CASE-BASED REASONING FRAMEWORK FOR GENERATING WIDE-RANGE DECISION SUPPORT SYSTEMS

V. Kurbalija, M. Ivanović, Z. Budimac
Department of Mathematics and Informatics, Faculty of Science,
Trg D. Obradovića 4, 21 000 Novi Sad, Serbia
e-mail:{kurba, mira, zjb}@im.ns.ac.yu

Abstract

Case-Based Reasoning (CBR) is a relatively new and promising technique of artificial intelligence. By CBR, every new problem is solved by adapting the solutions of the similar problems previously solved successfully. During the past few years CBR has become a very popular technique for application in knowledge-based systems for different domains because the experience has been included in solving every new problem. The intention of our research is to develop a robust and general framework which supports generation of wide-range decision support systems by using different CBR approaches. Presented framework integrates two previously developed CBR shells: CaBaGe and CuBaGe.

1. Introduction

Case-Based Reasoning (CBR) has become a successful technique applicable in different knowledge-based systems and different domains. This promising technique is based on use of previous experience in a form of cases to understand better and to solve new problems in particular domain. The main idea arises from the fact that in many particular domains similar problems usually have similar solutions.

Generally speaking, CBR systems provide solutions to given problems but also take care of other tasks which have occurred when this technique is used in practice. It means that CBR system can do several tasks depending on the intended use of reasoning:

- **Explain** current situation according to previously experienced similar situations,
- **Critique** current situation based on old cases,
- **Reason** from precedents to understand a current situation,
- **Combine** several old solutions in order to solve a current problem,
- **Learn** from the successfully solved new problems.

To summarize, mentioned aspects could be divided into two types: interpretative and problem solving CBR. In the interpretative CBR systems the essence is to achieve whether or not a current situation should be treated like previous ones based on similarities or differences among them. In problem solving CBR systems the main task is to propose a solution to a current situation based on the adaptation of solutions of past cases. However in practice, many problems have components of both types of CBR and combination of both methods is certainly employed.

In the last several years in the area of CBR technology at our Department, two independent shells dealing with different concepts and approaches of CBR are developed: *CaBaGe* [9] and *CuBaGe* [10]. The main difference between them is the form of case representation. In the *CaBaGe* shell the case is represented as a set of values of chosen attributes, while in the *CuBaGe* shell the case is represented as a curve or as time series - set of appropriate points in some space.

The paper presents a unique and robust framework which supports usage of these two shells in order to analyze, compare and implement different concepts of the CBR technique. This proposed framework will enclose a variety of case representation techniques and also different similarity measures defined for corresponding case representation techniques.

The rest of the paper is organized as follows. The section 2 elaborates necessary concepts of CBR technique [3, 4, 8, 11, 12]. The shells *CaBaGe* and *CuBaGe* are briefly described in the third and fourth sections, respectively. In the fifth section some characteristics and possibilities for application of the proposed framework are discussed, while the sixth section concludes the paper.

2. Foundations of CBR

Case-Based Reasoning is a relatively new and promising area of artificial intelligence and it is also considered as a problem solving technique. It is used for solving problems in domains where experience plays an important role [4, 11, 12].

Generally speaking, CBR is applied for solving new problems by adapting solutions that worked for similar problems in the past. The main supposition here is that similar problems have similar solutions. The basic scenario for mainly all CBR applications is: 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 for the actual problem.

In CBR systems experience is stored in a form of cases. The case is a recorded situation where the problem has been totally or partially solved, and it can be represented as an ordered pair *(problem, solution)*. The whole experience is stored in *case base*, and each case represents some previous episode where the problem was successfully solved.

The main problem in CBR is to find a good similarity measure – the measure that can tell in what extent the two problems are similar. In the functional way similarity can be defined as a function $sim : U \times CB \rightarrow [0 , 1]$ where U refers to the universe of all objects (from a given domain), 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 [11].

The CBR system has not the only goal of providing solutions to problems but also of taking care of other tasks occurring when it is used in practice. The main phases of the CBR activities are presented in the *CBR-cycle* (see Figure 1) [1].

