PERGAMON

Expert Systems with Applications 24 (2003) 49–57

Expert Systems with Applications

www.elsevier.com/locate/eswa

Automatic knowledge acquisition tool for irrigation and fertilization expert systems

Ahmed Rafea[a,*], Hesham Hassen[b], Maryam Hazman[c]

[a]American University in Cairo, P.O. Box 100, Dokki, Giza, Egypt
[b]Faculty of Computers and Information, Cairo University, P.O. Box 100, Dokki, Giza, Egypt
[c]Center Laboratory for Agriculture Expert System, P.O. Box 100, Dokki, Giza, Egypt

Abstract

Knowledge acquisition is commonly regarded as a major obstacle and bottleneck in the process of designing and implementing knowledge base system. Our approach to overcome this problem is to build a task specific knowledge acquisition tool for solving the irrigation and fertilization scheduling problems. The tool was tested using set of cases that was designed to measure the tool flexibility, usability, accuracy, and exception handling.

Keywords: Knowledge acquisition; Knowledge modeling; Reusability

1. Introduction

Knowledge acquisition is commonly regarded as a major obstacle and bottleneck in the process of designing and implementing knowledge base. Failure to acquire and encode appropriate amounts of relevant knowledge lead to limited consultation performance of the system (Eriksson, 1991). Knowledge typically could be acquired through one of two ways: either manual (through the knowledge engineer) or automatic. In the automated knowledge acquisition tools are developed to help the knowledge engineer or even the expert himself to build and maintain the required knowledge systems (Gruber, 1987).

Many Knowledge acquisition tools have been developed, e.g. MOLE (Eshelman, 1987), SALT (Marcus & McDermott, 1989), KCT (Fujihara, Simmons, Ellis, & Shannon, 1997), ITAKA (Edrees & Rafea, 1997), HCGT (Rafea & Rafea, 1998). In effect, these tools vary according to its strategies and approaches for solving the knowledge acquisition problems. Features that might be used to classify these tools might include: *automation degree*, *domain dependency*, *problem type*, *task dependency*, and *user type*. However, a specific knowledge acquisitions tool can be classified in more than one from these classes (Hazman, 1999). For instance, the HCGT is both domain independent tool and used in specific task (hierarchical classification) in the diagnostic problem.

Unfortunately, reusability degree is still limited, since no one of this tool integrates between task and domain. This issue was one of our motivations in developing our tool described in this paper especially reusable knowledge in the target domain (agriculture) is available in our research laboratory (CLAES).

The Central Laboratory for Agricultural Expert System (CLAES) has gained a considerable experience in developing knowledge based systems in agricultural domain. These knowledge based systems cover different agricultural production management problems and applied for different crops (Cucumber, Citrus, Tomato, Wheat, Lime) (Rafea, 1994). The overall production management problems involve, among other aspects, water and fertilizer requirements, pest control, farm evaluation, variety selection. These crops have different characteristics in developing its irrigation and fertilization schedulers, for Cucumber cultivation under Plastic Tunnels (CUPTEX) (Rafea, El-Azhari, Ibrahim, Edres, & Mahmoud, 1995), CITrus cultivation in open fields (CITEX) (Citex, 1997), and TOMATo cultivation (TOMATEX) (Tomatox, 1995), respectively.

The rest of this paper is organized as follows: Section 2 describes the abstract architecture of the proposed tool. Sections 3–6 present the library's components, the knowledge elicitation modules, the knowledge base generator, and the verification knowledge base, respectively. Section 7

* Corresponding author. Fax: +20-202-360-4727.
E-mail addresses: rafea@mail.claes.sci.eg (A. Rafea), hesham@mail.claes.sci.eg (H. Hassen), maryam@mail.claes.sci.eg (M. Hazman).

