLEAN &stop);

IMPORT DIALOG depositMoney(teleBankOptions &option,
                                        INTEGER &accountNumber,
                                        INTEGER &amount,
                                        status  &dialogStatus,
                                        BOOLEAN &stop);

IMPORT DIALOG bankAccountInformation(teleBankOptions &option,
                                                INTEGER &accountNumber,
                                                                                    status  &dialogStatus,
                                                                                          BOOLEAN &stop);

IMPORT DIALOG teleBankInformation(teleBankOptions &option,
                                                                                  status  &dialogStatus,
                                                                                  BOOLEAN &stop);



/* input:   none
 * output:  none
 * this dialog asks the user for the option and calles the appropriate dialog.
 */
DIALOG applicationOptions()

        RULES
                INCLUDE "teleBankOptionsRules.rhdl";
        END RULES;

        PREFERENCES
        END PREFERENCES;

        VARIABLES

                BOOLEAN stop;

                teleBankOptions option
                {
                        <- <teleBankOption>.option
                };

                INTEGER accountNumber
                {
                        <- <teleBankOption>.accountNumber
                };
                INTEGER amount
                {
                        <- <teleBankOption>.amount
                };

                MEANINGFUL_FILLER
                {
                        <- <greeting>,
                        <- <please>,
                        <- <wantStatement>,
                        <- <mine>,
                        <- <meaningfulFiller>
                };


                INTEGER newAccountNumber; // used in openBankAccount, must be declared here.
                status  subDialogStatus;  // used when calling a subdialog.


        END VARIABLES;
        KNOWLEDGE
        END KNOWLEDGE;

        EVENT HANDLER
        END EVENT HANDLER;

        VERIFICATION
        END VERIFICATION;

        USES
                openBankAccount                 openBankAccountDialog;
                closeBankAccount            closeBankAccountDialog;
                depositMoney                depositMoneyDialog;
                bankAccountInformation   bankAccountInformationDialog;
                teleBankInformation         teleBankInformationDialog;
        END USES;

        TRANSACTIONS
        END TRANSACTIONS;


        ACTIONS

                COND( ^^ stop ) PRIORITY > MULTIPLE
                {
                        IF (DEBUG) { OUTPUT { "applicationOptions.hdl: stop is verified";   }}
                        RETURN;
                }


                COND ( MULTIPLE(option) )  // very simple treatment.
                {
                        IF (DEBUG) { OUTPUT { "applicationOptions.hdl: option variable has
    multiple entries in the status graph.(debug)"; }}

                        CLEAR option; // clears the variable.

                        QUESTION(option)
                        {
                                INIT
                                {
                                        "kunt u alstublift alleen een optie nogmaals kiezen";
                                }

                                MyEvents( NUstatements;
                 ,
                                                "als u hulp nodig hebt kunt u de hulp optie kiezen";
                                                 "u kunt help zeggen";
          ,
                                                NRstatements;
                         ,
                                                "u kunt alleen een van de opties kiezen ";  ,
                                                "welke optie wenst u";
                                                        IF (DEBUG) { "applicationOptions"; }
                                               );
                        }
        }




        COND ( SINGLE(option,option == OPEN_BANK_ACCOUNT) )
        {
                IF (DEBUG) { OUTPUT { "applicationOptions.hdl: SINGLE option is
OPEN_BANK_ACCOUNT";     }}

                ACTIVATE    openBankAccountDialog

    (option,subDialogStatus,newAccountNumber,stop);
                CALL        openBankAccountDialog;
                DEACTIVATE  openBankAccountDialog;

                IF ( subDialogStatus == OK )
                {
                    IF (DEBUG) { OUTPUT { "subDialogStatus OK"; }}
                    RETURN;
                }
                ELSE
                IF ( subDialogStatus == TRANSACTION_ERROR )
                {
                    IF (DEBUG) { OUTPUT { "subDialogStatus ==
TRANSACTION_ERROR "; }}
                    RETURN;
                }

        }

        COND ( SINGLE(option,option == CLOSE_BANK_ACCOUNT) )
        {
                IF (DEBUG) { OUTPUT { "applicationOptions.hdl: SINGLE option is
CLOSE_BANK_ACCOUNT";    }}

                ACTIVATE    closeBankAccountDialog

    (option,accountNumber,subDialogStatus,stop);
                CALL        closeBankAccountDialog;
                DEACTIVATE  closeBankAccountDialog;


                IF ( subDialogStatus == OK )
                {
                    IF (DEBUG) { OUTPUT { "subDialogStatus  OK"; }}
                    RETURN;
                }
                ELSE IF ( subDialogStatus == TRANSACTION_ERROR )
                {
                    IF (DEBUG) { OUTPUT { "subDialogStatus ==
TRANSACTION_ERROR "; }}
                    RETURN;
                }
        }



        COND ( SINGLE(option,option == DEPOSIT_MONEY ) )
        {
                IF (DEBUG) { OUTPUT { "applicationOptions.hdl: SINGLE option is
DEPOSIT_MONEY ";      }}

                ACTIVATE        depositMoneyDialog

    (option,accountNumber,amount,subDialogStatus,stop);
                CALL            depositMoneyDialog;
                DEACTIVATE  depositMoneyDialog;

                IF ( subDialogStatus == OK )
                {
                    IF (DEBUG) { OUTPUT { "subDialogStatus  OK"; }}
                    RETURN;
                }
                ELSE IF ( subDialogStatus == TRANSACTION_ERROR )
                {
                    IF (DEBUG) { OUTPUT { "subDialogStatus ==
TRANSACTION_ERROR "; }}
                    RETURN;
                }
        }


        COND ( SINGLE(option,option == WITHDRAW_MONEY ) )
        {
                IF (DEBUG) { OUTPUT { "applicationOptions.hdl: SINGLE option is
WITHDRAW_MONEY ";}}

                CLEAR option; // clears the variable.

                OUTPUT { "de optie"; "geld afhalen"; "is niet geimplementeerd"; }
        }


        COND ( SINGLE(option,option == BANK_ACCOUNT_INFORMATION ) )
        {
                IF (DEBUG) { OUTPUT { "applicationOptions.hdl: SINGLE option is
BANK_ACCOUNT_INFORMATION";      }}

                ACTIVATE        bankAccountInformationDialog

    (option,accountNumber,subDialogStatus,stop);
                CALL            bankAccountInformationDialog;
                DEACTIVATE  bankAccountInformationDialog;

                IF ( subDialogStatus == OK )
                {
                    IF (DEBUG) { OUTPUT { "subDialogStatus  OK"; }}
                    RETURN;
                }
                ELSE IF ( subDialogStatus == TRANSACTION_ERROR )
                {
                    IF (DEBUG) { OUTPUT { "subDialogStatus ==
TRANSACTION_ERROR "; }}
                    RETURN;
                }
        }




        COND ( SINGLE(option,option == TELEBANK_INFORMATION) )
        {
                IF (DEBUG) { OUTPUT { "applicationOptions.hdl: SINGLE option is
TELEBANK_INFORMATION";  }}

                ACTIVATE
    teleBankInformationDialog(option,subDialogStatus,stop);
                CALL            teleBankInformationDialog;
                DEACTIVATE  teleBankInformationDialog;

                RETURN;
        }



        COND( UNDEFINED(option) )
        {
            QUESTION(option)
            {
                INIT
                {
                    "welke optie wenst u";
                }

                MyEvents( NUstatements;
         ,
                        "als u hulp nodig hebt kunt u de hulp optie kiezen";
                            "u kunt help zeggen";
         ,
                        NRstatements;
             ,
                        "uw kunt alleen een van de opties kiezen ";  ,
                            "welke optie wenst u";
                            IF (DEBUG) { "applicationOptions"; }
                       );

            }
        }


    END ACTIONS;

END DIALOG applicationOptions.
```

```
// openBankAccount.hdl

INCLUDE "declarations.hdl";


/* input:  option        - the option that was choosen, so we can verify
 *                       if needed that OPEN_BANK_ACCOUNT was chosen.
 *                       option must be passed by reference if we
 *                       want to ask if it is verified.
 * output: dialogStatus   - dialogStatus the return status of this dialog
 *                                                       will be OK if user opened an account,
 *                              TRANSACTION_ERROR if there was a transaction error
 *                         newAccountNumber - an integer with the new bank account number.
 *                 stop          - a variable that will be defined and verified when
 *                                      the user wants to stop.
 */
DIALOG openBankAccount(teleBankOptions &option,
                                          status  &dialogStatus,
                                          INTEGER &newAccountNumber,
                                          BOOLEAN &stop)

        RULES
                INCLUDE "yesNoRules.rhdl";
                INCLUDE "teleBankOptionsRules.rhdl";
        END RULES;

        PREFERENCES
        END PREFERENCES;

        VARIABLES

                YES
                {
                        <- <yes>
                };

                NO
                {
                        <- <no>
                };

                PARAMETER stop
                {
                        <- <stop_rule>
                };

                PARAMETER option
                {
                        <- <teleBankOption>.option
                };

                MEANINGFUL_FILLER
                {
                        <- <greeting>,
                        <- <please>,
                        <- <wantStatement>,
            <- <mine>,
            <- <meaningfulFiller>
        };

    END VARIABLES;

    KNOWLEDGE
    END KNOWLEDGE;

    EVENT HANDLER
    END EVENT HANDLER;

    VERIFICATION

        stop
        {
        }

        option
        {
            OPTION_CONVERSIONS(option);
        }

    END VERIFICATION;

    USES
    END USES;

    TRANSACTIONS
        teleBankDB teleBankDBinstance IMPORT;
    END TRANSACTIONS;

    ACTIONS


        STOP_COND("openBankAccount.hdl"); // PRIORITY > MULTIPLE


        // this will be delt with in applicationOptions
        COND ( MULTIPLE(option) )
        {
            IF (DEBUG) { OUTPUT { "openBankAccount.hdl: option has multiple values
return; "; }}
            RETURN;
        }
        COND ( SINGLE(option,option != OPEN_BANK_ACCOUNT) )
        {
            IF (DEBUG) { OUTPUT { "openBankAccount.hdl: option has single value Not
OPEN_BANK_ACCOUNT, return; "; }}
            RETURN;
        }
        COND ( UNDEFINED(option) )
        {
            IF (DEBUG) { OUTPUT { "OpenBankAccount.hdl: option undefined, return; ";
}}
            RETURN;
```

```
}



// option
//   verifiable   -> verify
//   verified     -> do it.


