# A Case-based Collaborative Knowledge acquisition Tool

**Mahmoud Rafea**
*Central Lab. for Agricultural Expert System CLAES*
*mahmoud@claes.sci.eg*

Abstract: We describe a fault tolerant client-server application that supports collaboration between knowledge engineers for developing case-based expert systems. The server holds knowledge bases and performs cases acquisition-verification, cases maintenance, cases reasoning, multimedia acquisition, and security checking. The clients are a number of specialized components which have the following functionalities: creating new knowledge base, acquiring cases, attaching multimedia files to cases, defining users for security checking, and deducting solutions using generic reasoning component for the purpose of application testing. 

## 1. INTRODUCTION

The issue of collaboration is very important in complex knowledge bases. In such knowledge bases, the domain is multi-disciplinary with the involvement of a group of domain experts and a group of knowledge engineers. Using such a collaborative tool will enhance productivity and quality of final application. This is because it will not only enable acquiring knowledge in parallel, but also enable reuse of shared concepts, and ensure consistency through online verification. Further, the ability of using Internet/Intranet to access a knowledge-base-server is very useful; domain experts can review knowledge remotely; and working location is selected flexibly.

The case-based reasoning model that has been used is described in [Rafea, 1995]. This model is re-implemented using Mozart programming system [Mozart Consortium, 1999]. The Mozart programming system has been used because it completely separates application functionality from distribution structure, and provides primitives for fault-tolerance, open computing, and partial support for security [Haridi & Franzén, 1999]. Also, the language library contains useful modules for a number of techniques of distributed programming and fault-tolerant programming, beside a multitude of useful abstractions, such as cached objects, stationary objects, fault-tolerant stationary objects, mobile agents, and fault-tolerant mobile agents, and facilitates developing of new abstractions [Haridi, et al, 1998].

In this article, we describe the implemented tool. In the next section, the research goal, objectives and motivation are stated. In section 3, the overall tool architecture is described. The functionalities supported by the tool are delineated, in section 4. Finally, hints about deployment of final application are given.

## 2. GOAL, OBJECTIVES AND MOTIVATION

The research goal of this work is to overcome the problem of knowledge acquisition and verification (KAKV) in complex domains and/or when more than one person is involved in encoding a knowledge-base. The problem of KAKV is worse in complex domains due to difficulties in knowledge management. To achieve this goal, the first and most important objective, is to enable sharing of acquired knowledge between different participants involved in the knowledge acquisition process so that each authorized participant can add, edit, and/or review a piece of knowledge that is acquired by him or by others. This prevents surprises which usually appear when integration is done later.

The second objective is to automate KAKV process. In order for automation to be successful, we found that:

- Knowledge Representation (KR) should be suitable to domain features. For instance, in clinical diagnosis, a case is suitable as a model, while in plant diagnosis and engineering fault diagnosis, rules are better. This is because, in the former, diagnosis usually leads to a single disorder (or syndrome), but in the latter, diagnosis involves different plants/faults, which may lead to explosion in the number of cases which represent combination of disorders.
- Communication Model (CM) should be suitable to KR. This, not only facilitates knowledge encoding, but also help the development of efficient system. Examples of CMs that we usually use in tools developed in our laboratory are formulae, tables, frames (a concept with facets), rules, cases, etc.
- Details of KR are better to be hided so that users can concentrate on the quality of knowledge without being distracted by syntax.

The motivation is to develop clinical expert systems in the veterinary domain. CBR have been used after several trials in which different knowledge engineering (KE) methodologies have been used. It should be remarked that the reason of using CBR is the selection of the associated CM by a consortium of domain experts and knowledge engineers. The KE methodologies that have been tried are Hierarchical Classification Generic Task [Gomez and Chandrasekaran 1981; Mittal, 1980]., and common-KADS methodology [Waern et al. 1993].

## 3. ARCHITECTURE

The tool consists of a set of components with client-server architecture. The server holds knowledge bases and performs cases acquisition-verification, cases maintenance, cases reasoning, multimedia acquisition, and security checking. The clients are specialized for the following functionalities: creating new knowledge base, acquiring cases, attaching multimedia files to cases, defining users for security checking, and deducting solutions using generic reasoning component for the purpose of application testing. In fact, clients are shadows of server components. So, clients have only communication models: one to communicate with a server, and the other to communicate with users.

