eXiT*CBR: A framework for case-based medical diagnosis development and experimentation

Beatriz López[*,a,b], Carles Pous[a,b], Pablo Gay[a], Albert Pla[a], Judith Sanz[c], Joan Brunet[b,d]

[a] *Control Engineering and Intelligent Systems Research Group, Universitat de Girona, Campus Montilivi, edifice P4, 17071 Girona, Spain*
[b] *Girona Biomedical Research Institute, Av. de França, s/n, 17007 Girona, Spain*
[c] *Unitat de Consell Genètic en Càncer Hereditari, Servei d'Oncologia Mèdica, Hospital de la Santa Creu i Sant Pau, Sant Antoni Maria Claret, 167, 08025 Barcelona, Spain*
[d] *Medical Oncology Department, Catalan Institute of Oncology, Av. França s/n, 17007 Girona, Spain*

Abstract

Objective: Medical applications have special features that require the development of particular tools. The eXiT*CBR framework is proposed to support the development of and experimentation with new case-based reasoning (CBR) systems for medical diagnosis.

Method: Our framework offers a modular, heterogeneous environment that combines different CBR techniques for different application requirements. The graphical user interface allows easy navigation through a set of experiments that are pre-visualized as plots (receiver operator characteristics (ROC) curves and other kinds of charts). This user-friendly navigation allows physicians to analyze the experiments easily. Experiment replication

[*]Corresponding author at: Universitat de Girona, Campus Montilivi, edifice P4, 17071 Girona, Spain. Tel.: +34 972 418 880; fax: +34 972 418 259.

Email addresses: beatriz.lopez@udg.edu (Beatriz López), carles.pous@udg.edu (Carles Pous), pgay@eia.udg.edu (Pablo Gay), apla@eia.udg.edu (Albert Pla), juditsanzbuxo@gmail.com (Judith Sanz), jbrunet@iconcologia.net (Joan Brunet)

is managed automatically by the system. Used as a plug-in on the same interface, eXiT*CBR can work with any data mining technique such as learning the relevance of features.

Results: The results show that eXiT*CBR is a user-friendly tool that facilitates physicians to use the CBR method to determine a diagnoses in the field of breast cancer, dealing with different patterns implicit in the data.

Conclusions: Although several tools have been developed to facilitate the rapid construction of prototypes, none of them has taken into account the particularities of medical applications. eXiT*CBR aims to fill this gap. It uses CBR methods and common medical visualization tools, such as ROC plots, that facilitate the interpretation of the results. The navigation capabilities of this tool facilitate the tuning of the different CBR parameters using experimental results. In addition, the tool allows reproducibility, as the experiments can be replicated as many times as required.

Key words:

Case-based reasoning, experimentation supporting tool, development supporting tool, medical diagnosis, breast cancer diagnosis.

---

1. Introduction

Case-based reasoning (CBR) has evolved over the last years, and there are currently a considerable number of techniques available for each phase of CBR system development, although these are unevenly distributed. Each particular domain requires that appropriate techniques can be selected for each phase and the parameters involved can be appropriately tuned. Several methodologies have been designed to guide CBR system development [1,

2], and some of these are accompanied by software tools that aim to build prototypes quickly, which can then be progressively corrected [3–5]. Some general purpose tools such as cf. [6], CBR* [3], CBR Shell [4] and jCOLIBRI [7] are available and can be used to develop a cancer diagnosis system as well as an electric fault diagnosis system.

Although there should always be a compromise between genericity and specificity when designing software tools, medical applications have particular requirements that justify the development of a specific CBR platform. In [8], for example, the authors highlight the need to study what is common to all successful applications of CBR systems and propose a search for common representations of CBR data and methods in this particular field. Other studies have been carried out in relation to the special cases of CBR [9, 10] and data mining [11] in the medical domain. In [10] we describe our experience of working together with several teams of physicians, and we agree with the problems presented in [8] and [11]. From that analysis, we detected the need to deploy a particular tool for CBR in medicine that can cope with the challenges identified. Specifically, we have dealt with the following issues: attaining a probabilistic interpretation of CBR, improving the interface to interpret the physicians' results, and the hybridization of CBR with other techniques.

First, a way to provide a probabilistic interpretation of CBR is closely related to the kind of visualization tools physicians usually work with: mainly receiver-operator characteristic (ROC) and cost curves. ROC curves depict the tradeoff between hit rate and false alarm rate [12, 13]. In cost curves, classifier performance is shown through the class distribution changes [14].

So the incorporation of these visualization tools in a CBR environment (although not being a CBR technique per se) should facilitate the use of CBR in medical domains.

Second, the use of ROC curves can also improve the interface with physicians. Moreover, result reproducibility, so that the results obtained by different work teams are the same given the same method and identical test material, should be guaranteed to offer physicians confidence. And since repeating experiments is often a critical and necessary process to obtain the parameters that provide the best results in a CBR system, special care should be taken to adequately store all the data used when the system is tested so they are available to the research community.

And third, the complementarity of other methods for feature learning, techniques for dealing with data dependencies [8] and other methods should be contemplated, so that CBR can be hybridized with other data mining and reasoning tools.

To our knowledge, no current tool supports all these features, in particular the two former ones. The final CBR performance depends on the skills of the engineer choosing the corresponding parameters guided by the physician's interpretation of the results obtained so far. Unless we provide the appropriate visualization tools for result interpretation and reproduction, it will be difficult to adequately tune the CBR parameters.

In this work a specific CBR tool for medical applications is presented to support CBR system development, experiment replication and result inter-

pretation. The framework, eXiT*CBR[1] is a modular environment that combines different CBR techniques for different application requirements. The graphical user interface allows easy navigation through a set of experiments that are pre-visualized as plots (ROC curves and other kinds of charts). This allows a knowledge engineer and a physician working side by side to test different CBR parameters and visualize the results in a user-friendly way. Experiment replication is assured, since experiments are managed automatically by the system. The methodology is based on keeping datasets, configuration files of the experiments, and other data.

Finally, any data mining method can be plugged into the system as a preprocessing step to treat the medical data, according to the current trends in knowledge transfer [15]. Thus, both the knowledge engineer and the physician have a single interface to deal with complex data. eXIT*CBR is based on CBR as a decision-making methodology, although it can be complemented by any other technique.

Therefore, the eXiT*CBR platform goes beyond the rapid prototyping of medical CBR to offer an extensible architecture for overall experimentation, and to support experiment replication and interpretation via common medical graphics tools. eXiT*CBR does not build a new system from scratch, but rather integrates existing CBR methods with new ones as required, and facilitates the integration of other data mining mechanisms.

This article is organized as follows. First, we introduce the eXiT*CBR framework. Next, we give brief descriptions of the eXiT*CBR functionalities.

[1]eXiT stands for the acronym of our research group.

Then, an example is used to describe the main steps required to develop a CBR application with eXiT*CBR, obtain results, and have them evaluated by the physicians. Finally, we compare our work with several previous studies and end the paper with some conclusions.

2. The eXiT*CBR framework

The eXiT*CBR framework was designed to bring together CBR methods currently used in medical applications, other data mining mechanisms, and visualization techniques that can be plugged into it. eXiT*CBR also facilitates the incorporation of new techniques, if required. Our architecture follows a modular approach based on the different phases in a CBR system.