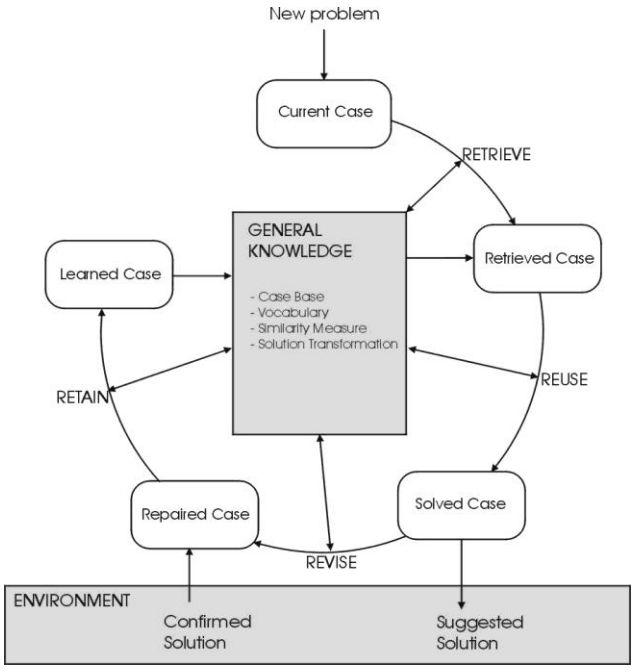

Figure 1. The *CBR-Cycle* after Aamodt and Plaza (1994)

In the *retrieve* phase the most similar case (or *k* most similar cases) to the problem case, is retrieved, while in the *reuse* phase some modifications of the retrieved case has been 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.

The main advantage of this technology is that it can be applied to almost any domain. 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 the similar problems 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. Furthermore, in more examined domains integration of CBR in classical rule-based reasoning systems brings some efficiency. The second very important advantage is that CBR approach to learning and problem solving is very similar to human cognitive processes – people take into account and use past experience to make future decisions.

3. *CaBaGe* – Case Base Generator

CaBaGe (Case Base Generator) is a general shell for generating an arbitrary decision support system based on CBR technique. In such shell previous cases as well as new problems (cases) are represented as a set of values of some selected, most important attributes. The shell was completely implemented in Java mainly because it supports all concepts of object-oriented technology, but also because it's main characteristic – platform independence. Currently it was realized as an application, but the small modifications are necessary in order to make an applet or servlet.

The main advantage of this shell is that it is domain independent and can be used to instantiate appropriate decision-support system. For a particular domain of application, for which decision-support system (DSS) is instantiated, the user can select the most important and characteristic cases from the available set of previously solved cases. These cases are to be stored in an appropriate form in the base of cases. According to that, the input for the DSS is a two folded one: the description of the previously solved cases which are stored in the appropriate case-base (file) and the data for a new case from the domain, which has to be solved. Using these data, DSS creates Case Retrieval Net (CRN) [11, 12] and it becomes capable to solve a new problem.

In the first input file - "Case Pattern File", the description of case is stored. Case pattern file contains the list of the attributes which completely represent essential features of the case, 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 of them must be enlisted in the desired order.

The second file - "Case Base File", contains the list of, in advance selected, representative 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. Type of the solution can also be: *int*, *float* or *string* and it is determined dynamically, when all cases are parsed. At this moment it is assumed that the case base file is a textual file where every case is listed in one line, and the values of the attributes and solution are separated with commas.

When the case base (i.e. appropriate text file) is created, DSS generates a case retrieval net. The basic structure of CRN is given in Figure 2. Two main parts of the CRN are an array of attributes and a list of solutions. The array of attributes is created by using case pattern file, while the list of solutions is created from a case base file. Each value for each attribute represents one information entity i.e. an ordered pair *(attribute, value)*. Each value (or information entity) contains a list of arcs to the solution nodes. The arc is pondered by its weight and has 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* and the solution node *c* are simply calculated as a number of cases that contain the information entity *e* and whose solution is *c*.

