Vladimir Kurbalija

IMPLEMENTATION OF DECISION SUPPORT SYSTEMS USING CASE BASED REASONING TECHNOLOGY

~ Master Thesis ~

Mentor:
dr Zoran Budimac

Novi Sad
2006.
## Contents

1. Introduction ............................................................................................................................... 7

2. Foundations of CBR .................................................................................................................... 11

2.1. Knowledge ............................................................................................................................... 11
2.2. Basic Concepts ....................................................................................................................... 11
2.3. Extending Basic Concepts of Case-Based Reasoning ............................................................ 14
   2.3.1. Case completion .................................................................................................................. 14
   2.3.2. Information Entities ............................................................................................................ 15
   2.3.3. Acceptance ........................................................................................................................... 16
   2.3.4. The general case .................................................................................................................. 17
2.4. Case Retrieval Nets ................................................................................................................... 19
   2.4.1. Formal Model of CRNs ........................................................................................................ 21
2.5. Types of CBR Applications and Current State ...................................................................... 22
   2.5.1. Classification ....................................................................................................................... 23
   2.5.2. Diagnosis ............................................................................................................................... 23
   2.5.3. Configuration and Design .................................................................................................. 24
   2.5.4. Planning ............................................................................................................................... 24
   2.5.5. Decision support .................................................................................................................. 25
   2.5.6. Information Searching ........................................................................................................ 25

3. CaBaGe – Case Base Generator .................................................................................................. 27

3.1. Implementation of CaBaGe ...................................................................................................... 29
   3.1.1. Class List ............................................................................................................................... 29
   3.1.2. Class Element ...................................................................................................................... 30
   3.1.3. Class IEEElement ................................................................................................................ 30
   3.1.4. Classes stringIEElement, intIEElement and floatIEElement ............................................ 31
   3.1.5. Class CaseElement .............................................................................................................. 32
   3.1.6. Class Arc ............................................................................................................................... 33
   3.1.7. Class activationElement ...................................................................................................... 33
   3.1.8. Class Attribute ..................................................................................................................... 34
   3.1.9. Class Lexer ........................................................................................................................... 38
   3.1.10. Class CaseRetrievalNet ..................................................................................................... 39
   3.1.11. Class Frame1 .................................................................................................................... 44
   3.1.12. Class Application1 .......................................................................................................... 49
3.2. An Example ............................................................................................................................. 49
3.2.1. The Structure of Case Retrieval Net ................................................................. 50
3.3. Application of the System .................................................................................. 53

4. Case Based Reasoning in Medical Domain ............................................................ 55

4.1. Characteristics of Medicine .............................................................................. 55
4.2. Advantages of CBR in Medicine ..................................................................... 57
  4.2.1. Similarity to Human Cognitive Processes ..................................................... 57
  4.2.2. Usage of Experience .................................................................................... 57
  4.2.3. Duality of Objective and Subjective Knowledge ......................................... 57
  4.2.4. Automatic Learning of Subjective Knowledge .......................................... 58
  4.2.5. Easy System Integration .............................................................................. 58
4.3. Problems of CBR in Medical Applications ....................................................... 58
4.4. Techniques for Medical CBR Systems .............................................................. 59
  4.4.1. Prototypes ................................................................................................... 59
  4.4.2. The Abstraction of Time Depended Parameters ......................................... 59
4.5. CBR Systems in Medicine .............................................................................. 60
  4.5.1. Diagnosis ................................................................................................... 60
  4.5.1.1. CASEY ................................................................................................... 60
  4.5.1.2. MEDIC .................................................................................................. 60
  4.5.1.3. PSIQ ....................................................................................................... 60
  4.5.1.4. GS.52 ...................................................................................................... 61
  4.5.1.5. Other Diagnostic Systems ....................................................................... 61
  4.5.2. Classification .............................................................................................. 62
  4.5.2.1. ProtoISIS ................................................................................................ 62
  4.5.2.2. MacRad .................................................................................................. 62
  4.5.2.3. MNAOMIA ............................................................................................. 62
  4.5.2.4. Other Classification Examples .................................................................. 63
  4.5.3. Planning ...................................................................................................... 63
  4.5.3.1. FLORENCEnet ....................................................................................... 63
  4.5.3.2. ICONS: Antibiotic Therapy Advice ......................................................... 64
  4.5.3.3. Therapy Support ...................................................................................... 65
  4.5.3.4. Planning of Personal or Material Resources ............................................. 65
  4.5.4. Tutoring ..................................................................................................... 65
  4.5.4.1. PlanAlyzer .............................................................................................. 65
  4.5.4.2. Neurology Trainer .................................................................................. 65
  4.5.5. Prognosis .................................................................................................... 65
  4.5.5.1. COSYL ................................................................................................... 65
  4.5.5.2. ICONS: Prognosis of Kidney Function Courses .................................... 66
  4.5.5.3. TeCoMED ............................................................................................... 67
  4.5.5.4. DIRAS .................................................................................................... 68

5. CaBaGe in Multiple Sclerosis Diagnosis .............................................................. 69

5.1. Multiple Sclerosis ............................................................................................ 69
  5.1.1. What is Multiple Sclerosis? ....................................................................... 69
  5.1.2. Symptoms and Signs ............................................................................... 72
  5.1.3. Diagnosis of MS ......................................................................................... 73
5.2. Selection of Features ....................................................................................... 74
5.3. The Application of CaBaGe in MS Diagnosis .................................................. 78
  5.3.1. Data Analysis .............................................................................................. 79
  5.3.2. The CaBaGe with Real World Data ............................................................. 82
6. Conclusion ............................................................................................................................... 85

Appendix I – UML Diagram of CaBaGe ................................................................. 89

Sažetak........................................................................................................................................ 91

Index of Figures ...................................................................................................................... 95

Sadržaj slika............................................................................................................................. 97

Index of Tables and Listings .................................................................................................... 99

Sadržaj tabela i listinga ........................................................................................................... 101

References ............................................................................................................................... 103

Biografy ..................................................................................................................................... 109

Biografija.................................................................................................................................... 111
1. Introduction

The main topic of this master thesis is the utilization of particular technique of artificial intelligence, called “Case Based Reasoning” (CBR) [Lenz et al. 1998], [Budimac, Kurbalija 2001], [Kurbalija, Ivanović 2002], [Kurbalija, Ivanović, Budimac 2003], [Burkhard 2001], [Burkhard, Richter 2001], [Bartsch-Spörl 1999], [Lenz, Glintschert 1999] for realization of a basic system, which can be used for implementation of different real world decision support systems. At the beginning, the foundations of CBR technology will be given. After that, a basic system will be completely described. At the end, a realized real world application will be shown.

Speaking in general, case based reasoning is a problem solving technique, where new problems are solved by adapting solutions that worked for similar problems in the past. Cases, generally, can be represented in the form \((\text{problem, solution})\) where the first element represents a description of the problem, and the second one a successful solution of the same problem in the past. The basic scenario for mainly all CBR applications looks as follows:

\[
\text{In order to find a solution of an actual problem, one looks for a similar problem in an experience base, takes the solution from the past, which is the most similar to the actual problem, and uses it as a starting point to find the solution of the actual problem.}
\]

The main advantage of this technology is that it can be applied to almost any domain [Rey 2002]. CBR system does not try to find rules between parameters of the problem; it just tries to find similar problems (from the past) and to use solutions of them as a solution of an actual problem. So, this approach is extremely suitable for less examined domains – for domains where rules and connections between parameters are not known. The second very important advantage is that CBR approach to learning and problem solving is a very similar to human cognitive processes – people take into account and use past experiences to make future decisions.

The system which is the main result of master thesis, “CaBaGe – Case Base Generator”, is designed as a decision support system totally independent of the domain. In this system, the case is described as an array of attributes and for every case the values of some (or all) attributes is given. This means that the description of the problem is given with the values of some attributes depending how the certain domain is modelled. The solution of the case is one additional attribute. Cases can be taken from any domain (medicine, administration, engineering…) but they must be correctly modelled in order they can be represented with the array of attributes.

At the beginning of application, a set of solved problems (cases) is presented to the system in order to create some initial knowledge of the system. On the basis of these presented cases,
the system creates *Case Retrieval Net (CRN)* [Lenz, Burkhard 1996], [Lenz, Burkhard 1997], [Lenz 1996] which is the basic component of the system. CRN has been developed at the Humboldt University, Berlin, and represents a set of data structures and algorithms for a very efficient search of large databases.

After loading cases and creating CRN, the system expects that the user enters a current problem which is structurally equal to all previous cases – an array of values of the attributes. It is not obligatory to enter the values of all attributes (if some values are not known) because the system has capability of information completion. The process of proposition of the solution of an actual problem is realized as the process of activation propagation through the case retrieval net.

The rest of thesis has been arranged as follows:

- **Chapter 2 - Foundations of CBR**
  In this chapter, the foundations, basic concepts, some basic definitions and necessary facts of CBR technology are given. Also, the memory structure used in the implementation (*Case Retrieval Net*) is introduced here. At the end of the chapter, a short overview of the realized CBR applications according to their application type is given.

- **Chapter 3 - CaBaGe – Case Base Generator**
  In chapter 3, the implementation of *CaBaGe* system has been described in detail. All classes have been described, but the listings include the most important methods. At the end of the chapter, the description of the use of system including one illustrative example has been shown.

- **Chapter 4 - Case Based Reasoning in Medical Domain**
  In chapter 4 the possibilities for applying CBR technology in medical domain is presented. Some advantages and problems are also discussed here. Also, some techniques which are characteristic for an application of CBR in medical domain are shown. At the end of the chapter, different application types are described, and some illustrative real world CBR medical applications are shown.

- **Chapter 5 - CaBaGe in Multiple Sclerosis Diagnosis**
  In this chapter, the application of *CaBaGe* with a real world data in diagnosis of multiple sclerosis is presented. At the beginning of chapter, some characteristics of multiple sclerosis, together with signs and symptoms are given. After that, the problem of selection the features is described. At the end of the chapter, the data analysis and the application of the system is described.

- **Chapter 6 – Conclusion**
  This chapter concludes this thesis and gives some possibilities for the further work.

At this moment, I would like to thank warmly to all the members of the committee for their patience and valuable suggestions regarding this thesis. But, my special thanks have been addressed both to prof. dr. Mirjana Ivanović and prof. dr. Zoran Budimac who patiently guided me through my research work in this area and provided me with a lot of literature, equipment and a large number of their advices pending my professional career.
I would also like to express my special gratitude to prof. dr. Hans-Dieter Burkhard, who provided me with a lot of information during my five months stay at the Humboldt University, Berlin in 2002. His great knowledge and excellent experience in the field of case-based reasoning were of extreme value to me.

I would like to thank warmly to all of my colleagues at the Computer Science Lab at the Department of Mathematics and Informatics in Novi Sad for their help in solving some of the problems relating to this respective matter. Also my thanks go to the colleagues from the Institute of Neurology, Novi Sad, especially to dr. Marija Semnic who provided the valuable data for the application of the system.

Finally, I would also like to acknowledge my special thanks to my wife Tatjana for her endless patience, understanding and great help in professional matters. My special thanks are also traced to my mother Mirjana for her constant support, encouragement and a great help in solving a number of the English language dilemmas. At last, I am very thankful to the Makuh family for their support.

Novi Sad
05. September 2006

Candidate
Vladimir Kurbalija
2. Foundations of CBR

Generally speaking, case-based reasoning (CBR) is applied for solving new problems by adapting solutions that worked for similar problems in the past.

This technique is relatively novel and promising. In the past ten years, CBR technique was frequently used in different domains. Furthermore, CBR approach is used in different problem types and systems that require some kind of intelligent behaviour [Burkhard 2001], [Burkhard, Richter 2001], [Bartsch-Spörl 1999].

In this chapter, some formal or informal definitions of the basic concepts will be given. Definitions are taken from [Lenz et al. 1998].

Knowledge

Knowledge can be understood as an informal notion describing something that a human, a formal system or a machine can possibly use in order to perform a certain task (to solve a problem). In order to use knowledge some entities need to have an access to it and to know how to apply it in solving problems.

The way the knowledge is expressed is the type of knowledge representation, which consists of certain data structures and some additional operators that allow changes on the data structure. The most common data structure in case-based reasoning is the attribute-value representation. Every attribute is given by:

- a name \( A \)
- a usually finite set \( \text{DOM}(A) \) called the domain of the attribute \( A \)
- a variable \( x_A \)

For a finite set \( A_i, 1 \leq i \leq n \), of attributes an attribute-value vector is an n-tuple \( (a_1, \ldots, a_n) \) such that \( a_i \in \text{DOM}(A_i) \). However, if one wants to deal with incomplete knowledge (which is occurring frequently in case-based reasoning), one must allow that some variables exist without values (value unknown).

Basic Concepts

Case-based reasoning is a problem solving technology. The basic scenario for case-based reasoning, from the simplified point of view, looks as follows:
In order to find a solution of an actual problem one looks for a similar problem in an experience base, takes the solution from the past and uses it as a starting point to find a solution of the actual problem.

A general desire in every knowledge-based system is to make use of the past experience. An experience may be concerned with what was true or false, correct or incorrect, more or less useful. It can be represented by a rule, constraint, some general law or advice or simply by saving a past event. From all of this the main idea of the case can be obtained. The case is some recorded situation where the problem was totally or partially solved [Minor 1998], [Minor 1999], [Minor 2000]. In its simplest form, the case is represented as an ordered pair:

\((\text{problem}, \text{solution})\)

The existence of the case means that the corresponding episode happened in the past. This episode contains some decisions that a decision maker finds useful. However, somebody else may not be happy with such a case and neglect it. From this follows that cases must be selected carefully, so different categories of cases can exist: good, typical, important, misleading or unnecessary.

A case base is a set of cases, which is usually equipped with some additional structure. A structured case base is usually called a case memory.

In many practical applications, one deals with problems with incomplete information. Both, the problem and the solution part in the case may be incompletely described. In these situations we talk about incomplete cases. The case completion is another task that can be solved by using case-based reasoning technology.

The next very important concept in the case-based reasoning is similarity. While in classical databases information can be retrieved by using only exact matches, in the case-based reasoning cases can be retrieved by using even inexact matches. The notion of similarity is equivalent to a dual mathematical concept - distance.

In the functional way similarity can be defined as a function:

\[ \text{sim} : U \times CB \rightarrow [0, 1] \]

where \(U\) refers to the universe of all objects, while \(CB\) refers to the case base (just those objects which were examined in the past and saved in the case memory). The higher value of the similarity function means that these objects are more similar. The boundary case is \(\text{sim}(x,x) = 1\), which means that each object is the most similar to itself.

Retrieval is a basic operation in databases and therefore in the case base too. A query to a database retrieves some information by an exact match by using a key, while a query to a case-based reasoning system presents a problem and returns a solution by using inexact matches with the problems from the cases in the case base.

As in databases, trees play a major role in efficient retrieval. Some examples of retrieval structures are: kd-trees (k-dimensional trees), case retrieval nets, discrimination nets etc.

The simplest way to use retrieved case is simply to take the unchanged solution of that case as the solution to the actual problem. However, in many applications even small differences
between the actual and the case problem may require significant modifications to the solution. Making the appropriate changes to the case solution is called a case adaptation. A general demand is that the solution of a similar problem should be easily adapted to a solution of an actual problem.

The knowledge container is the structural element, which contains some quantity of knowledge. The idea of the knowledge container is totally different from the traditional module concept in programming. While the module is responsible for a certain subtask, the knowledge container does not complete the subtask but contains some knowledge relevant to many tasks. Even small tasks require the participation of each container. The concept of the knowledge container is similar to concepts of the nodes and propagation rules in neural networks [Park 2004].

In case-based reasoning we identify the following knowledge containers [Richter 1995]:

a) the vocabulary used;
b) the similarity measure;
c) the case base; and
d) the solution transformation.

In principle, each container can carry almost all knowledge available. From a software engineering point of view there is another advantage of case-based reasoning - the content of the containers can be changed locally. This means that manipulations on one container have little consequences on the other ones. As a consequence, maintenance operations [Iglezakis 2001], [Reinartz 2000], [Wendler 2001] are easier to be performed then on classical knowledge based systems.

The task of machine learning is to improve a certain performance by using some experience or instructions. In inductive learning, problems and good solutions are presented to the system. The major desire is to improve a general solution method in every inductive step. Machine learning methods can be used in order to improve the knowledge containers of a case-based reasoning system (the case base, similarity measures and the solution transformation). However, one of the greatest advantages of the case-based reasoning system is that it can learn even through the work with users modifying some knowledge containers.

The case based reasoning system has not only to provide solutions to problems but also to take care of other tasks occurring when it is used in practice. The main phases of the case-based reasoning activities are described in the CBR-cycle of [Aamodt, Plaza 1994] in Figure 2.1.

In the retrieve phase the most similar case (or k most similar cases), to the actual problem case, is retrieved, while in the reuse phase some modifications to the retrieved case is done in order to provide better solution to the problem (case adaptation). As the case-based reasoning only suggests solutions, there may be a need for a correctness proof or an external validation. That is the task of the phase revise. In the retain phase the knowledge, learned from this problem, is integrated in the system by modifying some knowledge containers.
Extending Basic Concepts of Case-Based Reasoning

The case-based reasoning was developed in the context and in the neighbourhood of problem solving methods, learning methods (Machine Learning, Statistics, Neural Networks) and retrieval methods (Data Bases, Information Retrieval). It has inherited the concepts of "problem" and "solution" and a notion of "similarity" based on the distance.

The basic concepts of the case-based reasoning can be extended in the following way:

- we will use acceptance instead of similarity, because acceptance includes similarity but also other approaches related to "expected usefulness", "reminds on" etc.
- we will use the term case completion instead of solution (including solutions of problems but also a proposal of intermediate problem solving steps).
- we will consider cases as sets of information entities instead of vectors (whereby this approach includes vectors as well as textual documents).

Case completion

Case-based reasoning is considered as a problem solving method; given a problem we have to find the solution. This leads to a view where cases are split into the problem and the solution part. Given a new problem we search for related problems in the case memory and adapt their solution for the new problem. However, problem solving usually does not start with a complete problem description, which makes the identification of a final solution more difficult.
Nevertheless, the new case is often talked of as a given entity when only the first impression of the underlying task is given. Instead of this, we want to keep attention to the fact that the whole process of completing the task up to the final solution. The consequence is that the resulting case usually depends on a number of decisions. These decisions are initially open; they depend on future human decisions. Depending on different possible decisions, we can end up with different cases.

Most practical tasks are performed as processes with a lot of intermediate steps. Each of these steps could be considered as a single new problem-solving step. The question arises, as to whether we need different sets of cases to support each of these steps. This would mean splitting the whole story of the process into different cases for a later usage by the case-based reasoning. Case completion is an attempt to avoid such approach. A case should be a description of the whole performance of the task with all steps, and it should be useful for later tasks at intermediate situations, too. The main consequence of the case completion is that we do not actually need any distinction between the problem and the solution part in the case.

**Information Entities**

Information entities are atomic constituents of cases and queries. We consider a case as the result of the case completion process. Each step of that process adds some information entities. The current situation during the elaboration of a task is described by the information entities known at the time point. The final case, as it later may appear in the case memory, is a completed set of information entities.

The collected information entities result from the real world (an outcome of the test, a decision in an intermediate design step, etc). They are not a direct result of the case-based reasoning process - case-based reasoning is used to propose the next step (some test, the next design decision etc).

The number of information entities in a case may be variable. It is up to a human decision at which time point the task is finished.

The information entities, which are later used for retrieval, (which appear in the case memory) may be only a subset of the information entities collected during the case completion. These information entities (in the case memory) serve as the indexes for retrieval. The case memory consists of cases, which are sets of such information entities. These cases may then point to related complete descriptions in a collection of "full cases".

The information entity is an atomic part of a case or query. \( E \) denotes the set of all information entities in a given domain.

- A case is a set of information entities: \( c \subseteq E \).
- The set of cases (in the case memory) is denoted by \( C, C \subseteq P(E) \).
- A query is a set of information entities: \( q \subseteq E \).

In many applications, the information entities are simply attribute-value pairs. Some examples of information entities are:

\(<\text{price}, 1000\>, \ <\text{price}, 324\>, \ <\text{colour}, \text{blue}\>, \ <\text{mass}, 54\ \text{kg}\>.

We say that the first two information entities are comparable (because they have the same attribute) while the other information entities are not comparable.

This causes a structuring of the set $E$ into disjoint sets $E_A$, where $E_A$ contains all attribute-value pairs from $E$ for a certain attribute $A$.

If cases and queries are considered as attribute-value vectors over a finite set of attributes $A_1, \ldots, A_n$, then each case or query may contain at most one information entity from each $E_A$.

**Acceptance**

We want to use the association of information entities for reminding cases with the expectation that these cases are useful for a given query. Usefulness of a case in the case completion process depends on real world circumstances that are not completely known at the retrieval time. This means that usefulness is only a posterior criterion. The retrieval from the case memory will be based on matching of certain information entities. Usefulness of former cases is not restricted to those cases that are similar to a given query for all information entities. Cases may contain information entities that have no counterpart in the query. It is also possible that some information entities of the query are not present in the useful case.

Some special desirable properties of acceptance are following:

**P1:** A case might be acceptable for a query even if there exists some information entities that are not comparable.

**P2:** A case might be unacceptable for a query if there exists an unacceptable information entity (a fix budget may forbid expensive offers).

**P3:** The same information entity may have different importance for different cases (for example information entity $<\text{sex, male}>$ has different importance in pregnancy testing and in testing for influenza).

**P4:** The same information entity may have different importance for different queries according to the user's intentions (for example material has different priorities in design queries).

**P5:** Information entities may not be independent of each other.

In order to provide better understanding of acceptance we will define the preference relation $(\geq_q)$, over the set of all potential cases, in the following way:

$c' \geq_q c''$ if case $c'$ is preferable to case $c''$ in regard to the query $q$.

At the beginning, the definitions of acceptance functions will be given for cases and queries represented as feature vectors (and not as a set of information entities) because it is a more convenient form of representation. Both the queries and the cases are considered as feature vectors $(a_1, \ldots, a_n)$, where $a_i$ specifies the value for the $i$-th attribute $A_i$.