For a reader unfamiliar with the CBR methodology, a basic CBR system relies on a case base in which past experiences are stored. For example, in a breast cancer diagnosis, a case base contains information about patients that have been diagnosed in the past with the illness and about healthy people. Using this case base, a CBR system is able to diagnose future cases by following four main phases of action: retrieve, reuse, revise and retain [16]. First, in the retrieve phase, the current case is compared with all the past experiences in the case base, and the most similar are recovered. Next, in the reuse phase, a solution (diagnosis) to the current case is determined based on the solutions found in the retrieved cases. Third, the computed solution is evaluated in the revise phase. Finally, the retain phase analyzes whether to retain the case in the case base.

eXiT*CBR follows a modular approach: the different CBR phases are implemented as generic classes. When a new method is required but is not

provided in the system, it can be assembled as a particular instance of the generic class. It is important to distinguish between classic methods and methods that involve either the integration of other data mining tools or the hybridization with other artificial intelligence (AI) techniques. Thus, when users wish to add new CBR methods, they only need to instantiate the generic classes defined for this purpose. However, when other techniques should be integrated or hybridized, the corresponding executable codes should also be included. With this approach the system has the properties of modularity, reusability and extensibility.

As our framework goes beyond pure CBR prototyping and aims to support experimentation, other elements are required in addition to the basic CBR modules. Five main experiment-related components are distinguished: the experiment interpreter core, the CBR engine, the pre-process and post-process elements and the experiment navigator. In order for any method to manipulate the data, a common representation of cases is required. As the experiment is the key issue in eXiT*CBR, the architecture has been organized into different components around the *experiment interpreter core* (see Figure 1). All the components and common data representation are explained in the remainder of this section.

eXiT*CBR has been developed using the Java language and the jfreechart library[2]. It is compatible with the Linux and Windows operating system platforms. The software allows users to select a language to work in. Currently these languages include English and Catalan. We are in the process of mak-

[2]http://www.jfree.org/jfreechart/ (Accessed: 8 April 2010).

ing it possible to download the platform for free from our web page. In the meantime, however, any interested researcher can request a copy from the authors.

2.1. Case representation

With any method used eXiT*CBR requires a plain csv file to handle the data. Each row corresponds to a case, and each column to attributes of the cases. One of the attributes should be marked as the one containing the diagnosis information (class).

The first four rows describe the attributes as follows:

- The first row corresponds to the attribute descriptions (for example, "Age when first child was born")
- The second row corresponds to the attribute name (usually in a cryptic form, as for example, "age1stborn").
- The third row corresponds to the attribute type (-1 ignore, 0 discrete, 1 numerical, 2 textual, 3 date)
- The fourth row corresponds to the attribute weight (relevance).

We believe that this simple, plain representation covers most of the data used in medical applications[3] and is easy to manage and general enough to be used by any of the current CBR techniques (mainly distance functions).

[3]In fact, physicians usually collect data in Microsoft Excel or SPSS files, with a similar format.

From the original data file in csv format, and using, for example, the facility described in section 3, several datasets can be generated for different experiment methodologies. All of the datasets keep the same case representation.

*2.2. Experiment interpreter: the eXiT*CBR core*

The core of the framework is the experiment interpreter. A configuration file that contains all the information required to replicate an experiment is defined. The eXiT*CBR core interprets the configuration file, and applies the corresponding methods, or calls the corresponding eXiT*CBR components: first, the preprocess, then the CBR engine, then, the post-process. Only the experiment navigator acts interactively with the user.

The different parameters involved in an experiment depend on the method employed. Currently, there are two main methods: batch and cross-validation.

In eXiT*CBR batch processing means performing a single run of a CBR in a case base (see Figure 2, *top*). The input to this kind of experiment is a single dataset with two files: the training and the test files. The experiment interpreter core takes the training data to generate the case base, and uses the test file to obtain the results of the CBR system defined in the configuration file. These results are given in the form of CBR performance according to the experiment measures selected by the user. Note that the solutions of the test cases are known in advance, and they are used to verify that the CBR system, configured as it is, works appropriately.

Stratified cross-validation processing implies multiple runs of a CBR configuration with different datasets (see Figure 2, *bottom*). Input data in this kind of experiment are k datasets, "set_1", ... "set_k", each composed of at

least one training file and one test file. Thus, the experiment interpreter core takes the training data of a given set "set_i" as the case base, and uses the test data to obtain the results of the CBR system. Then, the performance of the CBR system with all of the datasets is averaged.

eXiT*CBR also allows a multi-agent system (MAS, [17]) method in which several CBR systems cooperate to obtain a diagnosis. Thus, for example, an experiment can be MAS-cross-validation or MAS-batch. For the sake of simplicity, in this paper we focus on the single (non-MAS) methods.

In the configuration file the data to be used, the pre-processing techniques, the CBR methods, and how the results are post-processed are defined. First, batch processing involves the specification of a single dataset, while cross validation requires the specification of a set of them. Other information related to the data is the name of the attribute that contains the solution of the case (i.e. class information since we are interested in case-based medical diagnosis). Second, pre-processing methods, as discretization of numerical attribute values, normalization of numerical attribute values, or filtering out a subset of case attributes, can be defined. In this case, the methods are the same for any kind of experimental methodology. Third, the particular techniques to be used in each of the four main phases of CBR should be specified. Both, classical CBR methods as well as other AI techniques can be included. For example, in Figure 3 the *retrieve_selection* method employed is *MajorityK* with the parameter $k = 5$ [4] but a fuzzy system that determines case similarities can be also used (see for example [18]). Combining different

[4]The details of the file contents are described in section 4.

AI techniques in hybrid systems seems to be a way to tackle the complexity and multi-faceted aspects of medical data. Therefore, in eXiT*CBR it is possible to develop a decision support system based on a single technique such as CBR, that is complemented by other AI techniques. And fourth, post-processing depends on the environment where the CBR system is designed to work. If the system is developed to cooperate with other systems to solve a given problem (on-line mode), the output of the CBR system should go through a communication port; on the other hand, if the purpose of the CBR system is to obtain results in isolation (off-line mode), visualization tools could be more useful. In both cases, several performance measures can be specified to show the results to the user.

When an experiment runs, all the files associated with it are stored in a folder with the name of the method used to conduct the experiment (cross-validation or batch), and the day and the time it took place. For example, in Figure 5 the experiment folder with the name "CrossValidation-CBR-20080516-124248" means that this folder contains the files belonging to an experiment carried out at 12:42:48 PM, on 16 May 2008, using cross-validation as the experimental method. This label ensures that repeating the same experiment with the same data a couple of seconds later does not overwrite the previous experiment results, even though the same experiment file name is used. The information stored in the result folder is used by the post-processor component to plot the results of the current experiment for the user, and by the navigator tool to compare different experiments.

2.3. The pre-process element

The first component of the framework that the experiment interpreter core calls to perform an experiment is the pre-process element which takes the description of the discretization, normalization and feature selection slots of the configuration file and applies them. Discretization methods convert numerical data into categorical data [19]. Normalization methods assure that all the numerical values are comparable in the [0,1] interval. Finally, feature selection methods determine which of the available features (attributes) are relevant for the application. Relevant and irrelevant features can be labeled with high and low weights, respectively, assigned to them [20, 19].

After the pre-process is completed, the experiment core takes back control and passes it to the CBR engine.

2.4. The CBR engine

All the information required to set up a CBR system according to user requirements is stored in the configuration file. The CBR engine is responsible for reading this file, extracting the selected methods and parameters and, finally, calling and executing the related algorithms. At the end of this process a set of files is obtained as output. These files are used by the post-process element of the architecture to display the results in the format the user selects.

The following paragraphs describe the modules that the CBR engine calls when launched.

2.4.1. Retrieve module

The retrieve module compares the cases with a given test case, and selects the most similar cases from the case base. Three key methods are involved in this process:

- The distance method or similarity measure employed to compare cases
- The methods employed to handle missing information when the similarity measures are applied
- The selection procedure to determine the most similar cases.

All these methods are defined in a generic way and can be instantiated as required or as they become available as depicted in Figure 4.