0957-4174/03/$ - see front matter © 2002 Elsevier Science Ltd. All rights reserved.
PII: S0957-4174(02)00082-9

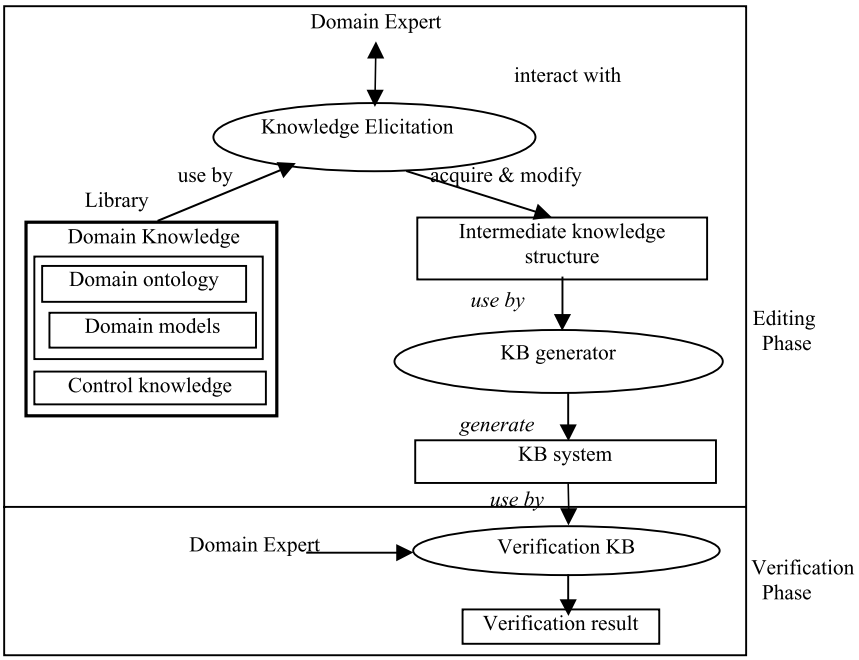

Fig. 1. The proposed tool architecture.

presents a case study and the evaluation of the tool. Section 8 provides the conclusion for our research.

2. Abstract architecture

The proposed tool includes four main components namely: knowledge elicitation module, library, knowledge base generator, and verification knowledge base (Hazman, 1999). The architecture of the proposed tool is graphically shown in Fig. 1. Ellipses correspond to modules, whereas rectangles represent input and output for these modules. As shown in Fig. 1 our proposed tool's architecture includes the following components

- *Library*. Contains both reusable domain knowledge and control knowledge.
- *Knowledge elicitation*. Serves as intermediate layer between the domain experts and the other components of the tool. Its main functions are to create, maintain, and store knowledge elicited from the domain expert, fetch the relevant knowledge components from the library, and transform this knowledge into the appropriate intermediate knowledge structure.
- *Knowledge base generator*. Automatically generates an executable system, which corresponds to the intermediate knowledge structure generated above.
- *Verification knowledge base*. Allows the domain expert to test and verify the stored knowledge, and study the effect of changing some input parameters on the final system results.

3. Library

Reusing previously developed components, is a major concern in any business or industry. The benefits of reusability are particularly important when components are expensive or take time to design and validate. The identification of reusable components has been a recurrent activity in knowledge engineering. In particular, reusing knowledge base or inference structure has long ago been considered as a way to overcome the knowledge acquisition bottleneck (David, Krivine, & Simmons, 1993).

The reusability has many facilities, it saves the time and cost for developing the main components in the knowledge base. Actually the cost of building a software system from reusable components is the cost of locating the most appropriate reusable software component for the current application, and the cost of filling selected components together so that the output of one component matches input of another (Abdelhamid, Hassan, & Rafea, 1997). In addition to each components of the library has been tested before.

The original idea of library based approach for knowledge base system construction is to build a new knowledge base system out of components drawn from pre-define knowledge component libraries (Krueguer, 1992). A knowledge base system is a collection of different component types, each of which has its own nature, and its own competence of reuse (Abdelhamid, 1998). Since the tool is designed to use in a specific domain using a specific problem solving, the library will contain both knowledge related to the application domain and knowledge related to problem solving. In this tool, we support the following reusable knowledge types:

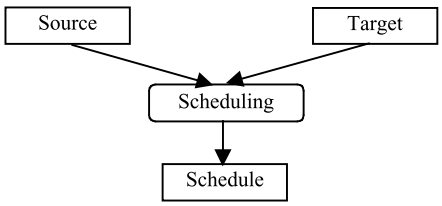

Fig. 2. The scheduling function.