// option:        verifiable
COND ( MyVerifiable(option); )
{
        IF (DEBUG) { OUTPUT { "openBankAccount.hdl: option: verifiable" ; }}

        QUESTION(YES,NO,option)
        {
                INIT
                {
                        "wilt u een nieuwe bank rekening";
                        CONFIRMATION(option);
                }

                MyEvents( NUstatements;  ,
                                YNstatements;  ,
                                NRstatements;  ,

                                YNstatements;  ,
                                        "wilt u een nieuwe bank rekening";
                                        CONFIRMATION(option);
                              );
        }
}


COND ( ^^ option && option == OPEN_BANK_ACCOUNT )
{
        BOOLEAN res;
        STRING C; STRING X; STRING I;

        teleBankDBinstance.openAccount(newAccountNumber,res);

        DIGIT_2_CONV_STRING_1(C,X,I,newAccountNumber);

        IF ( res )        {
                dialogStatus := OK;
                OUTPUT { "uw bank rekening is nu geopend. ";
                                "u nieuwe bank rekening nummer is";
                                DIGIT_2_CONV_STRING_2(C,X,I);
                        }
        }
        ELSE {
                dialogStatus := TRANSACTION_ERROR;
                OUTPUT { "uw bank rekening is niet geopend, systeem fout"; }
        }

        RETURN; // terminating the subdialog.
```

```
                }


        END ACTIONS;


END DIALOG openBankAccount.






// closeBankAccount.hdl

INCLUDE "declarations.hdl";


/* input: option         - the option that was choosen, so we can verify
 *                   if needed that CLOSE_BANK_ACCOUNT was chosen.
 *                   option must be passed by reference if we
 *                   want to ask if it is verified.
 *                     accountNumber  - an integer, will be defined if user said allready
 *                   a number. ( might not be defined ).
 *                   it is passed by reference so we can ask if it
 *                   is verified.
 * output: dialogStatus  - dialogStatus the return status of this dialog
 *                                                   will be OK if user closed an account,
 *                   TRANSACTION_ERROR if there was a transaction error
 *       stop        - a variable that will be defined and verified when
 *                   the user wants to stop.
 */
DIALOG closeBankAccount(teleBankOptions &option,
                                        INTEGER &accountNumber,
                                        status  &dialogStatus,
                                              BOOLEAN &stop)

        RULES
                INCLUDE "yesNoRules.rhdl";
                INCLUDE "teleBankOptionsRules.rhdl";
        END RULES;


        PREFERENCES
        END PREFERENCES;

        VARIABLES


                YES
                {
                        <- <yes>
                };

                NO
                {
```

```
                <- <no>
        };

        PARAMETER stop
        {
                <- <stop_rule>
        };

        PARAMETER option
        {
                <- <teleBankOption>.option
        };

        PARAMETER accountNumber
        {
                <- <teleBankOption>.accountNumber,
                <- <number>.value
        };

        MEANINGFUL_FILLER
        {
                <- <greeting>,
                <- <please>,
                <- <wantStatement>,
                <- <mine>,
                <- <meaningfulFiller>
        };

END VARIABLES;

KNOWLEDGE
END KNOWLEDGE;

EVENT HANDLER
END EVENT HANDLER;

VERIFICATION

        stop
        {
        }

        accountNumber + option
        {
                DIGIT_CONVERSIONS(accountNumber);
                OPTION_CONVERSIONS(option);
        }

        accountNumber
        {
                DIGIT_CONVERSIONS(accountNumber);
        }

        option
        {
                OPTION_CONVERSIONS(option);
        }


    END VERIFICATION;

    USES
    END USES;

    TRANSACTIONS
        teleBankDB teleBankDBinstance IMPORT;
    END TRANSACTIONS;

    ACTIONS

        STOP_COND("closeBankAccount.hdl"); // PRIORITY > MULTIPLE


        // the following event are delt in applicationOptions
        // (mq)
        COND ( MULTIPLE(option) )
        {
            IF (DEBUG) { OUTPUT { "closeBankAccount.hdl: option has multiple values
return; "; }}
            RETURN;
        }
        COND ( SINGLE(option,option != CLOSE_BANK_ACCOUNT) )
        {
            IF (DEBUG) { OUTPUT { "closeBankAccount.hdl: option has single value Not
CLOSE_BANK_ACCOUNT, return; "; }}
            RETURN;
        }
        COND ( UNDEFINED(option) )
        {
            IF (DEBUG) { OUTPUT { "closeBankAccount.hdl: option undefined return; ";
}}
            RETURN;
        }

        ////////////////////////////////////////////////////////////////////
        // there are four events defined each of them has a condition
        // dependent on both option and accountNumber variables:
        // option        || accountNumber
PRIORITY
        //   verifiable     (brs1)    undefined    -> get number   < UNDEFINED
        //   verifiable                (brs2)    verifiable   -> verify             < UNDEFINED
        //   verified                                    undefined             < UNDEFINED
        //   verified                                    verifiable                        <
UNDEFINED

        //   verified          verified     -> do it.                    < UNDEFINED



        // option:       verifiable
        // accountNumber: undefined.
        COND ( MyVerifiable(option); &&
                          MyUndefined(accountNumber);  )   PRIORITY < UNDEFINED
                {
                        IF (DEBUG) { OUTPUT { "closeBankAccount.hdl: option: verifiable,
accountNumber: undefined." ; }}


                        QUESTION(option,accountNumber)
                        {
                                INIT
                                {
                                        "wat is het bank rekening nummer dat u wilt ";

                                        VERIFICATION(option);
                                }

                                MyEvents( NUstatements;  ,
                                                 NUstatements;  ,
                                                 NRstatements;  ,

                                                 NUstatements;  ,
                                                        "wat is het bank rekening nummer dat u wilt
";
                                                        VERIFICATION(option);
                                                );
                        }
                }


                // option:        verifiable
                // accountNumber: verifiable.
                COND ( MyVerifiable(option); &&
                          MyVerifiable(accountNumber);  )  PRIORITY < UNDEFINED
                {
                        IF (DEBUG) { OUTPUT { "closeBankAccount.hdl: option: verifiable,
accountNumber: verifiable." ; }}

                        QUESTION(YES,NO,option,accountNumber)
                        {
                                INIT
                                {
                                        "wilt u de bank rekening met nummer";
                                        CONFIRMATION(accountNumber + option);
                                }
                                MyEvents( NUstatements;  ,
                                                 YNstatements;  ,
                                                 NRstatements;  ,

                                                 YNstatements;  ,
                                                        "wilt u de bank rekening met nummer";
                                                        CONFIRMATION(accountNumber +
option);
                                                );
                        }
                }
```

```
        // option:      verified
        // accountNumber: verifiable.
        COND ( ^^option &&
                 MyVerifiable(accountNumber);  )   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "closeBankAccount.hdl: option: verified,
accountNumber: verifiable." ; }}

            LANGUAGE <- "TeleBankYesNoNumbers";

            QUESTION(YES,NO,accountNumber)
            {
                INIT
                {
                    "wilt u de bank rekening met nummer ";
                    CONFIRMATION(accountNumber);
                    "sluiten";
                }
                MyEvents( NUstatements;  ,
                         YNstatements;  ,
                         NRstatements;  ,

                         YNstatements;  ,
                            "wilt u de bank rekening met nummer ";
                            CONFIRMATION(accountNumber);
                            "sluiten";
                        );

            }

            LANGUAGE <- "TeleBank";

        }



        // option:      verified
        // accountNumber: undefined.
        COND ( ^^option &&
                 MyUndefined(accountNumber);  )   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "closeBankAccount.hdl: option: verified,
accountNumber: undefined." ; }}

            LANGUAGE <- "TeleBankNumbers";

            QUESTION(accountNumber)
            {
                INIT
                {
                    "welke bank rekening wilt u sluiten";
                    IF (DEBUG) { "closeBankAccount"; }
                }
                MyEvents( NUstatements;  ,
                         NUstatements;  ,
                         NRstatements;  ,
                         NUstatements;  ,
                             "welke bank rekening wilt u sluiten";
                             IF (DEBUG) { "closeBankAccount"; }
                        );
            }

            LANGUAGE <- "TeleBank";

        }




        // option:      verified
        // accountNumber: verified.
        // Do it  and return from sub dialog.
        COND ( ^^ option &&
               option == CLOSE_BANK_ACCOUNT &&
               ^^ accountNumber )      PRIORITY < UNDEFINED  // smallest priority.
        {

            IF (DEBUG) { OUTPUT { "closeBankAccount.hdl: option: verified,
accountNumber: verified." ; }}

            BOOLEAN res := TRUE; // Silencing a warning.
            STRING C; STRING X; STRING I;

            DIGIT_2_CONV_STRING_1(C,X,I,accountNumber);

            teleBankDBinstance.closeAccount(accountNumber,res);


            IF ( res )      {
                dialogStatus := OK;
                OUTPUT { "uw bank rekening ";
                         DIGIT_2_CONV_STRING_2(C,X,I);
                         "is gesloten"; }
            }
            ELSE {
                dialogStatus := TRANSACTION_ERROR;
                OUTPUT { "uw bank rekening";
                        DIGIT_2_CONV_STRING_2(C,X,I);
                        "is niet gesloten";
                        "er is geen open bank rekening met dit nummer"; }