There are local and global similarity measures (Figure 4). Local similarity measures compare two attribute values. There can be as many kinds of local similarity measures as there are operands available. For example, the Euclidean distance is the measure proposed most often to handle numeric data, while the Hamming distance is set for categorical (discrete) data [21]. Other local similarity measures regarding data trees (to handle inheritance information, for example), series, and dates can also be used. Global similarity measures combine different local similarity outcomes to determine the similarity between two cases. An example is the weighted average, but many others can be considered [22].

Local distance methods have to be able to deal with missing values. They are cost sensitive and used widely in medical applications. According to [23], they can be classified as missing completely at random (MCAR), missing at random (MAR), and not missing at random (NMAR). MCAR is used when

the probability of missing a value is the same for all attributes, MAR is used when the probability of missing a value is only dependent on another attribute, and NMAR is used when the probability of missing a value is also dependent on the value of the missing attribute. Any of these methods can be implemented and added to eXiT*CBR.

Finally, the selection method to be used should also be chosen. Possibilities include selecting the k-nearest neighbors or selecting the cases with a similarity degree higher than a pre-fixed threshold.

As output of the retrieve phase, the CBR engine creates a file containing the distance matrix that depicts the similitude between each test case and the cases in the case base.

2.4.2. Reuse module

In the reuse module, a method for adapting the solutions of the retrieved cases to a given test case needs to be specified. This phase constitutes one of the main limitations of CBR in medical diagnosis [24]. The majority of medical CBR systems suggest past solutions without a further adaptation process. The Bilska-Wolak method [25] proposes a probabilistic approach; however, this method has only been applied with a low number of features, and much more research is needed to deploy it in real environments. eXiT*CBR allows users to create and test their own reuse algorithm easily.

2.4.3. Revise module

Most of the current medical CBR systems rely on human feedback in the revise phase. There are currently no other alternatives. However, in other environments, simulators are also possible [26].

In the experiments used with eXiT*CBR, namely, batch and cross-validation, the case solution is known in advance. Thus, two main results are generated:

- The classification matrix, which contains information about the real classification of the cases and the prediction given by the system (as output of the reuse phase);
- The confusion matrix, which contains information about true positive, true negative, false positive and false negative rates.

2.4.4. Retain module

Once the solution proposed for the new case is revised, a decision has to be made about whether to retain the new case or not. This decision will depend on whether the cases already in the case base produce a correct solution or not. Since the CBR system's core is a case base that can be very large, it has a lot in common with the data mining methods for data processing. When the case base grows, a good maintenance policy is necessary. This means that we have to delete, add or modify cases to keep the system performing well. Instance-based learning algorithms such as IB3 [27] or DROP4 [28] can be applied in this stage to decide whether to keep the new case, forget it, or keep it and delete other cases.

Adding cases automatically to a medical domain without any human control can be somewhat dangerous, since the system can increase the information about a particular case or situation, leading to a large bias in the system. An alternative is to temporarily store new cases until a physician checks them. CBRworks [29] has implemented this approach. In eXiT*CBR we follow this conservative view of the development of CBR systems, and we

leave the study of the integration of algorithms such as IB3 and DROP4 for the future.

2.5. The post-process element

The task of the post-process element starts when the CBR engine finishes. The post-process component of eXiT*CBR is related to the type of measures the user is interested in (confusion matrix, success rate, specificity, etc.) and how to visualize them. In addition, it processes the information according to the working mode selected by the user: on-line or off-line. More details are given in the following subsections.

2.5.1. Visualization results

In the configuration file, the user decides in advance on the method and measures to apply to the current experiment, as well as how to view the results. Then, the post-process component produces the corresponding plot from the information provided by the CBR engine (confusion matrices). A *Figure.png*, generated as output, depicts the corresponding plot derived from the experiment.

In the current implementation the *experiment navigator tool* uses the confusion matrix and the plots to help the user choose and compare experiments (see next section).

eXiT*CBR visualizes the results with ROC curves by default (like the one in Figure 10). However, there are certain situations in which visually comparing ROC curves is not the best way to select the best diagnosis system; in these cases using an area under curve (AUC) value can help make the decision. The larger the AUC value, the better the performance of the

diagnosis system. Therefore, the ROC figures are complemented with AUC information. Of course, the ROC curves and the AUC values have to be checked by physicians to certify that these results agree with their medical criteria.

2.5.2. Working mode

We distinguish between the off-line and the on-line modes. In an off-line mode, several batch processes and validation procedures are applied according to the experimental parameters. The system's answers in this context are visualization plots. In the on-line mode the system cooperates with other systems to solve a given problem. It uses several approaches based on ensemble learning techniques, including distributed and agent-based approaches [30].

2.6. The experiment navigator

Repeating experiments and interpreting their results are the two key points for medical diagnosis. Reproducibility should guarantee that when the same method and the same test material are used, identical results are obtained. Therefore, all the data used when the system is tested has to be adequately stored in a file structure (see Figure 5).

As shown in Figure 5, all the experimental outcomes are stored in the folder *results*. Inside this folder, users can organize the results into several destination folders ("Destination folder 1" and "Destination folder 2" in the case in Figure 5) according to selected criteria. For example, users could be interested in putting all the experiments carried out with a particular database or a set of experiments carried out using only a subset of the data

in the same destination folder.

The experiment navigator works interactively, so that when the mouse is positioned on an element of the file structure, the results of the experiments are automatically visualized. This is possible with the pre-computed images and the information kept after running an experiment.

3. eXiT*CBR functionalities

The following functionalities have been developed for user interaction: *create/edit configuration file*, *data conversion*, *modification csv*, *dataset generation*, *CBR application*, and *navigation*. They are made available as buttons in the left area of the main window frame of the platform. In addition, all of the other data mining applications integrated into the platform are available through the "plug-in" option. Finally, it is possible to change the installation of the tool with the program configuration facility.

3.1. Create/Edit configuration file

Writing a configuration file by hand can be so unpleasant it would make anyone give up using the tool. Therefore, we have developed an alternative editor interface so that users can define a configuration file in a user-friendly way, in which all the available techniques are displayed in a pop-up menu.

The directory which contains the datasets, the methods used for each CBR cycle, and the type of experiment can be defined using this graphic window.

3.2. Modify csv

Machine learning methods in general and CBR in particular assume that examples are independent. However, medical data cannot always satisfy this constraint. For example, in breast cancer data, two members of the same family can also be in the same dataset. Note that this situation is different from deduplication or record linkage [31], since the individuals are different but related.

This clear dependency can be removed if family relationships are identifiable in the dataset. Of course, other kinds of hidden dependencies are more difficult to avoid.

Thus, this facility has links to the current available, domain-dependent methods to deal with cases independency.

3.3. Data conversion

The data provided by physicians generally have different file formats and contain dissimilar information. Hence, the first step is to transform original datasets into csv files, which is the format that our framework is able to read. A breast cancer relational database has been converted into a csv file as an example. In general, every medical database has its own structure, format and type of data. Therefore, it is very difficult for the method to read all the available medical databases. It is often necessary to check new databases manually so that they are correctly transformed into a csv file.

3.4. Dataset generation

This function defines datasets for experiments. The input is the original database (in csv form) in which the medical information is contained (cases).

The output is a set of different datasets for training and testing the system according to the experimentation procedure (for example, cross-validation).

There are two kinds of procedures to determine the length of the test files: fixed (often used by physicians when, for example, ten tests cases are required), and percentage-based (usually in cross-validation).