Formally, there is no difference between case vectors $c = (c_1, \ldots, c_n)$ and query vectors $q = (q_1, \ldots, q_n)$.
Definition 1. (Global Acceptance Function): Let \( U := \text{dom}(A_1) \times \ldots \times \text{dom}(A_n) \) denotes the set of all queries and cases. The acceptance of a case for a query is expressed by a global acceptance function

\[
\text{acc} : U \times U \rightarrow \mathbb{R}
\]

such as that a higher value \( \text{acc}(q, c) \) denotes a higher acceptance of the case \( c \) for the query \( q \). The preference relation \( \preceq_q \subseteq U \times U \) induced by a query \( q \in U \) is defined by

\[
c' \preceq_q c'' \iff \text{acc}(q, c') \geq \text{acc}(q, c'').
\]

Definition 2. (Local Acceptance Functions for Attributes): A local acceptance function \( \sigma_i \) for the attribute \( A_i \) is defined over the domain \( \text{dom}(A_i) \):

\[
\sigma_i : \text{dom}(A_i) \times \text{dom}(A_i) \rightarrow \mathbb{R}
\]

such as that a higher value \( \sigma_i(q_i, c_i) \) denotes a higher acceptance of the value \( c_i \) (of a case \( c \)) for the value \( q_i \) (of a query \( q \)).

Global acceptance function can be obtained from local acceptance functions by a related composition function as follows:

Definition 3. (Composite Acceptance Function): A global acceptance function \( \text{acc} \) is called composite if it is composed by a composite function \( \Phi : \mathbb{R} \times \ldots \times \mathbb{R} \rightarrow \mathbb{R} \) from related local acceptance functions \( \sigma_i \):

\[
\text{acc}((q_1, \ldots, q_n), (c_1, \ldots, c_n)) = \Phi(\sigma_1(q_1, c_1), \ldots, \sigma_n(q_n, c_n)).
\]

The natural demand is that composition functions must be monotonously increasing.

An example for the composition of local acceptance values is given by a weighted sum with only positive weights \( g_i \) (because of monotonously increasing composition function):

\[
\text{acc}((q_1, \ldots, q_n), (c_1, \ldots, c_n)) = \sum g_i \cdot \sigma_i(q_i, c_i)
\]

Addition is widely used for the combination of local values. It has an intuitive interpretation concerning acceptance in the sense of "collecting arguments" in favour of something. Positive arguments have positive values, while negative arguments are expressed by negative values. The value 0 does not change the result, so unimportant or unknown attributes can be treated as value 0. Therefore, properties \( P_1 \) and \( P_2 \) are satisfied. However, a more general combination is necessary to satisfy properties \( P_3, P_4 \) and \( P_5 \).

The general case

In this section we will generalize the previous concepts in order to satisfy the properties \( P_1, \ldots, P_5 \).

Here, queries and cases are considered as sets of information entities. A weighted query is the generalization of this concept.

Definition 4. (Weighted query): The weighted query assigns an importance value to each information entity by a function:
\[ \alpha_q : E \to R, \]  

(6)

where \( \alpha_q(e) \) denotes the importance of the information entity \( e \) for the query \( q \).

High values indicate a high importance; negative values indicate the rejection of related cases. The value 0 is used as a neutral element (\( \alpha_q(e) = 0 \), means that information entity \( e \) is unimportant to the query \( q \)). Of course, values for \( \alpha_q(e) \) can be simply taken from the set \( \{0, 1\} \), where the 0 value means "\( e \) is unimportant for the \( q \)", while the 1 value means "\( e \) is important for the \( q \)".

By using \( \sigma \) (Definition 2.) we can compute the acceptance of the information entity \( e' \) from the case for a single information entity \( e \) of a query. However, a query may contain several information entities \( e \) such that \( \sigma(e, e') \) is defined for the single information entity \( e' \). The question is: how these values can be combined to a single value for \( e' \) which expresses the resulting acceptance value of \( e' \) for that query.

**Definition 5. (Local Accumulation Function):** Let \( E_e = \{e_1, \ldots, e_n\} \) denote the set of all information entities to which the information entity \( e \) is comparable concerning acceptance (\( E_e = \{e' | \sigma(e', e) \) is defined\)). The local accumulation function \( \pi_e \) for \( e \) is a function:

\[
\pi_e : R \times ... \times R \to R 
\]

such that \( \pi_e(a_1, \ldots, a_n) \) denotes the accumulated acceptance in \( e \). The values \( a_i \) denote the contributions of the information entities \( e_i \in E_e \) according to their occurrence in the query \( q \) and their local acceptance computed by \( \sigma(e, e) \).

The contributions are computed by a function:

\[
f : R \times R \to R 
\]

such that \( a_i = f(\alpha_q(e_i), \sigma(e, e)) \).

We consider the retrieval of the cases as a process of reminding. Reminding may be of a different strength; cases are in competition for retrieval according to the query. The cases receiving more reminders of more strength are the winners. The strength (importance, relevance) of reminding for an information entity \( e \in c \) is given by a relevance function:

**Definition 6. (Relevance Function):** The relevance between information entities and cases is described by a relevance function:

\[
\rho : E \times C \to R. 
\]

(9)

The relevance \( \rho(e, c) \) is considered as a measure for the relevance of information entity \( e \) for the retrieval of a case \( c \). \( \rho(e, c) \) is defined if and only if \( e \in c \).

Negative values \( \rho(e, c) \) may be used in the meaning "do not retrieve the case \( c \) if one asks for the information entity \( e \)".

The acceptance of a case \( c \) for the query \( q \) is accumulated from the contributions of the information entities \( e \in c \) according to their relevancies \( \rho(e, c) \). The contributions \( p_e \) of the
information entities are computed by their accumulation functions $\pi_e$ as described in the definition 5. The accumulation in the cases is evaluated by Global accumulation function.

**Definition 7. (Global Accumulation Function):** The global accumulation function $\pi_c$ has the form:

$$\pi_c : R \times \ldots \times R \rightarrow R$$  \hspace{1cm} (10)

for $c = \{e_1, ..., e_k\}$. The accumulated acceptance of the case $c$ regarding its constituting information entities is then computed by $\pi_c(p_1, ..., p_k)$, where $p_i$ is the contribution of the information entity $e_i \in c$. This contribution $p_i$ depends on $\rho(e_i, c)$ and another real value $x_i$ assigned to $e_i$ ($x_i$ is the accumulated local acceptance value computed by $\pi_{e_i}(a_1, ..., a_n)$ from definition 5). The contributions $p_i$ are computed by a function:

$$g : R \times R \rightarrow R,$$  \hspace{1cm} (11)

such that $p_i = g(x_i, \rho(e_i, c))$.

The global acceptance function, which satisfies properties $P1, ..., P4$, specified in definition 8, is calculated in the following way:

**Definition 8. (Extended Acceptance Function):** Acceptance between weighted queries and cases is expressed by an extended acceptance function:

$$\text{acc} : R^E \times P(E) \rightarrow R.$$  \hspace{1cm} (12)

The acceptance $\text{acc}(\alpha_q, c)$ of a case $c$ for a weighted query $\alpha_q$ can now be accumulated by using the introduced functions:

$$\text{acc}(\alpha_q, c) = \pi_c(g(\pi_{e'}(f(\alpha_q(e_{1,1}), \sigma(e_{1,1}, e_{1,1}')), ..., f(\alpha_q(e_{1,n}, \sigma(e_{1,n}, e_{1,1}')), \rho(e_{1,1})), \ldots)$$

$$g(\pi_{e'}(f(\alpha_q(e_{k,1}), \sigma(e_{k,1}, e_{k,1}')), ..., f(\alpha_q(e_{k,n}, \sigma(e_{k,n}, e_{k,1}')), \rho(e_{k,1}))))$$

where $c = \{e_{1}', ..., e_{k}'\}$ and $E_{e_{i,1}} = \{e_{i,1}, ..., e_{i,n}\}$ for $i = 1, ..., k$.  \hspace{1cm} (13)

If, for example, we consider $f$ and $g$ as products and $\pi_c$ and $\pi_{e_i}$ as sums then we get:

$$\text{acc}(\alpha_q, c) = \sum_{e' \in c} \rho(e', c) \sum_{e \in E_{e'}} \sigma(e, e') \cdot \alpha_q(e)$$  \hspace{1cm} (14)

Here, the properties $P1, ..., P4$ are satisfied, but for satisfaction of the property $P5$, the appropriate selection of the functions $f$, $g$, $\pi_c$ and $\pi_{e_i}$ is needed.

**Case Retrieval Nets**

In this section a broad and a formal structure of Case Retrieval Net [Lenz et al. 1998] are given. Case retrieval net is the basic structure used for the implementation of the system described in this thesis. The implementation of case retrieval net is described in part 3 of this thesis in detail.
Case Retrieval Net (CRN) is a special memory structure that has been developed especially for being employed in large case bases. CRNs are able to deal with vague and ambiguous terms; they support the concept of information completion and can handle case bases of reasonable size efficiently.

![Part of Case Retrieval Net](image)

**Figure 2.2.** Part of Case Retrieval Net

CRN is a net structure with information entity node for each information entity and case node for each case. An example of CRN is shown on Figure 2.2. There exists an "acceptance" (more general term used for similarity as explained in 2.3.) arc from the information entity nodes \( e \) to the information entity node \( e' \) if \( \sigma(e, e') \) is defined, and there exists a "relevance" arc from the information entity node \( e \) to the case node \( c \) if \( \rho(e, c) \) is defined. The arcs in the net are weighted by the values \( \sigma(e, e') \) and \( \rho(e, c) \) respectively.

In practice, it is impossible to include all information entities, since the real world domains are almost always infinite. Usually, it is sufficient to build a net from the information entities which occur in the cases from the case base only.

Acceptance values are computed by a spreading activation process in the net as follows: Information entity nodes are initially activated by \( \alpha_q(e) \). The computation is performed by propagating along the acceptance arcs to further information entity nodes, and from these nodes over relevance arcs to case nodes. The functions \( f / \pi_e \) and \( g / \pi_c \) are responsible for the accumulation of activities in the information entity nodes and in the case nodes respectively. The final activation at the case nodes denotes the acceptance value of the case for the weighted query.

An example and a structure of a case retrieval net formed over the real data is given in the section 3.2.
Formal Model of CRNs

The formal definition of CRNs assumes that the information entities are represented with an attribute-value pairs, and that the cases and queries are given with the set of information entities [Lenz et al. 1998].

**Definition 9. (Case Retrieval Net):** A Case Retrieval Net is defined as a structure \( N=[E,C,\sigma,\rho,\Pi] \) where:

- \( E \) is the finite set of information entity nodes;
- \( C \) is the finite set of case nodes;
- \( \sigma \) is the similarity function defined between two information entity nodes:
  \[
  \sigma : E \times E \rightarrow R
  \]  
  \( (15) \)
- \( \rho \) is the relevance function which describes the relevance of the information entity node to the case node:
  \[
  \rho : E \times C \rightarrow R.
  \]  
  \( (16) \)
- \( \Pi \) is the set of propagation functions for each node \( IE \) or case node \( n \):
  \[
  \pi_n : R^E \rightarrow R.
  \]  
  \( (17) \)

CRN is given by a weighted directed graph whose nodes are from the set \( E \cup C \). The arc from \( e_i \in E \) to \( e_j \in E \) is weighted by \( \sigma(e_i, e_j) \), while the arc from \( e \in E \) to \( c \in C \) is weighted by \( \rho(e, c) \). The arcs whose weight is zero are omitted.

An information entity \( e \) belongs to a case \( c \) if \( \rho(e, c) \neq 0 \). The functions \( \pi_n \) are used to compute the new activation of node \( n \) depending on the incoming activations (a simple example may be the sum of all incoming activations).

**Definition 10. (Activation of a Case Retrieval Net):** An activation of a CRN \( N=[E,C,\sigma,\rho,\Pi] \) is a function:

\[
\alpha : E \cup C \rightarrow R,
\]  
\( (18) \)
Informally, the activation of an information entity represents the importance of that information entity to the actual problem query.

**Definition 11. (Propagation process in Case Retrieval Net):** Let \( \alpha_t \) be the activation at time \( t \) for CRN \( N=[E,C,\sigma,\rho,\Pi] \) with \( E=\{e_1, ..., e_s\} \). The activation of information entity nodes \( e \in E \) at time \( t+1 \) is given by:

\[
\alpha_{t+1}(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_t(e_1), ..., \sigma(e_s, e) \cdot \alpha_t(e_s)),
\]  
\( (19) \)

and the activation of case nodes \( c \in C \) at time \( t+1 \) is given by:

\[
\alpha_{t+1}(c) = \pi_c(\rho(e_1, c) \cdot \alpha_t(e_1), ..., \rho(e_s, c) \cdot \alpha_t(e_s)).
\]  
\( (20) \)

The initial activation nodes can gain according to the weighted query (Definition 4).

The activation propagation process is described as a three step process:
1. **Initial Activation:** According to the weighted query, the initial activation $\alpha_0$ is determined for all information entity nodes.

2. **Similarity Propagation:** The activation is propagated to all information entity nodes $e \in E$:

   \[
   \alpha_1(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_0(e_1), ..., \sigma(e_s, e) \cdot \alpha_0(e_s)), \quad (21)
   \]

3. **Relevance Propagation:** The resulting activation $\alpha_1$ is propagated to the case nodes $c \in C$:

   \[
   \alpha_2(c) = \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), ..., \rho(e_s, c) \cdot \alpha_1(e_s)). \quad (22)
   \]

The result of the case retrieval process for a given weighted query is the list of cases, sorted decreasingly by gained activations $\alpha_2(c)$.

Originally, in the formal definition of CRN, the term similarity is used for the function $\sigma$. Since the term acceptance is accepted as a more general (as described in 2.3. and in Definition 2), it will be used in the rest of the thesis.

**Types of CBR Applications and Current State**

In this section the basic characteristics for every task type, which can be potentially solved with CBR technology, will be given together with several realized applications [University of Kaiserslautern, homepage]. However, before explaining task types it is necessary to make a difference between domain and task type.

*Domain* of application represents the area or discipline in which the CBR technology is used for solving problems. For example, the domains are: mechanical engineering, business administration, medicine... Each domain has its own characteristics, and it strongly influences the choice of data structure for knowledge representation.

*Task type* represents the kind of problem which has to be solved in some domain. Some of the task types are:

- Classification,
- Diagnosis,
- Configuration,
- Planning,
- Decision Support,
- Information Searching,
- etc…

Task type determines the type of problems and solutions and, furthermore, the activities for solving problems. There is no one-to-one correspondence between task type and domain. Furthermore, every pair (domain, task type) is possible and it requires its own expertise.
Classification

Classification is a process of determining a class for some elements from a given set, using corresponding functions. The parameters for classification are: universal set $U$ and its subsets $K_i \subseteq U, i \in I$, called classes. A classifier is a function:

$$f: U \rightarrow I$$

such that $f(x) = i$ implies $x \in K_i$.

In CBR technology the classifier is defined as an ordered pair $(CB, \text{sim})$, where $CB \subseteq U$ is a case base, and $\text{sim}$ similarity measure defined on $U \times CB$. If classes of the elements from case base $CB$ are known, the class of the element from universal set $x \in U$ is computed by using $\text{sim}$ in the following way:

$$x \in K_i \iff \text{NN}(x) \in K_i$$

where $\text{NN}(x)$ is the nearest neighbour of $x$, but in the set CB. Of course, the nearest neighbour can be computed by using $k$ nearest neighbours.

The process of classification is straightforward and deterministic, and is not so attractive for application of CBR technology. Therefore, there are no significant CBR applications in this area.

Diagnosis

Diagnosis is considered as a cost sensitive classification with incomplete information [Lenz, Burkhard 1996], [Kamp, Pirk, Burkhard 1996], [Kurbalija 2003]. This means that establishing the diagnosis is only a final step of a diagnostic progress. The process of diagnosis if often interleaved with some additional tests or questions which adds some information entities in order to establish a better diagnosis. This is often called a differential diagnosis.

Some commercial applications for diagnosis, realized using CBR technology are:

- **Case Advisor 4 / Webserver** – diagnosis of malfunctions and solving problems for PC computers. This tool uses static database. Case Advisor Webserver is a tool that offers online support to the users. Here is also offered an online support to the users of the cable TV network. Detailed information can be found on: [http://www.cs.sfu.ca/~isa/isaresearch.html#systems](http://www.cs.sfu.ca/~isa/isaresearch.html#systems)

- **Case-Based Reasoning in Cardiovascular Disease** – diagnosis of some cardiovascular diseases on the basis of some symptoms. Case base consists of 240 characteristic cases. Also this system supports learning form the experience. Detailed information can be found on: [http://medg.lcs.mit.edu/projects/cbr.html](http://medg.lcs.mit.edu/projects/cbr.html)

- **MoCas** – diagnosis in technical domains. The main characteristic of this system is that the general knowledge from some technical domain is integrated in the system. Integration of the knowledge enables adaptation and transformation of the cases. Detailed information can be found on: [http://wwwagr.informatik.uni-kl.de/~lsa/LSABook-English/LSABook-E_20.html](http://wwwagr.informatik.uni-kl.de/~lsa/LSABook-English/LSABook-E_20.html)

- **SpectroRx Resolution Expert** – tool for diagnosis problems with computer network. Company "ENTERASYS", whose main area is production of network equipment, used
CBR technology to make a help desk system which can help clients with various problems. This system saves the expert time because experts help clients only when system can not, and the rest of the time spends on more complex tasks. Detailed information can be found on: http://www.cabletron.com/products/items/SA-CSI1016/

Configuration and Design

*Configuration* can be understood as the construction of the artifact from the given set of components, such that certain conditions are satisfied. *Design* introduces some degree of creativity because some components of the artifact are not known during design process. Depending on the degree of creativity required 3 types of design can be distinguished: routine design, innovative design and creative design. In real world applications the solution gained from CBR configuration or design system can almost never remain unmodified – the adaptation phase is necessary.

Some commercial applications for configuration and design, realized by using CBR technology are:

- **AIDA** (Artificial Intelligence supported Design of Aircraft) – tool for helping in airplane design. This tool helps in the first phase of design process – conceptual design. Human designer can pay attention on more creative tasks while the system keeps attention on less creative tasks. For realization of this tool several artificial intelligence approaches are used: Constraint-Based Reasoning, Case-Based Reasoning and Rule-Based Reasoning. The intention of the authors is to use the concept of AIDA in other domains such as: car or ship design. Detailed information can be found on: [http://www.kbs.twi.tudelft.nl/Research/Projects/AIDA/](http://www.kbs.twi.tudelft.nl/Research/Projects/AIDA/)

- **Archie** – tool for helping in conceptual building design. The aim of this tool is to make use of past experience in design of public buildings in order to avoid some previous mistakes in design process. Detailed information can be found on: [http://www.cc.gatech.edu/aimosaic/faculty/kolodner/archie.html](http://www.cc.gatech.edu/aimosaic/faculty/kolodner/archie.html)

- **BRUSH** – tool for designing bathrooms for the invalids. Cases in this tool represent episodes in design bathrooms for invalids. System, on the basis of those cases, designs a new bathroom which satisfies certain properties. Detailed information can be found on: [http://www.arch.su.edu.au/~kate/BathRedesign/Guestbookii/Welcome.html](http://www.arch.su.edu.au/~kate/BathRedesign/Guestbookii/Welcome.html)

- **CADET** – tool for conceptual design of electro-mechanical parts. CADET consists of several subsystems. Case-based reasoning and model based reasoning are integrated in this system. Detailed information can be found on: [http://www.cs.cmu.edu/afs/cs.cmu.edu/project/cadet/ftp/docs/CADET.html](http://www.cs.cmu.edu/afs/cs.cmu.edu/project/cadet/ftp/docs/CADET.html)

- **CBRTeam** – multiagent system for design of steam condenser. This system consists of 3 agents: motor-agent, pump-agent and vbelt-agent which are responsible for design of motor, pump and belt respectively. When a user makes a specification these agents, in cooperation and on the basis of available parts, tries to construct suitable condenser. Detailed information can be found on: [http://ksi.cpsc.ucalgary.ca/KAW/KAW96/prasad/](http://ksi.cpsc.ucalgary.ca/KAW/KAW96/prasad/)

Planning

Planning covers a great variety of tasks. Here, planning will be restricted on the *action planning*. The problem is to find a sequence of actions which transforms a given initial
situation into a desired goal situation. The number of selected actions is not restricted but the set of available actions is fixed. For CBR planning, the reuse aspect is important because the retrieved plans have to be adapted.

Some commercial applications for planning, realized by using CBR technology are:

- **Bioplan** – bioprocess planning in medicine production. Detailed information can be found on: [http://www.vtt.fi/bel/bio/process/bioplan.htm](http://www.vtt.fi/bel/bio/process/bioplan.htm)
- **Knowledge Based Mashing** – process planning in beer production. Detailed information can be found on: [http://www.vtt.fi/bel/bio/process/mashplan.htm](http://www.vtt.fi/bel/bio/process/mashplan.htm)
- **Prodigy** – domain independent action planning. Prodigy is architecture for learning and planning. At this moment Prodigy supports different concepts: example based learning, partial calculations, graphical learning, automatic abstraction, initial planning, CBR in many different domains. Detailed information can be found on: [http://www.cs.cmu.edu/afs/cs.cmu.edu/project/prodigy/Web/prodigy-home.html](http://www.cs.cmu.edu/afs/cs.cmu.edu/project/prodigy/Web/prodigy-home.html)
- **CHARADE** – planning of human and equipment resources in state of emergency (fire, flood…). Detailed information can be found on: [http://mnemosyne.itc.it:1024/avesani/html/charade.html](http://mnemosyne.itc.it:1024/avesani/html/charade.html)

**Decision support**

The task of a system for decision support system is to help to the person who has to make some important decisions, and to totally replace him [Ivanović 2002], [Kurbalija, Ivanović 2003]. The answer of the system is usually not the solution of the problem but some advice or some useful piece of information.