### *3.1 Server architecture*

The server is designed to support development of more than one expert system application in the same time. Consequently, the server creates a thread for each application. The client communicates with the created thread through a ticket provided by the server. The conceptual architecture of the server is depicted in figure 1.

### *3.2 Knowledge base architecture*

A knowledge base consists of three representations. The first is a table of properties. Properties represent a simplified ontology of terms. We define a property as an ordered-pair: (name, value). The table consists of two fields. The first field is the property-name. Sometimes we refer to property-name with the word attribute. The second field is a set of property-legal-values which are the possible values an order-pair can have.

The second representation is cases hierarchical tree. First we need to define what a case is. A case can be defined as a function that maps two sets of properties. The first set represents specifications of a problem and the second set represents specifications of a solution to this problem. One of the solution property-names is special because it is used to group cases together in a cluster. It is called the cluster's determining property (CDP). The top node of cases hierarchical tree declares these specifications so that it can be used by the KAKV algorithm and the reasoning algorithm. Nodes of the second level contain CDP. Hence they are called cluster nodes and represent all possible solutions in the problem domain. Nodes of next levels are called case nodes and contain: case properties and any other case-specific solution properties. Figure 2 depicts the graphical representation of the knowledge base.

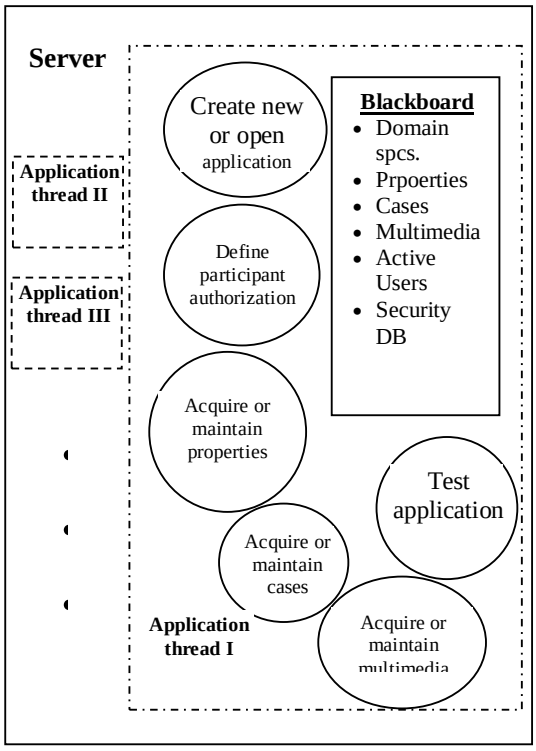


Figure 1: Conceptual server architecture

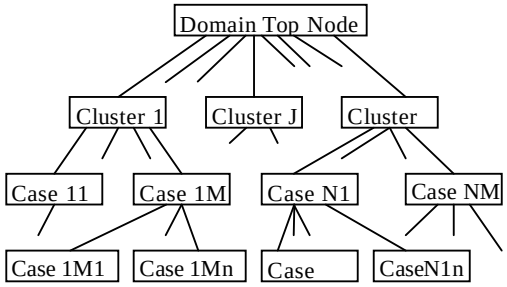


Figure 2: Knowledge architecture

Notice that all descendants inherit CDP. Consequently, for each value of this particular

property, cases cluster is created. This means that the number of clusters, in a particular domain, is the number of possible solutions. For example, if the domain is the diagnosis of some disorders, then the number of clusters is the number of disorders. Accordingly, the diagnosis property is the cluster's determining property while, for instance, treatment property will be related to a particular case or group of cases that can inherit this property. In this way, any real-world case complexity can be mapped easily to this representation.

The hierarchical organization of cases depends on the subset relation. Case properties, in a leaf node, must represent a proper subset of case properties in a super node. Therefore a leaf case is subsumed in its branch super cases. Furthermore, a node to the left (Figure 2) may be subsumed in a right sibling. This happens when the left case is acquired first. This feature is the basis for both indexing of cases and reasoning search algorithm.

