
# FI MU

**Faculty of Informatics**
**Masaryk University**

# Dialogue Interfaces for Library Systems

**by**

**Pavel Cenek**

**FI MU Report Series** **FIMU-RS-2001-04**

 **June 2001**

# Dialogue Interfaces for Library Systems

Pavel Cenek

Department of Information Technology, Faculty of Informatics
Masaryk University Brno, Czech Republic
xcenek@fi.muni.cz

**Abstract.** Basic principles and structure of a dialogue interface are introduced in the paper. Then features specific for library systems are described. Next part deals with the question how to use these features for the design of a dialogue interface for library systems. Finally some specific library systems are introduced and possibilities how to design a dialogue interface for them is discussed.

## 1 Introduction

Natural language processing, speech synthesis and speech recognition are nowadays in the centre of attention of many researchers and software companies. It is easy to understand – using speech is very natural for people and hence these technologies can bring computers closer to people. These technologies can be used for instance for creating so-called dialogue interfaces.

## 2 Dialogue Interface

### 2.1 Principles

To understand our approach to dialogue interfaces we need to mention the Pawlak's definition of the information system ([5]). According to Pawlak an information system consists of objects. These objects are described by means of attributes. A value can be specified for each attribute.

Task of the dialogue interface is to determine an object on the grounds of knowledge of values of some attributes specified by the user. There are many strategies how and in which order to ask the user for the values (see e.g. [2], [3], [4], [7]).

## 2.2 VoiceXML

We need a general purpose tool for flexible design of the dialogue interfaces. In this area there exists a standardised language for describing the dialogues needed for the dialogue interfaces – VoiceXML (see[6]). This markup language ensures platform independence and flexibility of design of dialogues. In VoiceXML, forms play the key role. Forms consist of fields and description how to fill in the fields. It is obvious that this concept is useful for our purposes.

## 2.3 Structure[1]

Structure of an information system with the dialogue interface is shown in Figure 1.

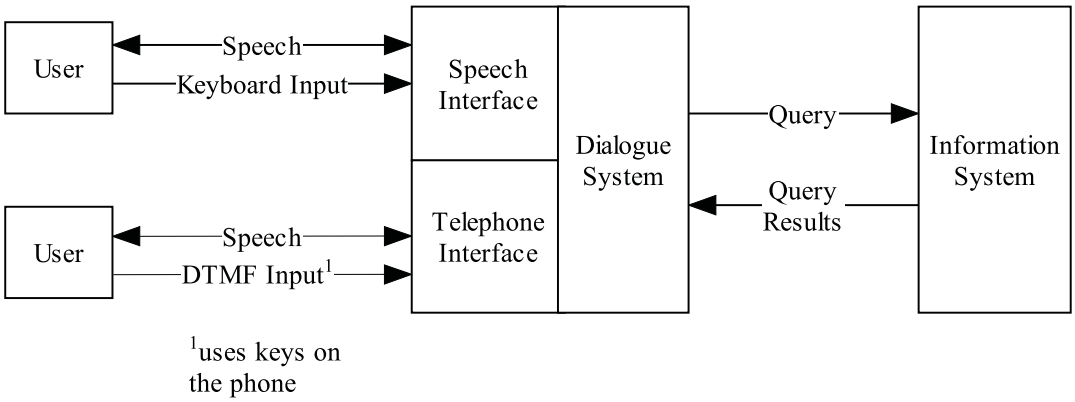


**Fig. 1.** Structure of an information system with the dialogue interface

User can use speech, keyboard or telephone keyboard for the communication with the system. System usually uses synthesised speech for its answers, sometimes it can use text output or a prerecorded sound. Dialogue system can contact the information system to get information about objects which satisfy user's demands. On the grounds of these results dialogue system decides next step.