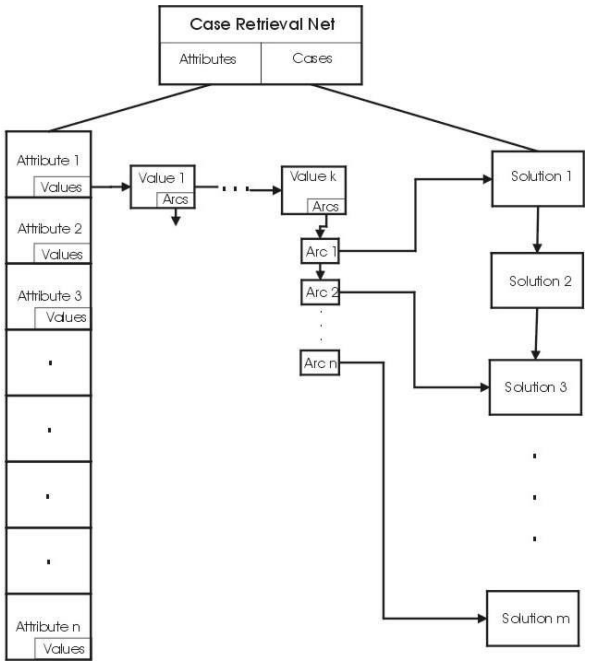

Figure 2. Structure of CRN

When the CRN is created the DSS can be used for solving new cases (queries) from the domain. Since the new case (query) and the case from the case base have the same structure, the user has to enter the values of attributes in an appropriate input form. In order to describe the problem better, the user should enter all known values of the attributes although it is not necessary. The input form contains one more field for every attribute – **importance**. The importance is a 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 denotes that he is 100% sure that the data are valid, while the value 0 denotes that he doesn't know the value of that attribute at all. The queries where attributes have some additional information (like importance) are called weighted queries.

After entering the query, the DSS starts searching the case base in order to find a 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 with the weight of the corresponding arc. Final activation of the solution nodes is calculated by summing all gained activations.*

3.1 Application of the *CaBaGe*

CuBaGe shell is supposed to be used for generating wide range of decision support systems for different domains. In this section, one successful application in medical domain, in diagnosis of Multiple sclerosis (MS) [6, 7, 9], will be presented. A set of attributes being important for diagnosis of MS (structure of the case) as well as data about previous patients (case base) are provided by medical experts from the Institute of Neurology, Novi Sad.

The structure of a case is described in the case pattern file "MSDiagnoses.key". The case is completely described by 72 different features/attributes which represent the most important observations and medical checkups in the diagnostic process of MS disease. All test cases are given in the case base file "MSDiagnoses.cbr". Last value in the description of the case is always the solution of the case (disease Diagnoses). These values are determined dynamically during the parsing of the case base file and can be: *Definitive MS, Probable MS, Possible MS or No MS.*

In Figure 3, the main window of the DSS for MS is shown. At the beginning the DSS expects the user to enter the names and select the appropriate case pattern and the case base files. If the paths are good, these two files are loaded and the case retrieval net is created. Also, the

dynamically created input form will appear in the middle of the window.

After loading files and creating CRN, the DSS expects the user to specify current case – query 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. When the entering the data is finished, the process of the spreading activation is started. In the right part of the window the solution is shown after completing the process. All possible solutions with corresponding activations are shown therein. The solution with the highest activation number is the "suggested solution". For this example it is most possible that for this "test" patient the diagnoses will be "No Multiple sclerosis" with the activation number 0.71465296.