Some commercial applications for decision support, realized by using CBR technology are:

- **Aircraft Conflict Resolution** – application which helps in flight control. This tool helps in solving conflicts in airplane traffic. The great significance of this application is that authors devoted a big effort in representing different kinds of cases. Detailed information can be found on: [http://www.cs.tcd.ie/research_groups/aig/cbr.html](http://www.cs.tcd.ie/research_groups/aig/cbr.html)
- **ALSTOM** – tool for better organization of trains in order to reduce costs. Detailed information can be found on: [http://www.acknosoft.com/alstom.html](http://www.acknosoft.com/alstom.html)
- **ANSALDO** – maintenance of subway in Napul. Detailed information can be found on: [http://www.acknosoft.com/ansaldo.html](http://www.acknosoft.com/ansaldo.html)
- **CBR Job Agent** – job search on the basis of entered skills. Detailed information can be found on: [http://minsk.informatik.uni-kl.de:8100/launch/JobbQueryInterface](http://minsk.informatik.uni-kl.de:8100/launch/JobbQueryInterface)
- **DESSERT** – decision support in obliging management. Detailed information can be found on: [http://www.broadcom.ie/partners/acts/race/dessert/dessert.html](http://www.broadcom.ie/partners/acts/race/dessert/dessert.html)

**Information Searching**

Inexact matches (finding similar documents when the exact document doesn't exist) are vital here, so the CBR is a natural technique in this area [Lenz, Hübner 1998], [Lenz 1998], [Minor, Carlos 2000]. Since exponential growth of information this is the promising area for
CBR, because CBR has ability to learn constantly and also has a good mechanism for indexing huge databases.

Some commercial applications for information searching, realized by using CBR technology are:

- **Broadway** – intelligent web browser. The system tries to use knowledge learned from some previous searches of one group of users. This browser tracks the user in his browsing, tries to discover his goals and proposes some potentially interesting documents. Detailed information can be found on: [http://www-sop.inria.fr/aid/broadway/](http://www-sop.inria.fr/aid/broadway/)

- **CaBaTa** – virtual tourist agency. On the basis of user's demands, system suggests the most similar offers. Also, the system contains the component for selling airplane tickets. Detailed information can be found on: [http://www.informatik.hu-berlin.de/~lenz/CheckIn/CABATA/cabata_e.html](http://www.informatik.hu-berlin.de/~lenz/CheckIn/CABATA/cabata_e.html)

- **Entree** – tool for finding optimal restaurant in Chicago. Detailed information can be found on: [http://infolab.cs.uchicago.edu/entree/](http://infolab.cs.uchicago.edu/entree/)

- **RECALL** – automatic saving and retrieving 'learned lesions' in "NASA Goddard Space Flight Center". Detailed information can be found on: [http://hope.gsfc.nasa.gov/RECALL/homepg/recall.htm](http://hope.gsfc.nasa.gov/RECALL/homepg/recall.htm)

- **FAQ Finder** – retrieving 'frequently asked questions'. Detailed information can be found on: [http://infolab.cs.uchicago.edu/faqfinder/](http://infolab.cs.uchicago.edu/faqfinder/)
3. *CaBaGe – Case Base Generator*

For application of CBR in any domain it would be very useful to have some basic, core framework that can produce decision support systems. “*CaBaGe*” (Case Base Generator) [Kurbalija, Ivanović, Budimac 2003], [Kurbalija 2005] was the direct result of these intentions. It was completely implemented in Java – JDK 1.3 mainly because it supports all concepts of object-oriented technology, but also because of its main characteristic – platform independence. The system was realized as an application, but the small modifications are necessary in order to make an applet or servlet. *javax.swing* components are used for creating a graphical user interface (GUI).

The one of important characteristics of this system is that it is domain independent. The input for the system is a description of the case and the data for particular cases from a chosen domain. On the basis of those data, system creates Case Retrieval Net (CRN) and it is capable to solve new problems from the same domain.

The system reads the data from two input files:

- **Case Pattern File**
- **Case Base File**

In the first input file - “*Case Pattern File*”, the description of case is stored. Case pattern file contains the list of the attributes, containing the name and the type of the attribute. The type of the attribute can be: *int, float* or *string*. Boolean type can be simulated, for example, with the string type where only two values (“Yes”,”No”) are allowed. The number of the attributes is arbitrary, but all attributes must be listed in the correct order.

The second file - “*Case Base File*”, contains the list of all already solved cases from the domain. Every case is described with the values of its attributes and with the final solution of that case. The final solution of the case is always listed at the end of the case (after values of all attributes). Type of the solution can also be: *int, float* or *string* and it is determined dynamically, when all cases are parsed. Case base file is an ordinary textual file where every case is listed in one line, and the values of the attributes and solution are separated with commas.

Together with the reading of the case base file, system creates case retrieval net. The broad structure of CRN is given in Figure 3.1. Two main parts of the CRN are array of attributes and list of solutions. The array of attributes is created by using case pattern file, while the list of solutions is created from case base file. Every value for every attribute represents one information entity because the information entity is an ordered pair *(attribute,value)*. Every value (or information entity) contains the list of arcs to the solution nodes. The arc is given with its weight and a pointer to the solution node; just the arcs whose weights are
different from zero are saved. Weights of the arcs between the information entity node $e$ to the solution node $c$ represents the value of the function $\rho(e, c)$. These weights are simply calculated as a number of cases (from the case base file) that contain the information entity $e$ and whose solution is $c$.

![Figure 3.1. Structure of CRN](image)

After creating the CRN the system expects from the user to enter the current problem (query). Since the query and the case have the same structure the user has to enter the values of attributes in an appropriate form. In order to describe the problem in better way, the user should enter all known values of the attributes although it is not necessary. The form contains one more field for every attribute – importance. The importance is the value from the interval $(0,1)$, and describes how much is the user sure in the validity of the value of the attribute he is entering. Value 1 means that he is 100% sure that the data are valid, while the value 0 means that he doesn’t know the value of that attribute at all. The value of importance corresponds to the previously described value of the weighted query.

After entering the query, the system searches for the possible solution in the following way:

*The information entities (attribute, value) that occur in the query are initially activated with the value of importance (weighted query). The activation is propagating through the arcs to the solution nodes, by multiplying the value of activation of the information entity node and the weight of the corresponding arc. Final activation of the solution nodes is calculated by summing all gained activations.*

CRNs can implicitly support the process of information completion. That means that not all values of all attributes must be entered in the input form (value unknown). That is the same
situation as when the value of importance for some attribute is equal to 0. The system will try to find a solution even if some of the attributes doesn’t have a value.

Since CRNs have this good property, the process of case completion is not supported in the system. Moreover, the cases for the process of case completion should not have strict distinction between problem and solution part, while in the CaBaGe system that distinction exists.

**Implementation of CaBaGe**

The implementation of the CaBaGe system will be described in detail in this section. The UML diagram of all classes in this project and their connections is given in Appendix I. All classes will be described, but, in listings, only some important and interesting parts of the source code will be shown here. The complete source code is given on CD, which is the part of this thesis.

**Class List**

Class List contains the implementation of sorted linked list. The name of methods and fields of this class are shown in Listing 3.1.

```java
public class List {
    List next;
    Element elem;

    public List(Element elem) {
        ...
    }

    public boolean HasNext() {
        ...
    }

    public static List Insert (List list, Element newelem) {
        ...
    }

    public static List Find (List list, Element wanted) {
        ...
    }
}
```

**Listing 3.1 Class List**

The container in the class List is an elem, while the other field (next) represents the pointer to the next element of the list.

The method List is a standard constructor of the class, while the method HasNext returns, whether the current list element has at least one element behind.
Method **Insert** inserts element `newelem` in the existing list `list` and returns the pointer to the beginning of the newly created list. Since the list is sorted, every inheritance of the class `Element` must implement a method `Greater`.

Method **Find** tries to find element `wanted` in the list `list` and returns pointer to it if, it is found, otherwise returns null.

**Class Element**

Class `Element` is an abstract class and represents a standard template for every type of data which has to be a part of the list. Every class which has to be stored in the list must inherit this abstract class. The methods of this class are shown in Listing 3.2.

```java
abstract class Element {
    public abstract boolean Greater (Element e);
    public abstract boolean Equal (Element e);
}
```

**Listing 3.2 Class Element**

Method `Greater` returns, weather the object `Element`, who calls this method, is greater then its parameter.

Method `Equal` returns, if the object, who call this method, and parameter have the same values.

**Class IEEElement**

Class `IEElement` defines the basic (template) class for elements of the list which represents information entities. Since this application supports three types of data (string, integer and float) for information entities, this class will have three child classes. The methods and fields of this class are shown in Listing 3.3.

```java
abstract class IEEElement extends Element {
    List arcs;
    float initialActivation;
    public IEEElement() {
        arcs = null;
        initialActivation = 0;
    }
}
```

**Listing 3.3 Class IEEElement**

According to a definition, every information entity must contain the list of arcs to all solution nodes. However, only the arcs, whose weight is different from zero, are stored in the field `arcs`. The field `initialActivation` is important in the process of solving problem. Here, the importance values, entered by the user when defining a problem, are stored.

The method `IEElement` is the standard constructor.
Classes stringIEElement, intIEElement and floatIEElement

These three classes correspond to information entities nodes. As already mentioned, information entities are defined as ordered pairs (attribute, value). However, since the information entities of one attribute are all kept in one list, only the values of information entities are stored in this class. Since this application supports three data types (string, integer and float), these classes hold the values of these data types. All three classes inherit the IEElement class. The methods and fields of these classes are shown in Listings 3.4, 3.5 and 3.6.

```java
public class stringIEElement extends IEElement {
    String name;
    public stringIEElement(String name) {
        super();
        this.name = name;
    }
    public boolean Equal(Element e) {
        return this.name.compareTo(((stringIEElement)e).name) == 0;
    }
    public boolean Greater(Element e) {
        return this.name.compareTo(((stringIEElement)e).name) > 0;
    }
    public String toString() {
        return name;
    }
}
```

Listing 3.4 Class stringIEElement

```java
public class intIEElement extends IEElement {
    int value;
    public intIEElement(int value) {
        super();
        this.value = value;
    }
    public boolean Equal(Element e) {
        return this.value == ((intIEElement)e).value;
    }
    public boolean Greater(Element e) {
        return this.value > ((intIEElement)e).value;
    }
    public String toString() {
        return Integer.toString(value);
    }
}
```

Listing 3.5 Class intIEElement

```java
public class floatIEElement extends IEElement {
    float value;
    public floatIEElement(float value) {
        super();
        this.value = value;
    }
    public boolean Equal(Element e) {
        return this.value == ((floatIEElement)e).value;
    }
}
```

Listing 3.6 Class floatIEElement
Each of these three classes have the field value (name in the stringIEElement) which holds a value of the corresponding data type.

The constructors call the constructor method of the IEEElement class and set initial value of the field. These classes implement methods Equal and Greater from the Element class because it is necessary for the implementation of sorted list.

Also, the method toString is implemented, for visual representation of objects, by converting the float value of the field value into string.

Class CaseElement

Class CaseElement represents solution nodes in the case retrieval net. All solution nodes are stored in the linked list so this class inherits the Element class. The methods and fields of this class are shown in Listing 3.7.

```java
public class CaseElement extends Element {
    String name;
    float activation;
    int maximumActivation;
    public CaseElement(String name) {
        this.name = name;
        activation = 0;
        maximumActivation = 0;
    }
    public boolean Equal(Element e) {
        return this.name.compareTo(((CaseElement)e).name) == 0;
    }
    public boolean Greater(Element e) {
        return this.name.compareTo(((CaseElement)e).name) > 0;
    }
}
```

Listed 3.7 Class CaseElement

Every solution has its name and it is stored in the field name. In field activation, the sum of all activations, gained during the activation propagation process for corresponding solution node, is stored. Since, the best solutions are chosen based on its relative activation; the maximum activation is also stored. The maximum activation, which corresponding solution node can gain, is stored in field maximumActivation.
The method CaseElement is the standard constructor, while the methods Equal and Greater are also implemented because this class must be a part of the sorted list.

**Class Arc**

Class Arc represents weighted arcs in case retrieval net. All arcs from one information entity are stored in a linked list, so the class Arc must also inherit the Element class. The methods and fields of this class are shown in Listing 3.8.

```java
public class Arc extends Element {
    public int weight;
    CaseElement caseNode;
    public Arc(CaseElement caseNode) {
        this.caseNode = caseNode;
        weight = 1;
    }
    public boolean Equal(Element e) {
        return caseNode == ((Arc)e).caseNode;
        //return caseNode.Equal(e);
    }
    public boolean Greater(Element e) {
        return caseNode.Greater(((Arc)e).caseNode);
    }
}
```

**Listing 3.8 Class Arc**

As already mentioned, all arcs from one information entity are stored in one linked list, so it is not necessary to keep a reference to information entity. In class Arc, only the reference to solution node is kept (in field caseNode) and the weight of the arc (in the field weight).

The constructor of the class has one parameter, which represents the reference to the solution node. When the arc is created, the weight is set to 1, and later, during the process of creating case retrieval net for concrete data, every time when the strength of connection between the information entity node and solution node has to be increased, the weight of the arc is increased by 1.

The method Equal returns true if two arcs refer to the same solution node, but not in the case when they refer to two solution nodes with the same name (like in the line which is commented).

Method Greater uses the method Greater from the CaseElement class, because it is expected that the arcs, in the list of one information entity, are sorted in the same order as solution nodes in their list.

**Class activationElement**

Class activationElement is a temporary class that is used in splitting activation process. This process will be described later in detail. Objects of this class are stored in a linked list, so the class activationElement must also inherit the Element class. The methods and fields of this class are shown in Listing 3.9.

```java
public class activationElement extends Element {
```

33
float similarity = 0;
IEElement which = null;
public activationElement(float sim, IEEElement which) {
    this.similarity = sim;
    this.which = which;
}
public boolean Equal(Element e) {
    return false;
}
public boolean Greater(Element e) {
    return similarity > ((activationElement)e).similarity;
}

Listing 3.9 Class activationElement

The similarity between one information entity from query and one information entity from case retrieval net is kept in the field similarity. Each information entity from query contains the list of activationElement objects – one object for each information entity from case retrieval net, which is bounded for the same attribute. This is necessary because all similarities must be computed firstly and compared in order to correctly split the initial activation. The field which is a reference to information entity in case retrieval net.

Method Equal always returns false because each information entity from the case retrieval net is treated as a different, no matter if the similarities to the information entity from the query are the same.

Method Greater compares two objects using their similarities, because it is convenient that at the beginning of the list stands the information entity with higher similarities.

Class Attribute

Since the case (and the query) is described with a set of attributes in this application, class Attribute represents one attribute of the case. The methods and fields of this class are shown in Listing 3.10.

```java
public class Attribute {
    String name;
    char type; //'f' for float, 'n' for integer, 'i' for ident
    List informationEntities;
    Element valueOfQuery; //holds a value of the actual query
    float valueOfActivation;

    public Attribute(String name, char type) {
        this.name = name;
        this.type = type;
        informationEntities = null;
        valueOfQuery = null;
        valueOfActivation = 0;
    }

    public void SetActivation() {
        //Distribution of initial activation on the similar IE nodes
    }
}
```
if (valueOfQuery == null) {
    System.out.println("Unknown value in the query for the attribute " + name);
    return;
}
if (informationEntities == null) {
    System.out.println("No IEs for the attribute " + name);
    return;
}
if (type == 'i') {
    List pointer = List.Find(informationEntities, valueOfQuery);
    if (pointer == null) {
        System.out.println("Value "+((stringIEElement)valueOfQuery).name+
        "unknown for the attribute "+name);
        return;
    }
    ((IEElement)pointer.elem).initialActivation = valueOfActivation;
}
else if (type == 'f') {
    List pointer = List.Find(informationEntities, valueOfQuery);
    if (pointer != null) {
        ((IEElement)pointer.elem).initialActivation = valueOfActivation;
        return;
    }

    //start of splitting the activation

    List tmp = informationEntities;
    int numOfIE = 0;
    List tmpList = null;
    while (tmp != null) {
        List.Insert(tmpList, new activationElement(
            ((floatIEElement)valueOfQuery).value>((floatIEElement)tmp.elem).value ?
            ((floatIEElement)valueOfQuery).value-((floatIEElement)tmp.elem).value :
            ((floatIEElement)tmp.elem).value-((floatIEElement)valueOfQuery).value,
            ((IEElement)tmp.elem)
        );
        numOfIE++;
        tmp = tmp.next;
    }
    numOfIE = (int) 0.2*numOfIE + 1; //20% of the best IEs
    int i = 1;
    tmp = tmpList;
    float sum = 0;
    while (i <= numOfIE) {
        sum = sum + 1 / ((activationElement)tmp.elem).similarity;
        i++;
        tmp = tmp.next;
    }
    sum = valueOfActivation/sum;
    i = 1;
    tmp = tmpList;
    while (i <= numOfIE) {
        ((activationElement)tmp.elem).which.initialActivation = sum *
            ((activationElement)tmp.elem).similarity;
        i++;
        tmp = tmp.next;
    }
}
else if (type == 'n') {
    List pointer = List.Find(informationEntities, valueOfQuery);
    if (pointer != null) {
        ((IEElement)pointer.elem).initialActivation = valueOfActivation;
        return;
    }

    // start of splitting the activation

    List tmp = informationEntities;
    int numOfIE = 0;
    List tmpList = null;
    while (tmp != null) {
        tmpList = List.Insert(tmpList, new activationElement(
            ((intIEElement)valueOfQuery).value>((intIEElement)tmp.elem).value ?
            ((intIEElement)valueOfQuery).value-((intIEElement)tmp.elem).value :
            ((intIEElement)tmp.elem).value-((intIEElement)valueOfQuery).value ,
            ((IEElement)tmp.elem)
        )
        );
        numOfIE++;
        tmp = tmp.next;
    }
    numOfIE = (int) (0.2*numOfIE + 1); // 20% of the best IEs
    int i = 1;
    tmp = tmpList;
    float sum = 0;
    while (i <= numOfIE) {
        sum = sum + 1 / ((activationElement)tmp.elem).similarity;
        i++;
        tmp = tmp.next;
    }
    sum = valueOfActivation/sum;
    i = 1;
    tmp = tmpList;
    while (i <= numOfIE) {
        ((activationElement)tmp.elem).which.initialActivation = sum *
            ((activationElement)tmp.elem).similarity;
        i++;
        tmp = tmp.next;
    }
}

Listing 3.10 Class Attribute

Every attribute is uniquely identified by its name, which is stored in a field `name`. The type of
the attribute is stored in a field `type`, with the following meaning:

- character “f” for the float data type,
- character “n” for the integer data type,
- character “i” for the string data type.

Boolean type can be simulated, for example, with the string type where only two values
(“Yes”/”No”) are allowed, or with the integer type where 0 means “false”, 1 means “true”.
Attribute class also contains the list of values for that attribute. In this list, only the values
that appear in the case base are stored (it would be impossible to store all values, for example, of numeric data type). These values, in fact, represent information entity nodes, because information entity is defined as ordered pair \((\text{attribute}, \text{value})\). The list of values is stored in the field \text{informationEntities}. Members of this list are instances of class \text{stringIEElement}, \text{intIEElement} or \text{floatIEElement}, depending whether the type of attribute is string, integer or float, respectively. In this application the query is also defined as an array of attributes. The value of the corresponding attribute of query, which has to be solved, is stored in the field \text{valueOfQuery}. Field \text{valueOfActivation} contains the importance of corresponding attribute in the query, which is defined by the user.

Method \text{Attribute} is the standard constructor of the class. The attribute is defined with its name and a type, while the other fields will get their values after object creation.

The task of the method \text{SetActivation} is to distribute the initial activation (stored in the field \text{valueOfActivation}) on the information entity nodes, concerning their similarity with the value of the query. This method is the implementation of the first iteration in the activation propagation process in case retrieval net, where the activation is propagating from information entity node of the query to most similar information entity nodes of case retrieval net. At the beginning, it is checked that value of query and the list of information entities contains some values. If not, a warning message occurs and no activation is distributed.

After that, there are three possibilities depending of the type of attribute. If the type of the attribute is a string, only the exact matches are supported. This means that if there exists an information entity whose value is equal to the value of query, it gets all the activation. If such information entity doesn’t exist, the activation for that attribute is not distributed. The process of inexact matches for textual data needs more sophisticated methods of knowledge management [Minor 2001], [Minor 2004].

For the integer and float data types the process is equal. Firstly, the exact match is tried to be found. If it is found it get whole activation. Otherwise, the temporary list of information entities (class \text{activationElement}), sorted by similarity with the value of query, is created. The similarity is calculated as an absolute value of difference between the value of query and the value of corresponding information entity. If \(qv_i\) denotes the value of query of the \(i\)-th attribute, and \(va_{i,j}\) denotes \(j\)-th value (information entity) of the \(i\)-th attribute, then the similarity, \(\text{sim}_{i,j}\), between \(i\)-th value of query and \(j\)-th value of the \(i\)-th attribute is calculated:

\[
\text{sim}_{i,j} = |qv_i - va_{i,j}|
\]  

(25)

After that, only the first 20% of the most similar values (information entities) are selected. In this moment the value called \text{factor}_{i} is calculated. If \(n_i\) denotes the number of all values of the \(i\)-th attribute, \text{factor}_{i} is calculated:

\[
\text{factor}_{i} = \frac{1}{\sum_{j=1}^{0.2n_i} \text{sim}_{i,j}}
\]  

(26)

Finally, if \(vact_{i}\) denotes the value of activation which has to be distributed on \(i\)-th attribute, the activation of \(j\)-th value of \(i\)-th attribute \(\text{act}_{i,j}\) is calculated:

37
\[ act_{i,j} = \text{sim}_{i,j} * \frac{vact_i}{\text{factor}_i} \] (27)

### Class Lexer

Class Lexer is a standard lexical analyzer, which reads an input file, recognizes and returns tokens from it. Since the algorithms of lexical analysis are well known, in Listing 3.11 only the main fields and the headings of methods, needed for further classes, are given.

```java
public class Lexer {
    private char ch;
    java.io.BufferedReader f;
    public String name;
    public int ivalue, line, col;
    public float fvalue;
    public boolean commaread, EOLread;

    public Lexer(String fileName) {
        ...
    }
    private void NextCh() {
        ...
    }
    private void Skip() {
        ...
    }
    public char Next() {
        ...
    }
}
```

**Listing 3.11 ClassLexer**

Field `ch` is used as a lookahead character for the lexer. Field `f` is used as a file input stream. The input is buffered for a better performance. Field `name` contains the value of string when string token is recognized. Similarly, fields `ivalue` and `fvalue` contains the values when integer or float token were recognized, respectively. Fields `line` and `col` are used for line and column counting for user-friendly error reporting. Fields `commaread` and `EOLread` are boolean values which are set to true when a comma or “end of line” characters are read.

Method `Lexer` is a constructor of the class. It tries to open the file with the name `filename`, and reports error if file was not found. If the file is found, it reads the first character and sets fields `line` and `col` to 1.

Method `NextCh` reads the next character from the input file and takes care of the line and column counting.

