An important aspect of knowledge base systems (KBSs) development is the continuous adaptation and extension of already deployed knowledge bases. Often, these modifications consider the improvement of the knowledge base (KB). Furthermore, examination of the KBSs in its actual user environment could broaden the KB. The presented research work focuses on enhancing evolutionary generic domain specific KBSs development process through an integrated web-based environment. The environment covers many aspects related to the maintenance and refinement facilities and exchange of knowledge between different system users. Two main users groups are involved in the proposed system: end user group and domain expert group. End users may require knowledge modifications according to experiment of KBSs in real life. These modifications should be approved by specific domain experts according to their area of expertise before applying it to KBSs at the server side. The proposed framework is generally applicable for KADS-based KBSs. As a matter of fact, we have applied it to agriculture diagnostic KBSs. Knowledge acquisition and model maintenance are key problems in knowledge engineering to improve the productivity in the development of intelligent systems. Traditionally, a KBS is built by a knowledge engineer (KE) who needs to acquire the knowledge from a subject matter expert and to encode it into the knowledge base. This is a very difficult process because the domain expert express their knowledge informally, using natural language, visual representations, and common sense, often omitting many essential details they regard as being obvious. Moreover, changing existing and often large knowledge bases is not a simple problem, since unstructured modifications can cause unexpected deficiencies and errors. When a KBS is built in an evolutionary way the content of the knowledge base needs to be improved from time to time.

The knowledge-engineering community has identified and developed Problem-Solving Methods (PSMs) for specific high-level tasks such as planning, diagnosis, assessment, etc. Problem-solving methods describe the control knowledge independent of the application domain and thus enable the reuse of this strategically knowledge for different domains and applications. Several systems that used these PSMs as components which are combined and instantiated into complete systems have been developed [13]. Accordingly, generic agriculture models that enhance KBSs construction for different crops have been developed [8, 9, 10]. The aim of these models and the tools that based on it is to assist the rapid development of irrigation, diagnosis and treatment agriculture expert systems by offering the system builder a template that can be easily filled. The dynamic nature of knowledge leads to updating KB continuously. The main challenge here is to keep the knowledge repository useful by maintaining it as it sits there. This may involve the regular updating of its content as content (domain knowledge) changes. Researchers have agreed that end users are only allowed to specify certain kinds of modifications, i.e., domain-specific knowledge regarding instances and classes but not modifying problem-solving knowledge [16]. As a result, a need for specific domain experts to review
and confirm user modifications arises. The aim of this work is to facilitate the access of generic diagnosis agriculture KBSs residing on server and allow both end users and domain experts to apply her/his modifications in structured manner without needs of knowledge engineer. Feedback from the usage of the KBSs in real world environment is usually used as a source of information when analyzed by researchers. Then, specific domain experts extend the KB in their area of expertise based on experiment of KBSs in real life environment. Different maintenance and refinement activities take place according to the user role and view. Structured maintenance examines and modifies the original design, and then reworks the code to match it. While, refinement handles user’ modifications that affect the semantic of the knowledge. The motivation for this work is the increasing demand for applications that take advantage of the Internet for accessing domain specific knowledge bases. The need for automated support and operational models that allow workflow of domain specific KBSs and to coordinate the maintenance work across multiple users is becoming critical for the proper management of such activities. Structure of this paper is as follows: we start by describing different KB development environment (cf. section 2). Next, we propose the framework of knowledge base management system (cf. section3). Section 4 introduces the general architecture of the evolutionary model. The complete refinement process is discussed in section5. We conclude this article on the necessity to specify the different ongoing research activities.

2 Literature Review

In this section, we are reviewing how different current approaches have proposed to carry out the development process of KBSs. We start by presenting different KB development environment then investigating modeling of systems by reusing problem-solving methods (PSMs).

2.1 Knowledge base Development Environment

The Rapid Knowledge Formation project (RKF) aims at building power-tools for knowledge acquisition. The central objective of this project was to enable distributed teams of subject matter experts to enter and modify knowledge directly and easily, without the need of prior knowledge engineering experience. Disciple-RKF, the implementation of the most recent version of the Disciple approach, is an agent shell with a knowledge base structured into an object ontology that describes the entities from an application domain, and a set of task reduction and solution composition rules expressed with these objects. The development of the knowledge base of a Disciple-RKF agent is based on importing ontological knowledge from existing knowledge repositories, and on teaching the agent how to perform various tasks, in a way that resembles how subject matter experts teaches a human apprentice [7].