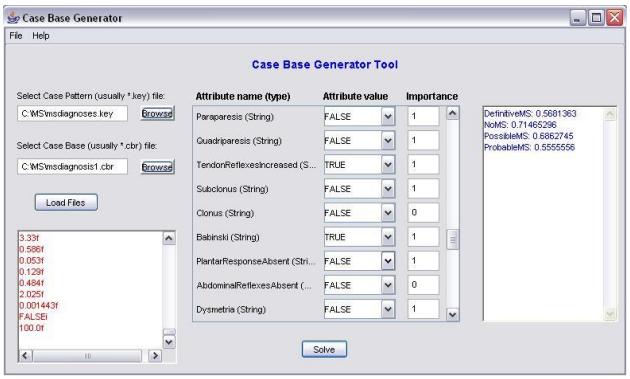

Figure 3. The main window of the "CaBaGe"

For every new case the system always suggests some solution, but the quality of the solution depends on the quality of the input data, which describes the case.

4. *CuBaGe* – Curve Base Generator

CuBaGe (Curve Base Generator) is another CBR shell in which, both the problem and the previous cases, are presented in a graphical manner [10]. The reasons, for implementing such shell, are that in many practical domains some decisions depend on behaviour of time series, charts and curves. The shell therefore analyses curves, compares them to similar curves from the past and possibly predicts the future behaviour of the current curve on the basis of the most similar curves from the past.

The main problem here, as almost in every CBR system, was to create a good similarity measure for curves, i.e. what is the function that can tell to what extent two curves are similar. Beforehand, it is necessary to choose an appropriate representation of the curve which is compatible with the desired similarity measure.

As already mentioned, in many practical domains data are represented with the set of points - an ordered pair *(x,y)*. Very often the pairs are *(t,v)* where *t* represents time and *v* represents some value in the time *t*. When the data is given in this way it can be represented graphically and when the points are connected they represent some kind of a curve. If the points are connected just with straight lines then they represent the linear interpolation, but if someone wants smoother curves then some other kind of interpolation with polynomials must be used. There was a choice between classical interpolating polynomial and cubic spline. The cubic spline was chosen as the curve representation technique for two main reasons:

- **Power**: for the n+1 points classical interpolating polynomial has the power n, while cubic spline always has the power 3.
- **Oscillation**: if only one point is moved (which can be the result of a bad experiment or measuring) classical interpolating polynomial significantly changes (oscillates), while cubic spline only changes locally (which is intuitively more appropriate for real world domains).

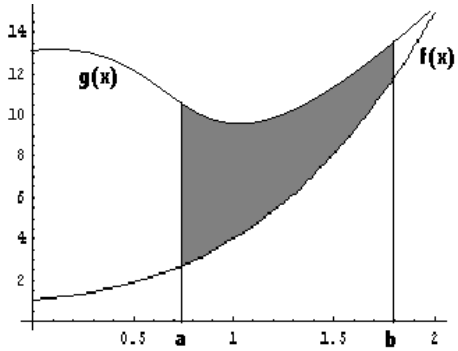

Figure 4. Surface between two curves

When the cubic spline is calculated for curves then one very intuitive and simple similarity (or distance – which is the dual notion for similarity) measure can be used. The distance between two curves can be represented as a surface between these curves as seen in Figure 4. This surface can be easily calculated using the definitive integral. Furthermore, the calculation of the definitive integral for polynomials is a very simple and efficient operation.

4.1 Application of the CuBaGe System

As *CuBaGe* is also implemented as a general shell, it could be used for a wide range of domains in which all cases can be represented by curves. In this section, a particular application and the utilisation of the *CuBaGe* shell will be shown. The shell was successfully applied in a financial domain for prediction future payment on the basis of previous payment and invoicing processes. However, this application of the *CuBaGe* shell is more complex and is beyond the scope of this paper. More

information can be found in papers [13, 14, 15]. For this reason, only some basic functionality of the shell will be described below.

In *CuBaGe* a cubic spline was chosen as a case representation technique, while the similarity measure is integral based. The polynomials are represented as arrays of its coefficients. Every curve is represented as a set of its points and a set of polynomials between points by a definition of the cubic spline. The distance between two curves is calculated by using a definitive integral of the function $h^2(x)$ [10]. However, the distance must be partially computed between every two points of both curves because the polynomials of the splines are changed in every point (Figure 5). To compute the global distance between two curves, all these partial distances have to be simply summed. *CuBeGe* shell can be used in a similar manner like *CaBeGe* shell, for generation of different DSS in a wide range of application domains.