Method `Skip` skips all separators.
Method `Next` recognizes the next token and returns:

- "i" if string token was recognized,
- "f" if float token was recognized,
- "n" if integer token was recognized,
- "e" if empty place was recognized (used for unknown values),
- "#" if end of file was reached,
- "?" if unknown set of characters was recognized.

This method also sets fields: `name`, `ivar` and `fvar` when string, integer or float value was recognized. Also, method `NextCh` sets fields `commaread` and `EOLread` when comma or “end of line” characters are read after the token.

**Class `CaseRetrievalNet`**

Class `CaseRetrievalNet` represents the structure shown in Figure 3.1. It contains the list of solution nodes (elements of the list are instances of class `CaseElement`) and an array of attributes. Every attribute contains its name, type and list of values for that attribute. Every value of every attribute (which represents the information entity node) contains the list of weighted arcs to solution nodes appropriate for that attribute value. Of course, only the arcs whose weights are different from zero are saved. This organization represents the implementation of formal case retrieval net described in section 2.4.

This class also contains methods for reading the structure of the case and the database of cases from two input files. The structure of case is stored in “Case pattern file”. The organization of “Case pattern file” is shown in Listing 3.12. Every line corresponds to one attribute, and here the name and the type of attribute are given. Name of attribute is an arbitrary string, while the type takes a value from a set `{string, int, float}`. The number of attributes is also arbitrary, depending on how the cases are modeled.

```
AttributeName1, AttributeType1,
AttributeName2, AttributeType2,
AttributeName3, AttributeType3,
...,
AttributeNameN, AttributeTypeN,
```

**Listing 3.12 Structure of “Case pattern file”**

The database of cases, or the case base, is stored in “Case base file”. Organization of this file is shown in Listing 3.13. Every line of “Case base file” corresponds to one case. The case is described with the values of its attributes and final solution of that case.

So, the values: `Valuei,1`, `Valuei,2`, `Valuei,3`, ... , `Valuei,N` represent the values of corresponding attributes of the `i`-th case. If any value is not known (value unknown), its place should be left empty. The value `Valuei,N+1` represents the solution of the `i`-th case. Solution is always listed at the end of line. Type of solution can also be integer, string or float, and it is determined dynamically, after all cases are parsed.
The fields and methods of class `CaseRetrievalNet` are shown in Listing 3.14.

```java
public class CaseRetrievalNet {
    public Attribute[] attributes;
    public List cases;
    public CaseRetrievalNet() {
        attributes = null;
        cases = null;
    }
    private static Attribute[] ReadPattern(Lexer pattern, int which) {
        // Parses "Case Pattern File" (recursively)
        // and returns the list of attributes
        char ch = pattern.Next();
        if (ch == '#') {
            return new Attribute[which];
        }
        else {
            if (ch == 'i' || pattern.commaread) {
                return null;
            }
            String name = pattern.name;
            ch = pattern.Next();
            if (ch == 'i' || pattern.EOLread) {
                return null;
            }
            String type = pattern.name;
            Attribute[] array = ReadPattern(pattern, which + 1);
            if (array == null) {
                return null;
            }
            if (type.compareTo("int") == 0) {
                array[which] = new Attribute(name, 'n');
            } else if (type.compareTo("float") == 0) {
                array[which] = new Attribute(name, 'f');
            } else if (type.compareTo("string") == 0) {
                array[which] = new Attribute(name, 'i');
            } else {
                Frame1.textArea1.append("Type " + type + " isn't recognized. Must be
int, float or string\n"");
                return null;
            }
        }
        return array;
    }
    private boolean eof;
    private CaseElement ReadOneCase(Lexer caseBase, int which) {
        // Reads one line from the "Case Base File" (recursively)
        char ch = caseBase.Next();
        ...
if (ch == '#') {
    eof = true;
    return null;
}
if (which < attributes.length) {
    if (! (caseBase.commaread)) {
        SkipToEOL(caseBase);
        Frame1.textArea1.append("Error (comma expected) in line: " +
                               caseBase.line+"\n");
        return null;
    } else {
        String name = "";
        float fvalue = 0;
        int ivalue = 0;
        IEEElement ie = null;
        if (((attributes[which].type == 'i')&&(ch == 'i'))
            name = caseBase.name;
        else if (((attributes[which].type == 'f')&&(ch == 'f'))
            fvalue = caseBase.fvalue;
        else if (((attributes[which].type == 'n')&&(ch == 'n'))
            ivalue = caseBase.ivalue;
        else if (ch != 'e') {
            SkipToEOL(caseBase);
            Frame1.textArea1.append("Type mismatch in line: " +caseBase.line);
            Frame1.textArea1.append(" ; Information entity: " +
                                    attributes[which].name+"\n");
            return null;
        }
        CaseElement oneCase = ReadOneCase(caseBase, which+1);
        if (oneCase == null)
            return null;
        if (ch == 'e')
            return oneCase;
        else if (ch == 'i') {
            attributes[which].informationEntities =
                List.Insert(attributes[which].informationEntities,
                new stringIEElement(name));
            //Inserts IE in the list of IEs
            ie = (IEElement) List.Find(attributes[which].informationEntities,
                                           new stringIEElement(name)).elem;
        } else if (ch == 'f') {
            attributes[which].informationEntities =
                List.Insert(attributes[which].informationEntities,
                new floatIEElement(fvalue));
            ie = (IEElement) List.Find(attributes[which].informationEntities,
                                           new floatIEElement(fvalue)).elem;
        } else {
            attributes[which].informationEntities =
                List.Insert(attributes[which].informationEntities,
                new intIEElement(ivalue));
            ie = (IEElement) List.Find(attributes[which].informationEntities,
                                           new intIEElement(ivalue)).elem;
        }
        //ie holds a pointer to information entity
        List arc = List.Find(ie.arcs, new Arc(oneCase));
        if (arc != null)
            ((Arc)(arc.elem)).weight++;
        //if the arc exists, just increment a weight of the arc
else {
    ie.arcs = List.Insert(ie.arcs, new Arc(oneCase));
    //insert a new arc
    //when the arc between case and ie doesn't exists (weight=1)
    oneCase.maxActivation++;
    //increase maximum activation of case
    return oneCase;
}
}
else { //which == attributes.length
    if (! caseBase.EOLread) {
        SkipToEOL(caseBase);
        Frame1.textArea1.append("Error in line: "+caseBase.line);
        Frame1.textArea1.append(". Wrong number of IEs!
");
        return null;
    }
    else {
        cases = List.Insert(cases, new CaseElement(caseBase.name));
        return (CaseElement)List.Find(cases,
            new CaseElement(caseBase.name)).elem;
    }
}
}

private void SkipToEOL(Lexer file) {
    while (! file.EOLread)
        file.Next();
}

public boolean Make(String patternFile, String caseBaseFile) {
    //Creates CRN from two input files; Returns "false" if error happened
    Lexer pattern = new Lexer(patternFile);
    attributes = ReadPattern(pattern, 0);
    if (attributes == null) {
        Frame1.textArea1.append("Error reading pattern file
");
        return false;
    }
    Frame1.textArea1.append("Pattern file loaded\n");
    Lexer caseBase = new Lexer(caseBaseFile);
    int errors = 0;
    int goodCases = 1;
    do {
        CaseElement oneCase = ReadOneCase(caseBase, 0);
        if (eof)
            break;
        if (oneCase == null) {
            errors++;
            Frame1.textArea1.append("Error reading case base file! Case in 
line "+caseBase.line+" not loaded\n");
        } else 
            goodCases++;
        Frame1.textArea1.append("SLUCAJ "+errors+goodCases+"\n");
        if (5*goodCases<errors) {
            Frame1.textArea1.append("NUMBER OF ERRORS IN CASE BASE FILE 
TOO BIG!!!\n");
            Frame1.textArea1.append("Reading aborted!!!\n");
            return false;
        }
    } while (! eof);
As already mentioned, class CaseRetrievalNet contains an array of attributes (in field attributes), and a list of solution nodes (in field cases).

Method CaseRetrievalNet is a standard constructor in which the initialization of case retrieval net is done.

The task of the method ReadPattern is to parse “Case pattern file” and to create and return the list of attributes. It has two parameters: pattern – the reference to input file, and which – the index of attribute that is parsed at the moment. Since the number of attributes is not known at the beginning, the method is realized as recursive. First it reads all data from the input file, creates an array of appropriate size, and then, in returning from recursion, fills the fields of array.

At the beginning, the method reads the next token from the input and checks whether it is the “end of file” (EOF). If the EOF is reached, it means that all attributes are parsed and an array of size which is created. If EOF is not reached, method parses the name and type of which-th attribute, and saves them in local variables name and type. After that, method calls itself in order to read the rest of attributes. After returning from recursion, it is checked whether the field type contains one of the values int, float or string and if so, the attribute of appropriate type is created.

The task of method ReadOneCase is to read one case from “Case base file”, to insert new information entities and solution nodes in case retrieval net if needed, to insert or update the list of arcs from information entity nodes to solution nodes, and to return a pointer to the solution node. The method has two arguments: caseBase – a reference to input file, and which – the index of attribute whose value is parsed at the moment. Since the solution is always listed at the end of line, the method is also realized as recursive. It reads all information entities and a solution, and after returning from recursion connects information entities to the appropriate solution.

At the beginning method reads a next token and checks if EOF is reached. If EOF is not reached, it checks, whether parameter which reached the length of attributes array.

- If the length of attribute array is not reached, method checks whether the type of token corresponds to the type of which-th attribute, and saves the value of name, fvalue or ivalue from Lexer. If everything passes well, method calls itself in order to parse the rest of case. After returning from recursion, method inserts an instance of class stringIEElement, floatIEElement or intIEElement in the list of information entities, depending of the type of attribute. Method Insert (from class List) is implemented in such way, so that it doesn’t insert an element in the list if it already exists. After insertion, it is only left to connect the information
entity to the solution node, which is stored in local variable oneCase. If the information entity doesn’t contain an arc to this solution node, the arc is inserted, but if such arc already exists the weight of arc is increased by one. Also the maximum activation of solution node is increased.

- If the length of attribute array is reached, that means that we reached an end of line and that the solution of the case is read. The solution is inserted in the list of solutions (if it doesn’t exist), and returned as a result of the method.

Method SkipToEOL escapes everything until end of line is reached. This method is called every time when error occurs in purpose of error recovery.

The task of method Make is to create case retrieval net. It has two parameters: the names of “Case pattern file” and “Case base file”, and returns a boolean value if everything passed ok. At the beginning, it opens a “Case pattern file” and calls a method ReadPattern, which creates a list of attributes. After that, method opens a “Case base file” and calls method ReadOneCase several times until EOF is reached. Also, it counts the number of correctly and incorrectly read cases (variables goodCases and errors). If the number of incorrect cases becomes five times higher than the number of correct cases the method terminates its job with the meaning “something is wrong with the input file”.

Class Frame1

The main task of class Frame1 is to define graphical user interface and to control the behavior of the whole system. Before explaining the class, the form of CaBaGe system will be explained. The main window of the system with the names of main objects is given on the Figure 3.2.

In fields jTextField1 and jTextField2 the user should enter the paths to the “Case pattern file” and “Case base file”. Also the buttons Browse can be used for navigation in file system to find appropriate files. Text area textArea1 is used as an information field for different text messages, while textArea2 is used for showing the results of the system. Panel jPanel1 is used as a placeholder for a dynamically created form in which the user should enter the current problem. This form is created on the basis of “Case pattern file” and it will be shown after the files are loaded. The behavior of buttons JButton3 and JButton4 will be described in detail within the next few paragraphs.

Since the class Frame1 contains a lot of fields and methods that control the behavior of application, only the two main methods will be shown here. These methods are shown in Listing 3.15.
Figure 3.2. Main window of CaBaGe

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Frame1 extends JFrame {

    // Behavior of "Load Files" button
    void jButton3_actionPerformed(ActionEvent e) {
        // Creation of Case Retrieval net
        crn = new CaseRetrievalNet();
        boolean succes = crn.Make(jTextField1.getText(),
                                jTextField2.getText());
        if (! succes)
            textArea1.append("Error reading files
                                 ");
        else {
            // Creation of the middle form
            labele = new JLabel[crn.attributes.length];
            polja = new JComboBox[crn.attributes.length];
            valuesOfActivation = new JTextField[crn.attributes.length];
            jPanel1.removeAll();
            for (int i = 0; i<crn.attributes.length; i++) {
                labele[i] = new JLabel();
                List tmp = crn.attributes[i].informationEntities;
                int j = 0;
```
while (tmp != null) {
    j++;
    tmp = tmp.next;
}
objects = new Object[j];
tmp = crn.attributes[i].informationEntities;
j = 0;
while (tmp != null) {
    objects[j] = tmp.elem;
    j++;
    tmp = tmp.next;
}
polja[i] = new JComboBox(objects);
polja[i].setEditable(true);
polja[i].setSelectedIndex(0);
String tmpstr;
if (crn.attributes[i].type == 'i')
    tmpstr = "String";
else if (crn.attributes[i].type == 'n')
    tmpstr = "Integer";
else
    tmpstr = "Float";
labele[i].setText(crn.attributes[i].name + " (" + tmpstr + ")");
valuesOfActivation[i] = new JTextField();
valuesOfActivation[i].setText("1");
jPanel1.add(labele[i], new XYConstraints(5, 5 + 30 * i, 150, 25));
jPanel1.add(polja[i], new XYConstraints(165, 5 + 30 * i, 90, 25));
jPanel1.add(valuesOfActivation[i],
            new XYConstraints(270, 5 + 30 * i, 40, 25));
}
jButton4.setEnabled(true);
jPanel1.updateUI();
}

// Behavior of "Solve" button
void jButton4_actionPerformed(ActionEvent e) {
    //Initialization of case activation
    List tmplst = crn.cases;
    while (tmplst != null) {
        ((CaseElement)(tmplst.elem)).activation = 0;
        tmplst = tmplst.next;
    }
    for (int i = 0; i < polja.length; i++) {
        textArea1.append(polja[i].getSelectedItem().toString() +
                        crn.attributes[i].type + "\n");
        float tempf = 1;
        try {
            tempf = Float.parseFloat(valuesOfActivation[i].getText().trim());
        }
        catch (NumberFormatException ex) {
            textArea1.append("Importance must be positive float value less
or equal 1.0. Error in attribute: " +
                            crn.attributes[i].name + "\n");
            tempf = 0;
        }
        if ((tempf != 0) || (tempf > 1))
            crn.attributes[i].valueOfActivation = tempf;
    }
    List tmp = crn.attributes[i].informationEntities;
while (tmp != null) {
    ((IEElement)tmp.elem).initialActivation = 0;
    tmp = tmp.next;
}
int n = 0;
float f = 0;
if (crn.attributes[i].type == 'i') {
    crn.attributes[i].valueOfQuery =
        new stringIEElement(polja[i].getSelectedItem().toString());
}
else if (crn.attributes[i].type == 'n') {
    try {
        n=Integer.parseInt(polja[i].getSelectedItem().toString().trim());
    } catch (NumberFormatException ex) {
        textArea1.append("In the field "+crn.attributes[i].name+
            " must be an Integer value!");
        return;
    }
    crn.attributes[i].valueOfQuery = new intIEElement(n);
} else if (crn.attributes[i].type == 'f') {
    try {
        f=Float.parseFloat(polja[i].getSelectedItem().toString().trim());
    } catch (NumberFormatException ex) {
        textArea1.append("In the field "+crn.attributes[i].name+
            " must be an Float value!");
        return;
    }
    crn.attributes[i].valueOfQuery = new floatIEElement(f);
} crn.attributes[i].SetActivation();

//Propagation of activation
List ie = crn.attributes[i].informationEntities;
while (ie != null) {
    List arc = ((IEElement)ie.elem).arcs;
    while (arc != null) {
        ((Arc)arc.elem).caseNode.activation =
            ((Arc)arc.elem).caseNode.activation +
            ((IEElement)ie.elem).initialActivation*((Arc)arc.elem).weight;
        arc = arc.next;
    }
    ie = ie.next;
}

//Printing the results
textArea2.setText("");
tmplst = crn.cases;
while (tmplst != null) {
    textArea2.append(((CaseElement)tmplst.elem).name+
        ": "+Float.toString(((CaseElement)tmplst.elem).activation)+"\n");
    tmplst = tmplst.next;
} ie = crn.attributes[i].informationEntities;
while (ie != null) {
if (crn.attributes[i].type == 'i')
    System.out.println("  " + ((stringIEElement)(ie.elem)).name + " ACT= " + ((IEElement)(ie.elem)).initialActivation);
else if (crn.attributes[i].type == 'n')
    System.out.println("  " + ((intIEElement)(ie.elem)).value + " ACT= " + ((IEElement)(ie.elem)).initialActivation);
else if (crn.attributes[i].type == 'f')
    System.out.println("  " + ((floatIEElement)(ie.elem)).value + " ACT= " + ((IEElement)(ie.elem)).initialActivation);
List arc = ((IEElement)(ie.elem)).arcs;
while (arc != null) {
    System.out.println("    " + ((Arc)arc.elem).weight + " -> " + ((Arc)arc.elem).caseNode.name);
    arc = arc.next;
}
    ie = ie.next;
}

Listing 3.15 Class Frame1

The method jButton3_actionPerformed is executed when the user presses the JButton3 button. At the beginning, method reads the paths of “Case pattern file” and “Case base file” from fields jTextField1 and jTextField2, and creates case retrieval net if the paths and files are good. After that method creates form in the panel JPanel1, based on the structure of case retrieval net in which the user should enter the current problem (query). The form contains rows as much as case retrieval net contains attributes. Every row contains:

- Name and type of the attribute, stored in array of labels labelɛ[].
- List of all values of attribute that appear in case retrieval net, stored in array of combo boxes polja[]. For every attribute, method reads the list of values (from field informationEntities) and creates a combo box based on these values. These combo boxes allows user to select one of the values from case retrieval net, but also to enter some new value.
- Field for entering the importance of the current attribute, stored in array of fields valuesOfActivation[]. The importance is the float value from the interval [0,1], and the default value is 1.

After that, it is expected, that user enters the current problem in the created form. Pressing JButton4 button starts the process of solving the problem. When this button is pressed, the method jButton4_actionPerformed is invoked. At the beginning, method reads the value of attribute and an importance (stored in polja[i] and valuesOfActivation[i]) and sets the value of query and the initial activation for the corresponding attribute (i). If the type of the attribute isn’t equal to the type of the user's input, an appropriate message is shown and the execution is canceled. After setting the values of query and initial activation, the method SetActivation (from the class Attribute) is called, in order to split the initial activation from value of query to most similar information entities in the case retrieval net.
When the splitting of initial activation is completed, the process of propagation of activation is started. The activation is propagating from all information entities that have a nonzero initial activation to all case nodes which are connected with nonzero arcs, by multiplying the activation with the weight of arc. The final activation in case nodes is calculated by summing all gained activations. After that, it remains only to print the results – case nodes and their percentage of activation (final activation divided by maximal activation). Suggested solution is a case node with the highest percentage of activation.

**Class Application1**

Class Application1 is a class with the main method, and its only purpose is to initialize the main window.

**An Example**

The application of the system will be shown on one simple test domain: a domain for determination species of animals when some features of the animals are given. Database for this domain and also some others are freely available at [CBR at AIAI], [UCI MLR].

The structure of a case is described in case pattern file “default.key”, and it is given in Listing 3.16. All attributes except NumberofLegs are of string type. All other attributes except AnimalName are in fact of boolean type which is simulated with string type with just two values: “Yes” or “No”.

<table>
<thead>
<tr>
<th>AnimalName, string</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasHair, string</td>
</tr>
<tr>
<td>HasFeathers, string</td>
</tr>
<tr>
<td>LaysEggs, string</td>
</tr>
<tr>
<td>GivesMilk, string</td>
</tr>
<tr>
<td>CanFly, string</td>
</tr>
<tr>
<td>LivesinWater, string</td>
</tr>
<tr>
<td>Predator, string</td>
</tr>
<tr>
<td>HasTeeth, string</td>
</tr>
<tr>
<td>HasA Backbone, string</td>
</tr>
<tr>
<td>BreathesWithLungs, string</td>
</tr>
<tr>
<td>IsVenomous, string</td>
</tr>
<tr>
<td>HasFins, string</td>
</tr>
<tr>
<td>NumberofLegs, int</td>
</tr>
<tr>
<td>HasTail, string</td>
</tr>
<tr>
<td>IsDomestic, string</td>
</tr>
<tr>
<td>IsCat Sized, string</td>
</tr>
</tbody>
</table>

**Listing 3.16 File “default.key”**

The cases are given in the case base file “default.cbr”. Part of this file is given in Listing 3.17. Every case is given in a new line and a case base consists of 101 cases. The values of the attributes must be given in the correct order according to the case pattern file. Last value in the description of the case is always the solution of the case. These values are determined dynamically and in this domain can be: Amphibian, Anthropoid, Bird, Fish, Insect, Mammal or Snake.
The Structure of Case Retrieval Net

On the basis of input data the system should create case retrieval net. As already mentioned, case base consists of 101 cases, and all cases belong to 7 different groups of animals. The numbers of cases according to their group are:

- 4 cases are from **Amphibian** group
- 10 cases are from **Anthropod** group
- 20 cases are from **Bird** group
- 13 cases are from **Fish** group
- 8 cases are from **Insect** group
- 41 cases are from **Mammal** group
- 5 cases are from **Snake** group

From these facts it is obvious that sometimes the data in the case base can be disproportional. This is the main reason why the solutions are sorted according to their relative activation (percentage of activation described in the class **Frame1**) instead of the real activation. In this situation, if the real activations were used, the solution **Mammal** would almost always be the winner. The reason is that it occurs in the most of the cases, and the sum of the weights of the arcs leading to this solution in CRN is considerably larger than the sum of the arcs leading to the other solutions.

The structure of case retrieval net is very huge even for this simple domain. Considering that, only a part of case retrieval net containing 2 attributes and all the solutions will be shown here.

The attribute **NumberofLegs** describes the number of legs of the corresponding animal. It is the only attribute of the integer type. The values appearing in the case base are: 0, 2, 4, 5, 6 and 8. The number of cases considering their solutions and values of the attribute **NumberofLegs** is given in Table 3.1. The graphical representation of these data is given in Figure 3.3.
Table 3.1. Number of cases concerning attribute NumberofLegs.

