# A Breast Cancer Classifier based on a Combination of Case-Based Reasoning and Ontology Approach

Essam Amin M.Lotfy Abdrabou — Ph.D Candidate Faculty of Computer and Information Sciences – Ain Shams University, Abbassia, 11566, Cairo, Egypt

AbdEl-Badeeh M. Salem — Professor Faculty of Computer and Information Sciences – Ain Shams University, Abbassia, 11566, Cairo, Egypt

## Abstract

*Breast cancer is the second most common form of cancer amongst females and also the fifth most cause of cancer deaths worldwide. In case of this particular type of malignancy, early detection is the best form of cure and hence timely and accurate diagnosis of the tumor is extremely vital. Extensive research has been carried out on automating the critical diagnosis procedure as various machine learning algorithms have been developed to aid physicians in optimizing the decision task effectively. In this research, we present a benign/malignant breast cancer classification model based on a combination of ontology and case-based reasoning to effectively classify breast cancer tumors as either malignant or benign. This classification system makes use of clinical data. Two CBR object-oriented frameworks based on ontology are used* jCOLIBRI *and* myCBR. *A breast cancer diagnostic prototype is built. During prototyping, we examine the use and functionality of the two focused frameworks.*

Key-Words: *Breast Cancer/diagnosis; Breast Cancer/classification; Breast Cancer/classification; Breast Cancer/therapy; Medical Informatics; Medical Informatics Applications.*

## Resumen

Un Clasificador de Cáncer de Mama basado en la combinación de un enfoque ontológico y de Razonamiento Basado en Casos

*Las neoplasias de la mama son el segundo tipo más común de cáncer entre las mujeres y la quinta causa de muerte por cáncer en todo el mundo. En el caso de este tipo específico de neoplasia, la detección temprana es la mejor forma de cura, y para ello el diagnóstico oportuno y preciso del tumor es extremadamente importante. Se han hecho muchos estudios en automación de procedimientos de diagnóstico crítico con el desarrollo de varios algoritmos de aprendizaje máquina, para ayudar a los médicos a optimizar la tarea de toma de decisión eficazmente. En este estudio, presentamos un modelo clasificador de cáncer de mama benigno/maligno basado en una combinación de ontología y razonamiento basado en casos para clasificar eficazmente tumores de cáncer de mama como malignos o benignos. Este sistema de clasificación utiliza datos clínicos. Se utilizan dos armazones orientados a objetos CBR basados en la ontología,* jCOLIBRI *y* CBR. *Se construyó un prototipo de diagnóstico de cáncer de mama. Durante el prototipaje, examinamos el uso y la funcionalidad de los dos armazones enfocados.*

Palabras-clave: *Neoplasias de la mama/diagnóstico; Neoplasias de la mama/clasificación; Neoplasias de la mama/diagnóstico; Informática médica; Aplicaciones de informática médica.*

## Resumo

Um Classificador de Câncer de Mama baseado em uma combinação da abordagem de Raciocínio Baseado em Casos e a ontologia

*O câncer de mama é o segundo tipo de câncer mais comum entre as mulheres e também a quinta causa mais comum de morte por câncer no mundo. No caso deste tipo específico de neoplasia, a detecção precoce é a melhor forma de cura e por isso, um diagnóstico oportuno e preciso do tumor é extremamente importante. Muitas pesquisas têm sido feitas na automação de procedimentos de diagnóstico critico com o desenvolvimento de vários algoritmos de aprendizagem de máquina, para auxiliar os médicos na otimização da tarefa de decisão de forma eficaz. Neste estudo, apresentamos um modelo de classificação de tumores de câncer de mama benigno/maligno. Este sistema de classificação utiliza dados clínicos. Foram utilizados dois sistemas CBR orientados a objetos baseados na ontologia,* jCOLIBRI *e* myCBR. *Foi construído um protótipo de diagnóstico de câncer de mama. Durante a prototipagem, examinamos o uso e a funcionalidade dos dois sistemas focados.*

Palavras-chave: *Neoplasias da mama/diagnóstico; Neoplasias da mama/classificação; Neoplasias da mama/ Informática médica; Aplicação de informática médica.*

## INTRODUCTION

Breast cancer classification, diagnosis and prediction techniques have been a widely researched area in the past decade in the world of medical informatics. Several articles have been published which tries to classify breast cancer data sets using various techniques such as fuzzy logic, support vector machines, bayesian classifiers, decision trees and neural networks. Classification accuracy as high as 98.8% has been achieved using a learning algorithm combining simulated annealing with the perceptron algorithm. Another study involving fuzzy modeling and cooperative co-evolution has gained an accuracy of 98,98% over one of the widely studied Wisconsin breast cancer database.[1]

This research applies a new technique in the field of breast cancer classification. It uses a combination of ontology and case-based reasoning by using ontology based object-oriented Case-Based Reasoning frameworks. Two frameworks are examined building the classifier. One is the open source *jCOLIBRI* system developed by *GAIA* group and provides a framework for building CBR systems based on state-of-the-art software engineering techniques. The other is the novel open source CBR tool *myCBR* developed at the German Research Center for Artificial Intelligence (*DFKI*). The objective of this classifier is to classify the patient based on his/her electronic record whether he/she is benign or malignant.