3.1. Control knowledge

Tabulating irrigation and fertilization is a type of a scheduling problem which has been defined by Breuker and Velde (1994) as follows

- It operates on two sets of objects (source, target), see Fig. 2.
- The solution consists of establishing assignment relations between objects of different sets, such that defined requirements and constraints are satisfied.

In other words, their view indicated that the scheduling problem is no more than finding an assignment between two sets of objectives *source* and *target* satisfying a given set of constraints. According to this definition the set of fertilizers in the fertilization scheduling problem could be mapped to the source and the application time of these fertilizers is mapped to target. The specification of these fertilizer types and its quality should be done according to rules, and quantities of all fertilizers should satisfy a set of constraints. In the similar way, the water in the irrigation problem could map to the source and the application time of the water is mapped to target. The specification of the water quantity should be done according to rules and should satisfy a set of constraints.

Having agreed about the type of problem, the next step is to answer the question that what is the proper problem solving method used to achieve the solution of this problem. A well-known method called Propose and Revise problem solving method (Breuker & Velde, 1994) is successfully used to solve this type of problem. The inference structure describing this method is shown in Fig. 3.

The objective of the 'propose' function is to combine a unit of one or more source[1] with the target[2] to generate an initial schedule based on a collection of environmental parameters. The objective of the 'revise' function is to check for any constraints violations in the proposed schedule with regard to some environmental constraints—such as the existence of certain disorders in the farm- and report these violations to be fixed for producing the final, and acceptable schedule.

[1] Example of sources in our case are: fertilizers, and water quantities.
[2] Target in our case will be a time slice or interval.

3.2. Domain knowledge

This library component contains the fundamental constituent of the knowledge base. We can organize these knowledge components into two components namely: domain ontology and domain models. The following subsections describe these two constituents in much detail.

3.2.1. Domain ontology

The objective of this component is to make the job of the domain expert more easily to build his/her application, by allowing him/her identifying and selecting the appropriate domain terms for his/her applications. However, the tool allows the domain expert to maintain this knowledge store by allowing him/her adding or updating the knowledge and using it in his/her application whenever it is needed.

As we have pointed before, the domain ontology is a set of pre-defined ontological primitives. These primitives are the domain concepts as well as a set of defined relations in a specific domain. The concepts are the fundamental constituent of the domain ontology; a concept can stand for anything existed in a specific domain. It has structure, which describes the characteristic of the concept. Example of a concept stored in the library has been shown in Fig. 4 for the irrigation concept.

The relation is the second ontology primitive used in our knowledge store. In our application, we have identified two types of relations namely: mathematical relations and non-mathematical relations. The mathematical relations 'also called mathematical function relations', whereby a specific irrigation or fertilization parameter is computed. An example of this relation is shown in Fig. 5. This type of relation is heavily used in both irrigation and fertilization scheduling for representing the mathematical equations required to compute irrigation and fertilization parameters.

The non-mathematical relations, in turn, could be classified into two types of relations namely: relation between concepts and relation between concepts' properties. In a relation between concepts, the concepts are related to each other by 'is-a' relation forming classification tree structure. This type of relation is useful to allow the sub-concepts inherit from their supper concept. The relation between concepts' properties (which sometimes called relation between expressions) is considered as the common part in any knowledge base system. In which relation arguments consist of expressions. Fig. 6 shows a non-mathematical relation, which abstracts the soil type.