<table>
<thead>
<tr>
<th>Number of Legs</th>
<th>Number of Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 legs</td>
<td>3</td>
</tr>
<tr>
<td>2 legs</td>
<td>7</td>
</tr>
<tr>
<td>4 legs</td>
<td>31</td>
</tr>
<tr>
<td>5 legs</td>
<td>0</td>
</tr>
<tr>
<td>6 legs</td>
<td>0</td>
</tr>
<tr>
<td>8 legs</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Legs</th>
<th>Number of Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 legs</td>
<td>0</td>
</tr>
<tr>
<td>2 legs</td>
<td>20</td>
</tr>
<tr>
<td>4 legs</td>
<td>0</td>
</tr>
<tr>
<td>5 legs</td>
<td>0</td>
</tr>
<tr>
<td>6 legs</td>
<td>0</td>
</tr>
<tr>
<td>8 legs</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Legs</th>
<th>Number of Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 legs</td>
<td>0</td>
</tr>
<tr>
<td>2 legs</td>
<td>0</td>
</tr>
<tr>
<td>4 legs</td>
<td>2</td>
</tr>
<tr>
<td>5 legs</td>
<td>0</td>
</tr>
<tr>
<td>6 legs</td>
<td>0</td>
</tr>
<tr>
<td>8 legs</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Legs</th>
<th>Number of Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 legs</td>
<td>0</td>
</tr>
<tr>
<td>2 legs</td>
<td>0</td>
</tr>
<tr>
<td>4 legs</td>
<td>4</td>
</tr>
<tr>
<td>5 legs</td>
<td>1</td>
</tr>
<tr>
<td>6 legs</td>
<td>1</td>
</tr>
<tr>
<td>8 legs</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Legs</th>
<th>Number of Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 legs</td>
<td>0</td>
</tr>
<tr>
<td>2 legs</td>
<td>0</td>
</tr>
<tr>
<td>4 legs</td>
<td>0</td>
</tr>
<tr>
<td>5 legs</td>
<td>0</td>
</tr>
<tr>
<td>6 legs</td>
<td>0</td>
</tr>
<tr>
<td>8 legs</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Legs</th>
<th>Number of Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 legs</td>
<td>0</td>
</tr>
<tr>
<td>2 legs</td>
<td>0</td>
</tr>
<tr>
<td>4 legs</td>
<td>0</td>
</tr>
<tr>
<td>5 legs</td>
<td>0</td>
</tr>
<tr>
<td>6 legs</td>
<td>0</td>
</tr>
<tr>
<td>8 legs</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Legs</th>
<th>Number of Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 legs</td>
<td>0</td>
</tr>
<tr>
<td>2 legs</td>
<td>0</td>
</tr>
<tr>
<td>4 legs</td>
<td>0</td>
</tr>
<tr>
<td>5 legs</td>
<td>0</td>
</tr>
<tr>
<td>6 legs</td>
<td>0</td>
</tr>
<tr>
<td>8 legs</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3.3. Distribution of cases concerning attribute NumberofLegs.

The values given in Table 3.1 in fact represent the weights of relevance arcs between information entity nodes of the attribute NumberofLegs to the solution nodes (according to the algorithm described in the class CaseRetrievalNet). For example, the weight of the arc between information entity node <NumberofLegs, 2 legs> to the solution node Bird is 20.

From the Figure 3.3, it is obvious that for certain value of the attribute NumberofLegs the certain kind of animal is mostly activated. For example, if the information entity node <NumberofLegs, 6 legs> is activated, in that case the solution node Insect gets the biggest activation from the attribute NumberofLegs. But this is only the 17th part of all the activations that solution nodes get, if weighted query equally distribute activation to all attributes.

The second attribute, LaysEggs, is attribute which describes if the animal layes eggs. Its type is string, but in fact it represents a boolean value (values Yes/No). The distribution of the cases, concerning attribute LaysEggs is given in Table 3.2.
The structure of Case Retrieval Net, containing information entity nodes of two described attributes and all 7 solution nodes is given in Figure 3.4.

![Figure 3.4. A part of Case Retrieval Net](image)

Table 3.2. Number of cases concerning attribute LaysEggs.

<table>
<thead>
<tr>
<th></th>
<th>Mammal</th>
<th>Insect</th>
<th>Bird</th>
<th>Fish</th>
<th>Snake</th>
<th>Amphibian</th>
<th>Anthropod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>1</td>
<td>8</td>
<td>20</td>
<td>13</td>
<td>4</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>No</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The weights of relevance arcs are the values given in tables. The weights of acceptance arcs are computed according to the algorithm described in method SetActivation() in the class Attribute. These weights are implicitly calculated when the weighted query is entered. The acceptance arc weights are not static for a given case retrieval net, but they are computed dynamically when the importance values of the weighted query is known. This is the consequence, of a property, that the sum of the arcs between information entities of one attribute must be equal to the importance value of that attribute in the weighted query.

The acceptance arcs are generally defined between every two information entity nodes of one attribute. But in this implementation, the arc exists (the weight is greater than 0) between current information entity node and only 20% of the most similar information entity nodes to the current node. Their weights depend of values of weighted query so they are not shown on the figure. The arc between information entity nodes <LaysEggs,Yes> and
doesn’t exist, because these two information entities are not similar at all.

**Application of the System**

The main window of the system is already shown in Figure 3.2. The system expects that the user enters the paths of the case pattern and the case base file.

If the paths are correct, after clicking on the “Load Files” button, these two files are loaded and the case retrieval net is created. Also, the dynamically created form will appear in the middle of the window. This situation is shown in Figure 3.5.

![Case Base Generator](image)

*Figure 3.5. CaBaGe after loading files*

After loading files, creating CRN and the case-base for that particular domain, the system expects from the user to enter current problem – to enter all known values of the attributes. If some values are not known then the user should enter the zero value in the corresponding importance field. When the act of entering the data into the form is completed, the process of spreading activation starts by clicking on the “Solve” button. In the right part of the window the solution is shown after completing the process. There all possible solutions and their activations are shown as in Figure 3.6. The solution with the highest number is the suggested solution. For this example it is most possible that the *problem animal* is Mammal, since it has the highest activation.
Figure 3.6. CaBaGe after solving the problem.

Of course, at this moment, some new “problem animal” can be entered in the form. The system always suggests some solution, but the quality of the solution depends on the quality of the input data.
4. Case Based Reasoning in Medical Domain

Medicine is a rather suitable domain for application of CBR because the knowledge of experts consists of a mixture of textbook (objective) knowledge and experience (subjective) knowledge (consists of cases). The problem of updating changeable subjective knowledge can partly be solved by incrementally incorporating new up-to-date cases. Usually, two sorts of knowledge can be clearly separated, and represented in an appropriate manner. Objective knowledge can be represented in forms of rules or functions, while subjective knowledge is contained in cases. So, there are several obvious facts for usage of case-oriented methods in medicine:

1. Reasoning with cases corresponds to a decision making process of physicians.
2. Incorporating current cases obtains automatically updated parts of knowledge.
3. Two distinguished types of knowledge can be clearly separated.
4. Integration into a clinic communication system is easy (cases are routinely stored).

Characteristics of Medicine

Medicine differs from other knowledge domains by the interaction of research and practice. The objects are the patients - complex organisms with high biological variance and a lot of interaction. The knowledge of these processes and their interactions is often weak. It mostly depends on a high number of sometimes even contradictory signs and symptoms. Furthermore, individual processes are affected by changes of environmental situations. To discover new medical knowledge, traditional research is based on disease case descriptions, case collections, and biostatistical case studies.

In contrast to most of other knowledge domains, clinical practice is characterized by a professional documentation of cases.

On the one hand, knowledge of special topics is often highly centralized. This means that only one person is an expert in his field. On the other hand, medical knowledge in different clinics is distributed into specialists and knowledge sources, such as patient oriented documents, medical journals, data bases, text works, etc.

Physicians are overwhelmed by the amount of data, even for one patient (e.g., hundreds of parameter values from a clinical laboratory, numerous images from a radiology or endoscope, etc., and dozens of textual documents). Physicians are usually confronted with contradictory information. Moreover, they have to cope with vague data, e.g., complaints of a patient.
However, clinicians mostly work under stress. This means that, they have to choose rapidly from a large number of alternatives. For instance, they have to decide for an appropriate dosage of drugs in order to preserve a desired impact for a particular disease.

The process of decision making of a physician is unique for the medical domain. It is strongly associated with the object of the decision – the patient. Figure 4.1 roughly describes a general sequence of tasks which is devoted to any patient [Hadorn, Zoellner 1979]. The feedback arcs indicate that this process is cyclical.

![Figure 4.1. Main steps for treatment of patient](image)

Case based reasoning approach does not try to understand in detail how physicians make their decisions. Most of experienced physicians are unable to describe precisely the first step of their disease recognition which could explain the symptoms of a patient. In most situations, they obviously recall a prototype, a typical case or an exception of a disease.

Physicians and desired knowledge based systems use the following facts and knowledge in decision making:

- Symptoms of diseases, used as triggers to find hypotheses initially.
- Knowledge on increased risks for a patient associated with certain diseases.
- Frequency of diseases derived from epidemiological studies.
- Frequency of diseases derived from individual experience.

Although a diagnosis is the most common application type in medical knowledge systems, the other application types can be also modelled for utilization in a medical domain:

- Diagnosis
Advantages of CBR in Medicine

Case-based reasoning is appropriate for application in a medical domain for several important reasons:

- Similarity to Human Cognitive Processes
- Usage of Experience
- Duality of Objective and Subjective Knowledge
- Automatic Learning of Subjective Knowledge
- Easy System Integration

Similarity to Human Cognitive Processes

Case-based systems correspond to the incremental keeping in mind, abstraction and recognition of human knowledge in a natural manner. As another important aspect, we point out that physicians often use their case experiences from their clinical practice to treat a current case more efficiently, to diagnose similar exceptions, etc.

Usage of Experience

CBR systems use less formal knowledge (e.g., rules) but extend, replace, change, and forget knowledge by continuous integration of cases into the case base. Thus, each case-based system can adjust itself to its current environment and to specific requirements of a clinic or a surgery.

Duality of Objective and Subjective Knowledge

In medicine, it is conventional to discover new knowledge by evaluating data through the use of clinical cases, books, journals, etc. Such knowledge is objective; it does not depend on the explorer. This knowledge, contained in textbooks, norms and proceedings, is not specific enough and often it is not accepted because of different views of medical problems or different opinions and experiences in diagnosis and therapy.

Most often, the process of diagnosis and therapy planning in practice is different. Physicians usually use their subjective knowledge which is mainly the result of their own experience. This knowledge is limited in time and space to every individual physician. Nevertheless, such knowledge is applicable because it is specific to every patient and because it is rather accepted by physicians. Here is an obvious dilemma: which knowledge should be used in conventional expert systems. The general desire in medical knowledge-based systems is to combine objective and, time and space limited, subjective knowledge.
This dualism can be naturally integrated in a case-based system. Such system contains cases (subjective knowledge) and formal rules (generalized knowledge) as an objective knowledge. By this separation of objective and subjective knowledge, CBR leads to objectivity and to specificity of the knowledge base and helps in solving the dilemma mentioned above.

**Automatic Learning of Subjective Knowledge**

The continuous integration of correctly solved cases during the use of a CBR system leads to an incremental knowledge acquisition. Furthermore, the system is able to abstract knowledge by generalizing cases through case groups to prototypes.

**Easy System Integration**

The use of data and medical patient records, which have already been collected by hospitals and stored on machine readable mediums, allows an easy integration of CBR system in existing clinical communication systems.

Case-based systems, which combine knowledge acquisition as a machine learning process during the consultation process, can give an important contribution to the solution of the knowledge acquisition problem. Some authors [Lenz et al. 1998] certify that a knowledge base generated from cases, seen and documented in a clinic, can produce a more accurate consultation than a knowledge-base compiled from case descriptions in international medical literature of the domain.

**Problems of CBR in Medical Applications**

Some authors [Schmidt, Gierl] think that the main problem of CBR technology is the adaptation task. It still depends both on domain and application characteristics. In medicine, adaptation can be a serious problem, because cases often consist of an extremely large number of features. So some adaptation solutions, typical for medical domains, have been developed:

1. **Focusing on retrieval** – An idea to avoid adaptation problem is to build retrieval-only systems. Retrieval is used only to find similar cases and present them to the user. Some systems have possibility to point out important differences between current and previous cases. The justification for giving up the adaptation task is that physicians are interested to get information about former similar cases, but they wish to reason and bring diagnosis for a current patient by themselves. A similar approach is to combine CBR with rule-based methods. In some concrete systems if no similar case can be found or if all adaptation problems can not be solved, a separate rule-based component of the system is applied.

2. **Generalized cases** – In this solution, the main idea is to form abstracted prototypes by generalizing them from single cases. However, generalization obtains to structure the case base, to decrease the storage amount by erasing redundant and highly similar cases and additionally it at least can help partly to solve the adaptation problem.
Techniques for Medical CBR Systems

Prototypes

The prototypes are used as a form of knowledge representation that fills the gap between specific cases and general rules. Some cases are clustered. The most typical example for this cluster is called a prototype [Gierl, Stengel-Rutkowski 1992], [Hampton 1993], [Murphy 1993], [Schmidt, Gierl 1996]. Case-based knowledge allows the incremental modification of the memory structure, from single cases via case groups to prototypes and eventually to taxonomy of diseases or therapies in a medical domain. A prototype shows essential signs and symptoms.

Intuitively speaking, a prototype can be thought of as the result of grouping together and fusing of typical cases of a disease.

The knowledge of physicians consists of a general knowledge having been obtained from medical books plus their experiences represented by cases they have treated themselves or colleagues have told them about. They consider the differences between a current patient and typical or known exceptional cases.

The main purpose of such generalized knowledge is to guide the retrieval process and sometimes to decrease the amount of memory requirements by erasing redundant cases. In domains with rather weak domain theories, another advantage of case-oriented techniques is their ability to learn from cases [Schmidt, Gierl 1996]. Only gathering new cases may improve the system's ability to find suitable similar cases for current problems, but it does not bring out the essential knowledge of the stored cases. To learn the knowledge contained in cases, a generalization process is necessary.

Tversky [Tversky 1977] determined the similarity between a case C and a prototype P by adding up a number of shared features, subtracting the number of features of the prototype which the case does not share, and subtracting the number of features the case does not share with the prototype:

\[ D(P,C) = \alpha \cdot f(P+C) - \beta \cdot f(C-P) - \gamma \cdot f(P-C) \]  

where, the coefficients \( \alpha, \beta, \gamma > 0 \) are scaling values.

The purpose of prototypes is to structure the case base and to guide the retrieval. The general desire is that prototypes of the system correspond directly to the physician's sense of prototypes.

The Abstraction of Time Depended Parameters

Time-depended parameters are essential for diagnosis and therapy. Very often, more than one parameter has to be considered. Reasoning concerns, therefore, a very complex problem of multiparametric time courses. It is a problem each physician is confronted with. The more parameters have to be considered for decision making, the greater is the information overload of the physician.

Some authors declare that post-operation care of transplanted patients requires the observation of up to 60 parameters in parallel over time. It is obvious that a knowledge-based system
would be helpful in dealing with this problem. One approach could be to abstract context sensitive time course concerning data and time.

**CBR Systems in Medicine**

Thorough the development of artificial intelligence, many medical systems are developed. The situation is the same in the area of case based reasoning, although its history is much shorter. Many pure CBR systems, but also systems which combine CBR with other knowledge techniques, are developed with different success. Here, an overview of some interesting, realized systems concerning their application type will be given.

**Diagnosis**

Diagnosis is the most common application type for all knowledge based techniques in medical domain. Therefore, the numerous applications for diagnosis are developed.

**4.1.1.1. CASEY**

CASEY [Koton 1988] is an expert system in the domain of heart failure diagnosis that combines case-based and rule-based reasoning techniques. The system uses three steps: a search for similar cases, a determination of differences and their evidences, and a transfer from the diagnoses of similar cases or - if the differences are too important - an attempt to explain and repair them. If no similar case can be found or if all repair attempts fail, CASEY uses the rule-based domain theory.

The most interesting aspect of CASEY is the attempt to use more general adaptation operators. For medical applications, some CBR researchers have entirely given up the claim to perform any adaptation at all while others apply only very specialized operators. However, in CASEY all adaptation problems could not be solved.

**4.1.1.2. MEDIC**

The interesting aspect of MEDIC [Turner 1988] is not a medical application, but a memory organization. MEDIC is a schema-based diagnostic reasoner on the domain of pulmonology. Schemata represent the problem solver's knowledge. These are packets of procedural knowledge about how to achieve a goal or a set of goals.

The memory does not only consist of schemata, but additionally of diagnostic memory organization packets of individual cases of diagnosis and of scenes. A scene represents an instantiation of a schema in a particular case. This memory organization and retrieval allows a reasoner to find the most specific problem-solving procedures available.

**4.1.1.3. PSIQ**

Another example is PSIQ [Schwartz 1997] which gives diagnostic and therapeutic advice in the domain of mental disorders. After a language treatment, component has delivered ICD-10 categories from the patient's history and grouped them, concerning different aspects. These categories can be used to search for similar cases which are shown to the user (including their diagnoses and treatments).
4.1.1.4. GS.52

GS.52 [Gierl, Stengel-Rutkowski 1992]; [Gierl, Stengel-Rutkowski 1994] is a prototype-based expert system that has been routinely used in the children's hospital of the University of Munich for many years. It is a diagnostic support system for dysmorphic syndromes. Such a syndrome involves a non-random combination of different disorders. The major problems are a high variability of the syndromes (hundreds), a high number of case features (between 40 and 130) and continuous modifications of the knowledge about dysmorphic syndromes.

Each syndrome is represented by a prototype that contains its typical features. The prototypes are acquired during an expert consultation session. A physician selects a new or an existing syndrome and typical cases for this syndrome. Subsequently, GS.52 determines the relevant features and their relative frequency.

The diagnostic support occurs while searching for the most adequate prototypes for a current case. Similarity value between each prototype and the current case is calculated and the prototypes are ranked according to these values.

The consultation produces a list of suspicious syndromes. The authors defined that the system provides a correct diagnosis if the correct dysmorphic syndrome appeared on the first screen. This screen comprises the 10 most suspicious diagnoses. They count a situation as a failure of the system if it ranks the syndrome on the following screens. On the following screens the physician can not see the correct diagnosis at first sight. It could be possible that the physician selects a wrong diagnosis if the correct one is mixed up with many other possible diagnoses.

4.1.1.5. Other Diagnostic Systems

Grimnes and Aamodt [Grimnes, Aamodt 1996] have developed a prototype of an image interpretation system for computer tomography with two layer CBR architecture. They want to use two case-based reasoners, one for lower level segment identification and the other for a higher level interpretation and understanding of the image as a whole.

Also, one image understanding system is SCINA [Haddad et al. 1997] which derives an assessment concerning the presence of coronary artery disease from a scintigraphic image data set automatically. Each image consists of 6 planes; each plane is divided into 12 segments. For each segment, a value of the relative thallium activity obtained by polar map analysis is determined. The retrieval, which is performed by nearest neighbour match, considers these 72 integer values, those from segments of planes placed in the middle are weighted high, those from the edges are weighted low. As the case base comprises only 100 cases, an indexing mechanism is not necessary.

MERSY is a system for rural health care workers [Opiyo 1995]. This system utilizes the most convincing advantage of CBR - the relocation of expertise. This is especially valuable in countries where the health care workers are rare and perform with only modest skills. The knowledge-base can be automatically adapted to the special health care problems of a region by simply using the CBR system.

The system developed by Evans [Evans 1996] (diagnosis of dysmorphic syndromes) is based on an incremental algorithm to generate a hierarchical network of generalized cases from
prototypical signs of syndromes. This algorithm focuses on the semantics of the two highest levels of the hierarchy. They are built from trigger signs of a syndrome and from important but secondary signs. This is necessary to avoid the building of artefacts which a physician would assess as medical nonsense.

Some other diagnostic domains are bone healing [Seitz and Uhrmacher 1996], histopathology [Jaulent 1997], which uses a combination of surface and structural similarity to retrieve similar pathological cases, urology [Burkhard et al. 1996], which uses Case Retrieval Nets (described in this thesis) as a mechanism for retrieval of similar previous cases, and MEDUSA [Fathi-Tortsagham and Meyer 1994] a hybrid system using cases, rules and fuzzy sets in the domain of diagnosis acute abdominal pain. In [Stamper et al. 1994] the possibilities of CBR for explanation in medical diagnosis applied to a problem in gynecology are described.

**Classification**

Classification is also commonly used in medicine. The problem of classification is to find a correct class of the input data in a finite set of classes. Classification is most frequently used in different image classification and in some initial diagnosis (diagnosis without additional tests).

### 4.1.1.6. ProtoISIS

ProtoISIS [Kahn and Anderson 1994] is used to provide an intelligent retrieval for image studies. The features of a case are clinical indications and questions which have to be answered. A semantic network relates cases, features and imaging procedures to built explanations for the user's questions. Relations in the network are one of the following: "feature of", "causes," "has exemplar", or "visualizes". ProtoISIS does not use a similarity measure or adaptation of former cases.

### 4.1.1.7. MacRad

The goal of MacRad [Macura and Macura 1995] is to provide a feature-coded image resource that allows the user to formulate image content-based queries when searching for reference images. As it is implemented in 4th Dimension, a relational database management system, it provides multilevel access to data, uses some hierarchies for the indexing, applies a precise matching of constraints expressed in the query for the retrieval, but does no adaptation. Macura justifies the lack of adaptation with the statement that, in medical domains, the diagnostic judgment should remain the user's responsibility.

### 4.1.1.8. MNAOMIA

Bichindaritz [Bichindaritz 1994] developed a system called MNAOMIA for eating disorders, whose memory consists of models, concepts, prototypes and cases. An object-oriented methodology is used for generalization / specialization between these different knowledge structures.
4.1.1.9. Other Classification Examples

Classification for structuring the Case Base is used to structure the case base by prototypes. Malek [Malek 1995] proposed a memory structure combining one level of prototypes, each of which represents a group of cases, with an incremental neural network. She applied this approach to the initial diagnosis of the cause of toxic comas and to neuropathy diagnosis [Malek and Rialle 1994].

The knowledge-base in [Malek and Rialle 1994] consists of a two-level structure. The first level comprises automatically generated prototypes which point to a set of similar cases, each associated with a prototype. Prototypes are unique cases from a sufficient number of cases of the same diagnosis. Atypical cases, which cannot be associated with any prototype, are stored as a special group. During an evaluation and correction phase, new prototypes can be generated or the existing ones can be deleted. Cases can be shifted to the group of atypical cases. The physician, as evaluator, accepts or rejects diagnoses of cases and, thus, triggers the reorganization of the knowledgebase.