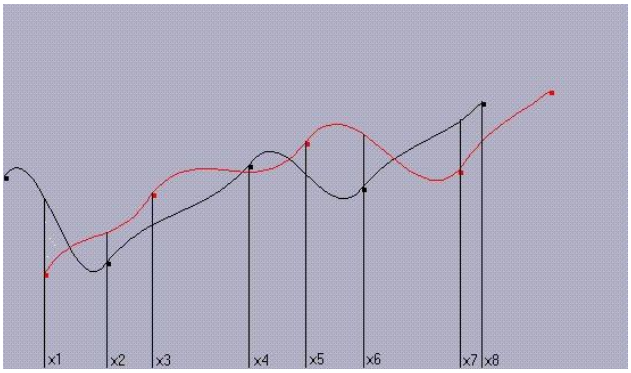

Figure 5. Computing distance between two curves.

For this particular application and generation of the appropriate DSS, a randomly generated database of the curves has been created. Database has consisted of 1200 curves. Each curve has been given with a set of points. The number of points is also random (more than 4 and less then 15). At the beginning, the generated DSS reads those sets of points (from selected input file), creates splines for each curve and internally stores the curves in a case base (curve base). Then, in the same way, DSS reads a curve (from the other input file), and that curve represents some current problem (in fact a new case) which has to be solved. After that, DSS tries to find the most similar curves to the problem curve. Retrieved curves, the most similar to the new one, are sorted by the distance and suggested as possible solutions.

Temporally, a graphical user interface (GUI) has been created so that the results of the DSS can be interactively seen. In Figure 6 some snapshots from the DSS can be seen. For the same problem curve, represented in red (lighter curve), the most similar curve from the curve base is shown in the upper picture (distance=26,19499). In the lower picture the 25th curve (sorted by distance) is shown (distance=129,6637). As can be seen the DSS really retrieves the most (intuitively) similar curve from the curve base.

Many of the task types, which are popular in the time series analysis, explicitly or implicitly use similarity measures. Some of them are [5]: Indexing, Clustering, Classification, Prediction (Forecasting), Summarization, Anomaly Detection, Segmentation, etc.

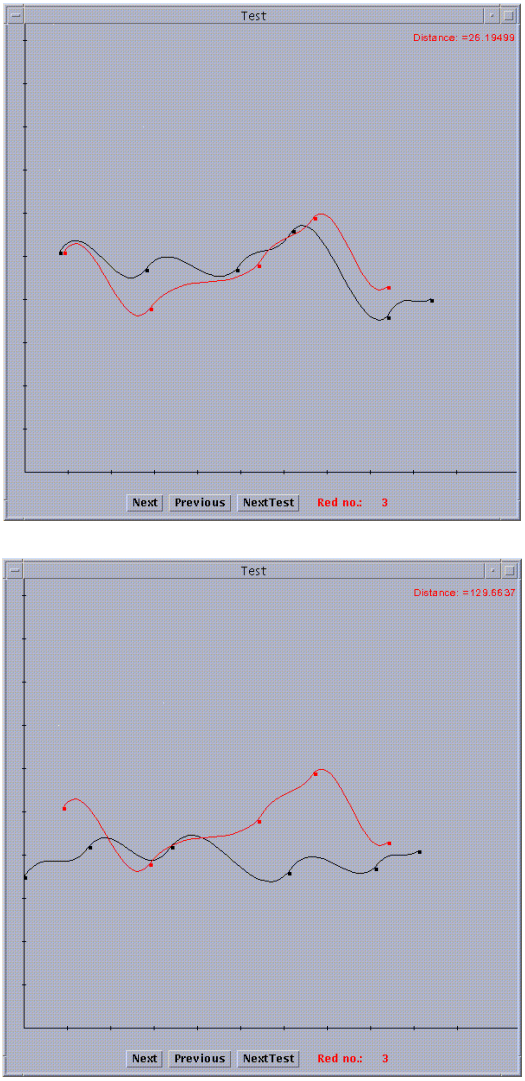

Figure 6. Most similar and 25th similar curve

CBR technology and *CuBaGe* shell can be used in most of these task types. One algorithm for prediction is already implemented in *CuBaGe* [13, 14].