This paper is organized in four sections. Section 1 is this introduction. Section 2 gives a theoretical background about breast cancer, ontology, CBR and object-oriented frameworks. Section 3 illustrates the implementation of the breast cancer classifier on the two frameworks. Finally, section 4 discusses and concludes the results.

## THEORITICAL BACKGROUND

### Breast Cancer

Breast cancer is the form of cancer that either originates in the breast or is primarily present in the breast cells. The disease occurs mostly in women but a small population of men is also affected by it. Breast cancer is the most common form of cancer amongst the female population as well as the most common cause of cancer deaths. Early detection of breast cancer saves many thousands of lives each year. Many more could be saved if the patients are offered accurate, timely analysis of their particular type of cancer and the available treatment options. Since the breast tumors whether malignant or benign share structural similarities, it becomes an extremely tedious and time consuming task to manually differentiate them. There is no visually significant difference between the fine needle biopsy image of the malignant and benign tumor for an untrained eye.

Accurate classification is very important as the potency of the cytotoxic drugs administered during the treatment can be life threatening or may develop into another cancer. Laboratory analysis or biopsies of the tumor is a manual, time consuming yet accurate system of prediction. It is however prone to human errors, creating a need for an automated system to provide a faster and more reliable method of diagnosis and prediction for the patients.

### Ontology

Ontology is a formal explicit description of concepts in a domain of discourse (classes – sometimes called concepts), properties of each concept describing various features and attributes of the concept (slots – sometimes called roles or properties), and restrictions on slots (facets – sometimes called role restrictions). Ontology together with a set of individual instances of classes constitutes a knowledge base. In reality, there is a fine line where the ontology ends and the knowledge base begins.

### Case-Based Reasoning

In Case-Based Reasoning (CBR) systems expertise is embodied in a library of past cases, rather than being encoded in classical rules. Each case typically contains a description of the problem, plus a solution and/or the outcome. The knowledge and reasoning process used by an expert to solve the problem is not recorded, but is implicit in the solution. To solve a current problem: the problem is matched against the cases in the case base, and similar cases are retrieved. The retrieved cases are used to suggest a solution that is reused and tested for success. If necessary, the solution is then revised. Finally the current problem and the final solution are retained as part of a new case.

The CBR process can be represented by a schematic cycle, as shown in Figure 1.[3]

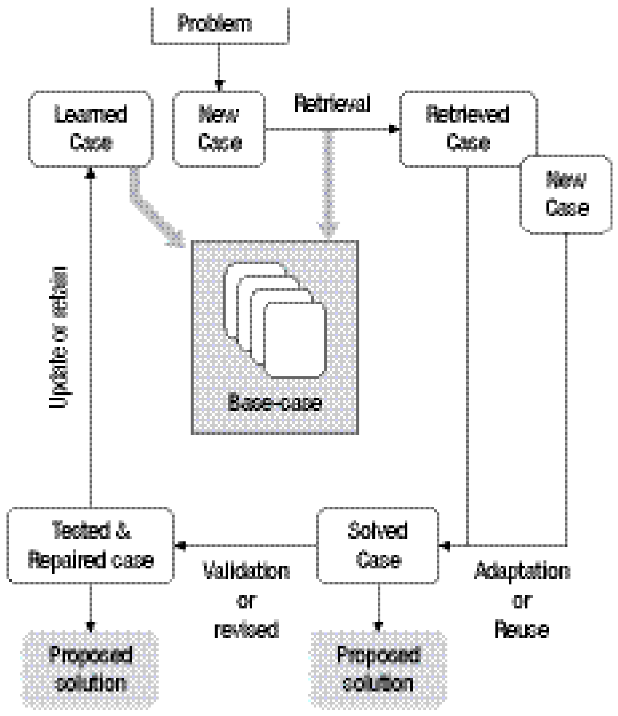


Figure 01 - The CBR Cycle.

Representation: Given a new situation, generate appropriate semantic indices that will allow its classification and categorization. This usually implies a standard indexing vocabulary that the CBR system uses to store historical information and problems. The vocabulary must be rich enough to be expressive, but limited enough to allow efficient recall.[4]

Retrieval: Given a new, indexed problem, retrieve the best past cases from memory. This requires answering three questions: What constitute an appropriate case? What are the criteria of closeness or similarity between cases? How should cases be indexed? Part of the index must be a description of the problem that the case solved, at some level of abstraction. Part of the case, though, is also the knowledge gained from solving the problem represented by the case. In other words, cases should also be indexed by some elements of their solution.[5]

Adaptation: Modify the old solutions to confirm to the new situation, resulting in a proposed solution. With the exception of trivial situations, the solution recalled will not immediately apply to the new problem, usually because the old and the new problem are slightly different. CBR researchers have developed and used various adaptation techniques. [5]