            }

            RETURN; // terminating the subdialog.
        }

    END ACTIONS;

END DIALOG closeBankAccount.
```

```
// depositMoney.hdl

INCLUDE "declarations.hdl";



/* input: option         - the option that was choosen, so we can verify
 *                       if needed that CLOSE_BANK_ACCOUNT was chosen.
 *                       option must be passed by reference if we
 *                       want to ask if it is verified.
 *                         accountNumber  - an integer, will be defined if user said allready
 *                       a number. ( might not be defined ).
 *                       it is passed by reference so we can ask if it
 *                       is verified.
 *       amount          - an integer, will be defined if user said allready
 *                       how much he wants to deposit. it is passed by
 *                       referance so we can ask if verified and change if
 *                       necessary.
 * output: dialogStatus  - dialogStatus the return status of this dialog
 *                                                                  will be OK if user closed an account,
 *                        TRANSACTION_ERROR if there was a transaction error
 *        stop           - a variable that will be defined and verified when
 *                       the user wants to stop.
 */
DIALOG depositMoney(teleBankOptions &option,
                                    INTEGER &accountNumber,
                                    INTEGER &amount,
                                    status  &dialogStatus,
                                    BOOLEAN &stop)

        RULES
                INCLUDE "yesNoRules.rhdl";
                INCLUDE "teleBankOptionsRules.rhdl";


                <bankAccountNumberSpecific> INTEGER accountNumber
                        : <bankAccount> <number>                        { accountNumber
:= #2.value; }
                        | <bankAccount> 'is' <number>               { accountNumber :=
#3.value; }
                        | 'op' <number>                                         {
accountNumber := #2.value; }
                        ;


                <amountSpecific> INTEGER amount
                        : <number> <gulden>                                     {
amount := #1.value;     }
                        ;


        END RULES;


        PREFERENCES
        END PREFERENCES;
```

```
VARIABLES


        YES
        {
                <- <yes>
        };

        NO
        {
                <- <no>
        };

        PARAMETER stop
        {
                <- <stop_rule>
        };

        PARAMETER option
        {
                <- <teleBankOption>.option
        };

        PARAMETER accountNumber
        {
                <- <teleBankOption>.accountNumber,
                <- <number>.value
MyAccountNumberNumberQualifier,
                <- <bankAccountNumberSpecific>.accountNumber
MyAccountNumberSpecificQualifier
        };


        PARAMETER amount
        {
                <- <teleBankOption>.amount,
                <- <number>.value
MyAmountNumberQualifier,
                <- <amountSpecific>.amount
MyAmountSpecificQualifier

        };


        MEANINGFUL_FILLER
        {
                <- <greeting>,
                <- <please>,
                <- <wantStatement>,
                <- <mine>,
                <- <meaningfulFiller>
        };

END VARIABLES;
```

```
KNOWLEDGE
END KNOWLEDGE;

EVENT HANDLER
END EVENT HANDLER;

VERIFICATION

        stop
        {
        }

        amount + option + accountNumber     // xxx gulden storten op bank rekening yyy
        {
                DIGIT_CONVERSIONS(amount);
                "gulden";
                OPTION_CONVERSIONS(option);
                "op bank rekening";
                DIGIT_CONVERSIONS(accountNumber);
        }

        amount + accountNumber     // xxx gulden storten op bank rekening yyy
        {
                DIGIT_CONVERSIONS(amount);
                "gulden storten op bank rekening";
                DIGIT_CONVERSIONS(accountNumber);
        }



        amount + option         // xxx gulden storten
        {
                DIGIT_CONVERSIONS(amount);
                "gulden";
                OPTION_CONVERSIONS(option);
        }

        accountNumber + option //  storten op bank rekening xxx
        {
                OPTION_CONVERSIONS(option);
                "op bank rekening";
                DIGIT_CONVERSIONS(accountNumber);

        }

        amount
        {
                DIGIT_CONVERSIONS(amount);
        }


        accountNumber
        {
                DIGIT_CONVERSIONS(accountNumber);
        }
```

```
                option                                          // storten
                {
                        OPTION_CONVERSIONS(option);
                }


        END VERIFICATION;

        USES
        END USES;

        TRANSACTIONS
                teleBankDB teleBankDBinstance IMPORT;
        END TRANSACTIONS;

        ACTIONS


                STOP_COND("depositMoney.hdl"); // PRIORITY > MULTIPLE


                // the following event are delt in applicationOptions
                // (mq)
                COND ( MULTIPLE(option) )
                {
                        IF (DEBUG) { OUTPUT { "depositMoney.hdl: option has multiple values
return; "; }}
                        RETURN;
                }
                COND ( SINGLE(option,option != DEPOSIT_MONEY) )
                {
                        IF (DEBUG) { OUTPUT { "depositMoney.hdl: option has single value Not
DEPOSIT_MONEY, return; "; }}
                        RETURN;
                }
                COND ( UNDEFINED(option) )
                {
                        IF (DEBUG) { OUTPUT { "depositMoney.hdl: option undefined return; "; }}
                        RETURN;
                }

                ////////////////////////////////////////////////////////////////

/*
                 there are xxxx events defined each of them has a condition
                 dependent on option, accountNumber and amount variables:

                 it will not be that accountNumber of amount are verified before option.

                 option         accountNumber               amount                   PRIORITY     ....

1.                 verifiable    undefined                   undefined                < UNDEFINED
+   option[V] or not
2.                 verifiable          undefined                        verifiable                    <
UNDEFINED   +   (option+amount)[V] or not
```

```
3.                  verifiable       verifiable                         undefined                   <
UNDEFINED   +   (option+accountNumber)[V] or not
4.                  verifiable       verifiable                         verifiable                  <
UNDEFINED   +   (option+accountNumber+amount)[V] or not

5.                  verified                 undefined                          undefined
        < UNDEFINED   +
6.                  verified                 undefined                          verifiable
        < UNDEFINED   +   (amount)[V]
7.                  verified                 verifiable                         undefined
        < UNDEFINED   +   (accountNumber)[V]
8.                  verified                 verifiable                         verifiable
        < UNDEFINED   +

9.                  verified                 undefined                          undefined
        < UNDEFINED  ( == 5 )
10.                 verified                 undefined                          verified            <
UNDEFINED   +
11.                 verified                 verified                  undefined                   <
UNDEFINED   +
12.                 verified                 verified                  verified            < UNDEFINED
( == 16. )

13.                 verified                 verifiable                         verifiable
        < UNDEFINED  ( == 8. )
14.                 verified                 verifiable                         verified            <
UNDEFINED   +   (accountNumber)[V]
15.                 verified                 verified                  verifiable                  <
UNDEFINED   +   (amount)[V]
16.                 verified                 verified                  verified            < UNDEFINED
+      do it

*/

                // 1.
                // option:                              verifiable
                // accountNumber:           undefined
                // amount:                              undefined
                COND ( MyVerifiable(option); &&
                          MyUndefined(accountNumber); &&
                          MyUndefined(amount);)   PRIORITY < UNDEFINED
                {
                        IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verifiable, ";
                                                                        "accountNumber: undefined,
amount: undefined" ; }}

                        ENABLE MyAccountNumberSpecificQualifier;
                        ENABLE MyAccountNumberNumberQualifier;// if one number is given it will
be understood

        // as the account number becouse that was the question.
                        ENABLE MyAmountSpecificQualifier;

                        QUESTION(option,accountNumber,amount)
                        {
                                INIT
                {
                    "op welke bank rekening wilt u geld ";
                    VERIFICATION(option);        // storten

                }

                MyEvents( NUstatements;  ,
                          NUstatements;  ,
                          NRstatements;  ,

                          NUstatements;  ,
                              "op welke bank rekening wilt u geld ";
                              VERIFICATION(option);        // storten
                          );
            }
            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAccountNumberNumberQualifier;
            DISABLE MyAmountSpecificQualifier;
        }



        // 2.
        // option:                        verifiable
        // accountNumber:         undefined
        // amount:                        verifiable
        COND ( MyVerifiable(option); &&
               MyUndefined(accountNumber); &&
               MyVerifiable(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verifiable, ";
                                                    "accountNumber: undefined,
amount: verifiable" ; }}

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAccountNumberNumberQualifier;  // if one number is given it will
be understood

    // as account number becouse that was the question.
            ENABLE MyAmountSpecificQualifier;

            QUESTION(option,accountNumber,amount)
            {
                INIT
                {
                    "op welke bank rekening wilt u het bedrag van";
                    VERIFICATION(amount + option); // xxx gulden storten;
                }
                MyEvents( NUstatements;  ,
                          NUstatements;  ,
                          NRstatements;  ,

                          NUstatements;  ,
                              "op welke bank rekening wilt u het bedrag
van";
```

```
                            VERIFICATION(amount + option); // xxx
gulden storten;
                        );
            }

            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAccountNumberNumberQualifier;
            DISABLE MyAmountSpecificQualifier;

        }



        // 3.
        // option:              verifiable
        // accountNumber:   verifiable
        // amount:              undefined
        COND ( MyVerifiable(option); &&
                 MyVerifiable(accountNumber); &&
                 MyUndefined(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verifiable, ";
                                "accountNumber: verifiable,
amount: undefined" ; }}

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAmountNumberQualifier;     // if one number is given it will be
understood

    // as amount becouse that was the question.
            ENABLE MyAmountSpecificQualifier;

            QUESTION(option,accountNumber,amount)
            {
                INIT
                {
                    "welk bedrag wilt u";
                    VERIFICATION(accountNumber + option); // storten op bank
rekening xxx
                }
                MyEvents( NUstatements;  ,
                        NUstatements;  ,
                        NRstatements;  ,