[1] based on the work described in [1]

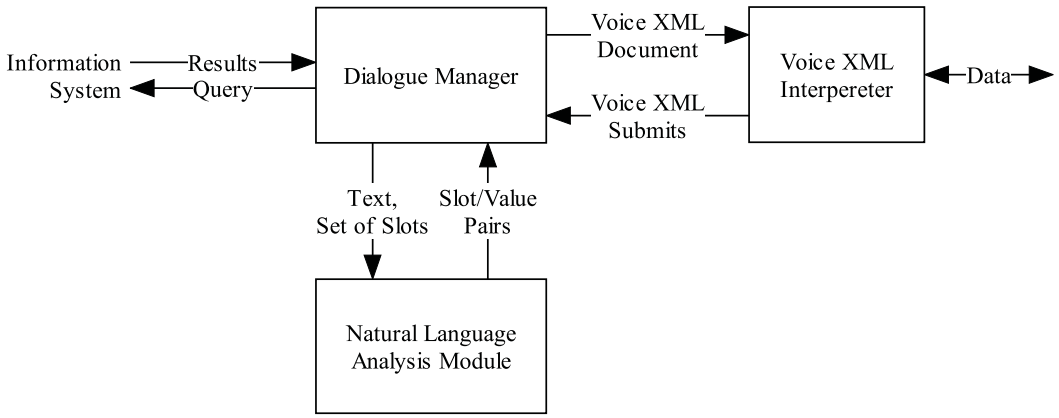


**Fig. 2.** Structure of the dialogue system

**Dialogue System (see Figure 2)** The dialogue manager generates VoiceXML documents which are interpreted by the VoiceXML interpreter and the filled form is returned back to the dialogue manager. Texts from the filled fields are sent to the natural language analysis module. Important information about demanded objects is extracted there. Then the dialogue manager contacts the information system and obtains information about objects which satisfy conditions demanded by the user. On the grounds of these results and selected dialogue strategy the dialogue manager generates next VoiceXML document, which should refine user's demands.

**Natural Language Analysis Module (see Figure 3)** Text of a field of the VoiceXML document is used as input to the NLP controller which is the heart of this module. In addition to the text, also information about slots which should be filled is supplied for this module. Roughly said slots are attributes of some objects from the information system. They are called slots because some values of the attributes should be inserted (filled) into them. To fill them, NLP controller collaborates with the slot manager which knows some information about the slot (e.g. possible values, etc.). If the values cannot be extracted, the text is passed to the error handling module. This module tries to correct mistakes caused by misspelling or incorrect syntax and tries to use thesaurus to replace some words by other words with the equivalent meaning,