Validation: After the system checks a solution, it must evaluate the results of this check. If the solution is acceptable, based on some domain criteria, the CBR system is done with reasoning. Otherwise, the case must be modified again, and this time the modifications will be guided by the results of the solution's evaluation. [5]

Update: If the solution fails, explain the failure and learn it, to avoid repeating it. If the solution succeeds and warrants retention, incorporate it into the case memory as a successful solution and stop. The CBR system must decide if a successful new solution is sufficiently different from already-known solutions to warrant storage. If it does warrant storage, the system must decide how the new case will be indexed, on which level of abstraction it will be saved, and where it will be put in the case-base organization.[5]

Retaining the case is the process of incorporating whatever is useful from the new case into the case library. This involves deciding what information to retain and in what form to retain it; how to index the case for future retrieval; and integrating the new case into the case library.

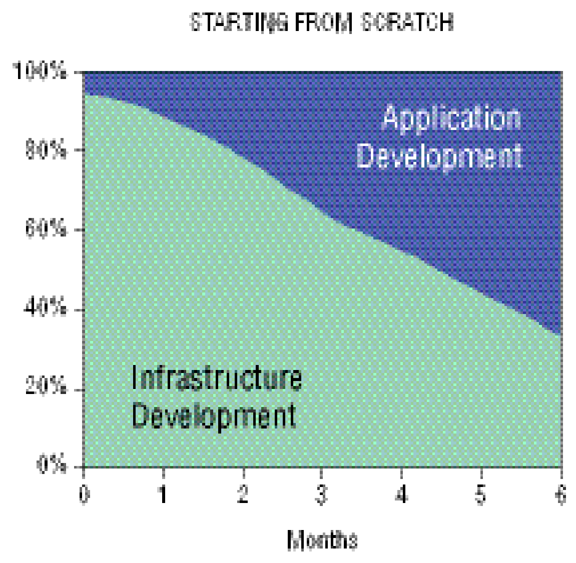
## CBR Object-Oriented Frameworks

The concept of object-oriented frameworks has been introduced in the late 80's and has been defined as "a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuses at a larger granularity than classes".

The goal of a framework is to capture a set of concepts related to a domain and the way they interact. In addition, a framework is in control of a part of the program activity and calls specific application code by dynamic method binding. A framework can be viewed as an incomplete application where the user only has to specialize some classes to build the complete application.[6]

Frameworks allow the reuse of both code and design for a class of problems, giving the ability to non-expert to write complex applications quickly. Frameworks also allow the development of prototypes which could be extended further on by specialization or composition. A framework once understood, it can be applied in a wide range of domain, and can be enhanced by the adding of new components. [6]

Using frameworks for development of new applications helps improve software quality. It improves programmer's productivity and quality, performance, and reliability of software. It also enhances extensibility by providing the required methods that allow applications to extend its stable interfaces.[20] Figure 2 clearly shows the difference of the effort required for developing an application from scratch and using a framework.[7]



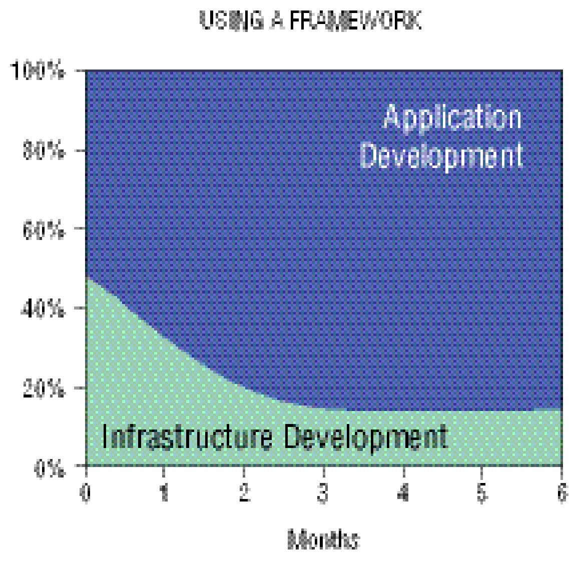


Figure 02 - Development Effort Reduction by using Frameworks.

CBR researchers agree that the best way to satisfy the increasing demand of developing CBR application is by development of frameworks. Recently, some efforts within the CBR community have developed CBR frameworks.[8] This paper focuses on two of them *jCOLIBRI* developed by *GAIA* group and *myCBR* developed by *DFKI* group.

## EXPERIMENTS

### Breast Cancer Classifications

Breast cancer has become the number one cause of cancer deaths amongst women. Once a breast cancer is detected, it can be classifieds benign (not cancerous tissue) or malignant (cancerous tissue). In this study, the two compared CBR frameworks are tested by developing a CBR application that classifies the condition of the breast cancer tumor whether it is benign or malignant. Wisconsin breast cancer data set was used for building the case-bases. It is obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.[9] Samples inside the data set arrive periodically as Dr. Wolberg reports his clinical cases. The number of instances inside the dataset is 699 (as of 15 July 1992). Each record contains ten attributes plus the class attribute. Table 1 shows the attributes and their possible values. 65.5% of the elements belong to the benign class and 34.5% to the malignant class. 16 elements are incomplete (an attribute is missing) and have been excluded from the database.