                        NUstatements;  ,
                            "welk bedrag wilt u";
                            VERIFICATION(accountNumber + option);
// storten op bank rekening xxx
                        );
            }

            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAmountNumberQualifier;
            DISABLE MyAmountSpecificQualifier;

        }
        // 4.
        // option:              verifiable
        // accountNumber:   verifiable
        // amount:              verifiable
        COND ( MyVerifiable(option); &&
                MyVerifiable(accountNumber); &&
                MyVerifiable(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verifiable, ";
                                            "accountNumber: verifiable,
amount: verifiable" ; }}

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAmountNumberQualifier;     // if one number is given it will be
understood

    // as amount becouse we say 'het bedraag'
            ENABLE MyAmountSpecificQualifier;

            QUESTION(YES,NO,option,accountNumber,amount)
            {
                INIT
                {
                    "wilt u het bedrag van";
                    CONFIRMATION(amount + option + accountNumber);
                        // xxx gulden storten op bank rekening yyy
                }
                MyEvents( NUstatements;  ,
                         YNstatements;  ,
                         NRstatements;  ,

                         YNstatements;  ,
                            "wilt u het bedrag";
                            CONFIRMATION(amount + option +
accountNumber);
                        );
            }

            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAmountNumberQualifier;
            DISABLE MyAmountSpecificQualifier;
        }



        // 5.
        // option:              verified
        // accountNumber:   undefined
        // amount:              undefined
        COND ( ^^option &&
                MyUndefined(accountNumber); &&
                MyUndefined(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
                                                                "accountNumber: undefined,
amount: undefined" ; }}

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAccountNumberNumberQualifier;// if one number is given it will
be understood

    // as the account number becouse that was the question.
            ENABLE MyAmountSpecificQualifier;

            QUESTION(accountNumber,amount)
            {
                INIT
                {
                    "op welke bank rekening wilt u geld storten";
                }
                MyEvents( NUstatements;  ,
                          NUstatements;  ,
                          NRstatements;  ,

                          NUstatements;  ,
                              "op welke bank rekening wilt u geld
storten";
                          );
            }
            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAccountNumberNumberQualifier;
            DISABLE MyAmountSpecificQualifier;
        }




        // 6.
        // option:                  verified
        // accountNumber:       undefined
        // amount:                  verifiable
        COND ( ^^option &&
             MyUndefined(accountNumber); &&
             MyVerifiable(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
                                                                "accountNumber: undefined,
amount: verifiable" ; }}

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAccountNumberNumberQualifier;  // if one number is given it will
be understood

    // as account number becouse that was the question.
            ENABLE MyAmountSpecificQualifier;

            QUESTION(accountNumber,amount)
            {
                INIT
                {
                    "op welke bank rekening wilt u het bedrag van";
                    VERIFICATION(amount);
                    "gulden storten";
                }
                MyEvents( NUstatements;  ,
                                         NUstatements;  ,
                                         NRstatements;  ,

                                         NUstatements;  ,
                                             "op welke bank rekening wilt u het bedrag
van";
                                             VERIFICATION(amount);
                                             "gulden storten";
                                        );
            }

            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAccountNumberNumberQualifier;
            DISABLE MyAmountSpecificQualifier;

        }


        // 7.
        // option:                 verified
        // accountNumber:      verifiable
        // amount:                 undefined
        COND ( ^^option &&
                 MyVerifiable(accountNumber); &&
                 MyUndefined(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
                                                "accountNumber: verifiable,
amount: undefined" ; }}

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAmountNumberQualifier;        // if one number is given it will be
understood

    // as amount becouse that was the question.
            ENABLE MyAmountSpecificQualifier;

            QUESTION(accountNumber,amount)
            {
                INIT
                {
                    "welk bedrag wilt u storten op bank rekening";
                    VERIFICATION(accountNumber);
                }
                MyEvents( NUstatements;  ,
                                         NUstatements;  ,
                                         NRstatements;  ,

                                         NUstatements;  ,
                            "welk bedrag wilt u storten op bank
rekening";
                            VERIFICATION(accountNumber);
                        );
            }

            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAmountNumberQualifier;
            DISABLE MyAmountSpecificQualifier;

        }


        // 8.
        // option:              verified
        // accountNumber:   verifiable
        // amount:              verifiable
        COND ( ^^option &&
               MyVerifiable(accountNumber); &&
               MyVerifiable(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
                                        "accountNumber: verifiable,
amount: verifiable" ; }}

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAmountNumberQualifier;     // if one number is given it will be
understood

    // as amount becouse we say 'het bedraag'
            ENABLE MyAmountSpecificQualifier;

            QUESTION(YES,NO,accountNumber,amount)
            {
                INIT
                {
                    "wilt u het bedrag van";
                    CONFIRMATION(amount + accountNumber);
                        // xxx gulden storten op bank rekening yyy
                }
                MyEvents( NUstatements;  ,
                         YNstatements;  ,
                         NRstatements;  ,

                         YNstatements;  ,
                            "wilt u het bedrag";
                            CONFIRMATION(amount + option +
accountNumber);
                        );
            }

            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAmountNumberQualifier;
            DISABLE MyAmountSpecificQualifier;
        }
        // 10.
        // option:                verified
        // accountNumber:     undefined
        // amount:                verified
        COND ( ^^option &&
             MyUndefined(accountNumber); &&
             ^^amount)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
                                    "accountNumber: undefined,
amount: verified" ; }}

            LANGUAGE <- "TeleBankNumbers";

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAccountNumberNumberQualifier;  // if one number is given it will
be understood

    // as account number becouse that was the question.

            QUESTION(accountNumber)
            {
                INIT
                {
                    "op welke bank rekening wilt u geld storten";
                }
                MyEvents( NUstatements;  ,
                         NUstatements;  ,
                         NRstatements;  ,

                         NUstatements;  ,
                            "op welke bank rekening wilt u geld
storten";
                        );
            }

            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAccountNumberNumberQualifier;

            LANGUAGE <- "TeleBank";
        }



        // 11.
        // option:                verified
        // accountNumber:     verified
        // amount:                undefined
        COND ( ^^option &&
             ^^accountNumber &&
             MyUndefined(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
```

```
                                                                "accountNumber: verified,
amount: undefined" ; }}


            LANGUAGE <- "TeleBankNumbers";

            ENABLE MyAmountNumberQualifier;        // if one number is given it will be
understood

    // as amount becouse that was the question.
            ENABLE MyAmountSpecificQualifier;

            QUESTION(amount)
            {
                INIT
                {
                    "welk bedrag wilt u storten";
                }
                MyEvents( NUstatements;  ,
                          NUstatements;  ,
                          NRstatements;  ,

                          NUstatements;  ,
                              "welk bedrag wilt u storten";
                        );
            }

            DISABLE MyAmountNumberQualifier;
            DISABLE MyAmountSpecificQualifier;

            LANGUAGE <- "TeleBank";

        }




        // 14.
        // option:                  verified
        // accountNumber:       verifiable
        // amount:                  verified
        COND ( ^^option &&
                MyVerifiable(accountNumber); &&
                ^^amount)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
                                                                "accountNumber: verifiable,
amount: verified" ; }}

            LANGUAGE <- "TeleBankYesNoNumbers";

            ENABLE MyAccountNumberSpecificQualifier;
            ENABLE MyAccountNumberNumberQualifier;

            QUESTION(YES,NO,accountNumber)
            {
                INIT
                {
                    "wilt u geld storten op bank rekening";
                    CONFIRMATION(accountNumber);
                }
                MyEvents( NUstatements;  ,
                         YNstatements;  ,
                         NRstatements;  ,

                         YNstatements;  ,
                             "wilt u geld storten op bank rekening";
                             CONFIRMATION(accountNumber);
                        );
            }

            DISABLE MyAccountNumberSpecificQualifier;
            DISABLE MyAccountNumberNumberQualifier;

            LANGUAGE <- "TeleBank";
        }



        // 15.
        // option:                  verified
        // accountNumber:       verified
        // amount:                  verifiable
        COND ( ^^option &&
             ^^accountNumber &&
             MyVerifiable(amount);)   PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
                                        "accountNumber: verified,
amount: verifiable" ; }}

            LANGUAGE <- "TeleBankYesNoNumbers";

            ENABLE MyAmountNumberQualifier;
            ENABLE MyAmountSpecificQualifier;

            QUESTION(YES,NO,amount)
            {
                INIT
                {
                    "wilt u het bedrag van";
                    CONFIRMATION(amount);
                    "gulden storten";
                }
                MyEvents( NUstatements;  ,
                         NUstatements;  ,
                         NRstatements;  ,

                         NUstatements;  ,
                             "wilt u het bedrag van";
                             CONFIRMATION(amount);
                             "gulden storten";
```

```
                                        );
                }

                DISABLE MyAmountNumberQualifier;
                DISABLE MyAmountSpecificQualifier;

                LANGUAGE <- "TeleBank";

        }



        // 16.
        // option:                        verified
        // accountNumber:         verified
        // amount:                        verified
        COND ( ^^option &&
                 option == DEPOSIT_MONEY &&
                 ^^accountNumber &&
                 ^^amount                               )   PRIORITY < UNDEFINED
        {
                IF (DEBUG) { OUTPUT { "depositMoney.hdl: option: verified, ";
                                                        "accountNumber: verified,
amount: verified" ; }}

                BOOLEAN res := TRUE; // Silencing a warning.
                STRING Caccount; STRING Xaccount; STRING Iaccount;
                STRING Camount;  STRING Xamount;  STRING Iamount;


                DIGIT_2_CONV_STRING_1(Caccount,Xaccount,Iaccount,accountNumber);
                DIGIT_2_CONV_STRING_1(Camount ,Xamount ,Iamount ,amount);


                teleBankDBinstance.deposit(accountNumber,amount,res);

                IF ( res )          {
                        dialogStatus := OK;
                        OUTPUT { "u heeft";
                                        DIGIT_2_CONV_STRING_2(Camount ,Xamount
,Iamount);
                                        "guldens gestort";
                                        "op bank rekening nummer";

DIGIT_2_CONV_STRING_2(Caccount,Xaccount,Iaccount);
                                        }
                }
                ELSE {
                        dialogStatus := TRANSACTION_ERROR;
                        OUTPUT { "u geld is niet gestord";
                                        "er is geen open bank rekening met dit nummer"; }

                }

                RETURN; // terminating the subdialog.
        }
```

```
            END ACTIONS;

END DIALOG depositMoney.