TEST1 (Troubleshooting Expert System Tool) [14] is an application shell, providing a domain-independent diagnostic problem-solver together with a library of schematic prototypes. These prototypes constitute the object types and the structure required by each domain specific TEST knowledge base. Several TEST applications, including those aimed at diagnosing engine problems, VAX/VMS performance, and factory floor machine failures were develop and tested.

A conceptual framework for facilitating the modelling and development of heuristic reasoning services such as diagnosis, therapy planning and patient monitoring in clinical domains was developed in [27]. The purpose of that framework was to provide support for 1) the representation of medical knowledge in a structured manner, 2) the design of clinical tasks, 3) simplifying the development of executable clinical reasoning services from their design, and 4) making available its use on the web, with two main objectives: helping to increase the use of decision-aid systems in the clinical practice; facilitating the design of these systems by reusing information and knowledge sources distributed in the Web.

2.2 System Modeling by reusing Knowledge Components

Many approaches have been exploited for accelerating the process of building KB systems, some of these approaches were concentrating on the portability of ontologies, so that it can be reused over a wide range of domains e.g. KACTUS [28]. Some other approaches were focusing on constructing a library of problem solving methods, that is somehow, generic enough to be reused in different applications e.g. Generic Tasks [5,6], Components of Expertise [26], CommonKADS [23,24]. Another approach was to build a complete framework for building knowledge based systems from generic library components e.g. PROTÉGE-II. The PROTÉGE-II [12] approach distinguishes between tasks and problem-solving methods, and the key steps in the development it are three: 1) task analysis, 2) method selection and 3)
method configuration. In this way, problem-solving methods can be described in a higher level without including specifications about problem commitments and so, separately from task description.

For Akkermans et al. [3], an application task is described by means of a set of assumptions about the problem space, a problem-solving method by a functional specification and method configuration is a process consisting of refining an application task taking into account the method functional specification and the available domain theory. This process is called assumption driven activity. Nevertheless, when this approach has been followed on [24], the refined methods are much too generic (such as the 'propose and revise' method), and their application still requires much adaptation effort.

3 A framework for Knowledge-based management system (KBMS)

The proposed KBMS framework is capable of representing multiple and concurrent views of features. The main idea behind this framework is to create a community of end users and domain experts specialized in different agriculture diagnostic field through World Wide Web. As end user deploys his own local KBS and goes in different sessions, he can apply modifications to his local KB. As a result, a refinement module runs and advises the user about her/his modification request. Furthermore, when the user submits the modification to the local KB, a mirror of these modifications is transferred to the server. At the server, the knowledge manager keeps history of all modifications applied by each end user, filters all these modifications, and assigns it regularly to specific domain expert. The domain expert(s) reviews all these modification and find out whether it is applicable to be included in the original (centralized) KB. The basic idea of this framework is that applied refinements are essentially based on real life experience which coincides with the idea of constructing KBS from different resources [17]. This framework is essentially based on:

- Domain specific model reusability
- Multiple system views
- Knowledge Refinement

3.1 Reusability of domain specific model

Domain specific knowledge and problem-solving methods are considered the backbone for building structured and reusable knowledge models [22]. Domain specific knowledge concerns declarative knowledge and it is the most fundamental part of a KBS. Problem-solving methods are building blocks for realizing the reasoning components of KBSs. Each of these elements is described independently to enable the reuse of problem-solving methods for different tasks and domains, and the reuse of domain knowledge for different tasks and problem-solving methods. The generic diagnostic model [9] was developed based on CommonKADS model of expertise [23] that integrates the domain knowledge and the problem solving knowledge. This generic model has the ability to derive diagnoses for differently structured individual systems from the agriculture domain as it contains knowledge that is the basics for the construction of any agriculture diagnosis expert system. The content of the KB must be able to be modified by the user. For this purpose, the KBS is distinguished into the following types of parts:

(1) Static parts, that represents the PSM which defines the reasoning process of a knowledge-based system, and
(2) User, dynamic areas that can be updated by users (domain knowledge).

3.2 Multiple user views

A number of successful agricultural expert systems were developed and deployed to a large number of users [18, 20]. These applications are traditional standalone systems. Recently, agent technology has enable constructing a community of cooperating agents capable of diagnosing disorders in the agriculture domain [15, 25]. By implementing expert systems as multi-agents that perform their tasks remotely, the expertise can be published on the Web. Expert systems running on the Internet can support a large group of users (clients) who communicate with the system over the network. Accordingly, the proposed framework distinguishes among two user access levels and provides an architecture to coordinate the behavior of several specific user views. These views are domain expert view, and end user view. The KBSM framework is oriented to facilitate the addition and modification of the domain knowledge for the end user. On the other hand, in order to carry out an agriculture diagnostic service, the domain expert must validate the required KB modifications. Each system participant or stakeholder have different roles, responsibilities and concerns based on his own view which will be described in the following subsections.