Once the technique for generating the dataset is selected, the system generates as many datasets as necessary. For each dataset, four files are produced:

- Patient[5] test file: includes cases of patients to be tested.
- Control test file: includes cases of persons that do not have the illness to be tested.
- Training file: includes the cases which form the case base. There are as many patients as healthy people.
- Remaining file: includes the cases that exceed the proportion in the training file. That is, if there are many more patients than healthy people, the generated files are stratified, and some remaining patient cases would not be considered in this set.

Figure 6 illustrates the process when a stratified cross validation methodology is used with a fixed length of test files (n cases per file) and a fixed number of datasets (m). Note that these datasets are generated apart from

[5]These are cases in medical terms, but in order to avoid confusion with the CBR terminology we prefer to use the term "patient".

the CBR application. Therefore, it is possible to define several CBR configurations and test them with the same datasets.

3.5. CBR application

The CBR application facility runs the CBR experiment according to a given configuration file. As a result, several internal files that contain the outputs of the application and other result-visualization supporting information are generated, as explained in the previous section.

3.6. Navigation

The navigation facility is perhaps the most innovative aspect of our framework. It allows the user to navigate in a user-friendly way through the different experiments run so far. In the left panel, the experiments executed are listed according to the file structure presented previously (see Section 2.6). When the user moves the cursor across the experiment tree, the corresponding results are shown in the top right panel The system also displays the dataset and methods used in the selected experiment.

To facilitate choosing from among all the experiments, the navigator allows users to hold a particular experiment figure while navigating across the experiment tree. With this utility, users can explore and graphically search for experiments they are interested in. These figures (as many as required) can also be expanded and overlapped in a single graphic, as shown in Figure 7. Each curve is labeled with the name of the experiment. Several types of lines and colors are available.

When a figure is expanded, the confusion matrix is also shown. If the figure is related to a cross-validation experiment, the confusion matrix obtained

for each fold (dataset) is shown in a tab structure.

It is clear that the experiment navigator greatly facilitates the comparison of experiment results, which speeds up the tuning of the diagnosis system parameters.

3.7. Plug-ins

To add other data mining or computing techniques (as for example, a feature learning mechanism), a step before CBR pre-processing might be necessary. Therefore, eXiT*CBR allows to plug-in any technique in the same interface. For example, Figure 8 shows the effects on the dataset after applying a plug-in called *family risk calculator*, which calculates for each case a new attribute related to family information.

4. Application to breast cancer diagnosis

The first application eXiT*CBR has been used with is a breast cancer case-based system. The public database, the Breast Cancer Wisconsin (Diagnostic) Dataset [32, 33], was used to conduct preliminary tests of the framework functionalities.

Then, the CBR system was developed for our own breast cancer dataset, which consists of 871 cases, 628 corresponding to healthy women and 243 to women with breast cancer. There are 1199 attributes for each case. They correspond to people's habits (smoker or not, diet, sport habits, etc.), disease characteristics (type of tumor, etc.), and gynecological history, among others.

Therefore, a complex database helps to illustrate the capabilities of the platform when real, complex data is dealt with. In addition, the physicians

involved in gathering the data are also involved in developing the CBR system and the experiments with eXiT*CBR, so they can also provide useful feedback on the platform.

4.1. Main steps for developing a CBR application

Figure 9 illustrates the process followed. The unfilled boxes show optional steps, while filled ones should be considered mandatory.

First, the destination directory of all the experiments to be carried out with this database is defined with the program configuration facility.

Second, our original access file is converted into a csv file with the data conversion procedure. Third, different datasets are built to perform a cross-validation experiment. Up to ten folders have been generated with test cases composed of ten patients plus ten healthy people, and the remaining cases as training cases.

Now everything is ready to start experimentation. The first CBR system was set up with the edit configuration facility. The resulting configuration file, seen in Figure 3, shows that the results will be stored in the "results" directory, the name of this first experiment is "original", and the datasets are taken from the directory "Datasets10". Although the name of the class attribute written in the textual file of Figure 3 is cryptic (`A1_1_1#8`), note that it has been selected via pop-up menus set up according to the attribute description provided in the csv file. Other options, such as the different methods involved in each phase, are also provided via pop-up menus. In the retrieve phase of the CBR system, the experiment has been set up with a Euclidean distance function for numerical values, a Hamming distance function for categorical ones, the average (mean) as the global similarity measure, and

the selection being the "majority" rule in which the five most similar cases would be selected (five is the parameter of the method). In the reuse phase, the probabilistic-Pous method is selected, and the threshold required to use it is set to 0.7 (see [34] for further details on this method). In order to generate results and plots in the experiments, information about the variation of the parameter (i.e. the threshold) must also be provided. Assuming that the parameter varies in the interval $[0, 1]$, variations would be studied from 0.0, and with a step of 0.1. Finally, there is no revise phase, since the experiment deals with a cross-validation experience, and the test cases are not retained in the system.

The default values for the experiment have been set, namely, the true positive and false positive rates of measures, and ROC curves have been used as the visualization plots.

The off-line working mode is selected as post-processing, which means that the output will be sent to the user, instead of being reported on-line to another system.

Once the configuration file is defined, it is possible to call the eXiT*CBR core to obtain an instantiation of the CBR system defined in it, and thus obtain the corresponding results. The CBR application facility is used for this and, after running it, the user obtains the results graphically as shown in Figure 10.

4.2. Using the experiment navigator tool

Next, as the original dataset has dependent data, a particular process for making cases independent, which is linked to the modify csv facility is run to remove the dependent cases and obtain a new dataset. Since the csv file

changes, it is necessary to execute the dataset generation facility again to obtain the files for cross validation (see functional dependency interactions in Figure 9).

In the configuration file, it is only necessary to change the information about where the new data are, and the name of the experiment: "originalIndp". The same edit configuration facility allows the existing configuration file to be loaded and modified.

Note, however, that the information from the previous experiments is not "changed", even though it seems like the user "updates" the files. As previously explained, this is because the eXiT*CBR core always maintains a backup of the different files involved in an experiment.

After the modifications have been performed, the experiment is run again with the new data, and new results are obtained.

The experiment navigator facilitates the interpretation of the results. Both experiments can be enlarged with the corresponding button of the experiment navigator, and compared as seen in Figure 11. The user can see whether the new experiment runs better, and continue to improve the results from this configuration.

4.3. Improving the case base with another data mining tool

If the physician now realizes that there is additional information in another file regarding family risk that could improve the results if incorporated in the data, then a procedure for learning the family risk is first defined and added to the existing data as a new attribute for each case. This procedure is implemented, and as a "jar" file can be used directly in eXiT*CBR through the plug-in facility, which means other researchers can also use it.

After adding this information, the experiment is repeated as in 4.2 and the results are improved again.

4.4. Changing the experimental parameters

Now, the physician tries to improve the results by increasing the "k" parameter of the retrieval method: from 5 to 7. This experimental parameter can be modified in the configuration file thanks to the edit configuration facility. The experiment is repeated and the new results are then compared with the previous ones thanks to the navigation tool.

Note then, that all the parameters of the configuration file can be changed. They include pre-processing (discretization, normalization and feature selection), retrieval, reuse, revise and retain methods, as well as visualization and post-processing methods. However, the use of some of the eXiT*CBR facilities, beyond the scope of the configuration file, could involve the generation of a new dataset, in addition to repetition of the experiment, as shown in sections 4.2 and 4.3. All of them, when based on a cross-validation method, are comparable, and the navigation tool can help to determine the best set of parameters for a given application.

4.5. Feedback from physicians

To evaluate this framework qualitatively, physicians filled out a questionnaire. The following conclusions can be drawn from their answers.