Prototype-based CBR systems are an appropriate answer to the problem of the large number of cases that, even for a small medical domain, a physician or a clinical ward encounters in a few years.

CBR methods have also been used to implement knowledge based systems for scientific projects. For example, in [Wendel 1995] cases are experiments for simulating a neural network.

Planning

Compared to CBR systems for diagnosis and classification, the problem of planning attracted fewer researchers. The planning is most commonly used in patient's therapy planning and in planning of personal and material recourses in hospitals and clinics.

4.1.1.10. FLORENCE

The FLORENCE system [Bradburn and Zeleznikow 1993] deals with a health care planning for nursing, which is a less specialized field. It fulfils all three basic planning tasks; diagnosis, prognosis and prescription. Diagnosis is not used in the common medical sense as the identification of a disease but it seeks to answer the question "What is the current health status of this patient?" Rules concerning weighted health indicators are applied. The health status is determined as the score of the indicator weights.

Prognosis seeks to answer the question "How may the health status of this patient change in the future?" Here a case-based approach is used. The current patient is compared to a similar previous patient for whom the progression of the health status is known. Similar patients are searched for first concerning the overall status and subsequently concerning the individual health indicators. As the further development of a patient does not depend on his situation only (current health status, basic and present diseases), but additionally on his further treatments, several individual projections for different treatments are generated.

Prescription seeks to answer the question "How the health status of this patient may be improved?" The answer is given by utilizing general knowledge about likely effects of
treatments and also by considering the outcome of using particular treatments for similar patients. It means that it is a combination of rule-based and case-based approaches. The best prognoses and their treatments are considered. As the treatments may come from different patients, they may be contrasting or forbidden to combine. Additionally, to deal with time and duration problem of treatments, rules about treatment constraints are applied.

4.1.1.11. ICONS: Antibiotic Therapy Advice

ICONS [Gierl et al. 2003] is an antibiotics therapy adviser for intensive care patients who develop an infection as additional complication. The identification of the pathogen that causes the infection needs at least 24 hours in the laboratory. In contrast to normal patients, where physicians usually can wait for the results from the laboratory, intensive care patients need an immediate introduction of an appropriate antibiotics therapy. As the real pathogen is still unknown, a spectrum of probable pathogens has to be calculated. The aim of ICONS is to give rapid antibiotic therapy advice. CBR techniques are used to speed up the process of finding suitable antibiotics therapies and to update those parts of the knowledge-base that are modified frequently.

ICONS has some interesting features:

- **Antibiotics Selection Strategy.** ICONS is not a diagnostic system. It is not used to attempt to deduce evidence for the diagnosis, but instead pursue a strategy that can be characterized as follows: Find all possible solutions and reduce them using the patient's contra-indications and the complete coverage of the calculated pathogen spectrum (establish-refine strategy). The first list of antibiotics is generated by a "weak" relation that returns, for each group of pathogens, all antibiotics which usually have therapeutic effects. The second list of antibiotics is generated by reducing the first one by applying criteria such as the patient's contra-indications and the desired sphere of activity. Before the user chooses one therapy, he can investigate potential side effects of the antibiotics. Moreover, he can obtain information about the calculated spectrum and the daily costs of each suggested therapy. After the physician has chosen one therapy, ICONS computes the recommended dosage.

- **Adaptation of a Similar Case.** The principal argument for CBR, that it is often faster to solve a new problem by modifying the solution of a similar case, also applies to the process of finding adequate therapies. Considering the close relation concerning the group of patients and the affected organ, a similar case is retrieved from a hierarchical and generalizing memory structure containing prototypes as well as cases. Furthermore, a criterion for adaptability is used during the retrieval, because similar cases with additional contra-indications in comparison to the current patient are not adaptable. The adaptation is performed by a solution transfer of advisable therapies of a similar case and subsequently by a reduction concerning additional contra-indications of the current patient.

- **Adaptation to the Results of the Laboratory.** Identifications of pathogens and sensitivity tests (antibiograms) made in the laboratory are used as control mechanisms. When the actual pathogen is identified, the calculated pathogen spectrum is replaced by this pathogen and the already started therapy has to be checked to see if it fits for the identified pathogen. The set of possible antibiotic therapies is reduced by the antibiogram to those antibiotics the pathogens are not resistant to.
4.1.1.12. Therapy Support

The authors in [Petersen et al. 1994] developed a system for postoperative pain therapy. In [Jurisica, Shapiro 1995] CBR methods are applied to therapy problems in gynaecology. ROENTGEN supports radiation therapy planning [Berger 1989].

4.1.1.13. Planning of Personal or Material Resources

The CAMP system is an experimental prototype for daily menu planning which meets individual, nutritional and personal preference requirements [Kovacic et al. 1992].

Tutoring

Education in medicine is largely based on confronting students with actual cases in clinics. In some countries, even in the first more theoretical phase of the study, practical teaching in clinics takes place. Theoretical and practical teaching, as well as textbooks, uses cases to explain medical problems. Therefore, CBR should be particularly suitable for knowledge-based instruction. Unfortunately, only a few systems have been published.

4.1.1.14. PlanAlyzer

An early example of a tutorial system is PlanAlyzer [Beck et al. 1989]. It is one of the early products of a tutorial system developed as the part of a long term effort at Dartmouth College using computers in medical education. PlanAlyzer has been applied to anaemia and coronary heart diseases. It has been evaluated in a controlled clinical study. The result was that the using of the PlanAlyzer significantly improves efficiency in studying special medical domains.

4.1.1.15. Neurology Trainer

Another CBR System for instruction in Medicine is the Neurology Trainer [Puppe et al. 1995]. It supports the training of students in neurological diseases. Neurology Trainer is based on the hybrid diagnostic shell D3 using rules and cases.

Prognosis

Prognosis is the task type which has great potentials in medicine. Usually, the prognosis of a patient's state and his/her risks is assumed. Furthermore, some researchers have produced systems for prognosis of spread of diseases and epidemics on the global level. In both cases, a good prognosis could help in preventing some risks and bigger damages.

4.1.1.16. COSYL

Liver-Transplantation is a highly efficient therapy for well-selected patients with an acute liver-failure or an end-stage chronic liver disease with poor prognosis. However, low 2-year mortality rates enforce a search for the best treatment strategies. Furthermore, this strategy is very expensive and, therefore, requires a detailed cost/benefit analysis. Transplantations are limited to a restricted number of patients and performed only in very specialized centres.
Usually, the cases are well documented. This allows easy acquisition of an adequate knowledge-base.

In the Hospital of the University of Munich about 200 liver-transplanted patients with more than 700,000 data records have been documented. Since in many aspects the underlying concepts of relevant complications are not well understood and the amount of data is enormous, the authors of this system [Swoboda et al. 1994] decided to apply the method of case-based reasoning to design an expert system on this domain.

The outcome of liver-transplantation is highly influenced by the medical history of the transplant donor and recipient. Furthermore, surgical techniques and postoperative management of complications are of great importance. In this context, complications are defined as the prognosis of diseases which are closely correlated to liver transplantation. Many of these complications influence each other. To avoid an acute or chronic rejection an immune-suppressive therapy is necessary. On the other hand, due to this suppression of the immune system, the occurrence of infectious diseases is markedly increased. These two types of complications are the most important ones during the immediate period after transplantation.

To evaluate the appropriate diagnostic and therapeutic strategy for these complications, it is important not only to register these diagnoses but also to determine all necessary examinations and their corresponding findings, as well as the applied drug therapy and their time course.

Knowledge-based expert system – COSYL (COnsiliar SYstem for Liver-transplanted patients) – uses a database of cases of liver transplanted patients. First of all, COSYL imports the data from this database, grades them according to postoperative complications and saves a pattern of data for every complication. When the system is consulted, it compares the pattern of the current patient with the saved patterns and displays possible complications with the highest similarity.

COSYL imports the data of the patients and matches them with a tree of complications. The leaves of this tree contain a set of complications found in the corresponding treatment group.

4.1.1.17. ICONS: Prognosis of Kidney Function Courses

At intensive care unit, a renal function monitoring system, NIMON, was developed to print daily a renal report consisting of 13 measured and 33 calculated parameters of those patients to whom renal function monitoring is applied. However, the interpretation of all reported parameters is quite complex and needs special knowledge of the renal physiology.

The aim of the ICONS, [Schmidt, Gierl 2002] system is to give an automatic interpretation of the renal state of the kidney function on time. In particular, knowledge about the behaviour of the various parameters over time is still incomplete.

The method consists of three main steps; two abstractions plus CBR retrieval.

The procedure for interpretation of the kidney function corresponds to a linear model. Firstly, the monitoring system NIMON gets 13 measured parameters from the clinical chemistry and calculates 33 meaningful kidney function parameters. As the interpretation of all parameters is
too complex, the authors decided to abstract them. For this data abstraction, they have defined states of the renal function, which determine states of increasing severity, beginning with a normal renal function and ending with a renal failure. Based on these definitions, the state of the kidney function is determined per day.

Based on the sequence of assessments of transitions of the state of a day to the state of the next day, four different trends are generated. These trends, which are temporal abstractions, describe courses of states. The CBR retrieval methods are used to search for similar courses. The current course, in comparison to similar ones, is presented to the user. The course continuations of the similar courses serve as prognoses. As there may be too many different aspects between both patients, the adaptation of the similar to the current course is not done automatically. ICONS offers only prognostic support; the user has to decide about the relevance of all displayed information.

4.1.1.18. TeCoMED

TeCoMED (Tele-Consultation to Monitor Emerging Diseases) [Schmidt, Gierl 2003], [Schmidt, Gierl 2002] is an early warning system based on network to discover health risks, to forecast temporal and spatial spread of epidemics, and to estimate the consequences of an epidemic.

The public health observation is a complex problem of multiparametric time courses of diseases, pathogens, resistances etc, in a geographical region. Furthermore, there are time-spatial patterns of the spread of diseases which are too complicated to be described by means of statistics. Nevertheless, there are successful applications that use case based techniques for monitoring complex processes in time and space in other domains. According to that, it is found out [Schmidt, Gierl 2003] that there exist similar epidemic waves of influenza according to the time-spatial dynamic.

In this project, the considered geographic region is divided into a finite number of disjoint geographical units. Also, a time scale is divided in equidistant time steps (e.g. one week). As a scenario, it is used a concept which describes the public health situation and the activities of the health service in the considered region during one period.

Forecasting of a regional health situation means the prediction of a future scenario. The system retrieves all scenario sequences from the case base so that the differences between them and the current scenario sequence are minimal.

By using the retrieved sequences, the system adapts a current scenario to forecast a future scenario. Here, some background knowledge must be taken in consideration. The resulting data will be stored in a case-based fashion and will be available for graphical presentation in the user interface. After a certain time period, it becomes known what has really happened in a scenario. In comparing the forecast scenario with this scenario the system evaluates the forecast mechanism. If the difference of these two scenarios is too large then the system has to learn this new scenario sequence or the adaptation mechanism must be changed. In the CBR framework of Aamodt and Plaza (1994) this is the revision step.
4.1.1.19. DIRAS

DIRAS [Armengol et al. 2001] is an application whose goal is to predict the risk of complications for diabetic patients. Diabetes is one of the most frequent human clinic diseases since it affects approximately 3% of the European population and around one hundred million people in the world. There are two major types of diabetes: type 1 (or insulin-dependent) and type 2 (or non insulin-dependent).

A bad management of both forms of diabetes will produce microcomplications (such as blindness, renal failure or polyneuropathy), and macrocomplications (such as gangrene and amputation, coronary heart disease or stroke). Therefore, the main concern in managing the diabetes is to reduce both the risk of the patient to develop a new long-term complication and the risk of progression of the complications having already been present.

The prediction of the individual risk to develop long-term complications is based on the analysis of a large quantity of data (e.g. diabetes type, diabetes duration, age, cholesterol, and metabolic control degree) that have to be continuously evaluated.

The goal of DIRAS is to determine the risk of each complication for individual diabetic patients. For each patient, DIRAS works with five kinds of data: Personal-Data, Basic-Diabetes-Data, Info-Patient-Consultation, Assessment and Risk-Pattern, where the Risk-Pattern represents some kind of a solution.

DIRAS uses a case base where each case is a patient described as a risk pattern, i.e. the patient's data plus the solution for that patient. The goal of DIRAS is to obtain the full risk pattern for the current patient. The risk pattern of each diabetic patient is obtained by using a case based reasoning method – LID (Lazy Induction of Descriptions).
5. *CaBaGe* in Multiple Sclerosis Diagnosis

As already mentioned, medicine is a suitable domain for application of CBR because the knowledge of experts consists of mixture of textbook (objective) knowledge and experience (subjective) knowledge, which mostly consists of cases. Usually, these two sorts of knowledge can be clearly separated and represented in an appropriate manner.

As it is well known a medical diagnosis, from surface aetiology, is difficult since it involves lots of complications. A physician has to investigate carefully patient’s symptoms and test results, major complaints and pathology examination in order to make a differential diagnosis. It is a rather difficult process and it takes years of training and practice for a physician to make correct decisions.

Implemented *CaBaGe* system is a general tool which can be applied in different domains in order to improve various activities and actions. But our initial intention was to apply it in a neurology domain, especially in diagnoses of Multiple Sclerosis (MS) disease. In cooperation with our colleagues from the Institute of Neurology, Medical School, Novi Sad, the main features being important for Multiple Sclerosis diagnosis are extracted and the data about several patients are prepared as a training set for the system [Kurbalija 2003], [Ivanović 2002], [Kurbalija, Ivanović 2003].

**Multiple Sclerosis**

Multiple sclerosis (MS) is a chronic disease of unknown cause that affects the nervous system (brain, spinal cord and peripheral nerves) in form of multifocal lesions of a myelin nerve sheet. It causes a damage of vision, muscle strength, sensation, coordination, speech, and bladder control and may affect cognitive functions. MS usually appears in young adulthood (more in females) and Vojvodina is the region with a high incidence of disease.

In this section, the characteristics of MS, some symptoms and signs and the process of diagnosis, will be briefly explained. More information can be found in [Eccles Health Web], [About MS Web], [National MS Web].

**What is Multiple Sclerosis?**

MS is thought to be an autoimmune disease that primarily affects the brain and spinal cord (central nervous system). It is usually diagnosed between the ages of 20 and 50. In MS, the covering of the nerves (myelin sheath) is destroyed (demyelination). The underlying nerve fiber may also be damaged or severed.

Like insulation on electrical wires, healthy myelin insures rapid transmission of nerve impulses. When myelin or nerve fibers are damaged, the messages from the brain, for
example, to move a body part, and messages to the brain, for example, to interpret sensations, are not transmitted effectively. Body movement may be slow or uncoordinated and body sensations may be altered.

While healing and return to normal function (remission) may occur. Scars (called plaques) may occur and permanently interfere with motor and sensory control. Damage of the myelin sheath can occur at any time and affect any part of the brain or spinal cord. The disease is called multiple sclerosis because there are multiple areas of scarring (sclerosis) (Figure 5.1.).

![MRI Scan of a patient with MS](image)

**Figure 5.1.** MRI Scan of a patient with MS - Multiple areas of demyelination are present in cerebral hemispheres

Each person with MS has a unique set of symptoms depending on where in the central nervous system the demyelination and fiber damage occurs. Common symptoms include weakness, numbness, incoordination, loss of balance, visual problems, loss of bladder or bowel control, mood swings, cognitive problems, difficult speaking, and fatigue. MS is not fatal or contagious. However, a small number of people have a severe type of MS that may shorten life expectancy.

There are four main varieties (types) of MS as defined in an international survey of neurologists [Lubin and Reingold, 1996]. In Figure 5.2. the graphs presenting level of disability over time are shown. Where two lines appear in the graph they denote two possible courses of that form of MS.
Figure 5.2. Types of MS: a) Relapsing/Remitting; b) Secondary Progressive; c) Progressive Relapsing; d) Primary Progressive.

Relapsing/Remitting (RRMS): This is characterized by relapses (also known as exacerbations) during which time new symptoms can appear and old ones resurface or worsen. The relapses are followed by periods of remission, during which the person fully or partially recovers from the deficits acquired during the relapse. Relapses can last for days, weeks or months and recovery can be slow and gradual or almost instantaneous. The vast majority of people presenting with Multiple Sclerosis are first diagnosed with relapsing/remitting. This is typically when they are in their twenties or thirties, though diagnoses much earlier or later are known. Approximately twice as many women as men are affected with this variety.

Secondary Progressive (SPMS): After a number of years many people who have had relapsing/remitting MS will pass into a secondary progressive phase of the disease. This is characterized by a gradual worsening of the disease between relapses. In the early phases of Secondary Progressive, the person may still experience a few relapses but after a while these merge into a general progression. People with secondary progressive may experience good and bad days or weeks, but, apart from some remission following relapsing episodes, no real recovery. After 10 years, 50% of people with relapsing/remitting MS will have developed secondary progressive.
Progressive Relapsing Multiple Sclerosis (PRMS): This form of MS follows a progressive course from onset, punctuated by relapses. There is significant recovery immediately following a relapse but between relapses there is a gradual worsening of symptoms.

Primary Progressive (PPMS): This type of MS is characterized by a gradual progression of the disease from its onset with no remissions at all. There may be periods of a levelling off of the disease activity and, as with secondary progressive, there may be good and bad days or weeks. PPMS differs from Relapsing/Remitting and Secondary Progressive in that onset is typically in the late thirties or early forties, men are as likely women to develop it and initial disease activity is in the spinal cord and not in the brain. Primary Progressive MS often migrates into the brain, but is less likely to damage brain areas than relapsing/remitting or secondary progressive. People with Primary Progressive are less likely to develop cognitive problems.

Symptoms and Signs

Common symptoms of MS include fatigue, weakness, spasticity, balance problems, bladder and bowel problems, numbness, vision loss, tremors and depression. Not all symptoms affect all MS patients. No two persons have the same complaints. No one develops all of the symptoms.

Symptoms may be persistent or may cease from time to time. Most patients have episodic patterns of attacks and remissions throughout the disease course. Symptoms may remit completely, leaving no residual damage, or partially leaving degrees of permanent impairment.

Because the symptoms that define the clinical picture of MS are the result of nerve lesions causing disturbances in electrical conduction in one or more areas of the CNS, the nature of the symptoms that occur is determined by the location of the lesion. For example: an optic nerve lesion may cause blurred vision; a brain stem lesion may cause dizziness or double vision; a spinal cord lesion may cause coordination/balance problems.

Some of the most common symptoms are listed below. These are not the only symptoms to affect those with MS. These symptoms may be sporadic or persistent. Not all of these symptoms affect all patients.

Fatigue: The most common complaint of MS patients is fatigue. It occurs in as many as 78% of patients, usually in the late afternoon and often subside in the early evening.

Numbness, Tingling, Burning Sensations: Sensory complaints occur in up to 55% of patients and are often the earliest symptoms of MS. Disturbances of feeling in the extremities or the trunk such as tingling, crawling sensations, feelings of swelling or numbness. Numbness also depends upon its cause. If severe neurological damage to the myelin sheath takes place, then numbness may remain.

Tremors: Shaking or trembling of a limb or occasionally the head. Up to 50% report extremity ataxia (shaky movements or unsteady gait) or tremors. Tremors may come and go. This symptom of MS impairs mobility and often is associated with difficulty in balance and coordination.
**Balance/Coordination:** Gait and balance disturbances are common with MS. Balance problems without vertigo may be more constant, causing the person to sway or stagger.

**Depression:** As in most cases with the onset of an illness, depression is a frequent reaction. MS-related lethargy and fatigue may also be mistaken for depression or heighten its effects.

**Spasticity:** Occurs with the initial attack of MS in up to 41% of patients and is present in about 62% of patients with progressive disease. Occurs when opposing groups of muscles contract and relax at the same time. When spasticity is present, the increased stiffness in the muscles means that a great deal of energy is required to perform daily activities.

**Bladder:** Increased frequency of urination, urgency, dribbling, hesitancy, and incontinence.

**Bowel:** Constipation, diarrhea and incontinence. Dysfunction occurs in almost two thirds of patients during the disease course.

**Vision Loss:** Rarely involves both eyes simultaneously, usually starts with blurred vision followed by vision loss.

**Cognitive and Emotional Dysfunction:** Affects approximately 50% of MS patients. Involves memory, reasoning, verbal fluency and speed of information processing. Emotional changes include euphoria, depression. Memory problems are fairly common among people with MS. Memory and reasoning problems may affect between two thirds and three fourths of those diagnosed with MS to varying degrees.

**Sexual Difficulties:** More than 90% of men and 70% of women with MS report some change in their sexual life after the onset of the disease. Some problems include decreased sexual drive, impaired sensation, diminished orgasmic response, and loss of sexual interest.

**Diagnosis of MS**

Diagnosing multiple sclerosis is anything but easy. There is no specific test for multiple sclerosis and, anyway, it is not even certain that it is only one disease. To an extent, getting an MS diagnosis is a process of eliminating all other possibilities. Typically, people who have finally been diagnosed with definite MS will have been through several diagnostic stages. This process is often drawn out over months or years.

It is not possible to diagnose definite MS from a single episode. In order to diagnose MS, there must be at least two episodes separated by at least one month and the location of the lesions must be in a least two distinct sites in the central nervous system. This means that the diagnosis will, by definition, have to wait at least the period of time that separate the first two relapses that cause clinical symptoms. This could be as little as one month but is more likely to be several months or even years.

Neurologists use two kinds of criteria to confirm a diagnosis of multiple sclerosis: **Schumacher criteria** and **Poser criteria**.

Although Schumacher criteria [Schumacher et al 1965] are now largely outdated, an MS diagnosis remains a clinical one and they still form the basis for later revisions. They are also
worth looking at because they are the simplest statement of what MS is in the clinical sense. The Schumacher criteria are:

- Neurological examination reveals objective abnormalities of CNS function.
- History indicates involvement of two or more parts of CNS.
- CNS disease predominately reflects white matter involvement.
- Involvement of CNS follows one of two patterns:
  - Two or more episodes, each lasting at least 24 hours and at least one month apart.
  - Slow or stepwise progression of signs and symptoms over at least 6 months.
- Patient aged 10 to 50 years old at onset.
- Signs and symptoms cannot be better explained by other disease process.