3.2.1 Domain expert view

Diagnostic KBS combines different specializations that are well defined in the agricultural domain. The
key problem lies in deciding how the domain is to be partitioned. We figured out that the agricultural diagnosis domain can be classified into six groups, namely: fungal, insect agent, nematode, snail, mites, and nutrition deficiency [19]. This classification is done according to the diagnosis taxonomy in the agriculture domain. It has the following advantages that contribute to the suggested framework. First, it tries to simulate what may happen in real world in a very fine-grained manner. Second, partitioning of domain knowledge makes it easy to maintain knowledge. Each of diagnosis class is assigned to a specific domain expert (s) who is authored to review and confirm any user modifications. According to the domain expert specialization, set of guilty KB component is sent regularly to him to decide whether to apply it in the centralized KB or not.

3.2.2 End Users view

Users need to access and interact with KBSs from different perspectives depending on the tasks they are completing. In normal cases, end users (extensions, highly experienced growers) interact with the system to complete their sessions by applying their own diagnostic cases and obtaining the system result. In real life, the symptoms of disorders are likely to change. Also, when the system does not (does no longer) function as required, user may require to update any of the domain specific knowledge. The aim of the system is to guide end users during applying their modifications to avoid any knowledge inconsistency. The end user view consists of three parts: a complete diagnostic model which has the ability to accept user change, local knowledge refinement module which filters wrong knowledge and a log file to record user applied modifications. The user view also focuses on the interactions among different components in a dynamic fashion, which reflects the nature of the dynamic environment.

3.3 Knowledge refinement

The suggested knowledge refinement process is based upon two fundamental assumptions [1,2]:
1. It is assumed that the initial KB was constructed with the intention of being correct, and if it is not correct, then a close variant of it is.
2. It is assumed that each component of the knowledge base appears there for some reason. Therefore, if a domain knowledge component is invalid, it cannot just be removed. Rather, we have either
   (a) To find out its reason for being in the domain knowledge, and find one (or several) alternative domain knowledge components) that will satisfy that reason after the incorrect component has been discarded or
   (b) To develop ideas how to modify the domain knowledge with the objective of improvement.

Thus, the present approach focuses on the second initiative. The proposed refinement technique is a cooperative approach which suggests, at various stages, options and the user makes the selections. As a user submits his choice, the system has to cope with any modifications and has to ensure the KB consistency whenever it is changes. The complete design of the end user refinement module is discussed in section 5.

4 System Architecture

The environment consists of a central server, where the original (centralized) KB reside, in addition to a number of clients. The central server serves two types of users: the domain expert groups and the end users groups, each working on his corresponding side of the server. The word “side” in that context means a group of services offered to clients of this server. The KBMS comprises various components that work in coordination with each other to provide the services offered by the KBMS. Figure 1 depicts those different components and lays out the general scheme of interaction between them.

4.1 System components

KBMS has a modular object-oriented design, which support the independence of the centralized knowledge model to be reused, and the knowledge manager module which manage different human interaction with centralized KB. In the proposed environment, the
knowledge model represents a generic agriculture diagnostic model component to be reused. The knowledge manager is the core of the presented environment, as it provides the deployment and refinement services to its clients over the network.

4.2 A generic agriculture diagnostic model

Domain knowledge contains all knowledge about disorder diagnosis in the agriculture domain. It consists of: domain ontology and domain relation. Domain ontology is the most fundamental part of a KBS. It represents the basic terminology used by the application in a specific domain in form of concept hierarchy. Each concept is identified through its name and described in details in terms of properties. Each property is also described by facets, which specify its data type, legal values, and source of value. In addition to other facets necessary for handling these properties. Domain relation represents different interactions between the domain concepts and their properties. Domain relations are built on top of the declared concepts, in the form of rules clusters, tables, and mathematical functions. Each of these representations has its own internal structure that suits different kinds of domain relations. The problem solving knowledge has two main categories: knowledge about the problem solving steps, their interactions, and their relations to the domain knowledge (inference knowledge); and task knowledge. The task knowledge represents how the problem solving process is controlled.