Several tools currently support breast cancer diagnosis. A recent survey can be found in [35]. However, none of these models satisfies the current needs of physicians. The Gail model, for example, includes some epidemiological data and a single piece of data regarding family information. Other

models, such as Claus's model, uses more in-depth family information, while others focus on genetic information, like the BRCAPro program does. In general, the available tools deals with a single model. Conversely, the approach proposed with a CBR methodology, allows physicians to tackle all the data at once, in an integrated way, dealing with different patterns implicitly in the data to determine a diagnoses.

Visualization tools allow them to play with different parameters (case attributes or variables) of an illness. For example, information regarding family risk can be included and the results compared by means of the ROC curves. Future applications are bein contemplated in the area of preventive medicine, such as adding a new attribute of a surgical operation and testing the benefits obtained with it.

5. Related work

General CBR tools have been developed in several previous studies. For example, cf. [6] is a lisp environment designed for rapid prototyping. CBR* [3] and CBR Shell [4] were both designed following an object-oriented methodology and implemented in Java, like this system. Thus, the CBR frameworks have inherited the modularity, reusability and extensibility properties of the object-oriented paradigm. The main difference between this work and previous approaches is the aim. Our framework is designed to provide engineers and physicians with navigation functionalities across different experiments, to help them choose the appropriate CBR methods and parameters and assure experimental reproducibility. In addition, we focus on a particular case of CBR: medical diagnosis. CBR is used for classification and the visualiza-

tion techniques are mainly based on ROC graphs, the most commonly used technique in this domain.

Other interesting CBR frameworks are jCOLIBRI [7] and MyCBR [36]. The jCOLIBRI framework follows a modular approach similar to the one used here. It characterizes the different CBR phases according to an ontology. This approach is interesting as it provides researchers with a common ontology to define CBR systems. This ontological approach will be taken into account in future versions of eXiT*CBR. The scope of jCOLIBRI and eXiT*CBR, however, are different: like some previous approaches, jCOLIBRI supports the development of CBR systems, while eXiT*CBR supports experimentation.

MyCBR focuses on defining similarity measures for the CBR retrieval phase. MyCBR is designed as a plug-in in the PROTEGE[6] ontology editor. The case representation needed in MyCBR is based on a csv file like in the eXiT*CBR platform. However, we include information on the attribute types and weights, while MyCBR assumes that this information will be edited and provided by an ontology to be built in PROTEGE. The ontological approach to describe attributes could be complementary and may be used to extend eXiT*CBR in the future. However, we tried to load the breast cancer database into MyCBR and had some problems with the amount of attributes involved in the data. Therefore, MyCBR could be more useful to test some properties of similarity measures than as an experimental platform for real CBR systems.

[6]http://protege.stanford.edu/ (Accessed: 8 April 2010).

Regarding other general platforms for data mining, Weka [37] also offers an experimentation framework. It allows users to deal with different kinds of algorithms at the same time, and to compare the different results. However, it is a general purpose environment, unlike the tool presented in this article which uses a specific methodology (CBR) in a specific domain (medicine) and can visualize the results in a useful way for the domain it is designed. Moreover, Weka does not support a full CBR cycle, only the nearest neighbor approaches (only retrieval).

6. Conclusions

The development of a CBR system for medical diagnostic purposes is always a time-consuming activity. Although several tools have been developed to facilitate the rapid construction of prototypes, none of them have incorporated navigation through experiments with different CBR configurations. Experimentation is a cornerstone of the empirical approach used by engineers and physicians to obtain more in-depth knowledge about the data they are working with.

The tool developed here, eXiT*CBR, tries to fill this gap. The experimental framework facilitates interaction with physicians, which are not lost in the different experiments generated. The tool also allows reproducibility, since facilitate experiment replication as many times as necessary. This is important when the research results concern medical applications, and could be very useful for the medical community.

In this article we have shown how eXiT*CBR has been applied to develop a breast cancer CBR diagnostic system. Our next step is to collect the most

successful methods in the field of medicine to test the system further, and at the same time enrich the platform. We also need to study how eXiT*CBR can be applied to tasks other than diagnosis. For example, it can be appllied to therapy problems, depending on their complexity, by providing the appropriate reuse method. If the therapy consists of drug adjustments, value variations can be computed according to the methods studied in [38, 39]. Other complex adaptation processes, such as the one proposed in [40] for breast cancer treatment, could be studied with the hybridization and integration capabilities of eXiT*CBR. Some challenges could be made against the retrieval methods, since some authors distinguish more than one retrieval step based on either individual patient history (as in [41]) or problem types (as in [38]). On the other hand, diagnosis, as shown in this article, is a binary classification problem that can easily be extended into a multi-class problem by combining the results of binary classifiers as in [42]. We have started to develop an application with such features using eXiT*CBR (see [43]).

In future work, we should also analyze the extensibility of eXiT*CBR to handle other case representations that allow the development of structured and conversational CBR systems. Representations should also consider graph structures for dealing with complex medical concepts, as has been done in [44]. Moreover, the particularities of conversational CBR require the inclusion of question-answer pairs in addition to a textual description of problems [45]. Finally, textual CBR systems should also be developed due to their importance in medicine. The works of [46] could be a good starting point.

Acknowledgments

This research project has been partially funded by the Spanish MEC projects DPI-2005-08922-CO2-02 and CTQ2008-06865-C02-02, the Girona Biomedical Research Institute (IdiBGi) project GRCT41 and DURSI AGAUR SGRs grants 2005-00296 and 2009-00523 (AEDS). The Wisconsin breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

References

[1] R. Bergmann, W. Wilke, K. Althoff, S. Breen, and R. Johnston. Ingredients for developing a case-based reasoning methodology. In *R. Bergmann and W. Wilke (Eds.): Proceedings of the 5th German Workshop in Case-Based Reasoning (GWCBR'97), LSA-9701E, University of Kaiserslautern*, pages 49–58, 1997.

[2] R. Bergmann and K.D. Althoff. Methodology for building CBR applications. In *M. Lenz, B. Bartsch-Spörl, H.D. Burkhard and S.Wess (Eds.): Case-Based Reasoning Technology: From Foundations to Applications.* Springer, Berlin, Lecture Notes in Computer Science 1400, chapter 12, pages 299–326, 1998.

[3] M. Jaczynski. A framework for the management of past experiences with time-extended situations. In *F. Golshani and K. Makki (Eds.): Proceedings of the Sixth International Conference on Information and Knowledge Management (CIKM).* ACM Press, New York, pages 32–39, 1997.

[4] S. Aitken. CBR shell java-v1.0, http://www.aiai.ed.ac.uk/project/cbr/cbrtools.html. (Accessed: 8 April 2010).

[5] S. Bogaerts and D. Leake. IUCBRF: A framework for rapid and modular case-based reasoning system development. Technical Report TR617, Computer Science Department, Indiana University, 2005.

[6] J.L. Arcos. cf development framework, http://www.iiia.csic.es/˜arcos/. (Accessed: 16 June 2009)

[7] B. Diaz-Agudo, P.A. Gonzalez-Calero, J.A. Recio-García, and A.A. Sánchez-Ruiz-Granados. Building CBR systems with jCOLIBRI. *Science of Computer Programming*, 69:68–75, 2007.

[8] I. Bichindaritz and C. Marling. Case-based reasoning in the health sciences: What´s next? *Artificial Intelligence in Medicine*, (36):127–135, 2006.

[9] I. Bichindaritz, S. Montinali, and L. Portinali. Special issue on case-based reasoning in the health sciences. *Applied Intelligence*, (28):207–209, 2008.

[10] B. López and C. Pous. Applications in medical dababases. In *Saita, L. (Ed): BluePrint in Ubiquitous Knowledge Discovery KDUbic*. European project IST-6FP-021321 Coordination Action document, http://www.kdubiq.org/ (Accessed: 16 June 2009), pages 36–39, 2007.