3.2.2. Domain models

This library component contains a collection of different domain models schema, which represent knowledge of what are the domain relations required to fulfill a specific function within the domain of application. Domain models point to existing relations in the domain ontology library, which will be used by existing inference steps to solve the scheduling

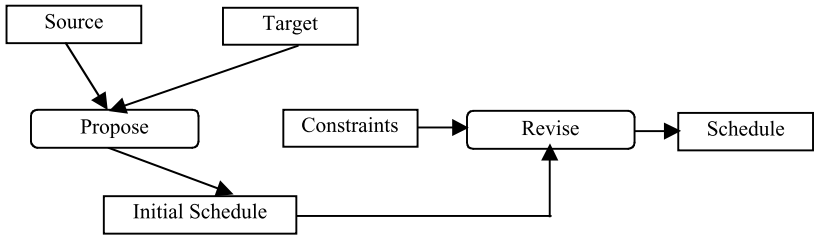

Fig. 3. A function structure for propose and revise.

Concept name: *Irrigation*;
Supper: *irrigation domain;*
Properties:
Property name: *Irrigation type;*
S/M value: *Single;*
Source of value: *User;* **Prompt:** *What type is your irrigation type?;*
Type: *Nominal;* **Property legal values:** *(flooding, drip);*
Property name: *Water_Qty;*
S/M value: *Single;*
Source of value: *driven;*
Type: *real;* **Lower Limit:** *100;* **Upper Limit:** *200.*

Fig. 4. An example to define a concept.

problem. The relations in a domain model are used to achieve its objective.

4. Knowledge elicitation module

Recalling from Section 2 that the main objective of this module (shown in Fig. 7) is to provide the user with set of facilities to retrieve, maintain, create, and store knowledge. As shown in Fig. 7 the reusability issue plays preliminary role in this tool.

Once this module is invoked by selecting one of the irrigation and fertilization tasks, the tool determines the corresponding collection of inference steps. Each inference step, in turn, focuses on a collection of domain models, which contain relation's schema that should be instantiated by the user. According to the relation type the tool will invoke the suitable relation editor to allow the developer creating the domain knowledge. Fig. 8 shows the relation between domain knowledge construct and their corresponding editor.

```
nu.n = (fnp.A *  sqrt(plant.age ) + fnp.B * sqrt(plant.efty )
```

Fig. 5. An example of a mathematical relation.

5. Knowledge base generator

The objective of the knowledge base generator is to automatically generate the target knowledge system using the relation instances obtained from the knowledge elicitation module. As shown in Fig. 9, the final target knowledge base system is obtained through running two sub-modules namely: KB source code generator and target KB generator.

5.1. KB source code generator

The objective of this module is to generate source code written in KROL (Rafea, Shaalan, & Rafea, 1997) and (KROL, 1998) format, which can be used by the domain experts to validate and test the system prior to its real usage. As shown in Fig. 10, the main components of knowledge base source code generator are concept converter, relation converter, and inference converter. Concept converter converts the concepts ontology constructs mentioned before to their corresponding knowledge source code using the intermediate knowledge representation. Relation converter

```
Relation: soil type
soil.type = fine
        If (soil.texture.= èclayê or èsily clayê or èclay loamê or èsily clay loamê)
soil.type = medium
        If (soil.texture.= èloamê or èsily loamê or èsandy clay loamê)
soil.type = coarse
        If (soil.texture.= èsandê or èsandy loamê)
```

Fig. 6. An example to non-mathematical relation.

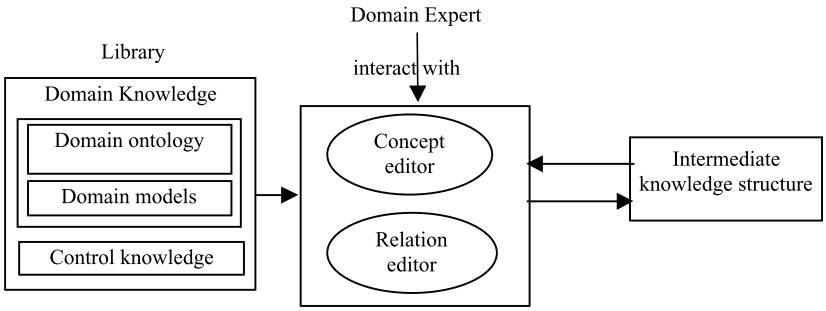

Fig. 7. The architecture of knowledge elicitation module.