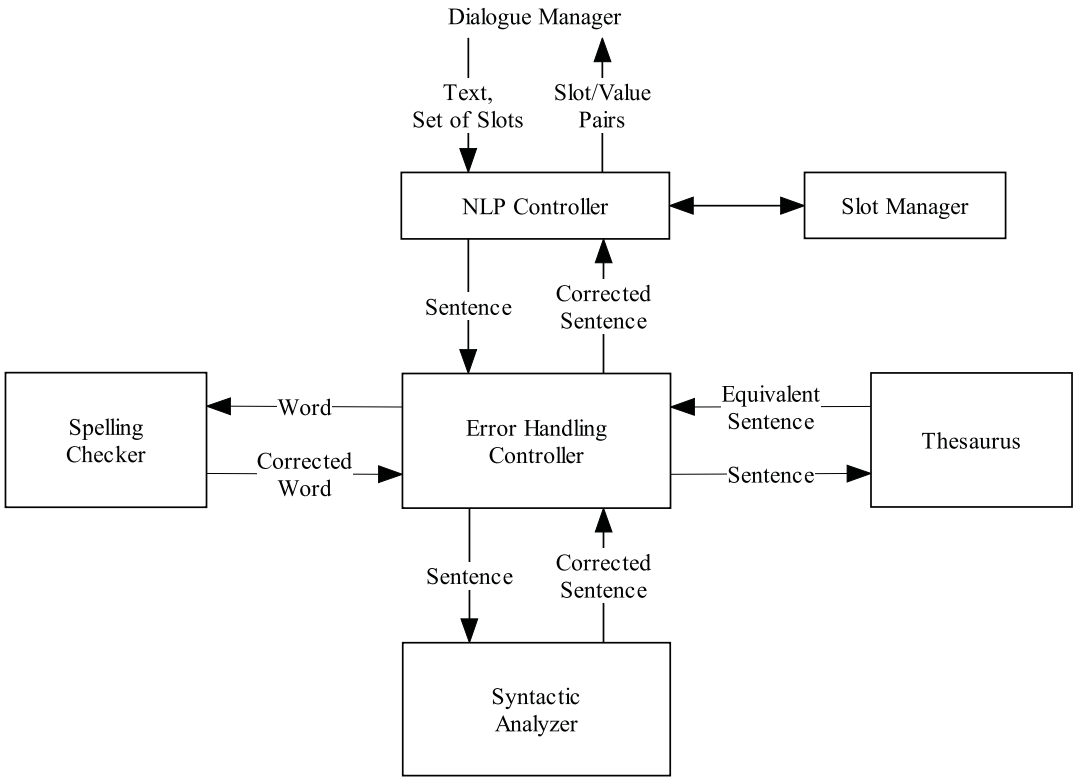


**Fig. 3.** Structure of the natural language analysis module

which the NLP controller understands. In this way corrected sentence is returned back to the NLP controller which tries to fill in the slots once more. If the slots are not filled correctly, an error is returned to the dialogue manager. The dialogue manager should generate another VoiceXML document, which should eliminate errors by restricting user's freedom of input.

# 3 Library Systems

Library systems are used for a complete management of a library. They are able to store and maintain book-funds, journals, articles, multimedia titles, etc. There are many library systems in the world. Although they are implemented in various ways, almost all follow the same system architecture.

## 3.1 Modules

Library systems are typically modular applications with a wide scale of modules. Not all of them have to be used in each system or some of them can be joined together. The most common modules are listed here:

**OPAC** OPAC means Online Public Access Catalogue. This module serves for search in the bibliographic data.

**Cataloguing** This module is intended for creating bibliographic records. A bibliographic record specifies a particular title but not a specific copy of this title.

**Items** Items (i.e.concrete copies of a title) are maintained here. Information like identification of collection, which determines store, signature of the item, which determines placement in the store or information who and for how long is allowed to lend this item, are stored here.

**Borrowing and Reader Tally** Personal data about readers and their borrowing are maintained by this module.

**Series** It is important to process some special information about series, such as time period between two issues, etc. This module is intended for this purpose.

**Interlibrary Borrowing Service** It is possible to borrow an item from another library. The data about this borrowing need a special treatment. Entry about such an item is not piece of the internal system, but the borrowing should be seen in other parts of the internal system. This feature is provided by this module.

**Other Modules** There can be other modules, which are not important for us. Into this group we can put modules for printing, system administration, support for tally of orders of new items, tally of invoices, etc.

# 4 Dialogue Interfaces for Library Systems

Now we will discuss the main features of the design of a dialogue interface for the library systems. Although the design is dependent on a specific library system we can find many common elements.

## 4.1 Access to the Data

A generally valid fact is that we need a good access to the data of the information system to create a dialogue interface. This aspect plays a key role for the successful design of the interface. In many systems we can easily access the database of the information system. The situation in the field of library systems is complicated by the fact, that bibliographic data are not suitable for storing in relational databases for their very variable length. In spite of this fact they are usually stored in a relational database. They are stored in a binary array with an internal structure which is unfortunately invisible from the point of view of the database system. Therefore it is not possible to use SQL for obtaining the structured data. Some systems use its own database engine, which offers some extensions suitable for maintaining bibliographic records. Usually there is no standard or even no documented language for querying the database. In both cases there is usually no easy way to obtain structured data from the library system.

Designers of the library systems were awake to this situation and hence the Z39.50 protocol was designed.

**Z39.50** The Z39.50 protocol is a standard which provides a system independent interface for searching, sorting, gathering information about items in library systems and more. Implementation of this protocol in a dialogue interface could be very valuable, because the interface would be usable for all library systems which support this protocol. Unfortunately not all library systems support it.

**WWW** Other way how to obtain structured data from a library system is to use it’s web-based interface. Today almost all library

systems dispose of this type of user interface. It is possible to send queries in URL (if the get-method for submitting form is used) and then parse returned html document to extract results of the query.

## 4.2 OPAC

As mentioned above OPAC serves for search in bibliographic data. There are usually three modes of the search.

- Search with unambiguous result – This mode applies to search by ISBN, ISSN, bar code, etc.
- List search – In this mode you need bibliographic records sorted by an item. Then you are entering some characters and a cursor is set to the first item which begins with the entered characters.
- Words – In this mode full-text search is performed on selected items. There can be specified other parameters to set more restrictive or attenuated requirements, e.g. wildcards.