[11] K.J. Cios and G.W. Moore. Uniqueness of medical data mining. *Artificial Intelligence in Medicine*, 26(1-2):1–24, 2002.

[12] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[13] J.A. Swets, R.M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, October 2000.

[14] C. Drummond and R. C. Holte. Explicitly representing expected cost: an alternative to ROC representation. In *S.J. Simoff and O.R. Zaïane (Eds.): Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM Press, New York, pages 198–207, 2000.

[15] T.G. Dietterich. Three challenges for machine learning research. Plenary talk at *The 9th Ibero-American Conference on AI (IBERAMIA)*, Puebla, México, 2004.

[16] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[17] G. Weiss. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence.* MIT Press, USA, 1999.

[18] K.S. Aggour, M. Pavese, P.P. Bonissone, and W.E. Cheetham. SOFT-CBR: A self-optimizing fuzzy tool for case-based reasoning. In *K.D. Ashley and D.G. Bridge (Eds.): Case-Based Reasoning Research and Development (ICCBR).* Springer, Berlin, Lecture Notes in Computer Science 2689, pages 5–19, 2003.

[19] H.F. Francisco Núñez. *Feature weighting in plain case-based reasoning.* PhD thesis, Technical University of Catalonia, Spain, 2004.

[20] T. Martinez. Selecció d'atributs i manteniment de la les base de casos per a la diagnosi de falles. Master's thesis, Universitat de Girona, Spain, 2007.

[21] D.R. Wilson and T.R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[22] V. Torra and Y. Narukawa. *Modeling Decisions: Information Fusion and Aggregation Operators.* Springer, Berlin, 2007.

[23] S. Zhang, Z. Qin, C. X. Ling, and S. Sheng. Missing is useful: Missing values in cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1689–1693, 2005.

[24] S. Montani. Exploring new roles for case-based reasoning in heterogeneous ai systems for medical decision support. *Applied Intelligence*, (28):275–285, 2008.

[25] A.O. Bilska-Wolak and C.E. Floyd. Development and evaluation of a case-based reasoning classifier for prediction of breast biopsy outcome with bi-radstm lexicon. *Medical Physics*, 29(9):2090–2100, September 2002.

[26] K. J. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence*, 45(1-2):173–228, 1990.

[27] D. W. Aha, D. Kibler, and M. Albert. Instance based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[28] D. Wilson and T. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[29] S. Schulz. CBR-works - a state-of-the-art shell for case-based application building. In *Proceedings of the 7th German Workshop on Case-Based Reasoning (GWCBR)*, Wurzburg, Germany, pages 3–5, 1999.

[30] E. Plaza and L. McGinty. Distributed case-based reasoning. *The Knowledge Engineering Review*, 20(3):261–265, 2005.

[31] J. Nin, J. Herranz, and V. Torra. On the disclosure risk of multivariate microaggregation. *Data & Knowledge Engineering*, 67:399–412, 2008.

[32] A. Asuncion, and D.J. Newman UCI Machine Learning Repository [http://www.ics.uci.edu/ mlearn/MLRepository.html (Accessed: 8 April 2010)]. Irvine, CA: University of California, School of Information and Computer Science, 2007.

[33] O.L. Mangasarian and W.H. Wolberg Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, September 1990.

[34] C. Pous, P. Gay, A. Pla, J. Brunet, J. Sanz, and B. López. Modeling reuse on case-base reasoning with application to breast cancer diagnosis. In *D. Dochev, M. Pistore and P. Traverso (Eds.): Artificial Intelligence: Methodology, Systems, and Applications (AIMSA).* Springer, Berlin, LNAI 5253, pages 322–332, 2008.

[35] C. E. Jacobi, G. H. de Bock, B.Siegerink, and C. J. van Asperen. Differences and similarities in breast cancer risk assessment models in clinical practice: Which model to choose? *Breast Cancer Research and Treatment*, 115(2):381–390, 2009.

[36] A. Stahl and T. Roth-Berghofer. Rapid Prototyping of CBR applications with the open source tool myCBR. In *K.D. Atholff, R. Bergmann, M. Minor, and A. Hanft (Eds): Advances in Case-Based Reasoning Technology (ECCBR)*. Springer, Berlin, LNAI 5239, pages 615–630, 2008.

[37] I. H. Witten and E. Frank. *Data mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, CA, USA, 2005.

[38] C. Marling, J. Shubrook, and F. Schwartz. Case-based decision support for patients with type 1 diabetes on insulin pump therapy. In *K.D. Atholff, R. Bergmann, M. Minor, and A. Hanft (Eds): Advances in Case-Based Reasoning Technology (ECCBR)*, Springer, Berlin, LNAI 5239, pages 325–339, 2008.

[39] C. Pous. *Case-based reasoning as an extension of fault dictionary methods for linear electronic analog circuits diagnosis*. PhD thesis, University of Girona, Spain, 2004.

[40] J. Lieber, M. d'Aquin, F. Badra, and A. Napoli. Modeling adaptation of breast cancer treatment decision protocols in the KASIMIR project. *Applied Artificial Intelligence*, (28):261–274, 2008.

[41] O. Vorobieva and R. Schmidt. CBR investigation of therapy inefficacy. In *2nd Workshop on CBR in the Health Sciences*,Madrid, Spain 2004. http://www.cbr-biomed.org/jspw/workshops/ECCBR04.jsp (Accessed: June 17th 2009).

[42] D.M.J. Tax and R.P.W. Duin. Using two-class classifiers for multi-class classification. In *R. Kasturi, D. Laurendeau, C. Suen (Eds.): Proceedings 16th International Conference on Pattern Recognition (ICPR)*, IEEE Computer Society Press (Los Alamitos, California), Volume 2, pages 124–127, 2002.

[43] B. López, C. Pous, A. Pla, and P. Gay. Boosting CBR agents with genetic algorithms. In *L. McGinty, and D.C. Wilson (Eds): Case-Based Reasoning Research and Development, 8th International Conference on Case-Based Reasoning, ICCBR.* Springer, Berlin, LNAI 5650, pages 195–209, 2009.

[44] E. Armengol and E. Plaza. Relational case-based reasoning for carcinogenic activity prediction. *Artificial Intelligence Review*, 20(1–2):121–141, 2003.

[45] D. W. Aha, L. A. Breslow, and T. Maney. Supporting conversational case-based reasoning in an integrated reasoning framework. *Applied Intelligence*, 14:9–32, 1998.

[46] M. Lenz, A. Hëbner, and M. Kunze. Textual CBR. In *M. Lenz, B. Bartsch-Spörl, H.D. Burkhard and S.Wess (Eds.): Case-Based Reason-*

ing Technology: From Foundations to Applications. Springer, Berlin, Lecture Notes in Computer Science 1400, pages 115–138, 1998.