There are two anomalies that can occur while acquiring a new case. The first anomaly is "redundancy" which means that the new case already exists. The second anomaly is "conflict". Conflict occurs when properties that will be stored as problem specifications are similar to properties in an already existing case-node, while properties that will be stored as solution specifications are different. The verification mechanism uses the reasoning algorithm to search for anomalies. The idea of the reasoning algorithm is to parse the knowledge-base tree until finding the best matching case. The parsing is directed using subset relation while a case-node is found through a mechanism for index generation. The algorithm details can be found in [Rafea, 1995].

The third representation is multimedia files associated with cases. The case-index is used, also, as a key in multimedia tables. The picture files associated with a particular case are indexed by that case-index. Similarly, video clips files are stored in the same way. Notice that cases also inherits multimedia associations from their super-case.

## 4. CLIENTS COMPONENTS

Client components communicate with the server through a fault-tolerant TCP/IP link. The fault tolerant connection is important because the Internet link can go down during a session and restart without affecting intermediate results of a client's session and a client may shutdown or crash without affecting a server performance with respect to ongoing sessions of other clients. Technically faults are detected, abstracted and reflected to a component-level using an orthogonal mechanism for lazy fault-detection and handling.

### *4.1 Tool manager*

The tool manager is designed to help application developer to administrate a number of expert system applications. The manager component, figure 3, is used either to run a server component on a server-machine, or retrieve a handle of an already remotely running server. Selecting a server handle will enable a user to run clients that communicate with a selected server.

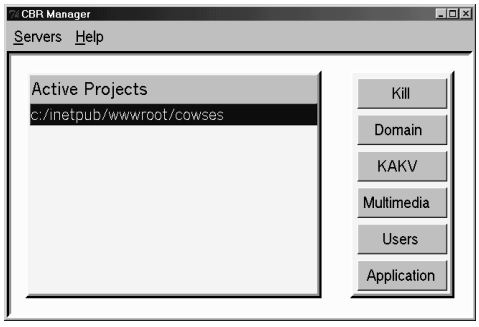


Figure 3: Manager dialogue showing one active project server: "c:/inetpub/wwwroot/cowses"

### *4.2 KAKV component*

The KAKV component is illustrated in figure 4. A developer can define or maintain cases-properties (e.g., symptoms and signs). The case properties are used to construct cases. A case consists of Conclusion-properties (solution properties), and problem properties which are captured as And-findings, and Or-findings. For instance, a problem properties of: (A And B) And (C Or D) generates two cases: (A And B And C) and (A And B And D).

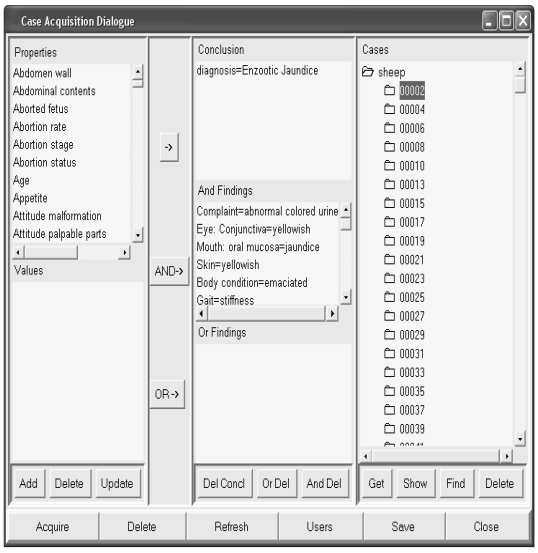


Figure 4: Case acquisition and verification dialogue

The communication model is designed to do all necessary maintenance actions while ensuring integrity of a knowledge base. For instance, if user wants to change a property name or one of its values, then all the cases containing this property/value are updated with newly entered name/value. Further, a user can browse cases or search for cases having a particular property/value.

*4.3 Multimedia component*

The multimedia component, figure 5, is used to attach multimedia files to a case. Two types of files are allowed: pictures and video clips. The communication model is designed to help a user in finding files on the local network, browse them, and do any necessary updating. The selected file is uploaded to the application directory in the server; then URI of the file is captured and encoded in KB with an appropriate description title.