## 4.3 UNIMARC

UNIMARC is a cataloguing standard which is used for cataloguing of titles in the Czech Republic. It consists of fields which describe the title. Besides for a laical user usual fields like name, author, etc. there are some fields which can be very useful for creating an intelligent dialogue interface. All these fields apply to characterising titles by means of some specific words or sorting them into categories.

- First of them is the field called subject headwords. This field is an analogy of the paper subject catalogue. These headwords should be organised in a hierarchic structure. This structure can be useful for search for similar titles.
- Second field are keywords. Creation of the keywords is more or less dependent on the consideration of the person who performs cataloguing. The keywords can be used for the definition of the problem scope.
- Third field is the information in UDC. UDC (universal decimal classification) is a formal language which defines some categories and rules for their concatenating. A description in the

natural language exists to each headword and it is possible to find an UDC expression to collocations created in the natural language.
– Last field is thesaurus. It is a lexicon ordered by branches with defined structure (e.g. supertype — subtype, related words, etc.).

Unfortunately the quality of these fields is individual for each library. There are even no standardised dictionaries of keywords, headwords and categories (except of UDC). Libraries define their own standards in the better case, they use no standards at all or completely ignore these fields in the worse case. Therefore using these fields is the question of the design of the dialogue interface for a particular library.

# 5 Aleph 500

Aleph 500 is a library system developed by the Israeli company ExLibris. It is one of the most widespread library systems in the world. It's popularity arises from its powerful functions and regular updates.

## 5.1 System Architecture (see Figure 4)

Queries of the modules are formulated in an universal language and then, after processing, translated to an SQL query. Obtained results are processed and returned to the caller.

## 5.2 Access to the Data

Modules provide a configurable user interface, but the API for them is not public and hence it is impossible to develop own interface and communicate through the modules. Other possibilities are described here.

**Communication with the Database Engine** All data are physically stored in a database. ExLibris chose the database engine from Oracle. Data are not encrypted and hence it is possible to maintain data directly by means of SQL. Each bibliographic record is

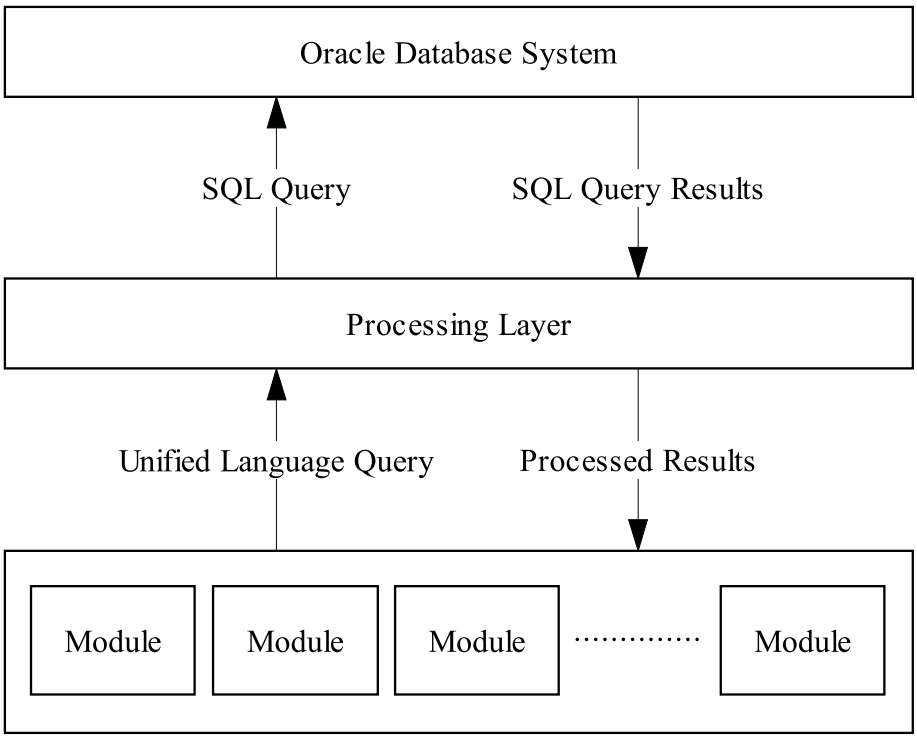


**Fig. 4.** Aleph 500 architecture

stored in a binary array and therefore it is not structured. So it is necessary to write some routines which will restore the structure. The processing layer in Aleph 500 serves for the same task. In the database, indexes are created to the binary records, so that it is possible to search for data efficiently, the problem is just the decoding of the structure.