Table 1 - Wisconsin Breast Cancer Dataset.

| No. | Atribute | Possible value |
|---|---|---|
| 1 | Sample code number | id number |
| 2 | Clump Thickness | 1 – 10 |
| 3 | Uniformity of Cell Size | 1 – 10 |
| 4 | Uniformity of Cell Shape | 1 – 10 |
| 5 | Marginal Adhesion | 1 – 10 |
| 6 | Single Epithelial Cell Size | 1 – 10 |
| 7 | Bare Nuclei | 1 – 10 |
| 8 | Bland Chromatin | 1 – 10 |
| 9 | Normal Nucleoli | 1 – 10 |
| 10 | Mitoses | 1 – 10 |
| 11 | Class | (2 for benign, 4 for malignant) |

### jCOLIBRI

#### Overview

*jCOLIBRI* is an evolution of the COLIBRI architecture[10], that consisted of a library of Problem Solving Methods (PSMs) for solving the tasks of a knowledge-intensive CBR system along with ontology, CBROnto[11], with common CBR terminology. COLIBRI was prototyped in LISP using LOOM as knowledge representation technology. This prototype served as proof of concept; was very useful but it is not helpful for non-expert users. Then, people at *GAIA* group have started to develop a new complete framework with the name of *jCOLIBRI*. It stands for Cases and Ontology Libraries Integration for Building Reasoning Infrastructures. CBR ontology assumes the same vocabulary provided by any CBR system. In *jCOLIBRI*, ontology is not represented as a new source. All concepts of CBR are mapped into classes and interfaces of framework. Classes that represent the concept of ontology serve as templates where new CBR types should be added. They also provide the tasks and abstract interface of the methods.

The design of the *jCOLIBRI* framework comprises a hierarchy of Java classes plus a number of XML files. The framework is organized around the following elements[4]:

- *Tasks and Methods*: The tasks supported by the framework and the methods that solve them are all stored in a set of XML files.
- *Case Base*: Different connectors are defined to support several types of case determination, from the file system to a database.
- *Cases*: A number of interfaces and classes are included in the framework to provide an abstract representation of cases that support any type of actual case structure.
- *Problem Solving Methods*: The actual code that supports the methods included in the framework.

The *jCOLIBRI* comes in two major releases version 1 and version 2. According to the tutorial[12], version 2 is a new implementation that follows a new and clear architecture divided into two layers: one oriented to developers and other oriented to designers. Unfortunately, the only available distribution of version 2 is the one that is oriented to the developers which is out of scope of this paper. *jCOLIBRI* version 1 is the first release of the framework. It includes a complete Graphical User Interface (GUI) that guides the user in the design of a CBR system. This version is recommended for non-developer users that want to create CBR systems without programming any code which is exactly the scope in this study. As a result, version 1 is selected to implement the required application.

Downloading of the *jCOLIBRI* is an easy task; it can be obtained through the web page of *GAIA* group. It comes in a compressed distribution that can be easily extracted to have the full package.

To run *jCOLIBRI*, there is a ready batch file (we are using *MS® Windows* platform) that can be invoked directly to run *jCOLIBRI*. It is required to have *JAVA®* Virtual Machine installed before running the batch file. By invoking this batch file we get the first screen of the framework GUI.

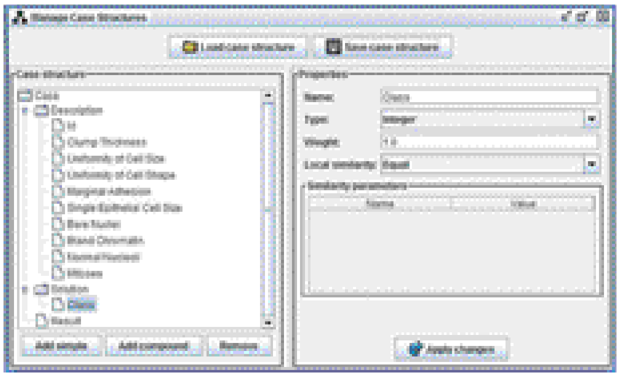
## Implementation

By the help of the multimedia tutorials provided and the GUI of the *jCOLIBRI*, users can go through five steps to implement and deploy a CBR System. These steps are

- Definition of case structures;
- Building the case-base;
- Managing similarity measures;
- Configuring the behavior of the CBR process;
- Testing and deploying the CBR application.

## Definition of Case Structures

By using *jCOLIBRI* GUI users are able to create the case structure defining simple and compound attributes that describe the cases together with their types, weights, similarity measure – that is chosen from a library of existing similarity functions and parameters. The case structure can be saved/ loaded in/ from a XML file. Figure 3 shows the definition of the patient case parameters.



Name: Class
Type: Integer
Weight: 1.0
Local similarity: Equal

Figure 03 - Patient Case Definition in *jCOLIBRI*.

