TeleBank

Contents

- Introduction
- Problem Definition
- The Application
- Co-operation and User friendliness
- Design
- Implementation

Introduction - TeleBank

- Automatic bank services
- Initiates a Dialog with the user

Problem definition

- **Requirements**
  - Open a new account.
  - Close an existing account.
  - Deposit money in an account.
  - Withdraw money from an account.
  - Inform how much money there is in an account.
  - Get help/information about the application.
- **Goal**
  - Design and implement TeleBank.
  - Implement TeleBank using SpeechMania framework.
  - TeleBank should be as cooperative and user friendly as possible.

The Application

- Coded in HDDL
  (Dialog Description Language)
- SpeechMania Online system
- Option withdraw money was not implemented.

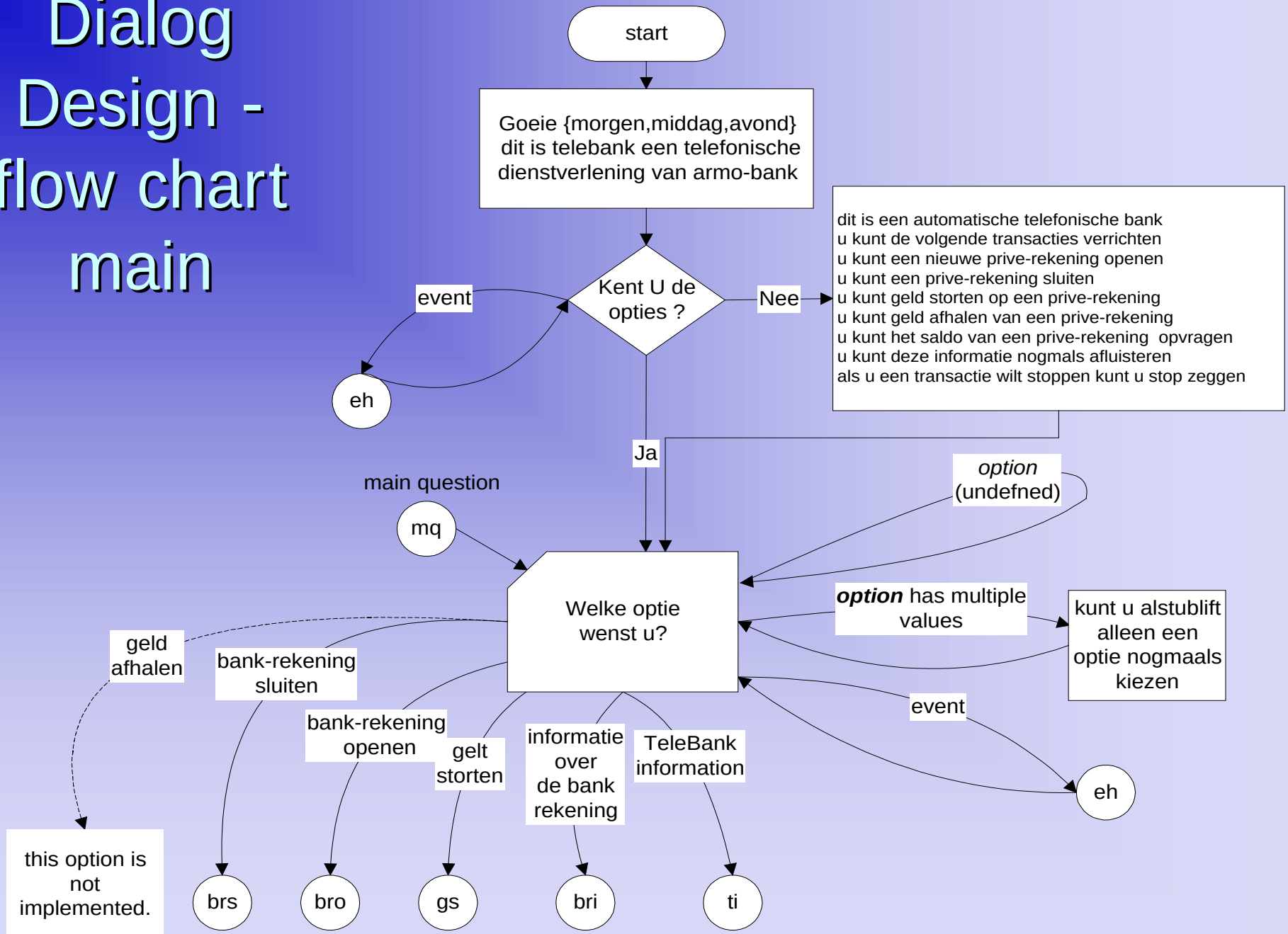
Co-operation and User friendliness

- Minimize closed questions (yes/no)
- Verification Vs confirmation questions
- Allow some user initiative
- Reset option