Other problem could be that the structure of the tables is not documented.

**Using Z39.50** Aleph 500 supports the Z39.50 protocol. This way of communication seems to be the best one. But the implementation of this protocol is not easy.

**Communication via the Web Interface** This way of communication is also possible. The communication goes through the modules of Aleph 500. This way has also its disadvantages. Input of the queries and parsing of the results is a little bit lumpish and more-

over we do not have the possibility to use all features of Aleph 500, if they are not accessible through the web interface.

# 6 Tinlib

Tinlib is a product of the American company Electronic Online Systems International (EOSi). Its development is stopped today and its successor T Series is developed. Nevertheless Tinlib is engaged in libraries in many countries. Popularity of this system arose from its implementation for many platforms (especially UNIX and MS-DOS) and relatively low price.

## 6.1 System Architecture (see Figure 5)

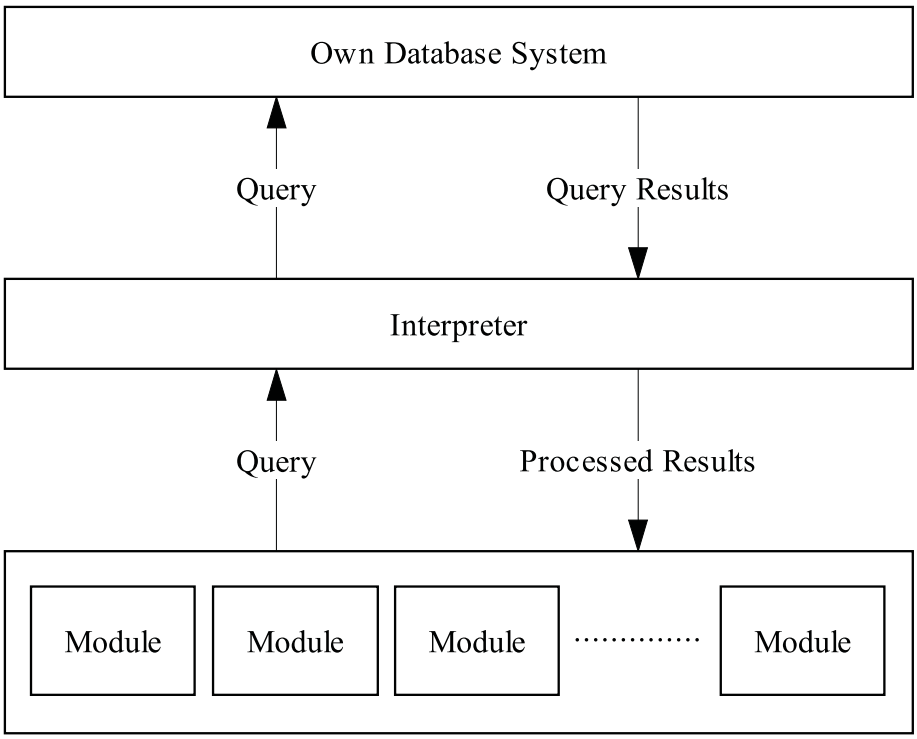


**Fig. 5.** Tinlib architecture

Queries of the modules are processed by the interpreter, which is responsible for fetching data from own non-standard database, its processing and returning back to the caller.

### 6.2 Access to the Data

Access to the data is a little bit problematic in this system. This system was designed for interactive work and provides no functions for non-interactive querying. Communication with the system is provided by means of a text based client.

**Communication with the Database Engine** All data are physically stored in a database, which is non-standard. It is not possible to use SQL. The database consists of many files. Each file contains records about one entity (e.g. all books of one author) and links to other files with relevant records. Thus each bibliographic record is stored in many files, on the other hand the system is hyper-text with many navigation possibilities.

For this approach it is necessary to write some routines which will be able to traverse through the records and collect all needed information.