5. Characteristics of proposed framework

All generic applications based on CBR can be divided in two subgroups: CBR tools, and CBR shells [2]. A CBR

tool is 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, the weight vectors for the retrieval etc. CBR shells are a kind of tools that can usually be used by a non-programmer user. Any extension or integration of new components in these tools is usually not possible.

The realized *CaBaGe* and *CuBaGe* are classical CBR shells. They are realized for easy use of non-programmers. The user doesn't have to know anything about the realization of the concrete shells. All he/she has to do is to prepare the input files in appropriate manner, in which the values for previous situations (cases) are stored. By using a prepared file with the information of previous cases and the file with data of a new case he/she can choose one of the proposed shells to generate a decision support system in a particular domain.

This proposed framework merges two independent CBR shells which support different approaches of the case representation and cover a wide range of possible domains of application (Figure 7).

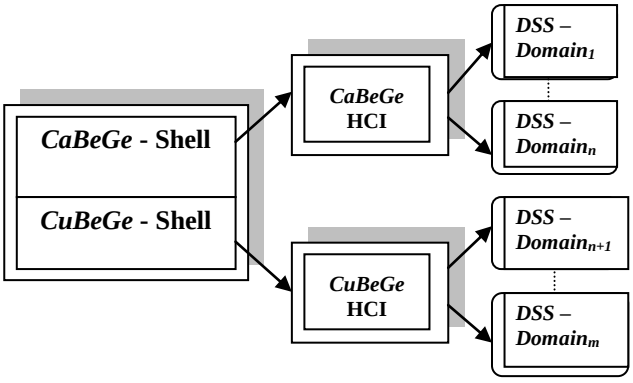

Figure 7. Architecture of proposed framework

This framework inherited numerous advantages of *CaBaGe* and *CuBaGe* shells:

- **Domain independence.** The shells calculate 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 or values in the time series), nor for the expertise of domain experts.
- **Incremental learning.** The shells can gain the new knowledge simply by saving the successfully solved case in the base of previously solved cases (a textual input file).
- **Platform independence.** The shells are implemented in Java so they can work equally well on all kinds of platforms. Furthermore, the GUI of the shells adapts to a target platform so that the user can work with the GUI he used to.
- **Fast retrieval algorithm *(CaBaGe)*.** 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 big is the case base.
- **Generality and robustness *(CuBaGe)*.** Architecture presented in the *CuBaGe* shell can be employed equally well in conventional time series (where all values are known), but also in some non-standard time series (sparse, vague, non-equidistant).

The framework could be useful for many reasons. Both of the shells support generation of a decision support system independent of the domain. So the resulting framework would be also independent on domain. Furthermore, the resulting framework would support generation of a greater variety of possible decision support systems: it will cover the domains where the situations could be specified with a set of attributes, but also the domains where the cases could be represented as curves or time series.

Additionally, the proposed framework could be useful as a good starting point and a core environment for future research. Some of possible utilisations could be done by:

- Testing the performance of different approaches,
- Comparing various approaches of solving the same problem,
- Analysing of the results gained from testing and comparing,
- Developing new applications...

6. Conclusions and Further Work

Case-based reasoning is a problem solving technique that in many respects is different from other major AI approaches. Instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize specific knowledge of previously experienced, concrete problem situations (cases). A new problem is solved by finding a similar past case, and reusing it in a new problem situation. The second important difference is that CBR also represents an approach to incremental, sustained learning, since a new experience is retained each time the problem has been solved, making it immediately available for future problems. The CBR field has grown rapidly over the last few years, as seen by its increased

share of papers at major conferences, available commercial tools, and successful applications in daily use.