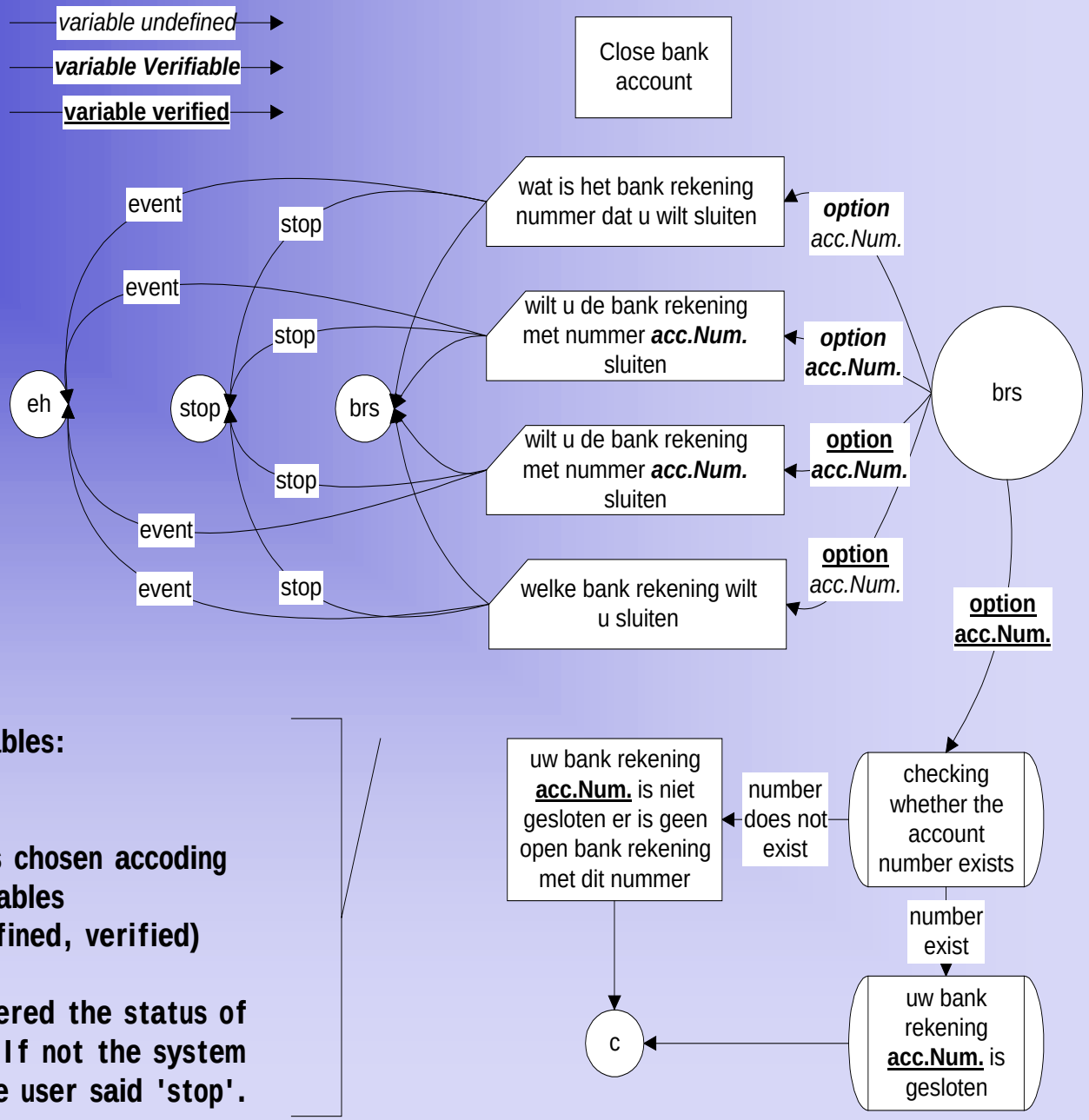
Design

- Dialog Design
- Architecture

Dialog Design - flow chart main



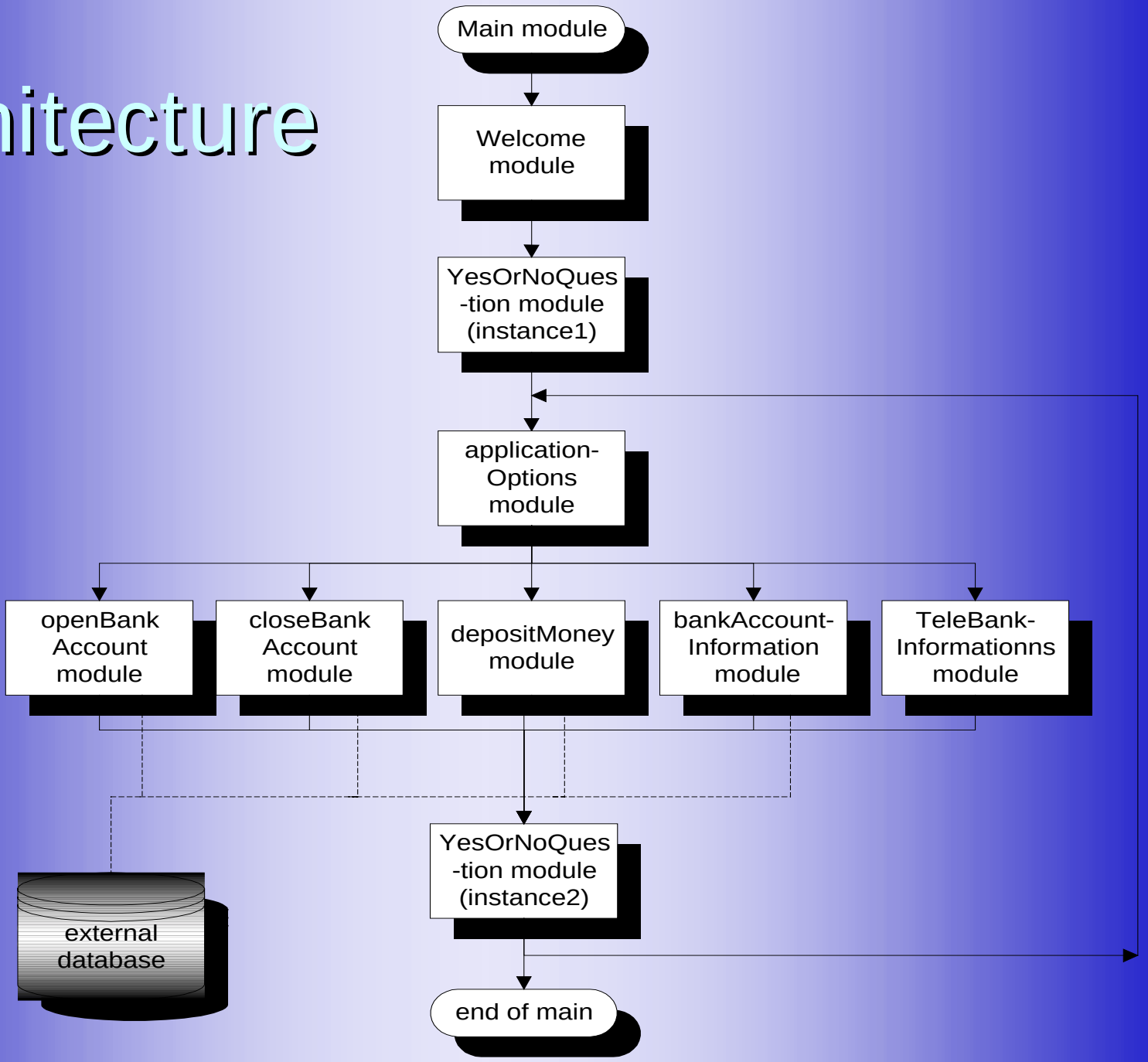
Dialog Design - flow chart close bank account



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'.

Architecture



Implementation

- Rules & variables
- Cond-actions
- Example: Reset option

Rules & Variables

- Variables are the interface between the speech understanding and dialog control
- Variables are linked to rules
- Variable can get values when the rule is ‘activated’
- Variable can be undefined, defined (single,multiple) or verified.
- Example:

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

Questions & Cond-action

- HDDL Dialog is a module
- Dialog is controlled using an event loop
- events can trigger an action (conditionally)
- in a cond-action a question can be asked or the data base can be accessed
- example:

```
COND ( MyVerifiable(option); ) { // option:    verifiable
    QUESTION(YES,NO,option)   {
        INIT {
            "wilt u een nieuwe bank rekening";
            CONFIRMATION(option);
        }
    }
}
COND ( ^^ option && option == OPEN_BANK_ACCOUNT )          {
    BOOLEAN res;       ...
    teleBankDBinstance.openAccount(newAccountNumber,res);
    ...
    OUTPUT { "uw bank rekening is nu geopend. "; .... }
    RETURN; // terminating the subdialog.
}
```

Reset Option

```
DIALOG closeBankAccount (teleBankOptions &option, ….
                                    BOOLEAN &stop)
    …
    ACTIONS
        STOP_COND("closeBankAccount.hdl");
        ….
        COND (MyVerifiable(option); &&
                MyUndefined(accountNumber);) PRIORITY<UNDEFINED {
                ….
                    QUESTION(option,accountNumber) { … }
        }
        …
     END ACTIONS;
END DIALOG closeBankAccount;
```

```
MACRO STOP_COND(fileName)
{
    COND ( MyVerifiable(stop); )  PRIORITY > MULTIPLE    {
        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 {
        RETURN;
    }
}
```

The End