## Building the Case-Base

*jCOLIBRI* introduces the concept of connectors which cases persistence is built around. Connectors are objects that know how to access and retrieve cases from the storage media and return those cases to the CBR system in a uniform way. Therefore connectors provide an abstraction mechanism that allows users to load cases from different storage sources in a transparent way.[13,14] Defined connectors can work with plain text files, XML files, or relational data bases. The graphical interface helps mapping the defined case structure with the tables and columns from the storage scheme. Figure 4 shows how the patient case structure is mapped to columns in a text file containing the Wisconsin data set patient records.

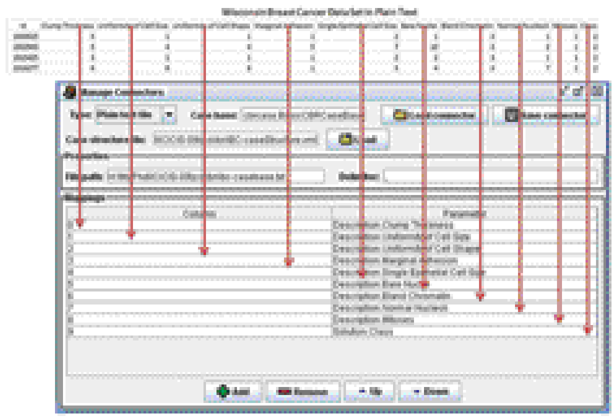


Figure 04 - Managing Connectors in *jCOLIBRI*.

### *Managing Similarity Measures*

When two cases are compared, the local similarity functions are used to compare simple attribute values. Global similarity functions are linked to compound attributes and are used to gather the similarities of the collected attributes in a unique similarity value. At last, the similarity value of two cases is computed as the similarity of their description concepts. The available similarity measures are listed in a configuration file, and can be managed using the GUI. Since our problem is simple, we leave the default similarity assigned by *jCOLIBRI*.

### *Configuring the Behavior of the CBR Process*

As introduced, *jCOLIBRI* formalizes the CBR knowledge using CBR ontology (CBROnto), a knowledge level description of the CBR tasks and a library of reusable Problem Solving Methods (PSMs).[13]

Configuration of tasks is done in an interactive approach by choosing from a library of reusable methods one that is suitable to solve the selected task. Constraints of the selected task are being tracked during the configuration process so that only applicable methods in the given context are offered to users. In our comparison we focus only on the retrieval task. Figure 5 shows the configured tasks in the breast cancer application.

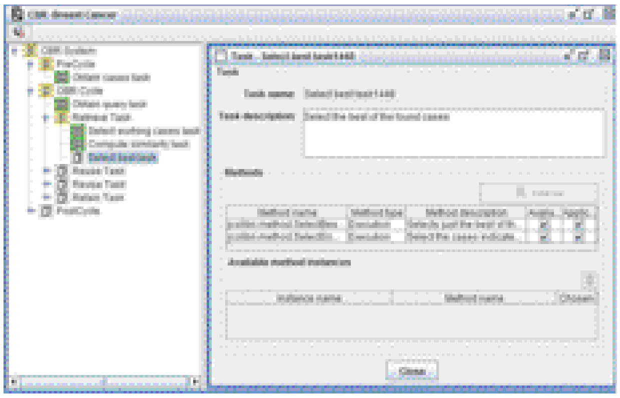


Figure 05 - Configuration of Tasks in *jCOLIBRI*.

### *Testing and Deploying the CBR Application*

The CBR application is finished when all the tasks have been configured. Users can test the system from inside the graphical interface. The first task of the CBR system, (Obtain query task), obtains the query that is going to be used to retrieve the most similar cases. Figure 6 shows the GUI after a query. We tested the 16 records that are excluded from the dataset according to one missing value. Only two missed classifications are obtained.

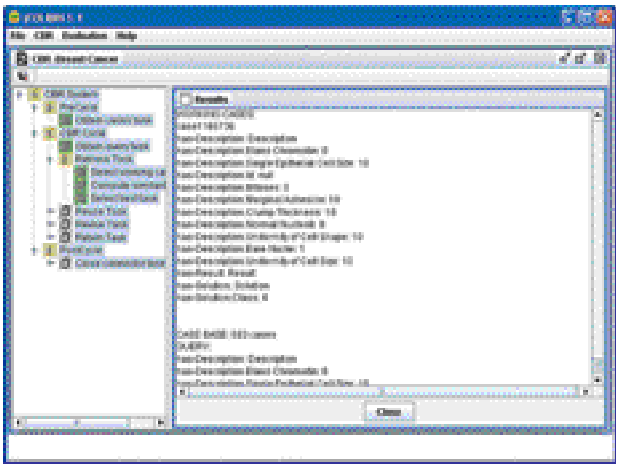


Figure 06 - *jCOLIBRI* Retrieval.

Documentation mentions that it is possible to deploy the developed CBR application by generating a code template with most of the code required to run the developed system as an independent application. We have tried this process but it is completely failed.