// bankAccountInformation.hdl

INCLUDE "declarations.hdl";



/* input: option        - the option that was choosen, so we can verify
 *                    if needed that BANK_ACCOUNT_INFORMATION was chosen.
 *                    option must be passed by reference if we
 *                    want to ask if it is verified.
 *                      accountNumber  - an integer, will be defined if user said allready
 *                    a number. ( might not be defined ).
 *                    it is passed by reference so we can ask if it
 *                    is verified.
 * output: dialogStatus  - dialogStatus the return status of this dialog
 *                                                              will be OK if user recieved the information,
 *                     TRANSACTION_ERROR if there was a transaction error
 *       stop          - a variable that will be defined and verified when
 *                    the user wants to stop.
 */
DIALOG bankAccountInformation(teleBankOptions &option,
                                                    INTEGER &accountNumber,
                                                         status  &dialogStatus,
                                                              BOOLEAN &stop)

            RULES
                        INCLUDE "yesNoRules.rhdl";
                        INCLUDE "teleBankOptionsRules.rhdl";
            END RULES;


            PREFERENCES
            END PREFERENCES;

            VARIABLES


                        YES
                        {
                                    <- <yes>
                        };

                        NO
                        {
```

```
		<- <no>
	};

	PARAMETER stop
	{
		<- <stop_rule>
	};

	PARAMETER option
	{
		<- <teleBankOption>.option
	};

	PARAMETER accountNumber
	{
		<- <teleBankOption>.accountNumber,
		<- <number>.value
	};

	MEANINGFUL_FILLER
	{
		<- <greeting>,
		<- <please>,
		<- <wantStatement>,
		<- <mine>,
		<- <meaningfulFiller>
	};

END VARIABLES;

KNOWLEDGE
END KNOWLEDGE;

EVENT HANDLER
END EVENT HANDLER;

VERIFICATION

	stop
	{
	}

	accountNumber + option  // saldo van de bank rekening nummer xxx
	{
		OPTION_CONVERSIONS(option);
		"van de rekening nummer";
		DIGIT_CONVERSIONS(accountNumber);
	}

	accountNumber
	{
		DIGIT_CONVERSIONS(accountNumber);
	}

	option
	{
```

```
                OPTION_CONVERSIONS(option);
            }

    END VERIFICATION;

    USES
    END USES;

    TRANSACTIONS
            teleBankDB teleBankDBinstance IMPORT;
    END TRANSACTIONS;

    ACTIONS

            STOP_COND("bankAccountInformation.hdl"); // PRIORITY > MULTIPLE


            // the following event are delt in applicationOptions
            // (mq)
            COND ( MULTIPLE(option) )
            {
                IF (DEBUG) { OUTPUT { "bankAccountInformation.hdl: option has multiple values return; "; }}
                RETURN;
            }
            COND ( SINGLE(option,option != BANK_ACCOUNT_INFORMATION) )
            {
                IF (DEBUG) { OUTPUT { "bankAccountInformation.hdl: option has single value Not BANK_ACCOUNT_INFORMATION, return; "; }}
                RETURN;
            }
            COND ( UNDEFINED(option) )
            {
                IF (DEBUG) { OUTPUT { "bankAccountInformation.hdl: option undefined return; "; }}
                RETURN;
            }

            ///////////////////////////////////////////////////////////////
            // option        accountNumber                         PRIORITY
            // 1.  verifiable     undefined    -> get number   < UNDEFINED
            // 2.  verifiable       verifiable   -> verify            < UNDEFINED
            // 3.  verified         undefined              < UNDEFINED
            // 4. verified          verifiable                        < UNDEFINED

            // 5.  verified      verified     -> do it.               < UNDEFINED


            // 1.
            // option:       verifiable
            // accountNumber: undefined.
            COND ( MyVerifiable(option); &&
                    MyUndefined(accountNumber);  )   PRIORITY < UNDEFINED
            {
                IF (DEBUG) { OUTPUT { "bankAccountInformation.hdl: option: verifiable, accountNumber: undefined." ; }}
            QUESTION(option,accountNumber)
            {
                INIT
                {
                    "wat is de bank rekening nummer waarvan u de";

                    VERIFICATION(option); // "saldo";
                    "wilt weten";
                }

                MyEvents( NUstatements;  ,
                        NUstatements;  ,
                        NRstatements;  ,

                        NUstatements;  ,
                            "wat is de bank rekening nummer waarvan u
de";
                            VERIFICATION(option); // "saldo";
                            "wilt weten";
                       );
            }
        }

        // 2.
        // option:       verifiable
        // accountNumber: verifiable.
        COND ( MyVerifiable(option); &&
             MyVerifiable(accountNumber);  )  PRIORITY < UNDEFINED
        {
            IF (DEBUG) { OUTPUT { "bankAccountInformation.hdl: option: verifiable,
accountNumber: verifiable." ; }}

            QUESTION(YES,NO,option,accountNumber)
            {
                INIT
                {
                    "wilt u de"; // saldo van de bank rekening nummer xxx
                    CONFIRMATION(accountNumber + option);
                }
                MyEvents( NUstatements;  ,
                        YNstatements;  ,
                        NRstatements;  ,

                        YNstatements;  ,
                            "wilt u de"; // saldo van de bank rekening
nummer xxx
                            CONFIRMATION(accountNumber +
option);
                       );
            }
        }
```

```
                // 3.
                // option:        verified
                // accountNumber: undefined.
                COND ( ^^option &&
                          MyUndefined(accountNumber);  )   PRIORITY < UNDEFINED
                {
                        IF (DEBUG) { OUTPUT { "bankAccountInformation.hdl: option: verified,
accountNumber: undefined." ; }}

                        LANGUAGE <- "TeleBankNumbers";

                        QUESTION(accountNumber)
                        {
                                INIT
                                {
                                        "over welke bank rekening wilt u informatie";
                                        IF (DEBUG) { "bankAccountInformation"; }
                                }
                                MyEvents( NUstatements;  ,
                                                 NUstatements;  ,
                                                 NRstatements;  ,
                                                 NUstatements;  ,
                                                        "over welke bank rekening wilt u
informatie";
                                                        IF (DEBUG) { "bankAccountInformation";
}
                                                );
                        }

                        LANGUAGE <- "TeleBank";
                }



                // 4.
                // option:        verified
                // accountNumber: verifiable.
                COND ( ^^option &&
                          MyVerifiable(accountNumber);  )   PRIORITY < UNDEFINED
                {
                        IF (DEBUG) { OUTPUT { "bankAccountInformation.hdl: option: verified,
accountNumber: verifiable." ; }}

                        LANGUAGE <- "TeleBankYesNoNumbers";

                        QUESTION(YES,NO,accountNumber)
                        {
                                INIT
                                {
                                        "wilt u de saldo van de bank rekening";
                                        CONFIRMATION(accountNumber);
                                        "weten";
                                }
                                MyEvents( NUstatements;  ,
                                                 YNstatements;  ,
                                        NRstatements;  ,

                                        YNstatements;  ,
                                                "wilt u de saldo van de bank rekening";
                                                CONFIRMATION(accountNumber);
                                                "weten";
                                       );

                        }

                        LANGUAGE <- "TeleBank";

                }




                // 5.
                // option:        verified
                // accountNumber: verified.
                // Do it  and return from sub dialog.
                COND ( ^^ option &&
                          option == BANK_ACCOUNT_INFORMATION &&
                          ^^ accountNumber )  PRIORITY < UNDEFINED  // smallest priority.
                {

                        IF (DEBUG) { OUTPUT { "bankAccountInformation.hdl: option: verified,
accountNumber: verified." ; }}

                        BOOLEAN res := TRUE; // Silencing a warning.
                        INTEGER amount := 0;
                        STRING C; STRING X; STRING I;

                        teleBankDBinstance.currentAmount(accountNumber,amount,res);

                        DIGIT_2_CONV_STRING_1(C,X,I,amount);

                        IF ( res )              {
                                dialogStatus := OK;
                                OUTPUT { "uw bank rekening saldo is";
                                          DIGIT_2_CONV_STRING_2(C,X,I);
                                          "gulden";
                                         }
                                IF (DEBUG) { OUTPUT { " "; AS_STRING(amount); }}
                        }
                        ELSE {
                                dialogStatus := TRANSACTION_ERROR;
                                OUTPUT { "er is geen open bank rekening met deze nummer"; }

                        }

                        RETURN; // terminating the subdialog.
                }
        END ACTIONS;
END DIALOG bankAccountInformation.
```

```
// teleBankInformation.hdl

INCLUDE "declarations.hdl";


/* input:  option        - the option that was choosen, so we can verify
 *                       if needed that TELEBANK_INFORMATION was chosen.
 *                       option must be passed by reference if we
 *                       want to ask if it is verified.
 * output: dialogStatus   - dialogStatus the return status of this dialog
 *                                                   will be OK if user wanted information an account,
 *       stop            - a variable that will be defined and verified when
 *                      the user wants to stop.
 */
DIALOG teleBankInformation(teleBankOptions &option,
                                                  status  &dialogStatus,
                                                  BOOLEAN &stop)