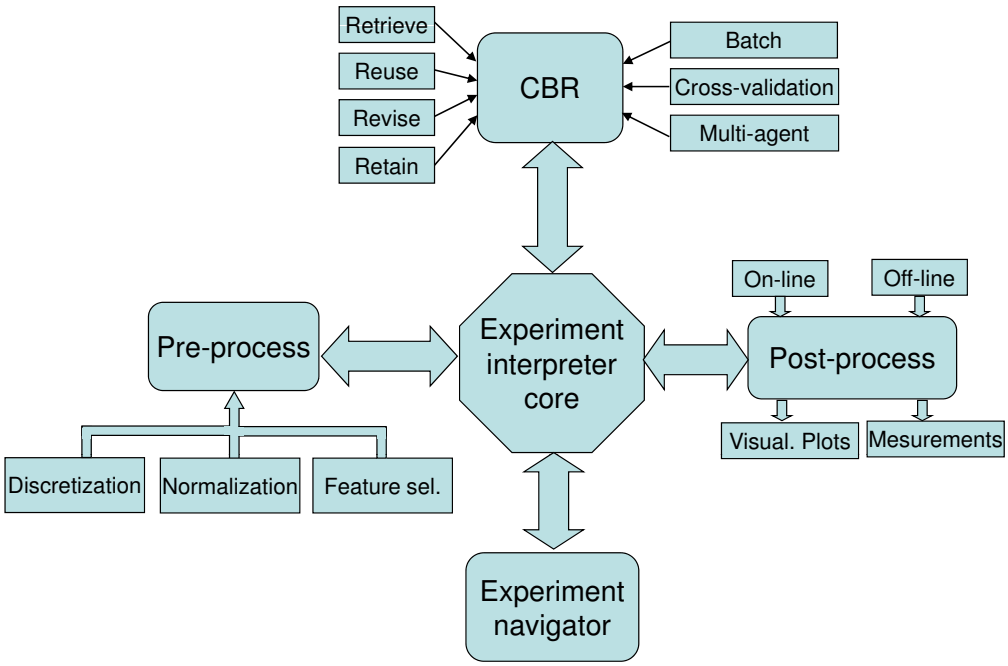

Figure 1: eXiT*CBR framework.

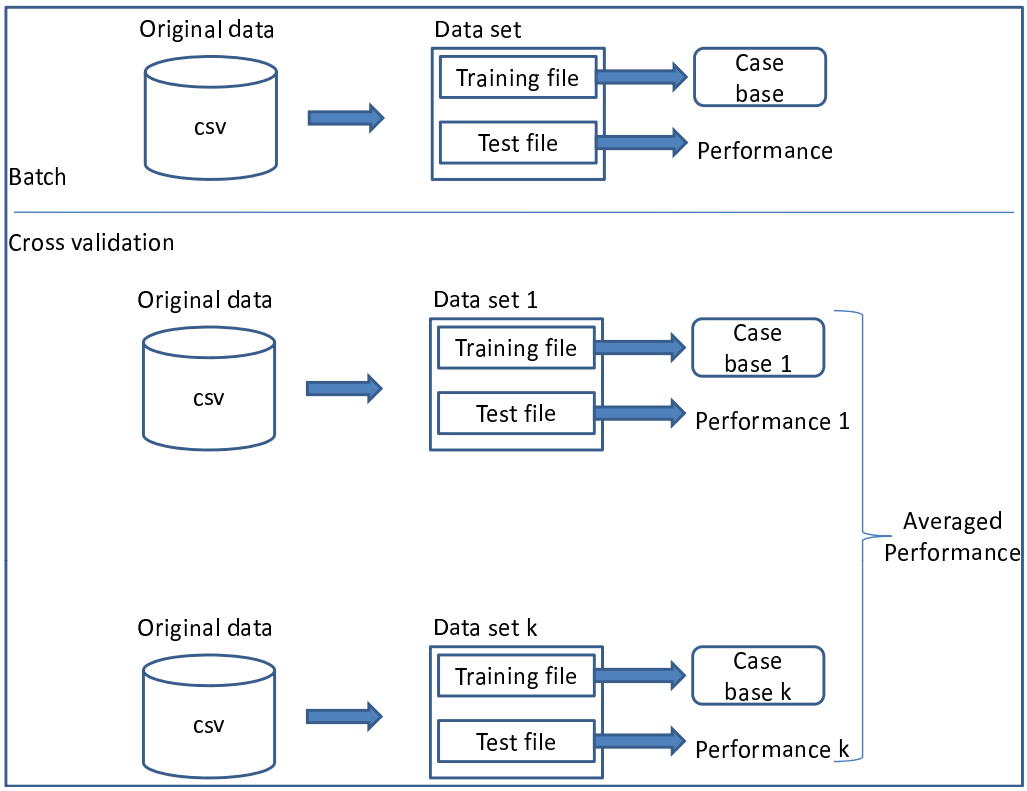

Figure 2: The two main experimental methods in eXiT*CBR.

```
######################################
### Configuration File  v 2.0     ####
######################################
# Data:
########
Directory = C:/Results
Name = Original
Data = /Datasets10
Class = A1_1_1#8
##############
# Pre-process:
##############
Discretization = EqualWidthIntervalBinning;B2_6#42/5.0
Normalization = MaxMin
AttributeSelection = null
##############
# CBR Model:
#############
Retrieve
    Retrieve_Unknown = SimpleAll
    Retrieve_Num_Local_Distance = Euclidean
    Retrieve_Cat_Local_Distance = Hamming
    Retrieve_Global_Distance = Average
    Retrieve_Selection = MajorityK;5
Reuse = ReuseProbabilisticPous;[0:0.1:1],0.7
Revise = null
Retain = Never
##################
# Experimentation:
##################
Measures = truePositiveFalsePositiveRates
Visualization = ROC
Method = CrossValidation
###############
# Post-process:
###############
Post-process = Off_line
### End
```

Figure 3: Configuration file.

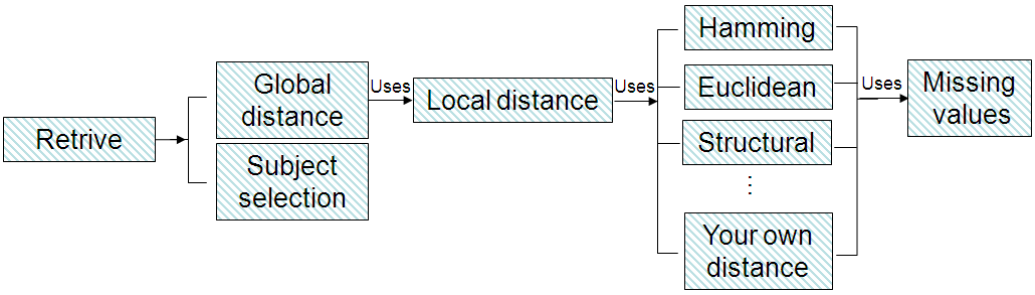

Figure 4: Retrieve modules.

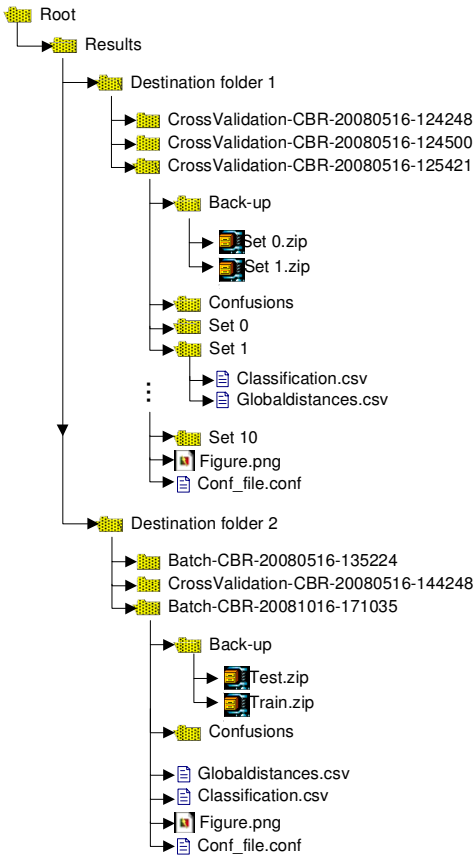

Figure 5: File structure.