## myCBR

### Overview

*myCBR* is an open-source plug-in for the open-source ontology editor *Protégé*. *Protégé* is based on Java, is extensible, and provides a plug- and-play environment that makes it a flexible base for rapid prototyping and application development.[15] *Protégé*[15] allows defining classes and attributes in an object-oriented way. Furthermore, it manages instances of these classes, which *myCBR* interprets as cases. [16] So the handling of vocabulary and case base is already provided by *Protégé*. The *myCBR* plug-in provides several editors to define similarity measures for an ontology and a retrieval interface for testing.[14]

As the main goal of *myCBR* is to minimize the effort for building CBR applications that require knowledge-intensive similarity measures, *myCBR* provides comfortable GUIs for modeling various kinds of attribute specific similarity measures and for evaluating the resulting retrieval quality. In order to reduce also the effort of the preceding step of defining an appropriate case representation, it includes tools for generating the case representation automatically from existing raw data.[16] The novice as well as the expert knowledge engineer are supported during the development of a *myCBR* project through intelligent support approaches and advanced GUI functionality. [16]

Downloading *myCBR* requires two steps of downloading. The first is to download *myCBR* plug-in files; this can be done directly through *myCBR* web page. The second step is to download the *Protégé* ontology editor; this can be done through the *Protégé* web page. Downloading *Protégé* is not an easy task. Users need to do some readings on the site to be able to select the suitable version to download.

Since *myCBR* is a plug-in inside *Protégé*, users need to install *Protégé* first. It is required to have *JAVA®* Virtual Machine installed before proceeding in installation, or users may choose to download the version that includes the *JAVA®*. To install the *myCBR* plug-in for *Protégé*, users need to copy the *myCBR* plug-ins into *Protégé*'s plug-ins directory. Then to start *Protégé* and create new projects, users need to enable the *myCBR* plug-ins from the configuration menu of *Protégé*. After installing and activating the *myCBR* plug-in, the user interface of *Protégé* is extended with additional tabs to access the *myCBR* modules.

After developing a CBR application using the *Protégé* plug-in, *myCBR* can also be used as a stand-alone Java module, to be integrated in arbitrary applications, for example, JSP5-based web applications. In this application phase, the retrieval engines of *myCBR* just read the XML files of the created project generated using the plug-in interface and perform the similarity-based retrieval.[14]

For *Protégé* manuals and tutorial, users may consult the documentation section of the *Protégé* web site for available documentation. Among other things, users may find the *Protégé* User's Guide, a "getting started" tutorial, and information on ontology development.

The manual for *myCBR* is available on its web page as *HTML* version or a *PDF* version. The manual covers installation and different usage issues. No multimedia tutorials are available for the usage of *myCBR*.

### Implementation

Four steps are required to develop a CBR application:

- Generation of case representations
- Modeling similarity measures
- Testing of retrieval functionality
- Implementation of a stand-alone application

### Generation of Case Representations

One powerful feature provided by *myCBR* is the easiness of the case representation by *CSV* data import module.[14] Users have the choice to import data instances in an existing *Protégé* class or to create a new class that is suitable for their raw data. Figure 7 shows how Wisconsin dataset is arranged in a *CSV* file.

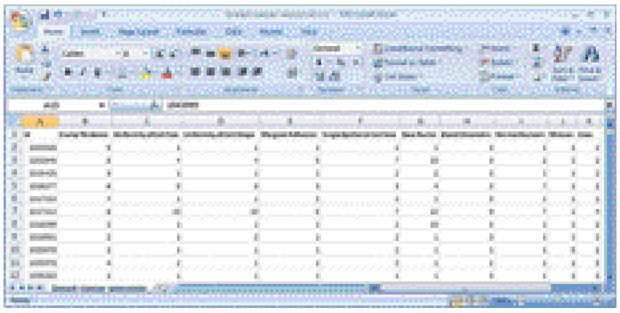


Figure 07 - Wisconsin Dataset in a *CSV* File.

The *myCBR* allows also slots to be added manually using *Protégé*. Figure 8 shows *myCBR* screen after importing the dataset into a new class Patient which will be used as query and case values for retrieval step.

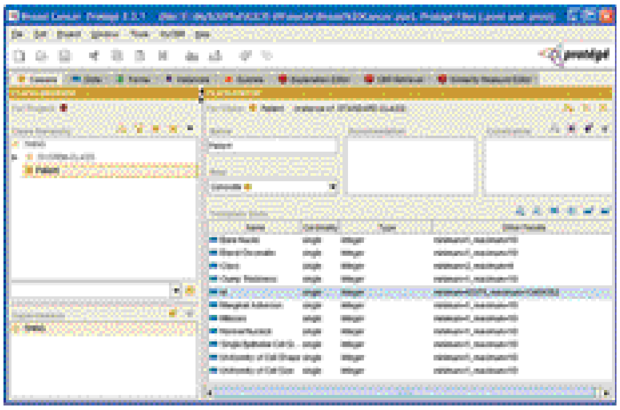


Figure 08 - Patient Case Data Representation in *myCBR*.