        RULES
                INCLUDE "yesNoRules.rhdl";
                INCLUDE "teleBankOptionsRules.rhdl";
        END RULES;

        PREFERENCES
        END PREFERENCES;

        VARIABLES

                YES
                {
                        <- <yes>
                };

                NO
                {
                        <- <no>
                };

                PARAMETER stop
                {
                        <- <stop_rule>
                };

                PARAMETER option
                {
                        <- <teleBankOption>.option
                };

                MEANINGFUL_FILLER
                {
                        <- <greeting>,
                        <- <please>,
                        <- <wantStatement>,
                        <- <mine>,
                        <- <meaningfulFiller>
        };

    END VARIABLES;

    KNOWLEDGE
    END KNOWLEDGE;

    EVENT HANDLER
    END EVENT HANDLER;

    VERIFICATION

        stop
        {
        }

        option
        {
            OPTION_CONVERSIONS(option);
        }

    END VERIFICATION;

    USES
    END USES;

    TRANSACTIONS
    END TRANSACTIONS;

    ACTIONS

        STOP_COND("teleBankInforamtion.hdl"); // PRIORITY > MULTIPLE


        // this will be delt with in applicationOptions
        COND ( MULTIPLE(option) )
        {
            IF (DEBUG) { OUTPUT { "teleBankInforamtion.hdl: option has multiple
values return; "; }}
            RETURN;
        }
        COND ( SINGLE(option,option != TELEBANK_INFORMATION) )
        {
            IF (DEBUG) { OUTPUT { "teleBankInforamtion.hdl: option has single value
Not TELEBANK_INFORMATION, return; "; }}
            RETURN;
        }
        COND ( UNDEFINED(option) )
        {
            IF (DEBUG) { OUTPUT { "teleBankInforamtion.hdl: option undefined, return;
"; }}
            RETURN;
        }
```

```
                // option
                //   verifiable   -> verify
                //   verified     -> do it.


                // option:        verifiable
                // (ro.a)
                COND ( MyVerifiable(option); )
                {
                        IF (DEBUG) { OUTPUT { "teleBankInforamtion.hdl: option: verifiable" ; }}

                        QUESTION(YES,NO,option)
                        {
                                INIT
                                {
                                        "wilt u de ";
                                        CONFIRMATION(option);
                                        "afluisteren";
                                }

                                MyEvents( NUstatements;  ,
                                                 YNstatements;  ,
                                                 NRstatements;  ,

                                                 YNstatements;  ,
                                                        "wilt u de ";
                                                        CONFIRMATION(option);
                                                        "afluisteren";
                                                );
                        }
                }


                COND ( ^^ option && option == TELEBANK_INFORMATION )
                {
                        // print telebank information.
                        PrintTeleBankInformation

                        dialogStatus := OK;

                        RETURN; // terminating the subdialog.
                }


        END ACTIONS;


END DIALOG teleBankInformation.




// welcome.hdl
```

```
INCLUDE "declarations.hdl";



DIALOG welcome()

        RULES
        END RULES;

        PREFERENCES
        END PREFERENCES;

        VARIABLES
        END VARIABLES;

        KNOWLEDGE
        END KNOWLEDGE;

        EVENT HANDLER
        END EVENT HANDLER;

        VERIFICATION
        END VERIFICATION;

        USES
        END USES;

        TRANSACTIONS
        END TRANSACTIONS;

        ACTIONS

                // welcome statment
                COND (ONCE)
                {
                        IF ( CURRENT_TIME > BEGIN_EVENING )
                        {
                                OUTPUT { "goede avond"; }
                        }
                        ELSE IF ( CURRENT_TIME > END_MORNING )
                        {
                                OUTPUT { "goede middag"; }
                        }
                        ELSE
                        {
                                OUTPUT { "goede morgen"; }
                        }

                        OUTPUT
                        {
                                "dit is telebank een telefonische dienstverlening van armo bank";
                        }
                }

        END ACTIONS;
END DIALOG welcome.
```

```
// yesOrNoQuestion.hdl

INCLUDE "declarations.hdl";

/* input: question - a string containing the text that will
 *                 be asked.
 * output: answer  - answer will be TRUE if the answer to
 *                 the question was 'yes' and FLASE if the
 *                 answer to the question was 'no'
 */
DIALOG yesOrNoQuestion(STRING question,BOOLEAN &answer)

        RULES
                INCLUDE "yesNoRules.rhdl";
        END RULES;

        PREFERENCES
        END PREFERENCES;

        VARIABLES
          /*
                * i am using her YES_ and NO_ becouse the system defined
                * YES and NO work diffrently: when the question is asked
                * the NOTHING_UNDERSTOOD event is activate (without any
                * parameters ) when none of the variables in the current
                * belief has changed except for the system YES and
                * NO. so to solve this either NOTHING_UNDERSTOOD should be
                * declared as follows NOTHING_UNDERSTOOD(YES,NO) or
                * new variable should be defined.
                * i defined new variables so i could use MyEvents macro
                * for simplicity.
           */
                BOOLEAN YES_
                {
                        <- <yes>
                };
                BOOLEAN NO_
                {
                        <- <no>
                };

        END VARIABLES;

        KNOWLEDGE
        END KNOWLEDGE;

        EVENT HANDLER
        END EVENT HANDLER;

        VERIFICATION
        END VERIFICATION;

        USES
        END USES;

        TRANSACTIONS
        END TRANSACTIONS;

        ACTIONS

                COND (! ^answer)
                {
                        // only YES/NO lexicon improves the understanding.
                   LANGUAGE <- "TeleBankYesNo"; // language resource name. TeleBank.lex

                        QUESTION(YES_,NO_)
                        {
                                INIT
                                {
                                        question;
                                }

                                MyEvents( NUstatements;  ,
                                                YNstatements;  ,
                                                NRstatements;  ,
                                                YNstatements;  ,
                                                        question ;
                                                        IF (DEBUG) { "yesOrNoQuestion.hdl"; }
                                                );
                        }

                        IF ( YES_ && NO_ )
                        {
                                OUTPUT { YNstatements; }
                                CLEAR YES_;
                                CLEAR NO_;
                                BREAK;
                        }

                        IF ( YES_ )
                        {
                                answer := TRUE;
                        }
                        IF ( NO_ )
                        {
                                answer := FALSE;
                        }

                        ASSERT( ^answer, "yesOrNoQuestion.hdl: answer not defined" );


                        // return to normal lexicon ( all words )
                        LANGUAGE <- "TeleBank"; // language resource name.

                }

        END ACTIONS;


END DIALOG yesOrNoQuestion.
```

```
// declarations.hdl

MACRO DEBUG
{
	FALSE		//TRUE	// FALSE
}


// minutes from midnight.
MACRO END_MORNING
{
	720		// 12:00
}

MACRO BEGIN_EVENING
{
   1080		// 18:00
}

MACRO NUstatements
{
	"uw antwoord op de vraag is onduidelijk";
	"wilt u nogmaals antwoord geven op de volgende vraag"
}

MACRO NRstatements
{
	"het systeem heeft uw antwoord niet gehoord";
	"probeer alstublieft nogmaals duidelijk antwoord te geven op de volgende vraag"
}

MACRO YNstatements
{
	"kunt u de volgende vraag alstublieft nogmaals beantwoorden met slechts ja of nee"
}

// assert
MACRO ASSERT(boolExpr,text)
{
	IF ( boolExpr != TRUE )
	{
		OUTPUT
		{
			NO_SPEECH;
			"ASSERTION FAILED ";
			text;
		}
		BYE;
	}
}



MACRO DIGIT_CONVERSIONS(number)
{
	CONVERSIONS numbers2text;
	IF ( number < 10 )
	{
		"n_" ~ AS_STRING(number);
	}
	ELSE
	IF ( number < 100 )
	{
		"n_" ~ AS_STRING(number / 10);
		"n_" ~ AS_STRING(number % 10);
	}
	ELSE
	{
		"n_" ~ AS_STRING(number / 100);
		"n_" ~ AS_STRING((number / 10) % 10);
		"n_" ~ AS_STRING(number % 10);
	}
}



/* C - number of hundreds
 * X - number of tens
 * I - number of ones
 * i.e. if the number is 806 than
 *  C = "n_8", X = "n_0", I = "n_6"
 */
MACRO DIGIT_2_CONV_STRING_1(C,X,I,number)
{
	IF ( number < 10 )
	{
		I := "n_" ~ AS_STRING(number);
		X := "";
		C := "";
	}
	ELSE
	IF ( number < 100 )
	{
		I := "n_" ~ AS_STRING(number % 10);
		X := "n_" ~ AS_STRING(number / 10);
		C := "";
	}
	ELSE
	{
		I := "n_" ~ AS_STRING(number % 10);
		X := "n_" ~ AS_STRING((number / 10) % 10);
		C := "n_" ~ AS_STRING(number / 100);
	}
}


MACRO DIGIT_2_CONV_STRING_2(C,X,I)
{
	CONVERSIONS numbers2text;
	C;
	X;
```

```
        I;
}


MACRO OPTION_CONVERSIONS(option)
{
        IF ( option == OPEN_BANK_ACCOUNT )
        {
                "openen";
        }
        ELSE
        IF ( option == CLOSE_BANK_ACCOUNT )
        {
                "sluiten";
        }
        ELSE
        IF ( option == DEPOSIT_MONEY )
        {
                "storten";
        }
        ELSE
        IF ( option == WITHDRAW_MONEY )
        {
                "afhalen";
        }
        ELSE
        IF ( option == BANK_ACCOUNT_INFORMATION )
        {
                "saldo";
        }
        ELSE // ( option == TELEBANK_INFORMATION )
        {
                "telebank informatie";
        }
}