4.3 The knowledge base manager

The knowledge base manager (KBM) plays a vital role in the overall process. Its main function is to control the access and the updates to the centralized knowledge base. As illustrated in figure2, the main components of KBM are:

- End users manager
- Domain expert manager
- End users feedback module

A detail description of each of these components is in the following subsections

4.3.1 The domain expert group manager

It is responsible for authenticating the domain expert of the KBMS and authorizing them to the appropriate resources. By partitioning the diagnostic domain into sub-domains that are as independent as possible, and assign each partition to specific domain experts (may be a group of experts in the same field). This expert (s) is responsible for approving user’s feedback. It manages these activities through a database of domain experts and groups.

4.3.2 The end users group manager

It is responsible for authenticating the users of the KBMS and authorizing them to the appropriate resources. i.e provide each of them with a modifiable local KBSs, user refinement module and log file to keep track of user modifications. It manages its activities through a database of users which is also used in broadcasting the approved modifications by experts to all clients. It is also responsible for applying the collaboration mechanism between the server and end users group.

4.3.3 End users feedback module

This module is responsible on gathering all user modifications stored in their local log files regularly through collect user modification process. These modifications are stored in a global modifications database. Since all these modifications are related to different diagnostic classes. User modifications should be applied to specified domain expert (s) according to her/his specialization in order to approve it. Therefore, it is applied to a classification module that classifies it before delivering them to the corresponding domain expert (s).
5 The refinement process

Refinement tools aim to incrementally modify knowledge based systems (KBSs) by identifying and repairing faults that are indicated by applied examples for which the KBS gives an incorrect solution [29]. Therefore, any required modification should be validated against inconsistency. We broadly classify inconsistencies in KB into two types: structural and semantic. We define structural inconsistencies as those arise as a result of user modifications affect the KB structure. Modifying a KB component often requires several individual changes to other components, and all those changes must be carefully coordinated. While semantic inconsistency appears as a result of user modifications affect the semantic of the knowledge. Since taxonomies are often organized based on the semantics of the taxonomic relationship [4]. The refinement module assists end users to accommodate a new item in an established KB. The user can commits or cancels her/his change. For example, when a user wants to add new diagnostic rule, and as he is not familiar with the classification criteria upon which the disorder taxonomy was built. As a result, he may classify the input rule at incorrect KB components. Each of these inconsistencies elimination will be discussed in this section.

5.1 Eliminate structure inconsistency

Structural inconsistency is detected and resolved automatically [11]. Structural inconsistencies appear as a result of removal of any domain KB components such as (concepts, rules, tables and mathematical functions). This operation not only affects the required component but also affect other related components. For example, when a concept in the hierarchy is deleted, there are several concerns to be resolved. One is what to do with the sub-concept; other is other domain relations that use this concept. For each concern, we provide a set of possible strategies for users to choose based on their own preferences. Therefore, this step implies understanding of the effect of different maintenance request on the whole KB, then apply the required changes. Once the change is requested, the system generates a list of suggestions and other affected KB components upon which the user resolve.

5.2 Eliminate semantic inconsistency

As describe earlier, semantic inconsistencies are caused by adding or updating different domain KB components such as concepts, rules, tables and mathematical functions. In the real world, the features of disorders are likely to change from time to time. Therefore, the user may require to update the symptoms that lead to certain disorder or to add new disorder. The main problem here is how to locate the newly added knowledge in the most appropriate component for the current system according to same criteria used in system development. Therefore, it is essential to understand how the domain knowledge components are related to each other. In fact, clusters of domain relations drawn from the KB are used together to fulfill a specific function at the knowledge level. In the generic agriculture diagnosis KBSs, disorder taxonomy is divided into sub-domains (groups) namely: fungal, insect agent, nematode, snail, mites, and nutrition deficiency. Each sub-domain has its ontology and rule clusters sharing the same semantic context. In order to fit the modified component together so that the semantic of the knowledge remain, it is required to calculate the best candidate sub-domain that accepts this modification. User refinement module is decomposed of the following phases: acquire, identify, localize, generate and select that will be discussed in the following subsections.

5.2.1 Phase1 Acquire (Get user modifications)

The user selects the KB component to be modified from a pull-down menu. Different modification operations are allowed such as add, edit and delete. Whenever deletion operations are required, the change needs to be propagated to the whole KB. A list of affected KB components appears to the user with complete details. The user can commit or cancel this operation. On the other hand, the addition and edition operations need running the next phases to validate that new (updated) added knowledge doesn’t affect the semantic of the whole KB.