Case-based reasoning can be used for solving problems in many practical domains such as: mechanical engineering, medicine, business administration, physics, meteorology etc. Furthermore, for each respective domain, various task types can be implemented. Some of them are: classification, diagnosis, configuration, planning, decision support etc. Moreover, all the domains are possible for each task type. That is exactly the case with our shells. Proposed framework, consisting of two shells *CaBaGe* and *CuBaGe*, with completely different manners of case representation, is suitable for building different decision support systems from a wide range of domains.

Proposed framework can be further improved in several directions. For example, the shell *CaBaGe* could be extended with some meta-language, in which the experts from a particular domain could specify some relations between attributes of the problem. This would further improve the retrieval process. Furthermore, the *CuBaGe* shell could be further improved by adding different well known case representation techniques and similarity measures. This would further increase the competence of the framework for testing, comparing and analysing.

7. References

[1] Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches, AI Commutations, pp. 39-58. 1994.

[2] An Object-Oriented Framework for the Design and the Implementation of Case-Based Reasoners, internet source: http://www-sop.inria.fr/aid/cbrtools/manual-eng/doc_web/Abstract98.html

[3] Budimac Z., Kurbalija V., Case-Based Reasoning – A Short Overview, Proceedings of Second International conference on Informatics and Information Technology "Molika 2001", Molika, Macedonia, December 20-23, 2001, pp. 222-234.

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

[5] Chotirat Ann Ratanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, Gautam Das: MINING TIME SERIES DATA, chapter in Data Mining and Knowledge Discovery Handbook: Maimon, Oded; Rokach, Lior (Eds.), ISBN: 978-0-387-24435-8, Springer, 2005.

[6] Ivanović M., Kurbalija V., Budimac Z., Semnic M., Role of Case-Based Reasoning in Neurology Decision Support, Proceedings of the Fifth Joint Conference on Knowledge-Based Software Engineering "JCKBSE 2002", Maribor, Slovenia, September 11-13, 2002, pp. 255-264.

[7] Kurbalija V., Ivanović M., Budimac Z., Semnic M., Multiple Sclerosis Diagnosis - Case-Based Reasoning Approach, Proceedings of Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS 2007), Maribor, Slovenia, June 20-22 2007, pp 65-71.

[8] Kurbalija V., Ivanović M., Case-Based Reasoning – A Powerfull Artificial Intelligence Approach, Journal of Electrical Engineering 12/s, Slovak Centre of IEE, Vol. 53, 2002, pp.20-24 (Proceedings of Conference of Applied Mathematics for undergraduate and graduate students "SCAM 2002", Bratislava, Slovak Republic, April 19-20, 2002).

[9] Kurbalija V., Ivanović M., CaseBaseGenerator for Intelligent Systems, Novi Sad Journal of Mathematics – NSJOM, vol. 35, no. 1, 2005, pp. 25-40.

[10] Kurbalija, V.: On Similarity of Curves – project report, Humboldt University, AI Lab, Berlin, 2003.

[11] Lenz M., Bartsh-Sporl B., Burkhard HD., Wess S.: Case-Based Reasoning Technology: From Foundations to Aplications, Springer Verlag, October 1998.

[12] Lenz M.: Case Retrieval Nets Applied to Large Case Bases, draft, Lenz's home-page.

[13] Simić D., Budimac Z., Kurbalija V., Ivanović M., Case-Based Reasoning for Financial Prediction, Lecture Notes in Artificial Intelligence (LNAI), ISSN: 0302-9743, vol. 3533/2005, pp. 839-842, (Proceedings of 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2005, Bari, Italy, June 22-24 2005).

[14] Simić D., Kurbalija V., Budimac Z., Improving the CBR System for Financial Predictions, Proceedings of the International Conference on Information and Knowledge Engineering "IKE 04", Las Vegas, Nevada, USA, June 21-24, 2004, pp. 561-567.

[15] Simić, D., Kurbalija, V., Budimac, Z.: An Application of Case-Based Reasoning in Multidimensional Database Architecture. In Proc. Of 5th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), Lecture Notes in Computer Science, Vol. 2737. Springer-Verlag, Berlin Heidelberg New York (2003) 66 - 75.