MACRO MyVerifiable(Expr)
{
        ( ^Expr && !^^Expr )
}

MACRO MyUndefined(Expr)
{
        ( ! ^Expr )
}


MACRO PrintTeleBankInformation
{
        OUTPUT
        {
                "dit is een automatische telefonische bank";
                "u kunt de volgende transacties verrichten";
                "u kunt een nieuwe prive-rekening openen";
```

```
                        "u kunt een prive-rekening sluiten";
                        "u kunt geld storten op een prive-rekening";
                        "u kunt geld afhalen van een prive-rekening";
                        "u kunt het saldo van een prive-rekening opvragen";
                        "u kunt deze informatie nogmals afluisteren";
                        "als u een transactie wilt stoppen kunt u stop zeggen";
            }
}




/* NUstatementsFirst - NOTHING_UNDERSTOOD statements that will be executed
 *                    the first time this event is generated, and after any
 *                    other event.
 * NUstatementsRest - NOTHING_UNDERSTOOD statements that will be executed
 *                                                        in each of the consecutive generations of same event.
 *                 ( this does not work as specified in  pg 87  )
 * NRstatements - NOTHING_RECORDED statements,
 * TLstatements - TOO_LONG statements,
 * Expretions  - these will be executed after the statement in each
 *              of the events. might be empty ;
 */
MACRO MyEvents(NUstatementsFirst,NUstatementsRest,
                                    NRstatements_,TLstatements_,Expretions)
{
            NOTHING_UNDERSTOOD
            {
                        IF (DEBUG) { "event handler NOTHING_UNDERSTOOD."; }

                        IF ( REPETITION == 1 )
                        {
                                    NUstatementsFirst
                        }
                        ELSE {
                                    NUstatementsRest
                        }

                        Expretions
            }
            NOTHING_RECORDED
            {
                        IF (DEBUG) { "event handler NOTHING_RECORDED."; }

                        NRstatements_

                        Expretions
            }
            TOO_LONG
            {
                        IF (DEBUG) { "event handler TOO_LONG."; }

                        TLstatements_

                        Expretions
```

```
    }
}



MACRO STOP_COND(fileName)
{
        COND ( MyVerifiable(stop); )  PRIORITY > MULTIPLE
        {
            IF (DEBUG) { OUTPUT { fileName; "STOP_COND: stop is verifiable"; }}

            QUESTION(YES,NO,stop)
            {
                INIT
                {
                    "wilt u deze transactie stoppen";
                    CONFIRMATION(stop);
                }
                MyEvents( NUstatements;  ,
                         YNstatements;  ,
                         NRstatements;  ,

                         YNstatements;  ,

                            "wilt u deze transactie stoppen";
                            CONFIRMATION(stop);
                        );
            }
        }


        COND ( ^^ stop )  PRIORITY > MULTIPLE
        {
            IF (DEBUG) { OUTPUT { fileName; "STOP_COND: stop is verified";}}
            RETURN;
        }

}



CONVERSIONS numbers2text
{
  "n_0"  -> "nul";
  "n_1"  -> "een";
  "n_2"  -> "twee";
  "n_3"  -> "drie";
  "n_4"  -> "vier";
  "n_5"  -> "vijf";
  "n_6"  -> "zes";
  "n_7"  -> "zeven";
  "n_8"  -> "acht";
  "n_9"  -> "negen";
}
```

```
ENUMERATION status
{
        OK,
        TRANSACTION_ERROR
}



ENUMERATION teleBankOptions
{
        OPEN_BANK_ACCOUNT,
        CLOSE_BANK_ACCOUNT,
        DEPOSIT_MONEY,
        WITHDRAW_MONEY,
        BANK_ACCOUNT_INFORMATION,
        TELEBANK_INFORMATION
}


/*
class accountsData {
private:
        account accounts[NumAccounts];  //we can have up to 1000 accounts.
        char fileName[MaxNameLength];   //the fileName contaning data.

public:
        accountsData(char* _fileName);  //loading the file. (into the array)

        bool openAccount (int &accountNumber); // if o.k. true & accountNumber initialized, else false
  bool closeAccount(int  accountNumber); // if o.k. true, false if accountNumber doesn't exist.
                                  // amount may be negative.
        bool deposit (int accountNumber,int amount); // adds amount to accountNumber, false if
accountNumber doesn't exist.
        bool withdraw(int accountNumber,int amount); // reduces amount from accountNumber,false if
accountNumber doesn't exist.

        bool currentAmount(int accountNumber,int &amount); // returns amount in accountNumber,false
if accountNumber doesn't exist.

        ~accountsData(); // writing the array into the file.

        void printAccounts(void);
};
*/

CLASS teleBankDB
{
        openAccount (INTEGER &accountNumber,BOOLEAN &res);
        closeAccount(INTEGER  accountNumber,BOOLEAN &res);
        deposit (INTEGER accountNumber, INTEGER amount, BOOLEAN &res);
        withdraw(INTEGER accountNumber, INTEGER amount, BOOLEAN &res);
        currentAmount(INTEGER accountNumber, INTEGER &amount, BOOLEAN &res);
        printAccounts();   // for debbuging
}
```

```
// yesOrNoRules.rhdl
// yes no rules


<yes> BOOLEAN
            : 'ja'                                                  { TRUE; }
            | 'graag'                                           { TRUE; }
            | 'goed'                                            { TRUE; }
            ;
<no>  BOOLEAN
            : 'nee'                                                 { TRUE; }
            | 'nee' 'dat' 'wil' 'ik' 'niet'             { TRUE; }
            | 'dat' 'wil' 'ik' 'niet'                       { TRUE; }
            | 'iets' 'anders'                                  { TRUE; }
            | 'ik' 'wil' 'deze' 'optie' 'niet'      { TRUE; }
            ;


<stop_rule> BOOLEAN
            : 'stop'                                            { TRUE; }
            | 'halt'                                            { TRUE; }
            ;




// telebankOptionsRules.rhdl

INCLUDE "yesNoRules.rhdl";


<negative>
            : <no>
            | 'niet'
            | 'geen'
            | 'helemaal' 'niet'
            ;

<digit> INTEGER value
            : 'nul'                 { value := 0; }
            | 'een'                 { value := 1; }
            | 'twee'        { value := 2; }
            | 'drie'        { value := 3; }
            | 'vier'        { value := 4; }
            | 'vijf'        { value := 5; }
            | 'zes'                 { value := 6; }
            | 'zeven'       { value := 7; }
            | 'acht'        { value := 8; }
            | 'negen'       { value := 9; }
            ;
```

```
<number> INTEGER value // defined in a very simple manner.
        : <digit>               { value := #1.value; }
        | <digit> <digit>       { value := #1.value * 10 + #2.value; }
        | <digit> <digit> <digit> { value := #1.value * 100 + #2.value  * 10 + #3.value; }
        ;


<gulden>
        : 'gulden'
        | 'piek'
        ;


<daytime>
        : 'morgen'
        | 'middag'
        | 'avond'
        ;


<greeting>
        : 'goede' <daytime>
        | 'dag'
        | 'hallo'
        ;


<please>
        : 'alstublieft'
        | 'alsjeblieft'
        | 'graag'
        | 'willen'   // goes with: 'ik zou graag'
        ;

<wantStatement>
        : 'ik' 'wil'
        | 'ik' 'wilde'
        | 'ik' 'zou'
        | 'ik' 'zou' 'graag'
        | 'willen'
        | 'graag'
        ;

<mine>
        : 'mijn'
        | 'van' 'mij'
        ;


<connector>
        : 'een'
        | 'de'
        | <mine>
        | <please>
        ;
```

```
<overVanOp>
        : 'over'
        | 'van'
        | 'op'
        ;


<help>
        : 'help'
        | 'hulp'
        | 'nu' 'wat'
        | 'wat' 'nu'
        | 'joehoe'
        | 'ik' 'weet' 'het' 'niet'
        | 'ik' 'weet' 'niet'
        | 'wat' 'moet' 'ik' 'nu' 'doen'
        ;


<meaningfulFiller>
        : 'mmm'
        | 'mag' 'ik'
        | 'met'
        | 'ik' 'ben'
        ;


<bankAccount>
        : 'rekening'
        | 'bank' 'rekening'
        | 'prive' 'rekening'
        | 'prive' 'bank' 'rekening'
        | 'rekening' 'nummer'
        | 'bank' 'rekening' 'nummer'
        | 'prive' 'rekening' 'nummer'
        | 'prive' 'bank' 'rekening' 'nummer'
        ;


<teleBankOption> teleBankOptions option,
                                 INTEGER accountNumber,
                                 INTEGER amount

        // open bank account
        /* examples:
         * 'ik wil [een] bank rekening openen'
         * 'ik zou [graag] een rekening [willen] openen'
         * 'ik wil een nieuwe rekening'
         * 'mag ik een nieuwe rekening openen alstublift'
         * 'ik wilde een rekening openen'
         */
        : 'openen'
        { option := OPEN_BANK_ACCOUNT; }
        | <bankAccount> 'openen'                  { option := OPEN_BANK_ACCOUNT; }
        | <bankAccount> <please> 'openen'         { option := OPEN_BANK_ACCOUNT; }
        | 'nieuwe' <bankAccount>                  { option := OPEN_BANK_ACCOUNT; }
        | 'nieuwe' <bankAccount> 'openen'                  { option := OPEN_BANK_ACCOUNT; }
        | 'nieuwe' <bankAccount> <please> 'openen'         { option := OPEN_BANK_ACCOUNT; }