The Poser criteria [Poser 1983] have updated the Schumacher criteria in recognition of the diagnostic benefits of laboratory data. They have not changed the fact that MS is still essentially a clinical diagnosis and are themselves about to be replaced by new criteria that acknowledge the importance of Magnetic Resonance Imaging (MRI). The Poser criteria are:

- Clinically definite MS
  - 2 attacks and clinical evidence of 2 separate lesions
  - 2 attacks, clinical evidence of one and paraclinical evidence of another separate lesion

- Laboratory supported Definite MS
  - 2 attacks, either clinical or paraclinical evidence of 1 lesion, and cerebrospinal fluid (CSF) immunological abnormalities
  - 1 attack, clinical evidence of 2 separate lesions & CSF abnormalities

- Clinically probable MS
  - 2 attacks and clinical evidence of 1 lesion
  - 1 attack and clinical evidence of 2 separate lesions
  - 1 attack, clinical evidence of 1 lesion, and paraclinical evidence of another separate lesion

- Laboratory supported probable MS
  - 2 attacks and CSF abnormalities

**Selection of Features**

The selection of features for diagnosis of MS was a significant problem, because the correct diagnosis depends on many different attributes (medical checkups), and because this illness can have many different indications.
By analyzing the diagnoses problem of MS disease, the colleagues from the Institute of Neurology prepared the appropriate case structure with characteristic features. Every case is characterized by 78 different features representing the most important observations in the diagnoses process of MS disease.

The data about the patient are divided in two groups:

- Data gained from conservation with the patient (Table 5.1.)
- Data gained from medical checkup (Table 5.2.)

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Name</td>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Family name</td>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Residence</td>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Nationality</td>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Education</td>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Age</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Year of First Diagnosis</td>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Sex</td>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Family history</td>
<td>BOOLEAN</td>
<td>If exists, increases certainty in diagnosis</td>
</tr>
<tr>
<td>10.</td>
<td>Weakness in 1 or more extremities</td>
<td>BOOLEAN</td>
<td>Indicates damage of one path in neural system</td>
</tr>
<tr>
<td>11.</td>
<td>Cramps in extremities</td>
<td>BOOLEAN</td>
<td>Indicates damage of one path in neural system</td>
</tr>
<tr>
<td>12.</td>
<td>Numbs in extremities</td>
<td>BOOLEAN</td>
<td>Indicates damage of one path in neural system</td>
</tr>
<tr>
<td>13.</td>
<td>Paroxysmal pain</td>
<td>BOOLEAN</td>
<td>Indicates damage of more functional systems</td>
</tr>
<tr>
<td>14.</td>
<td>Chronic pain</td>
<td>BOOLEAN</td>
<td>Indicates damage of more functional systems</td>
</tr>
<tr>
<td>15.</td>
<td>Facial pain</td>
<td>BOOLEAN</td>
<td>Damage of one nerve</td>
</tr>
<tr>
<td>16.</td>
<td>Dizziness</td>
<td>BOOLEAN</td>
<td>Damage of more systems</td>
</tr>
<tr>
<td>17.</td>
<td>Gait difficulties</td>
<td>BOOLEAN</td>
<td>Damage of one or more systems</td>
</tr>
<tr>
<td>18.</td>
<td>Speech difficulties</td>
<td>BOOLEAN</td>
<td>Damage of one or more systems</td>
</tr>
<tr>
<td>19.</td>
<td>Hearing difficulties</td>
<td>BOOLEAN</td>
<td>Damage of one or more systems</td>
</tr>
<tr>
<td>20.</td>
<td>Visual loss for one or both eyes</td>
<td>BOOLEAN</td>
<td>Damage of visual path</td>
</tr>
<tr>
<td>21.</td>
<td>Ocular pain</td>
<td>BOOLEAN</td>
<td>Inflammation of eye nerve</td>
</tr>
<tr>
<td>22.</td>
<td>Double vision</td>
<td>BOOLEAN</td>
<td>Damage of nerves responsible for eye movement</td>
</tr>
<tr>
<td>23.</td>
<td>Increased frequency of voiding</td>
<td>BOOLEAN</td>
<td>Damage of nerves responsible for voiding</td>
</tr>
<tr>
<td>24.</td>
<td>Hesitancy of voiding</td>
<td>BOOLEAN</td>
<td>Damage of nerves responsible for voiding</td>
</tr>
<tr>
<td>25.</td>
<td>Urge incontinence</td>
<td>BOOLEAN</td>
<td>Damage of nerves responsible for voiding</td>
</tr>
<tr>
<td>26.</td>
<td>Bowel disfunction</td>
<td>BOOLEAN</td>
<td>Damage of nerves responsible for bowel</td>
</tr>
<tr>
<td>27.</td>
<td>Impotence</td>
<td>BOOLEAN</td>
<td>Damage of nerves responsible for impotence</td>
</tr>
<tr>
<td>28.</td>
<td>Fatigue</td>
<td>BOOLEAN</td>
<td>General sign</td>
</tr>
<tr>
<td>29.</td>
<td>Depression</td>
<td>BOOLEAN</td>
<td>General sign</td>
</tr>
<tr>
<td>30.</td>
<td>Difficulty in sustaining attention</td>
<td>BOOLEAN</td>
<td>Difficulties in sustaining attention</td>
</tr>
<tr>
<td>31.</td>
<td>Memory disturbances</td>
<td>BOOLEAN</td>
<td>Indicates cognitive deficit</td>
</tr>
<tr>
<td>32.</td>
<td>Number of relapses</td>
<td>BOOLEAN</td>
<td>True if the number of relapses is bigger than 2</td>
</tr>
</tbody>
</table>

Table 5.1. Data gained from conservation with the patient.

In Table 5.1. the features gained from conservation with the patient are given. There are 32 attributes important for the MS diagnosis. In the table, the attribute names, types and a short description of the attributes are given. First 5 attributes are for general purpose only, and they are not important for the diagnosis.
In Table 5.2, the features gained from medical checkups are given. There are 46 attributes important for the MS diagnosis. Additionally, the diagnosis of each patient is given as a 79th attribute. In the table, the attribute names, types and a short description of the attributes are given.

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>Ambliopy</td>
<td>BOOLEAN</td>
<td>Having poor vision, damage of eye nerves, objectively verified by medical checkup</td>
</tr>
<tr>
<td>34</td>
<td>Loss of visual acuity</td>
<td>BOOLEAN</td>
<td>Lost of sight</td>
</tr>
<tr>
<td>35</td>
<td>Diplopia</td>
<td>BOOLEAN</td>
<td>Double vision</td>
</tr>
<tr>
<td>36</td>
<td>Nystagmus</td>
<td>BOOLEAN</td>
<td>Spasm of eyeballs, damage of eye nerves or some others functional systems</td>
</tr>
<tr>
<td>37</td>
<td>Central facial palsy</td>
<td>BOOLEAN</td>
<td>Paralysis of facial nerves</td>
</tr>
<tr>
<td>38</td>
<td>Peripheral facial palsy</td>
<td>BOOLEAN</td>
<td>Peripheral paralysis of facial nerves</td>
</tr>
<tr>
<td>39</td>
<td>Trigeminal neuralgia</td>
<td>BOOLEAN</td>
<td>Pain in the area of 5th nerve</td>
</tr>
<tr>
<td>40</td>
<td>Pareisis n. XII</td>
<td>BOOLEAN</td>
<td>Paralysis of 12th nerve</td>
</tr>
<tr>
<td>41</td>
<td>Lhermit sign</td>
<td>BOOLEAN</td>
<td>Strong pain (like electricity shock) during sudden neck curvetting, rather typical for this illness</td>
</tr>
<tr>
<td>42</td>
<td>Spasticity</td>
<td>BOOLEAN</td>
<td>High tonus of muscles</td>
</tr>
<tr>
<td>43</td>
<td>Monoparesis</td>
<td>BOOLEAN</td>
<td>Weakness of one extremity</td>
</tr>
<tr>
<td>44</td>
<td>Hemiparesis</td>
<td>BOOLEAN</td>
<td>Weakness of one half of the body</td>
</tr>
<tr>
<td>45</td>
<td>Triparesis</td>
<td>BOOLEAN</td>
<td>Weakness of three extremities</td>
</tr>
<tr>
<td>46</td>
<td>Paraparesis</td>
<td>BOOLEAN</td>
<td>Weakness of four extremities</td>
</tr>
<tr>
<td>47</td>
<td>Quadriparaparesis</td>
<td>BOOLEAN</td>
<td>Weakness of four extremities</td>
</tr>
<tr>
<td>48</td>
<td>Tendon Reflexes Increased</td>
<td>BOOLEAN</td>
<td>Increased tendon reflexes</td>
</tr>
<tr>
<td>49</td>
<td>Subclonus</td>
<td>BOOLEAN</td>
<td>Problems with posture</td>
</tr>
<tr>
<td>50</td>
<td>Clonus</td>
<td>BOOLEAN</td>
<td>Problems with posture</td>
</tr>
<tr>
<td>51</td>
<td>Babinski</td>
<td>BOOLEAN</td>
<td>Pathological reflex</td>
</tr>
<tr>
<td>52</td>
<td>Plantar response absent</td>
<td>BOOLEAN</td>
<td>Absence of reflexes as pathological sign</td>
</tr>
<tr>
<td>53</td>
<td>Abdominal reflexes absent</td>
<td>BOOLEAN</td>
<td>Absence of reflexes as pathological sign</td>
</tr>
<tr>
<td>54</td>
<td>Dysmetria</td>
<td>BOOLEAN</td>
<td>Difficulties when trying to touch the nose with the finger</td>
</tr>
<tr>
<td>55</td>
<td>Intention tremor</td>
<td>BOOLEAN</td>
<td>Shaking when performing fine movements</td>
</tr>
<tr>
<td>56</td>
<td>Ataxia</td>
<td>BOOLEAN</td>
<td>Unstability, as a sign of cerebellum damage</td>
</tr>
<tr>
<td>57</td>
<td>Romberg test</td>
<td>BOOLEAN</td>
<td>Sign of cerebellum damage</td>
</tr>
<tr>
<td>58</td>
<td>Hemihypoesthesia</td>
<td>BOOLEAN</td>
<td>Sign of sensibility path damage</td>
</tr>
<tr>
<td>59</td>
<td>Hyperaesthesia</td>
<td>BOOLEAN</td>
<td>Sign of sensitivity path damage</td>
</tr>
<tr>
<td>60</td>
<td>Dysarthria</td>
<td>BOOLEAN</td>
<td>Sign of damage of one or more systems</td>
</tr>
<tr>
<td>61</td>
<td>Nasalitiy of speech</td>
<td>BOOLEAN</td>
<td>Disturbance of speech, rather typical for multiple sclerosis</td>
</tr>
<tr>
<td>62</td>
<td>Scanning speech</td>
<td>BOOLEAN</td>
<td>Disturbance of speech, rather typical for multiple sclerosis</td>
</tr>
<tr>
<td>63</td>
<td>Euphoria</td>
<td>BOOLEAN</td>
<td>Euphoria</td>
</tr>
<tr>
<td>64</td>
<td>MRI evidence of demyelination</td>
<td>BOOLEAN</td>
<td>Visible damages on magnetic resonance checkup</td>
</tr>
<tr>
<td>65</td>
<td>9 T2 hyperintense lesions</td>
<td>BOOLEAN</td>
<td>Some characteristics of magnetic resonance checkup</td>
</tr>
<tr>
<td>66</td>
<td>1 or more infratentorial</td>
<td>BOOLEAN</td>
<td>Localization of lesions, less typical for multiple sclerosis</td>
</tr>
<tr>
<td>67</td>
<td>1 or more juxtacortical lesions</td>
<td>BOOLEAN</td>
<td>Localization of lesions</td>
</tr>
<tr>
<td>68</td>
<td>3 or more periventricular</td>
<td>BOOLEAN</td>
<td>Localization of lesions, rather typical for multiple sclerosis</td>
</tr>
<tr>
<td>69</td>
<td>VEP latences</td>
<td>INTEGER</td>
<td>More than 108 is pathological sign</td>
</tr>
<tr>
<td>70</td>
<td>CSF cell count</td>
<td>REAL</td>
<td>Normal value: up to 2x10^7</td>
</tr>
<tr>
<td>71</td>
<td>Total CSF protein</td>
<td>REAL</td>
<td>Normal value: 0.20-0.35</td>
</tr>
<tr>
<td>72</td>
<td>IgG</td>
<td>REAL</td>
<td>Normal value: 0.002-0.028</td>
</tr>
<tr>
<td>73</td>
<td>IgG/albumine</td>
<td>REAL</td>
<td>Normal value: 0.060-0.220</td>
</tr>
<tr>
<td>74</td>
<td>Link index</td>
<td>REAL</td>
<td>Normal value: 0.280-0.880</td>
</tr>
</tbody>
</table>
De novo synthesis IgG in CNS

Normal value: -9.9 to +3.3 mg/24h

MC index of synthesis Ig

Normal value: less than 0.001

Oligoclonal bands

Normal is that they are absent, the presence of this attribute is one of the signs for definitive MS

Blood barrier dysfunction

Normal value: greater than 170

Diagnosis

Four possible values:
- NoMS,
- PossibleMS,
- ProbableMS and
- DefinitiveMS

Table 5.2. Data gained from medical checkup.

Most of these attributes are the results of neurological checkups (tests). They are mostly of the boolean type with the meaning: the patient is positive or negative on corresponding test. The last 10 attributes (except one) are of numeric type. In the table, in a description column, the normal values are given.

Some of the most common neurological tests are:

- **Romberg's sign**: This is a test for ataxia (incoordination or clumsiness of movement that is not the result of muscular weakness) and involves patients standing with his/her feet together with his/her eyes closed. Ataxics have great problems standing still under these conditions.

- **Gait and coordination**: The neurologist evaluates ataxia in various parts of the body by observing the patient walking normally, walking heel-to-toe and finger-to-nose tests. The neurologist will also be looking for intention tremor (shaking when performing small motor movements) as well as ataxia in this last test.

- **L'Hermittes sign**: This is a test for lesions on the spinal cord in the neck. The neurologist will ask the patient to lower his/her head towards his/her chest. A positive L'Hermittes will generate buzzing, tingling or electrical shock sensations in one or more parts of the body.

- **Optic Neuritis**: This is a condition of the eye caused by inflammation and demyelination of the Optic Nerve and is perhaps the most commonly presenting symptom in MS. The tests involve the ubiquitous reading of letters from a board and a test for colour vision using an "Ishihara" colour chart.

- **Hearing Loss**: This is done by lightly clicking the fingers next to each ear and asking the patient which ear the click was done next to.

- **Muscle Strength**: This involves resisting the neurologist with various muscle groups. Differences in strength between left and right sides are easier to evaluate than symmetrical loss unless the weakness is severe.

- **Reflexes**: This is done with both ends of the hammer. The reflexes can be normal, brisk, i.e. too easily evoked, or non-existent.

- **Babinski’s sign**: A test for signs of a disease process in the motor neurons of the pyramidal tract. The test involves drawing a semi-sharp object along the bottom of the foot. The normal response in adults and children is for the toes to reflex downwards (flexor response). In babies and people with neurological problems of the corticospinal tract, the big toe moves upwards (extensor response).
The Application of CaBaGe in MS Diagnosis

When the structure of database is modelled (the most important attributes are selected), the next important activity was the extraction of the data from the patients histories. Most of the illustrative patient files are examined and necessary data were extracted. In the rest of this chapter the functionality of the system will be shown.

The structure of a case in the system is described in case pattern file “msdiagnosis.key” – Listing 5.1. This structure corresponds to the case structure presented in Tables 5.1. and 5.2. The Boolean type (from Tables 5.1. and 5.2.) is simulated with a string type with only two values TRUE and FALSE. The feature Diagnosis is not given in the case pattern file because the system assumes that the diagnosis will be given in the case base file after the last attribute.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name, FamilyName, Residence, Nationality, Education, Age, YearOfFirstDiagnosis, Sex, FamilyHistory, WeaknessInOneOrMoreExtremities, CrampsInExtremities, NumbersInExtremities, ParoxysmalPain, ChronicPain, FacialPain, Dizziness, GaitDifficulties, SpeechDifficulties, HearingDifficulties, VisualLossForOneOrBothEyes, OcularPain, DoubleVision, IncreasedFrequencyOfVoiding, HesitancyOfVoiding, UrgeIncontinence, BowelDisfunction, Impotence, Fatigue, Depression, DifficultyInSustainingAttention, MemoryDisturbances, NumberOfRelapses, Amblophy, LossOfVisualAcuity, Diplopia, Nystagmus, CentralFacialPalsy, PeripheralFacialPalsy, TrigeminalNeuralgia</td>
<td>string</td>
</tr>
<tr>
<td>ParesisNoXII, LhermitSign, Spasticity, Monoparesis, Hemiparesis, Triparesis, Paraparesis, Quadriparesis, TendonReflexesIncreased, Subclonus, Clonus, Babinski, PlantarResponseAbsent, AbdominalReflexesAbsent, Dysmetria, IntentionTremor, Ataxia, RombergTest, Hemihypoesthesia, Hyperaesthesia, Dysarthria, NasalityOfSpeech, ScanningSpeech, Euphoria, MRIEvidenceOfDemyelination, NineT2HyperstringenseLesions, OneOrMoreInfratentorial, OneOrMoreJuxtacorticalLesions, ThreeOrMorePeriventricular, VEPLatencies, CSFCellCount, TotalCSFProtein, IgG, Albumine, LinkIndex, DeNovoSynthesisIgGInCNS, MCIndexOfSynthesisIg, OligoclonalBands, BloodBarrierDysfunction</td>
<td>string</td>
</tr>
</tbody>
</table>

Listing 5.1 Case pattern file “msdiagnosis.key”
The test cases are given in the case base file "msdiagnosis.cbr". Since the process of data extraction is very time consuming (for every patient, it is necessary to examine several records and histories), so far it is fully extracted the data about 22 patients with different diagnoses.

The values of the features must be given in a correct order according to the case pattern file. Last value in the description of the case is always the solution of the case (Diagnosis). These values are determined dynamically during the parsing of the case base file and can be: Definitive Multiple Sclerosis – DefinitiveMS, Probable Multiple Sclerosis – ProbableMS, Possible Multiple Sclerosis – PossibleMS or No Multiple Sclerosis – NoMS.

Above, the diagnoses are listed according to their certainty to the illness. The patient with the DefinitiveMS and ProbableMS diagnosis has a greater probability of having multiple sclerosis than the patient with PossibleMS and NoMS diagnosis. On the other hand, the diagnosis can be changed over time.

The cases are not equally distributed according to their diagnosis. This is a consequence of an extraction of a real world clinical data. There are more cases with a DefinitiveMS and ProbableMS diagnosis and less with PossibleMS and NoMS diagnosis because on clinical examination comes patients with already several suspicious symptoms. The numbers of cases according to their diagnosis are:

- 13 cases are of DefinitiveMS diagnosis
- 5 cases are of ProbableMS diagnosis
- 2 cases are of PossibleMS diagnosis
- 2 cases are of NoMS diagnosis

Data Analysis

According to the supplied data, all of the attributes do not influence the process of diagnosis in the same way. Some of the attributes more exactly indicates one or two diagnoses, while the others are not so exact.

<table>
<thead>
<tr>
<th></th>
<th>DefinitiveMS</th>
<th>ProbableMS</th>
<th>PossibleMS</th>
<th>NoMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech difficulties</td>
<td>30.77%</td>
<td>40.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Urge incontinence</td>
<td>23.08%</td>
<td>40.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Fatigue</td>
<td>38.46%</td>
<td>20.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Number of relapses</td>
<td>100.00%</td>
<td>100.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Ambliopy</td>
<td>30.77%</td>
<td>20.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Spasticity</td>
<td>46.15%</td>
<td>40.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Triparesis</td>
<td>46.15%</td>
<td>20.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Clonus</td>
<td>23.08%</td>
<td>20.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Dysmetria</td>
<td>69.23%</td>
<td>60.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Intention tremor</td>
<td>53.85%</td>
<td>80.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Ataxia</td>
<td>69.23%</td>
<td>60.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>3 or more periventricular</td>
<td>61.54%</td>
<td>60.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Table 5.3. Attributes that more exactly indicates the diagnosis.
Some of the attributes which more exactly indicate the diagnosis process are given in Table 5.3. Most of these attributes are mentioned as common symptoms and signs for multiple sclerosis in sections 5.1. and 5.2. In the table, the percent of patients according to their diagnosis from the database with positive results on corresponding tests are shown.

Since these attributes more exactly indicates the diagnosis, the percentage of the patients being positive on the tests generally decreases from DefinitiveMS diagnosis to NoMS. This means that more patients with DefinitiveMS diagnosis are positive on these tests then the patients with some other diagnosis. This is the main reason why these attributes more exactly indicates the diagnosis. The patients with PossibleMS and NoMS diagnosis are not positive at all on these tests.

The graphical representation of these data is shown in Figure 5.3. Most of these diagrams are decreasing from DefinitiveMS diagnosis to NoMS.

![Graphical representation of attributes that more exactly indicates the diagnosis.](image)

**Figure 5.3.** Graphical representation of attributes that more exactly indicates the diagnosis.

Naturally, there exists attributes that are not so exact. Mostly they are not typical symptoms of multiple sclerosis. Some of them are shown in Table 5.4.

These attributes not so strongly influences the process of diagnosis. The percentage of patients positive on some test can be arbitrary for any diagnosis.
Table 5.4. Attributes that not so exactly indicates the diagnosis.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>DefinitiveMS</th>
<th>ProbableMS</th>
<th>PossibleMS</th>
<th>NoMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weakness in 1 or more extr.</td>
<td>76.92%</td>
<td>100.00%</td>
<td>50.00%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Dizziness</td>
<td>69.23%</td>
<td>100.00%</td>
<td>50.00%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Hearing difficulties</td>
<td>7.69%</td>
<td>0.00%</td>
<td>50.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Depression</td>
<td>38.46%</td>
<td>20.00%</td>
<td>0.00%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Difficulty in sustaining attention</td>
<td>23.08%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Diplopia</td>
<td>30.77%</td>
<td>80.00%</td>
<td>0.00%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Tendon Reflexes Increased</td>
<td>76.92%</td>
<td>100.00%</td>
<td>50.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Babinski</td>
<td>76.92%</td>
<td>100.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Hemihypoaesthesia</td>
<td>23.08%</td>
<td>40.00%</td>
<td>0.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>1 or more infratentorial</td>
<td>76.92%</td>
<td>100.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>1 or more juxtacortical</td>
<td>0.00%</td>
<td>60.00%</td>
<td>50.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Since these attributes are so random it is hard to say how they influence the diagnosis. It is possible that the positive result on some of these tests more indicates the NoMS diagnosis than DefinitiveMS or ProbableMS. That is the consequence of an activation propagation process in a case retrieval net. Some of such attributes, for example, are: Depression, Difficulty in sustaining attention and Hemihypoaesthesia attributes.
Nevertheless, this situation with attributes was derived from currently available data (22 records). When some new data will be available, maybe some of the attributes that more exactly indicates the diagnosis will be moved to the other group, or vice versa.