5.2.2 Phase2: identification (Determine class generalization and specialization)

Following the addition of the new disorder or disorder rule, the taxonomy may no longer be valid, in which case the system has to refine the taxonomy and possibly the disorder rule cluster to ensure the new component is appropriately located. Newly diagnostic rules are obtained from user in form of antecedence–consequence pairs. Antecedence represents the symptoms (finding) that lead to certain disorder, while consequence presents the disorder. The new rule must have at least one dominant finding match with the specified rule cluster from the user. Otherwise, it has no characteristics and cannot be added to that rule
cluster. Hence, the system searches for the most likely rule cluster and add the new rule to it. This is achieved by determine the generalization and specialization for each rule cluster class. Generalized symptoms (finding) are those common between all rules in the same rule cluster. While specialized symptoms (finding) are those unique to that class i.e appear only in this class. Algorithms used to determine rule cluster generalization and specialization are shown in figure 3,4 respectively.

5.2.3 Phase3 localization

We realize that it is important to assign confidence values to each finding, to differentiate them into dominant and subsidiary features w.r.t diagnosis class. Dominant finding are those prototypical [21] for the diagnosis class membership, more importantly, while subsidiary finding are those overlaps between classes. For example, stem shape is spindly appear only may in nutrition deficiency of nitrogen. On the other hand, findings always overlap between different disorders classes i.e. each finding may appear in more than one class like leaves color is yellow appear almost in all disorder classes. In order to fit the new rule to the nearest rule cluster, it is significant to rank it against all class. Rule localization is a two-step process. First, the technique discovers finding relevant to the cluster of interest, by comparing each input finding against class generalization and specialization. Figure5 illustrate the algorithm used to localize input finding w.r.t class generalization and the same algorithm is used for class specialization. Then, we calculate the confidence of each finding to all classes using the algorithms of figure6. In this second step, the confidence values are accumulated and the algorithm calculates the best candidate rule cluster.

```
EmptyList(RulesList);
EmptyList(FindingsList);
ClustersList = GetAllClustersFromKB(KB-ID);
For each cluster in ClustersList
    Begin
      RulesList = GetRulesFromCluster(Cluster-ID);
      Expanded-list= Cluster-ID;
      For each rule in RulesList;
          Begin
              FindingsList = GetFindingsFromRules(Rule-ID);
              FindingsList = Remove-redundancy (FindingsList);
          End
          Concat(Expanded-list, FindingsList);
      End
    End

Figure3: Algorithm used to determine rule cluster Generalization

Expanded-list;
EmptyList(RuleFindingsList);
EmptyList( classlist );
EmptyList( class-finding-compoundlist ) ;
Input-Rule = GetRuleFromUser();
RuleFindingsList = GetFindingsFromRule(Input-Rule);
For each cluster in Expanded-list;
Begin
    temp-list1=ClusterID-Finding;
    For each finding in ClusterID-Finding
        Templist2=searchclassfinding(Finding, Remainder-Finding);
        If(Templist2 !==[])
            %
            finding is unique to that class
            Begin
                Concat(class-uniqueness-list, Cluster-ID);
                Concat (class-uniqueness-list ,Templist2);
            End
        End
    End
Figure4: Algorithm used to determine rule cluster specialization

Figure5: Algorithm used to determine Rule localization
```
5.2.4 Phase 4: Generation and Selection

The last refinement phase prompts the user with a report containing detailed information about the newly added rule. The user submits his choice, it automatically updates her/his KB and the local log file as well.

Figure 6: Algorithm used to calculate the confidence of rule findings

6 Conclusion and Future Work

This paper describes a framework for facilitating the construction and maintenance of domain specific KBSSs. In order to assist knowledge dissemination, these generic models are deployed through web to be widely used. Our approach is oriented to re-configure domain specific knowledge by reusing generic models residing on server and allow end user to apply her/his modifications in structured way. The aim of this work is to allow both end users and domain experts scattered throughout the world to apply her/his modifications in structured manner without need of knowledge engineer. When the user request specific change that can affect both the structure and the semantic of the domain knowledge, he is prompted, by a list of options, to choose how to handle the modifications. Then, domain expert in different specific areas review and confirm user modifications. This is an important step in KBSSs development because it allows monitoring the system in its actual user environment. It is important to note that the approach followed by KBMS can be generalized to other KBSSs in different domains. Development of a complete knowledge acquisition environment that can be used to construct the domain specific knowledge based on PSM can enrich the proposed approach.

References