        // NOT open bank account
        /* examples:
         * 'ik wil[de] geen rekening openen'
         * 'ik wil niet een nieuwe rekening'
         */
        | <negative> <bankAccount> 'openen'                { option := NOT
OPEN_BANK_ACCOUNT; }
        | <negative> 'nieuwe' <bankAccount>                { option := NOT
OPEN_BANK_ACCOUNT; }
        | <negative> 'nieuwe' <bankAccount> 'openen'       { option := NOT
OPEN_BANK_ACCOUNT; }
        | <negative> <bankAccount> 'openen'                { option := NOT
OPEN_BANK_ACCOUNT; }
        | <bankAccount> <negative> 'openen'                { option := NOT
OPEN_BANK_ACCOUNT; }
        | 'nieuwe' <bankAccount> <negative>                { option := NOT
OPEN_BANK_ACCOUNT; }
        | 'nieuwe' <bankAccount> <negative> 'openen'       { option := NOT
OPEN_BANK_ACCOUNT; }


        // close bank account
        /* examples:
         * 'ik wil een bank rekening sluiten'
         * 'ik zou [graag] een rekening [willen] sluiten'
         * 'ik wil een rekening sluiten'
         * 'ik wilde mijn rekening sluiten'
         * 'mag ik een rekening sluiten alstublift'
         * 'ik wil bank rekening drie nul nul sluiten'
         * 'ik wil drie nul nul sluiten'
         */
        | 'sluiten'
        { option := CLOSE_BANK_ACCOUNT; }
        | <bankAccount> 'sluiten'                          { option := CLOSE_BANK_ACCOUNT; }
        | <connector> <bankAccount> 'sluiten'              { option := CLOSE_BANK_ACCOUNT; }
        | <bankAccount> <please> 'sluiten'                 { option := CLOSE_BANK_ACCOUNT; }
        | <bankAccount> <number> 'sluiten'                 { option := CLOSE_BANK_ACCOUNT;
                                                             accountNumber := #2.value; }
        | <number> 'sluiten'                               { option := CLOSE_BANK_ACCOUNT;
                                                             accountNumber := #1.value; }

        // do NOT close bank account
        /* examples:
         * 'niet een rekening sluiten'
         * 'ik wil niet een rekening sluiten'
         * 'ik wil de bank rekening niet sluiten'
         * 'ik wilde mijn bank rekening niet sluiten'
         */
        | <negative> <bankAccount> 'sluiten'               { option := NOT
CLOSE_BANK_ACCOUNT; }
        | <negative> <bankAccount> <number> 'sluiten'      { option := NOT
CLOSE_BANK_ACCOUNT; }
```

```
        | <bankAccount> <negative> 'sluiten'                { option := NOT CLOSE_BANK_ACCOUNT; }
        | <bankAccount> <mine> <negative> 'sluiten'         { option := NOT CLOSE_BANK_ACCOUNT; }
        | <bankAccount> <number> <negative> 'sluiten'       { option := NOT CLOSE_BANK_ACCOUNT; }
        | <negative> <connector> <bankAccount> 'sluiten'    { option := NOT CLOSE_BANK_ACCOUNT; }
        | <negative> <connector> <bankAccount> <number> 'sluiten'          { option := NOT CLOSE_BANK_ACCOUNT; }


        // deposit money
        /* examples:
         * 'ik wil geld storten alstublift'
         * 'ik zou [graag] geld [willen] storten'
         * 'ik wil drie gulden storten in mijn rekening'
         * 'mag ik een gulden storten'
         */
        | 'geld' 'storten'                                  { option := DEPOSIT_MONEY;  }
        | 'geld' <please> 'storten'                         { option := DEPOSIT_MONEY;  }
        | 'storten' 'op' <bankAccount> <number>             { option := DEPOSIT_MONEY;
                                                              accountNumber := #4.value; }
        | 'geld' 'storten' 'op'
                        <bankAccount> <number>              { option := DEPOSIT_MONEY;
                                                              accountNumber := #5.value; }
        | 'geld' <please> 'storten' 'op'
                        <bankAccount> <number>              { option := DEPOSIT_MONEY;
                                                              accountNumber := #6.value; }

        | <number> 'storten'                                { option := DEPOSIT_MONEY;
                                                              amount := #1.value;         }
        | <number> <please> 'storten'                       { option := DEPOSIT_MONEY;
                                                              amount := #1.value;         }
        | <number> <gulden> 'storten'                       { option := DEPOSIT_MONEY;
                                                              amount := #1.value;    }
        | <number> <gulden> <please> 'storten'              { option := DEPOSIT_MONEY;
                                                              amount := #1.value;    }
        | <number> <gulden> 'storten' 'op'
                        <bankAccount> <number>              { option := DEPOSIT_MONEY;
                                                              amount := #1.value;
                                                              accountNumber := #6.value;  }
        | <number> <gulden> 'storten' 'op'
                <mine> <bankAccount> <number>               { option := DEPOSIT_MONEY;
                                                              amount := #1.value;
                                                              accountNumber := #7.value;  }
        // do NOT deposit money.
        /* examples:
         * 'niet storten'
         * 'ik wil geen geld storten'
         * 'ik wil niet geld storten'
         */
        | <negative> 'geld' 'storten'                       { option := NOT DEPOSIT_MONEY; }
        | <negative> 'storten'                              { option := NOT DEPOSIT_MONEY; }
```

```
// withdraw money
/* examples:
 * 'ik wil geld afhalen alstublift van mejn rekening'
 * 'ik zou [graag] geld [willen] afhalen'
 * 'ik wil drie gulden storten in mijn rekening'
 * 'mag ik een gulden storten'
 */
| 'geld' 'afhalen'                             { option := WITHDRAW_MONEY;  }
| 'geld' <please> 'afhalen'                    { option := WITHDRAW_MONEY;  }
| <number> 'afhalen'                           { option := WITHDRAW_MONEY;
                                                 amount := #1.value;           }
| <number> <please> 'afhalen'                  { option := WITHDRAW_MONEY;
                                                 amount := #1.value;           }
| <number> <gulden> 'afhalen'                  { option := WITHDRAW_MONEY;
                                                 amount := #1.value;     }
| <number> <gulden> <please> 'afhalen'         { option := WITHDRAW_MONEY;
                                                 amount := #1.value;     }
| <number> <gulden> 'afhalen' 'van'
                 <bankAccount> <number>        { option := WITHDRAW_MONEY;
                                                 amount := #1.value;
                                                 accountNumber := #6.value;  }
| <number> <gulden> 'afhalen' 'van'
                 <mine> <bankAccount> <number> { option := WITHDRAW_MONEY;
                                                 amount := #1.value;
                                                 accountNumber := #7.value;  }
// do NOT withdraw money.
/* examples:
 * 'niet storten'
 * 'ik wil geen geld afhalen'
 * 'niet geld storten'
 */
| <negative> 'geld' 'afhalen'                  { option := NOT WITHDRAW_MONEY; }
| <negative> 'afhalen'                         { option := NOT WITHDRAW_MONEY; }


// information over bank account

/* examples:
 * 'ik wil weten wat mijn saldo is'
 * 'hoeveel geld heb ik staan [op mijn rekening]
 * 'ik wil informatie van mijn rekening'
 * 'welk bedrag staat er op bank rekening nummer ...'
 * 'wat is [het/de] saldo op bank rekening nummer ...'
 */
| 'informatie'                            { option := BANK_ACCOUNT_INFORMATION; }
| 'informatie' <overVanOp> <connector> <bankAccount>
                                          <number>
                                          { option := BANK_ACCOUNT_INFORMATION;
                                            accountNumber := #5.value; }

| 'informatie' <overVanOp> <bankAccount>  <number>
                                          { option := BANK_ACCOUNT_INFORMATION;
                                            accountNumber := #4.value; }


| 'saldo'                                 { option := BANK_ACCOUNT_INFORMATION; }
| 'saldo' <overVanOp> <connector> <bankAccount> <number>
                                        { option := BANK_ACCOUNT_INFORMATION;
                                          accountNumber := #5.value; }
| 'saldo' <overVanOp> <bankAccount> <number>
                                        { option := BANK_ACCOUNT_INFORMATION;
                                          accountNumber := #4.value; }
| 'staat' 'er' <overVanOp> <bankAccount> <number>
                                        { option := BANK_ACCOUNT_INFORMATION;
                                          accountNumber := #5.value; }
| 'stand'                               { option := BANK_ACCOUNT_INFORMATION; }
| 'stand' <overVanOp> <bankAccount> <number>
                                        { option := BANK_ACCOUNT_INFORMATION;
                                          accountNumber := #4.value; }
| 'stand' <overVanOp> <connector> <bankAccount>
                                        <number>
                                        { option := BANK_ACCOUNT_INFORMATION;
                                          accountNumber := #5.value; }


// do NOT want information of bank account
/* examples:
 * 'ik wil geen saldo weten'
 * 'ik wil geen informatie van mijn rekening'
 */
| <negative> 'informatie' <overVanOp> <connector><bankAccount>
                                { option := NOT BANK_ACCOUNT_INFORMATION; }
| <negative> 'informatie' <overVanOp> <bankAccount>
                                { option := NOT BANK_ACCOUNT_INFORMATION; }
| <negative> 'saldo'
                                { option := NOT BANK_ACCOUNT_INFORMATION; }
| <negative> 'stand'            { option := NOT BANK_ACCOUNT_INFORMATION; }


// information over telebank
/* examples:
 * 'ik wil graag weten hoe telebank werkt'
 * 'wat moet ik doen'
 */
| 'informatie' 'over' 'telebank'        { option := TELEBANK_INFORMATION; }
| 'telebank'                            { option := TELEBANK_INFORMATION; }
| <help>                                { option := TELEBANK_INFORMATION; }


// do NOT want information over telebank
/* examples:
 * 'ik wil geen informatie'
 * 'ik wil geen help'
 */
| <negative> 'informatie' 'over' 'telebank'     { option := NOT TELEBANK_INFORMATION; }
| <negative> 'telebank'                         { option := NOT TELEBANK_INFORMATION; }
| <negative> <help>                             { option := NOT TELEBANK_INFORMATION; }
;
```

# The End