**Using Z39.50** Unfortunately Tinlib does not support the Z39.50 protocol.

**Communication via the Web Interface** This way of communication is also possible thanks to people from the Palacký University in Olomouc. They implemented a web-based interface for Tinlib called TinWeb. This interface manipulates directly with the files and records stored in them. It uses the get-method for submitting form, so it is possible to formulate the query in URL. This interface supports a mode in which the results are presented in a very simple form suitable for an additional processing. This way is probably easier and better than a direct manipulation with the records.

## 7 Designing of the Dialogue Interface for the Library System of the Masaryk University

The library System of the Masaryk University consists of independent libraries created and maintained individually by each faculty.

All the libraries use TinLib, therefore it was possible to create an universal web-based interface using TinWeb. This interface offers an access to the library catalogues of all faculties. Unfortunately, since the catalogues are created completely independently, they do not share dictionaries of keywords, subject headwords, etc. or some of them even do not support them.

## 7.1 System Design

TinWeb will be used for communication with the library system. All queries will be sent via http protocol and the resultant html pages will be processed, all relevant information extracted and the structure of the bibliographic records restored. After finishing this operation, results of the query will be represented as a list of the bibliographic records.

Dialogues of the dialogue interface will be written in VoiceXML and interpreted by a VoiceXML interpreter. The whole dialogue with the user will be computer driven in the first phase of development. So the dialogue will start with a welcome message and first question, which will be pre-created. Other subdialogues will be created dynamically according to the current answer of the user and according to the results of the prior user's requirements. Dialogue templates will be used for the dynamic creation of the dialogues. It means subdialogues for various specific situations will be created, but the dialogues will not be complete. They will contain some empty places (often called slots) into which some information specific for the current dialogue will be inserted.

## 7.2 Dialogue Structure (see Figure 6)

In every state a grammar is active, which determines what can be said by the user. The user input is processed by means of the grammar, a query created, sent to the library system and the answer processed as described above. Then the dialogue system changes its state according to the arrows.

As said above the dialogue will start with a welcome message. Then the user will select a library in which he or she wants to search. In next step the dialogue can split. If the user knows just