### Modeling of Similarity Measure

The *myCBR* follows the local-global approach which divides the similarity definition into a set of local similarity measures for each attribute, a set of attribute weights, and a global similarity measure for calculating the final similarity value.

The dataset used in this experiment is simple so we leave the similarity measure definition as the default of *myCBR*. We only change the weight values of the Id and Class slots from one to zero. However, users may consult *myCBR* tutorial for more options in defining local and global similarity measure.

### Testing of Retrieval and Explanation

The *myCBR* includes an easy to use GUI for performing retrievals and for analyzing the corresponding results. By providing similarity highlighting and explanation functionality, *myCBR* supports the efficient analysis of the outcome of the similarity computation. We tested the 16 records that are excluded from the dataset according to one missing value. Only two missed classifications are obtained. Figure 9 shows one query of these records after retrieving the most similar cases.

Another alternative of performing case retrieval is to use a query from cases. This is also tested and gives a similar result as shown in Figure 10.

### Implementation of Stand-Alone Application

The *myCBR* can also be used as a stand-alone Java module, to be integrated in arbitrary applications. In this application phase, the retrieval engines of *myCBR* just read the XML files of the created project generated using the plug-in interface and perform the similarity-based retrieval. Figure 10 shows the breast cancer stand-alone application.

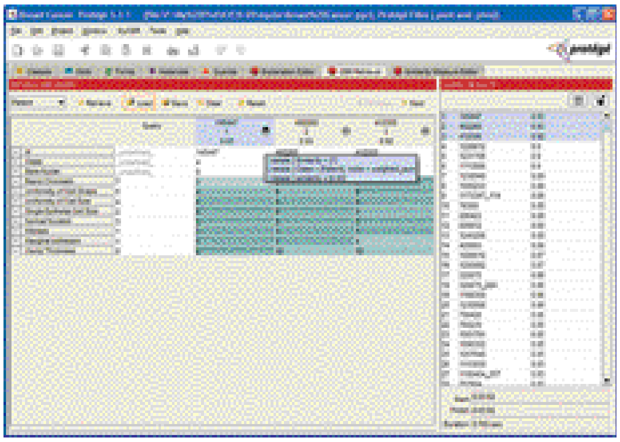


Figure 09 - Retrieval of a Case Query with a Missing Attribute Value.

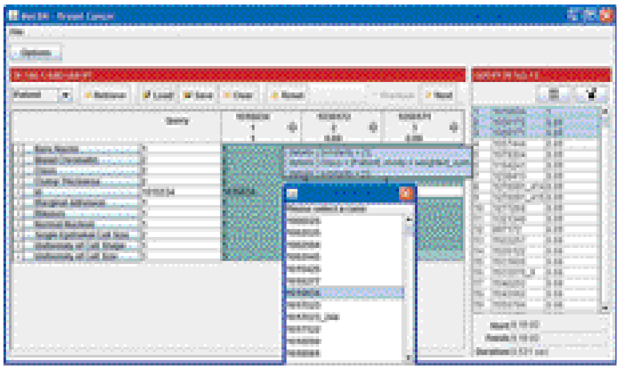


Figure 10 - Breast Cancer as a Stand-Alone Application.

## DISCUSSION AND CONCLUSION

In this paper, we examined two object-oriented ontology based CBR frameworks *jCOLIBRI* developed by *GAIA* group and *myCBR* developed by *DFKI* group. A breast cancer classifier is built by using the two selected frameworks.

During the implantation of the breast cancer diagnostic application using *jCOLIBRI* we found that *jCOLIBRI* is user-friendly and efficient to develop a quick application. The classifier was successful in classification of the selected data set.

During the implantation of the breast cancer classifier using *myCBR* we noticed that *myCBR* is a really a tool for rapid prototyping of a new CBR application. In seconds, users may have a running standalone CBR application by using the CSV importing feature. *myCBR* is intelligent enough to build the case structure and the case base by parsing the provided CSV file. *myCBR* avoids reinventing the wheel by making the development of a new CBR application done inside Protégé. The classifier was successful in classification of the selected data set.

In conclusion, two CBR frameworks are very useful to develop CBR base breast cancer classifier that can play a very important role to help for early detecting the disease and hence right medications can be used to save lives.