The CaBaGe with Real World Data

In Figures 5.5 and 5.6, the main window of the system is shown. At the beginning of execution the system expects that the user (physician) enters the paths of the case pattern and case base file. If the paths are good, these two files are loaded and the case retrieval net is created. Also, a dynamically created form will appear in the middle of the window.

After loading files and creating the case retrieval net, the system expects from the user to enter the data concerning the current problem (the data about the current patient) and to enter the known values of the attributes. If some values are not known then the user should enter the zero value in the corresponding importance field (like for the attributes Clonus and Abdominal reflexes absent on the figure 5.5).

When the process of entering the data into the form is finished, the process of spreading activation is started by clicking on the “Solve” button. In the right part of the window the solution is shown after completing the process. All possible solutions and their relative activations are shown there. The solution with the highest number is the “suggested solution”.

In Figure 5.5 the situation with suggested diagnosis NoMs is shown. In this situation, all attributes that more exactly indicates the diagnosis are set to false. That is the case, because the positive result on tests that more exactly indicates the diagnosis more "reminds" on the diagnoses DefinitiveMS and ProbableMS. The attributes from the other group are randomly set: some are set to true, some to false. The settings of these attributes are irrelevant because all two possible values (TRUE or FALSE) "reminds" on all four diagnoses.

![Figure 5.5. CaBaGe with MS data – diagnosis NoMS](image)
The situation in Figure 5.6 is opposite. There, the suggested diagnosis is *DefiniteMS*, and the next (sorted by relevance) is *ProbableMS*. As expected, the settings of the attributes are also opposite to the previous example. In this situation, all attributes that more exactly indicates the diagnosis are set to true. The attributes from the other group are set arbitrary, but in this example mostly to false.

![Figure 5.6. CaBaGe with MS data – diagnosis DefinitiveMS](image)

Since the *CaBaGe* is a decision support system, it only suggests the solution. It is not intended to replace a physician, but more likely to help him in further leading of the diagnosis process. The system, for each new patient, suggests some solution, but the quality of the solution depends on the quality and quantity of the input data.
6. Conclusion

During the past 15 years Case Based Reasoning proved to be a promising and very applicable artificial intelligence technique. That confirms a huge number of realized commercial and scientific applications in different areas. Probably, the most important reason for such success of CBR technique lies in its similarity with a human cognitive process. People take into consideration successfully solved past problems when trying to solve some new ones.

Case-based reasoning can be used for solving problems in many practical domains such as: mechanical engineering, physics, biology, medicine, business administration etc. Each domain has its own characteristics. The domain strongly influences the choice of data structures for the knowledge representation.

Furthermore, various task types can be implemented for each domain. Some of them are: classification, diagnosis, configuration, design, planning, decision support, information retrieval etc. The task type determines the type of the problems and solutions, but also the various activities that the system for problem solving has to be undertaken. There is no one to one correspondence between domains and task types, but for each such pair specific expertise is required.

In design of each CBR system, the choice of a memory organization is very important. The case base can consists of a huge number of cases with potentially large number of attributes, so it is important for the memory structure to be efficient with the appropriate similarity measure. In this application Case Retrieval Nets (CRN) [Lenz, Burkhard 1996], [Lenz et al. 1998] have been chosen.

Case Retrieval Net is a special memory structure with a number of good characteristics. They have been developed especially for being employed in large case bases. Also, they are applicable in domain independent generic systems, such as a system having already been proposed here. CRNs have following properties:

- They are able to deal with vague and ambiguous terms
- They support the concept of information completion
- They can handle case bases of reasonable size efficiently.

All generic applications based on CBR can be divided in two subgroups:

- CBR tools and
- CBR shells.
A CBR tool is a software that can be used to develop several applications that require case-based reasoning. The tool can be domain-independent or dedicated to an application domain or a type of problems (such as help-desk applications).

CBR shells are a kind of application generators with a sophisticated graphical user interface, where some parameters can be specified by the user to develop a new application. For example, it is possible to specify the fields of cases, the domain knowledge, and the weight vectors for the retrieval. CBR shells are a kind of tools that can usually be used by a non-programmer user. The extension or the integration of new components in these tools is usually not possible.

The realized CaBaGe system is a classical CBR shell. This tool is realized for easy use of non-programmers. The user doesn't have to know anything about the realization of the system. All he has to do is to prepare two input files, in which the previous situations are stored, so as to obtain a working decision support system in any kind of domain.

Based on available references it is clear that there exist many CBR tools but just a few CBR shells [Schultz 1999] [CBR at AIAI]. CBR shells are equally important as CBR tools since they are implemented directly for the users – no additional programming effort is needed. The main advantages of the system proposed here are:

- **Fast retrieval algorithm.** The Case Retrieval Nets are used as a basic memory structure since they have a very high retrieval performance. The solution is gained after just two iterations no matter how large is the case base.
- **Domain independence.** The system calculates similarity measures just on the basis of previous cases from any kind of domain. There is no need for some special domain knowledge (complex relations between attributes of the cases), nor for the expertise of the domain experts.
- **Incremental learning.** The system can acquire new knowledge simply by saving the successfully solved case in the base of previously solved cases (a textual "case base file").
- **Platform independence.** The system is implemented in Java so it can work equally well on all kinds of platforms. Furthermore, the GUI of the system adapts to the target platform so that the user can work with the GUI he used to.

As already mentioned, many different domains are possible for each task type. That is exactly the case with CaBaGe. CaBaGe is a framework suitable for creating different decision support systems from different domains.

In this thesis, the application of the CaBaGe in the medical domain is shown. More precisely, the decision support system that will help medical experts in determination of the Multiple sclerosis disease was build. The realization of this system came out as a result of cooperation with colleagues from the Institute of Neurology, Novi Sad. Also, the valuable data about previous patients, which is essential for the functionality of the system, was carefully prepared by colleagues from the same institution.

The multiple sclerosis disease is chosen mainly because its diagnosis is highly complicated. The diagnosis of multiple sclerosis depends on many different attributes which are mainly the results of medical tests and examinations. The physician has to keep in mind more than 80
attributes just for one patient. Furthermore, each patient has a different set of symptoms, and the knowledge of connections between symptoms is still rather poor.

The system was not intended to replace the physician completely. More likely, it is designed to help him to find a final decision. The physician can consult the system in any moment of the diagnostic process to see a "second opinion". He/She can then decide to make a final diagnosis or to perform some additional tests to be more sure.

Some of possibilities for further work can be the modification of the CaBaGe system to offer the help in the therapies planning. The colleagues from the Institute of neurology are already preparing the data about the therapies which gave good results to different types of patients. These data are also valuable and can be easily applicable in CBR system.

The CaBaGe system can be surely applied in other domains. The possible extension of the CaBaGe system could be in defining and implementing some kind of meta language for defining more sophisticated similarity measures. This meta language could be applied mostly in defining similarity measure for attributes of a string type. Such similarity measures would be the result of different data mining techniques.

Also some future work is focused in a direction of creating CBR decision support system, which can be applied in domains where some decisions depend on behaviour of curves, diagrams or time courses [Kim 2004].
Sažetak


Uopšteno, zaključivanje na osnovu slučajeva je tehnik a za rešavanje problema, gde se novi problemi rešavaju adaptacijom rešenja koja su odgovarala sličnim problemima u prošlosti. Slučaj se, generalno, može predstaviti kao uređeni par (problem, rešenje), gde prvi elemenat predstavlja opis problema, a drugi uspešno rešenje tog problema iz prošlosti. Osnovni scenario za gotovo CBR aplikacije izgleda ovako:

Da bi pronašao rešenje aktuelnog problema, sistem traži sličan problem u svojoj "iskustvenoj" bazi, uzima ispravno rešenje najsličnijeg problema iz prošlosti i koristi ga kao osnovu za pronalaženje rešenja aktuelnog problema.


Mreža za pronalaženje slučajeva je vrsta memorijske strukture sa velikim brojem dobrih karakteristika. Ovakve mreže su posebno razvijene za potrebe velikih baza slučajeva. Takođe, one su primenljive u generičkim sistemima koji su nezavisni od domena, kakav je i sistem prikazan u ovoj tezi. Mreže za pronalaženje slučajeva imaju sledeće važne karakteristike:

- U mogućnosti su da manipulišu sa neodređenim i dvosmislenim vrednostima
- Podržavaju koncept kompletiranja informacija
- Efikasno manipulišu sa velikim bazama slučajeva

91
Sistem, koji je glavni rezultat ove teze, *CaGaBe – Case Base Generator*, je dizajniran kao sistem za podršku odlučivanju u potpunosti nezavisan od domena. U ovom sistemu, slučaj je opisan kao niz atributa, a za svaki slučaj su date vrednosti nekih ili svih atributa. Ovo znači da je opis problema dat vrednostima atributa u zavisnosti od toga kako je domen modeliran. Rešenje slučaja je još samo jedan dodatni atribut. Slučajevi se mogu uzeti iz bilo kog domena (medicina, administracija, inžinjerstvo…), ali moraju biti korektno modelovani tako da bi se mogli predstaviti kao niz atributa.


Realizovan *CaBaGe* sistem je klasična CBR shell aplikacija. Sistem sadrži jednostavno grafičko okruženje i kreiran je za jednostavno korišćenje ne-programera. Korisnik ne mora ništ da zna oko realizacije sistema. Sve što on treba da uradi je da pripremi dva ulazna fajla, u kojima se nalazi opis slučaja i prethodni rešeni slučajevi, da bi dobio funkcionalan sistem za podršku odlučivanju u domenu koji on odabere.

Glavne prednosti predloženog sistema su sledeće:

- **Brz algoritam za pretraživanje.** Mreže za pronalaženje slučajeva su korišćene kao osnovna memorijska struktura pošto pokazuju dobre performanse. Rešenje se dobija kroz samo dve iteracije bez obzira kolika je baza slučajeva.
- **Nezavisnost od domena.** Sistem računa meru sličnosti samo na bazi predhodnih slučajeva nezavisno od domena. Nije potrebno neko specijalno znanje o domenu (kompleksne relacije između atributa), kao ni posebna ekspertiza stručnjaka iz datog domena.
- **Inkrementalno učenje.** Sistem stiče novo znanje jednostavnim čuvanjem uspešno rešenog problema u bazi predhodnih slučajeva.
- **Nezavisnost od platforme.** Sistem je implementiran u Java-i tako da podjednako kvalitetno može da funkcioniše na različitim platformama. Takođe, grafičko okruženje se adaptira ciljnoj platformi tako da korisnik radi u okruženju na koje je navikao.

U ovoj tezi, prikazana je primena sistema *CaBaGe* u medicinskom domenu. Preciznije, kreiran je sistem za podršku odlučivanju, koji bi pomogao lekarima u dijagnozi multiple skleroze. Ovakva primena *CaBaGe* sistema je proizašla iz saradnje sa kolegama sa Instituta za neurologiju u Novom Sadu. Tamo su takođe pripremljeni vredni podaci o prethodnim pacijentima, koji su esencijalni za funkcionisanje sistema.
Multipla skleroza je odabrana prvenstveno zato što je njena dijagnoza izuzetno komplikovana. Dijagnoza multiple skleroze zavisi od mnogobrojnih atributa, koji su uglavnom rezultati medicinskih testova i pregleda. Lekar mora da vodi računa o preko 80 atributa za samo jednog bolesnika. Takođe, svaki bolesnik ima jedinstveni skup simptoma, a znanje o zavisnostima između simptoma je još uvek dosta slabo.

Index of Figures

Figure 2.1. CBR-cycle of Aamodt and Plaza (1994)
Figure 2.2. Part of Case Retrieval Net
Figure 3.1. Structure of CRN
Figure 3.2. Main window of CaBaGe
Figure 3.3. Distribution of cases concerning attribute NumberofLegs.
Figure 3.4. A part of Case Retrieval Net
Figure 3.5. CaBaGe after loading files
Figure 3.6. CaBaGe after solving the problem.
Figure 4.1. Main steps for treatment of patient
Figure 5.1. MRI Scan of a patient with MS - Multiple areas of demyelination are present in cerebral hemispheres
Figure 5.2. Types of MS: a) Relapsing/Remitting; b) Secondary Progressive; c) Progressive Relapsing; d) Primary Progressive.
Figure 5.3. Graphical representation of attributes that more exactly indicates the diagnosis.
Figure 5.4. Graphical representation of attributes that not so exactly indicates the diagnosis.
Figure 5.5. CaBaGe with MS data – diagnosis NoMS
Figure 5.6. CaBaGe with MS data – diagnosis DefinitiveMS
Sadržaj slika

Figure 2.1. *CBR-krug* po Aamodt i Plaza (1994)

Figure 2.2. Deo mreže za pronalaženje slučajeva

Figure 3.1. Struktura mreže za pronalaženje slučajeva

Figure 3.2. Glavni prozor *CaBaGe*

Figure 3.3. Distribucija slučajeva po atributu *NumberOfLegs*.

Figure 3.4. Deo mreže za pronalaženje slučajeva

Figure 3.5. *CaBaGe* posle učitavanja fajlova

Figure 3.6. *CaBaGe* posle rešavanja problema.

Figure 4.1. Glavni koraci u tretmanu pacijenta

Figure 5.1. MRI snimak pacijenta sa MS – Višestruke demijelinizirajuće oblasti su prisutne u cerebralnim hemisferama

Figure 5.2. Tipovi multipleks skleroze: a) Relapsing/Remitting; b) Secondary Progressive; c) Progressive Relapsing; d) Primary Progressive.

Figure 5.3. Grafička reprezentacija atributa koji egzaktnije utiču na dijagnozu.

Figure 5.4. Grafička reprezentacija atributa koji manje egzaktno utiču na dijagnozu.

Figure 5.5. *CaBaGe* sa MS podacima – dijagnoza *NoMS*

Figure 5.6. *CaBaGe* sa MS podacima – dijagnoza *DefinitiveMS*
Index of Tables and Listings

Listing 3.2 Class Element
Listing 3.3 Class IEEElement
Listing 3.4 Class stringIEElement
Listing 3.5 Class intIEElement
Listing 3.6 Class floatIEElement
Listing 3.7 Class CaseElement
Listing 3.8 Class Arc
Listing 3.9 Class activationElement
Listing 3.10 Class Attribute
Listing 3.11 Class Lexer
Listing 3.12 Structure of “Case pattern file”
Listing 3.13 Structure of “Case base file”
Listing 3.14 Class CaseRetrievalNet
Listing 3.15 Class Frame1
Listing 3.16 File “default.key”
Listing 3.17 A part of file “default.cbr”
Table 3.1. Number of cases concerning attribute NumberofLegs.
Table 3.2. Number of cases concerning attribute LaysEggs.
Table 5.1. Data gained from conservation with the patient.
Table 5.2. Data gained from medical checkup.
Listing 5.1 Case pattern file “msdiagnosis.key”
Table 5.3. Attributes that more exactly indicates the diagnosis.
Table 5.4. Attributes that not so exactly indicates the diagnosis.
Sadržaj tabela i listinga

Listing 3.2 Klasa Element
Listing 3.3 Klasa IEEElement
Listing 3.4 Klasa stringIEElement
Listing 3.5 Klasa intIEElement
Listing 3.6 Klasa floatIEElement
Listing 3.7 Klasa CaseElement
Listing 3.8 Klasa Arc
Listing 3.9 Klasa activationElement
Listing 3.10 Klasa Attribute
Listing 3.11 Klasa Lexer
Listing 3.12 Struktura “Case pattern file”
Listing 3.13 Struktura “Case base file”
Listing 3.14 Klasa CaseRetrievalNet
Listing 3.15 Klasa Frame1
Listing 3.16 Fajl “default.key”
Listing 3.17 Deo fajla “default.cbr”

Table 3.1. Raspodela slučajeva po atributu NumberofLegs.
Table 3.2. Raspodela slučajeva po atributu LaysEggs.
Table 5.1. Podaci dobijeni iz razgovora sa pacijentom.
Table 5.2. Podaci dobijeni iz medicinskog pregleda.
Listing 5.1 Case pattern file “msdiagnosis.key”
Table 5.3. Atributi koji više utiču na dijagnozu.
Table 5.4. Atributi koji manje utiču na dijagnozu.
References


[CBR at AIAI] Case-Based Reasoning at AIAI, http://www.aiai.ed.ac.uk/project/cbr/


[University of Kaiserslautern home page] University of Kaiserslautern, Case-Based Reasoning homepage: http://www.cbr-web.org/


Vladimir Kurbalija was born on 12th May 1977, in Novi Sad. In the year 1992, he enrolled in the gymnasium "Jovan Jovanović Zmaj". After gymnasium he enrolled at the Faculty of Science, University of Novi Sad, Department of Mathematics and Informatics. He graduated in the year 2000, with the average grade 9.88 (max. 10). On October 23, 2000 he defended his graduation thesis *Realization of lexical and syntax analyzer for programming language Tiger* with the mark 10.

He is the winner of several faculty and university prizes for extraordinary success in the first, second and fourth year of studies, and also for his success during the whole studies. Also, he is the winner of the university prize named *Aleksandar Saša Popović* for the year 2000. (for the best graduation thesis in the area of Computer Science for the current year), and for the year 2002. (for exceptional scientific paper in the area of Computer Science).

He has been employed as the assistant at the Department of Mathematics and Informatics since 2000. He conducts exercises for the courses in *Internet Tools, Compiler Construction and Data Structures and Algorithms* for the students of computer science. During the same year he enrolled in post-graduate studies at the Department of Mathematics and Informatics of the Novi Sad University. All examinations, having been fixed and anticipated by the Plan and Programme of the studies he passed by 10.00.

He was a secretary and a member of organizing committees of several seminars and conferences. He also participated in these projects:

- **Development of (intelligent) techniques based on software agents for application in information retrieval and workflow, 2002 – 2004.** Supported by Ministry of Science and Technologies (Republic of Serbia), project no. 1844.

- **Abstract Methods and Applications in Computer Science**, Project no. 144017A Ministry of Science and Technologies (Republic of Serbia), 2006-2010.

During the time from September 2002 until January 2003, he stayed at the Humboldt University, Berlin, Research Group Artificial Intelligence, as a participant of the project "Utilization of Case-Based Reasoning Technology for implementation of Decision Support Systems". Furthermore, in June 2005, he stayed for one month in Linz, Austria at Johannes Kepler University, Institute for System Software as a participant of the project "Innovation of Compiler Construction Course".

He is the author or a co-author of 10 scientific papers, of which 7 are published internationally. His fields of interests are: Compiler Construction, Artificial Intelligence and Case-Based Reasoning.
Biografija


Dobitnik je univerzitetskih i fakultetskih nagrada za izuzetan uspeh postignut u prvoj, drugoj i četvrtoj godini studija, kao i za uspeh postignut u toku celokupnih studija. Takođe je dobitnik univerzitetske nagrade Aleksandar Saša Popović za 2000. godinu (za najbolji diplomski rad iz oblasti računarских nauka u tekućoj godini) i za 2002 godinu (za izuzetan naučni rad u oblasti računarских nauka).

Na Departmanu za matematiku i informatiku Prirodno-matematičkog fakulteta u Novom Sadu radi od 2000. godine kao asistent-pripravnik, gde drži vežbe iz predmeta Internet alati, Konstrukcija kompajlera i Strukture podataka i algoritmi. Iste godine upisao je magistarske studije na Prirodno-matematičkom fakultetu u Novom Sadu. Sve ispite predviđene planom i programom položio je sa prosečnom ocenom 10,00.

Bio je sekretar i član organizacionih odbora nekoliko konferencija u zemlji. Učešće na projektima:

- Razvoj (inteligentnih) tehnika zasnovanih na softverskim agentima za primenu u pretraživanju podataka i tokovima poslova, Ministarstvo za nauku, tehnologije i razvoj Republike Srbije – projekat broj 1844, 2002-2004.
- Apstraktni metodi i primene u računarskim naukama, Projekt 144017A Ministarstva nauke i zaštite životne sredine Republike Srbije (2006-2010)


Autor ili koautor je 10 naučnih radova, od kojih su 7 objavljenih u inostranstvu. Bavi se konstrukcijom kompajlera, veštakćom inteligencijom i zaključivanjem na osnovu slučajeva (eng. Case-Based Reasoning).
IMPLEMENTACIJA SISTEMA ZA PODRŠKU ODLUČIVANJU KORIŠČENJEM "CASE BASED REASONING" TEHNOLOGIJE


Datum odbrane:

Članovi komisije:

Predsednik: dr Mirjana Ivanović, redovni profesor, Prirodno-matematički fakultet, Novi Sad
Mentor: dr Zoran Budimac, redovni profesor, Prirodno-matematički fakultet, Novi Sad
Član: dr Dušan Tošić, redovni profesor, Matematički fakultet, Beograd
Član: dr Hans Dieter Burkhard, redovni profesor, Humboldt University, Berlin
Title: IMPLEMENTATION OF DECISION SUPPORT SYSTEMS USING CASE BASED REASONING TECHNOLOGY

Language of text: English

Language of abstract: Serbian/English

Country of publication: Serbia

Locality of publication: Vojvodina, Novi Sad

Publication year: 2006

Publisher: Author’s reprint

Pub. place: Novi Sad, Trg Dositeja Obradovića 4
Abstract: The main topic of this master thesis is the utilization of particular technique of artificial intelligence, called “Case Based Reasoning” (CBR) for realization of a basic system, which can be used for implementation of different real world decision support systems. At the beginning, the foundations of CBR technology is given. After that, a basic system is completely described. At the end, a realized real world application is shown.

Accepted by the Scientific Board on: 17.11.2005.

Defended:

Thesis defend board:

President: dr Mirjana Ivanović, full professor, Faculty of Science, Novi Sad
Mentor: dr Zoran Budimac, full professor, Faculty of Science, Novi Sad
Member: dr Dušan Tošić, full professor, Faculty of Mathematics, Belgrade
Member: dr Hans Dieter Burkhard, full professor, Humboldt University, Berlin