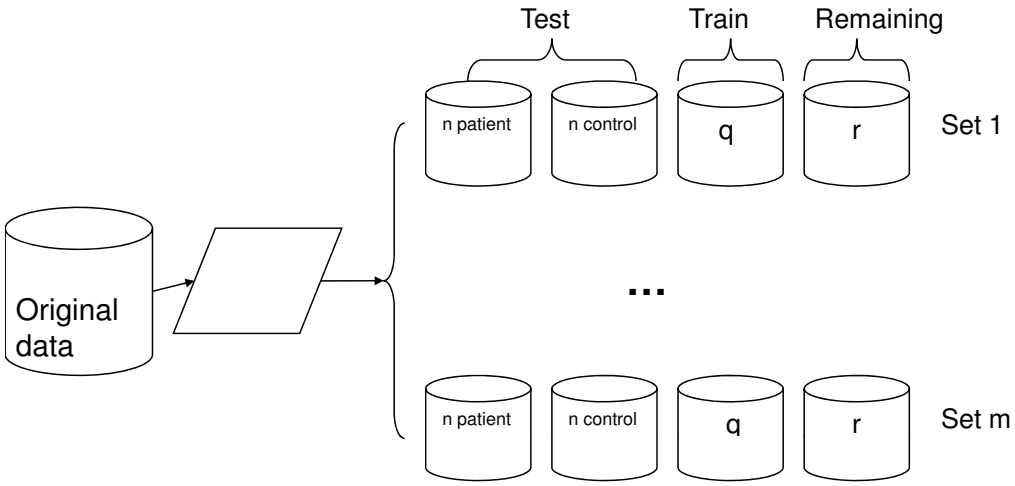

Figure 6: Datasets generated by stratified cross-validation.

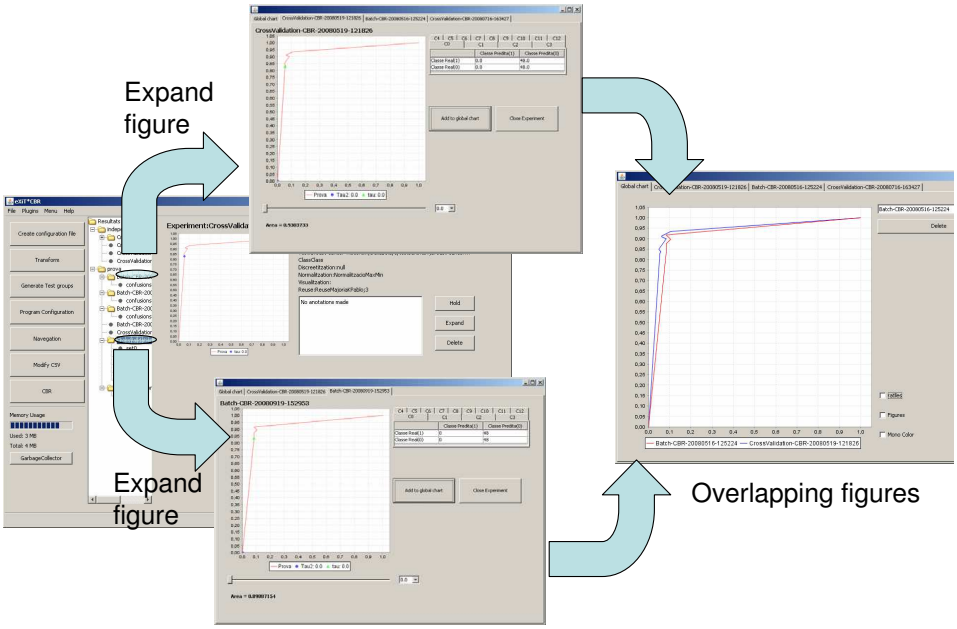

Figure 7: Expansion of different experiments and comparison.

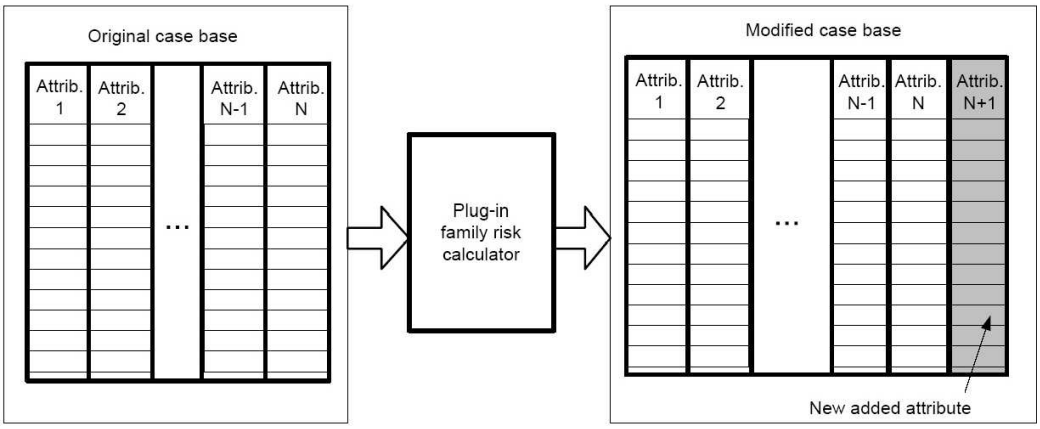

Figure 8: Integration of other techniques, such as plug-in, in eXiT*CBR.

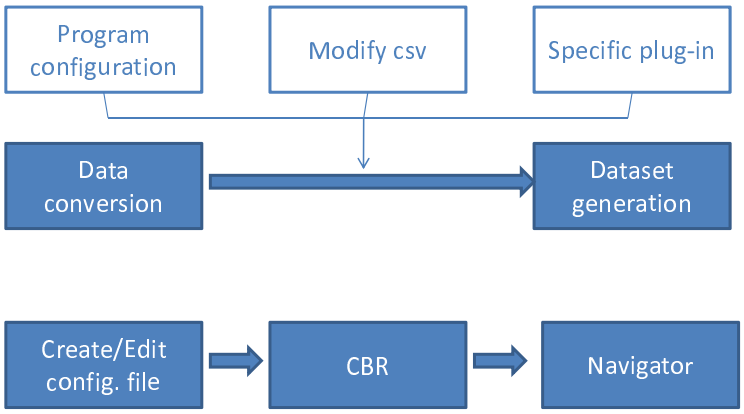

Figure 9: The main steps followed to develop the breast cancer application.

Figure 10: First graphical results obtained with the tool. The AUC value has been enlarged for clarity reasons.

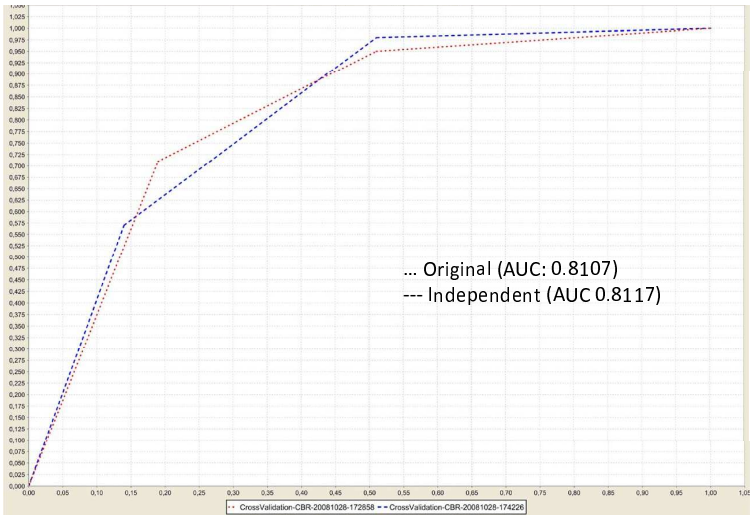

Figure 11: Comparison of two experiments through the experiment navigator enlargement facility.