## REFERENCES

1. Pena-Rayes CA, Sipper M. Applying Fuzzy CoCo to Breast Cancer Diagnosis.In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on IEEE. 16 Jul 2000 - 19 Jul 2000. La Jolla, CA , USA: IEEE Xplore; 2000. v.2, p.1168-75. Digital Object Identifier: 10.1109/CEC.2000.870780
2. Sewak M, Vaidya P, Chan CC, Duan ZH. SVM Approach to Breast Cancer Classification. IMSCCS. 2007; 2:32-7.
3. Aamodt A, Plaza E. Case-based reasoning: foundational issues, methodological variation and system approaches. AICOM. 1994; 7(1):39-58.
4. Bello-Tomás JJ, González-Calero PA, Díaz-Agudo B. JCOLIBRI: An Object-Oriented Framework for Building CBR Systems. Advances in Case-Based Reasoning. Lect Notes Computer Scie. 2004; 3155: 32-46.
5. Kolodner JL. Case-Based Reasoning. California: Morgan Kaufmann Publishers; 1993.
6. Jaczynski M, Trousse B. An Object-Oriented Framework for the Design and the Implementation of Case-Based Reasoners. In: Proceedings of the 6th German Workshop on Case-Based Reasoning. Berlin; 1998.
7. Mulder A. Developing a Reusable Application Framework. Hariot Solutions. [Cited 2010 nov. 15]. Available from: http://www.chariotsolutions.com/javalab/presentations.jsp, 2003
8. Recio-García J, Díaz-Agudo AB, Sánchez A, González-Calero PA. Lessons learnt in the development of a CBR framework. In: Petridis M, editor, Proccedings of the 11th UK Workshop on Case Based Reasoning. Greenwich: CMS Press, University of Greenwich; 2006. p. 60–71.
9. Mangasarian OL, Wolberg WH. Cancer diagnosis via linear programming. SIAM News. 1990; 23(5):1-18.
10. González-Calero JA, Díaz-Agudo B. An architecture for knowledge intensive CBR systems. In: Blanzieri E, Portinale L, editors. Advances in Case-Based Reasoning– (EWCBR'00). Berlin: Springer-Verlag; 2000.
11. González-Calero PA, Díaz-Agudo B. CBROnto: a task/method ontology for CBR. In: Haller S, Simmons G, editors, Proccedings of the 15th International FLAIRS'02 ConferenceMenlo Park, CA: AAAI Press; 2002 . Special Track on CBR, 101–106.
12. Recio-García JA, Bridge D, Díaz-Agudo B, González-Calero PA. CBR for CBR: A Case-Based Template Recommender System. In: Althoff K-D, Bergmann R, editors. Advances in Case-Based Reasoning, 9th European Conference, ECCBR 2008. LNCS. Springer. In Press.
13. Recio-García JA, Sánchez A, Díaz-Agudo B, González-Calero PA. jCOLIBRI 1.0 in a nutshell. A software tool for designing CBR systems. In: Petridis M, editor. Proccedings of the 10th UK Workshopon Case Based Reasoning, 2005, 20-28. Greenwich: CMS Press, University of Greenwich; 2005.
14. Stahl A, Roth-Berghofer TR. Rapid prototyping of CBR applications with the open source tool myCBR. In: Bergmann R, Altho KD, editors. Advances in Case-Based Reasoning. Berlin: Springer Verlag; 2008.
15. Bogaerts S, Leake D. A Framework for rapid and modular Case-Based Reasoning System Development. Bloomington, In: Computer Science Department, Indiana University; 2005. Technical Report TR 617.
16. Roth-Berghofer TR, Bahls D. Explanation capabilities of the open source case-based reasoning tool myCBR. 2008.
17. [Cited 2010 nov. 15]. Avalilable from: http://mycbr-project.net/download.html
18. Bogaerts S, Leake D. Increasing AI Project Effectiveness with Reusable Code Frameworks: A Case Study Using IUCBRF. Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference, 2005, 2-7, Menlo Park, CA: AAAI Press; 2005.
19. Díaz-Agudo B, González-Calero PA, Recio-García J, Sanchez-Ruiz A. Building CBR systems with jCOLIBRI. J Scie Comput Program. 2007; 69(1-3):68-75.
20. Gennari JH, Musen MA, Fergerson RW, Grosso WE, Crubezy M, Eriksson H, Noy NF, Tu SW. The evolution of Protege an environment for knowledge-based systems development. Int J Hum Comput Stud. 2003; 58(1):89-123.
21. Johnson R, Foote B. Designing reusable classes. J Object-Oriented Program. 1988; 1(5):22-35.
22. Leake D. Case Based Reasoning. Experiences, Lessons and Future Directions. Menlo Park, CA: AAAI Press, MIT Press, USA; 1997.
23. Manago M, Bergmann R, Conruyt N, Traphner R, Pasley J, Le Renard J, et al. CASUEL: a common case representation language. ESPRIT project 6322, 1994. Task 1.1, Deliverable D1. Kaiserslautern: University of Kaiserslautern; 1994.
24. Recio-García JA, Díaz-Agudo B, González-Calero PA. Prototyping recommender systems in jCOLIBRI. In: Proceedings of the 2008 ACM Conference on Recommender Systems (Lausanne, Switzerland, October 23 - 25, 2008). RecSys '08. New York, NY: ACM; 2008. p. 243-50.
25. Recio-García JA, Díaz-Agudo B, González-Calero PA. jCOLIBRI2 Tutorial, 2008. Group of Artificial Intelligence Application (GAIA). Madrid: University Complutense of Madrid; 2008. Document Version 1.2.
26. Schulz S. CBR-Works: A state-of-the-art shell for case-based application building. In: Melis E, editor. Proceedings of the 7th German Workshop on Case-Based Reasoning, GWCBR'99, Wurzburg, Germany; University of Wurzburg, 1999. p. 166-75.