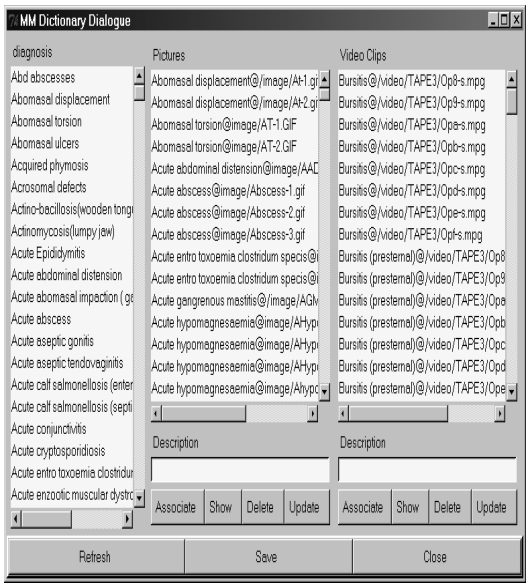


Figure5: Multimedia acquisition dialogue

*4.4 Security component*

The security component, figure 6, is used to define participants (users) working in the application. Four groups are pre-defined: administrator, knowledge engineer (KE), domain expert, and guest. Each group has its privileges. There are 4 privileges: Browse KB, Cases acquisition, Properties acquisition, and Server-thread management: KB saving and thread termination. A user privilege record is of the form (B/C/P/S). Guests can browse knowledge-base and run the generic component for testing. Consequently, a guest user privilege record will be (+/-/-/-). Experts can modify cases (+/+/-/-). Knowledge engineers can modify properties (+/+/+/-). Administrator can save knowledge-base and terminate a running server thread (+/+/+/+).

.


Figure 6: Users dialogue used to define new users for project authentication

## 5. APPLICATION DEPLOYMENT

A final application usually needs a user-interface that is designed according to specifications and requirements collected from end-users (and/or domain experts). Mozart library and COM component are provided to facilitate this important step. Mozart code is platform independent. Consequently, the tool or a final application can run in all platforms supported by Mozart. Meanwhile, a deployment programmer should use an appropriate technology for a front-end (screen forms, dialogues, etc.), use Mozart library, and acquired KB to develop a final application.

## 6. CONCLUSION

The tool has been used successfully in developing two commercial expert systems: Bovine expert system and Caprine expert system. The aim of those clinical expert systems is to diagnose affections including different age groups and diseases of infections, noninfectious, surgical and genital diseases. Both systems contain multimedia, which covers images of major ailments as well as video clips. In effect, they improve the knowledge of veterinarians as well as provide the line of treatment and control of all diseases.

In the near future, the knowledge-base will be represented using one of the Semantic Web languages. The properties will be part of ontology for veterinary domain. The tool components will be

implemented as web services. Consequently, final applications will be available as web services, too.

## 7. REFERENCES


Mozart Consortium, (1999), "The Mozart programming system", Available at http://www.mozart-oz.org/

Gomez, F., & Chandrasekaran, B, (1981)., "Knowledge organization and distribution for medical diagnosis", *IEEE Transaction on Systems, Man, and Cybemetics*, **SMC-11(1)**, 34-42 .

Mittal, S., (1980), "Design of a distributed medical diagnosis and data base systems", Computer and Information Science Department, Ohio State University.

Rafea, M., A, (1995) "computer Tool for Knowledge Acquisition in Different Domains". Ph.D. thesis, Institute of Statistical Studies and Research, Cairo University.

Haridi, et al, (1998) Haridi, S., Van Roy, P., Brand, P., & Schulte, C., Programming Languages for Distributed Applications. New Generation Computing, **n3 v16**, Omsha, Ltd. and Springer-Verlag

Haridi, S., & Franzén, N., (1999) Tutorial of Oz. http://www.mozart-oz.org/documentation/tutorial.

Waern, A., Hook, K., Gustavsson, R., & Holm, P., (1993 ) "The common-KADS communication model", KADS-II/M3/SICS/TR/006/V.2.0, http://swi.psy.uva.nl/projects/CommonKADS/publications.html.