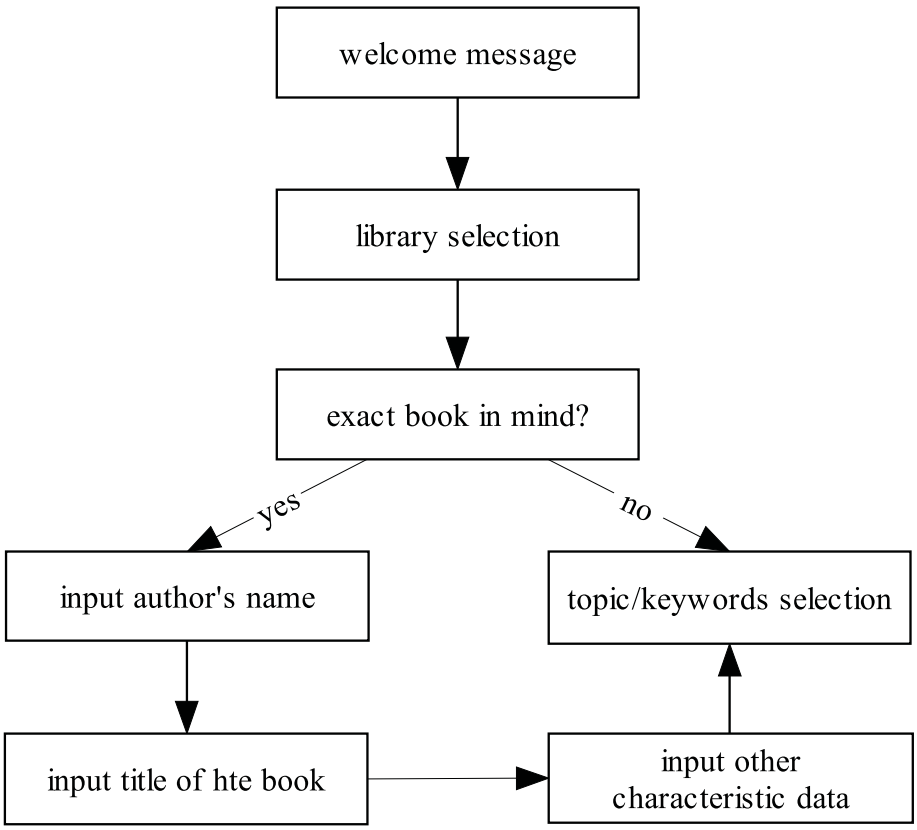


**Fig. 6.** Dialogue structure

a topic a book should be about, system chooses a dialogue which asks for some keywords, subject headwords or other words determining the book. Of course, this will be possible just in the libraries which support it. If the user has an exact book in mind the other branch will be entered. Here subdialogues for determining the author's name and title of the book will be used. These subdialogues should also detect incomplete names when the user is not certain or does not know the name exactly. When the information is not sufficient system should try to ask the user for other information such as producer, publication date, etc. When all alternatives are exhausted or the set of results is reasonably small, the system switches to the browse mode.

In this mode all results are presented as a list. There can be too many items to read them linearly. So user should have a possibility to sort, filter and summarise the results.

In every state, besides the grammar defining user's allowed utterances, there are other grammars active. These other grammars serve for the catch of special phrases which can cause the transition to another state. These transitions are not shown in the Figure 6. The special phrases include:

- Help – provides a help what to do in the current situation. The system tells the user what he or she is allowed or expected to say. Then the dialogue continues as before the help invocation.
- Change – calls a subdialogue which enables to change the value of an attribute. This serves for correction of potential mistakes. Such as correction can cause a big change of results. We need to deal with the problem (see [2]).
- New search – the dialogue will be restarted.
- Exit – the systems terminates the communication.
- Summarise – forces the system to switch to the browse mode. It can be useful at the moment when user does not know more information about the book and wants to browse through the results.

### 7.3 "Intelligence" of the System

The system should offer a better navigation in the library catalogues then today's web-based interface. Even if the web-based interface is often too complicated and confusing and it can be problem to formulate our requirements, it is not an easy objective to reach. There will be many obstacles. Language restriction will be one of them. It is necessary to allow the user to say as many variants as possible. Even if the allowed user's utterances are defined by a grammar, user must have enough freedom of phrasing.

To create a smart dialogue system it is important to be able to derive a logical coherence among the single bibliographic records in the result set. E.g. if the headwords are available, the system should try to detect to which discipline the books in the result set belong to and take note of it in the next part of the dialogue.

## 8 Conclusion

Design of the dialogue interfaces for information systems is a complex task. It requires usage of many technics from the field of the

natural language processing. The design itself is very dependent on the information system. Creating a dialogue interface for a library system is even more complicated, because the data structure and its cohesion is complicated. Aspects of the design were outlined in this work. Looking for other common features and principles of the dialogue interfaces, implementation of the interface for the library system of the Masaryk University and study of resulting dialogues of such systems is the plan for the future work.

## 9 Acknowledgements

I would like to thank Martin Vojnar from the State Research Library in Olomouc and Miroslav Bartošek from the Institute of Computer Science, Masaryk University for providing the information about library systems.

## References

1. Bártek, L.: Library dialogue information system. Proposal of the Design of a Dialogue Interface for Library Systems.
2. Cenek, P.: Design of Formal Dialogue Systems (in Czech). Master's thesis, Faculty of Informatics, Masaryk University Brno, Czech Republic, April 2000.
3. Constantinides, P., Hansma, S., Tchou, C. and Rudnicky, A: A schema-based approach to dialog control. Proceedings of ICSLP. 1998, Paper 637.
4. Heeman, P., Johnston, M., Denney, J., Kaiser, E.: Beyond Structured Dialogues: Factoring Out Grounding. In Proceedings of the International Conference on Spoken Language Processing (ICSLP-98) Sydney, Australia, December 1998, pp. 863-866.
5. Pawlak, Z.: Information Systems, ICS PAS Reports, 338, Warszawa 1978.
6. VoiceXML Forum: VoiceXML. March 2000, Version 1.00.
7. Walker, M., Fromer, J., Narayanan, S.: Learning optimal dialogue strategies: a case study of a spoken dialogue agent for email. In Proceedings of ACL/COLING 98, 1998.

**Publications in the FI MU Report Series are in general accessible via WWW and anonymous FTP:**

`http://www.fi.muni.cz/informatics/reports/`
`ftp ftp.fi.muni.cz (cd pub/reports)`

**Copies may be also obtained by contacting:**

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**