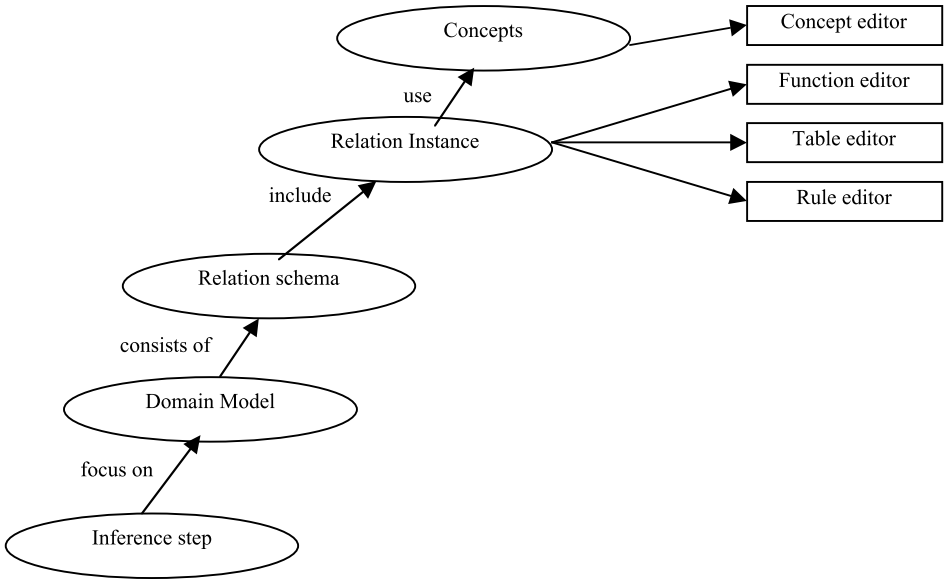

Fig. 8. Relation between domain knowledge construct and their corresponding editor.

is a set of sub-modules required for converting the relation knowledge representation schemas from their internal representation to the final required code. Inference converter converts the relation between the inference steps, domain models, relations from the intermediate knowledge structure to generate the corresponding knowledge source code for it.

5.2. Target KB generator

Once the domain expert has been finished with his/her testing, he/she can invoke this module to transform the knowledge base system from the source code Prolog format ('pl' format) to runtime Prolog format ('ql' format). The purpose of this process is to obtain a standalone system, which can be distributed for using by the end users.

6. Verification module

Adjusting the irrigation and fertilization parameters is not an easy task even for domain experts. This fact raises the importance of having a verification module, which provides some facilities to study the parameters interrelationship and their reflection on the final result. To achieve this objective, the verification module has been designed to run in two modes. In the first mode, the user selects items from two different sets namely: source parameters and target parameters. The source parameters, in this case, are all input properties that the users want to study their effect on some other parameters in the system. On the other hand, the target parameters set includes both the intermediate and output properties that the users wants to study their result

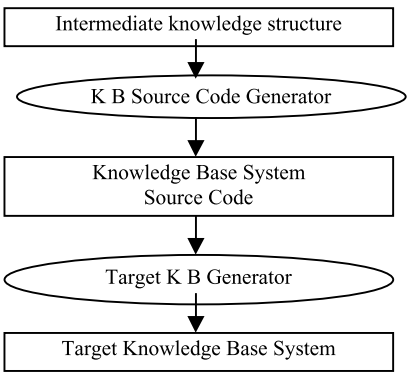

Fig. 9. The knowledge generator structure.

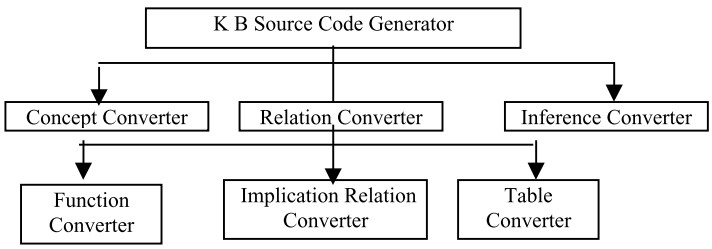

Fig. 10. The main knowledge generator components.

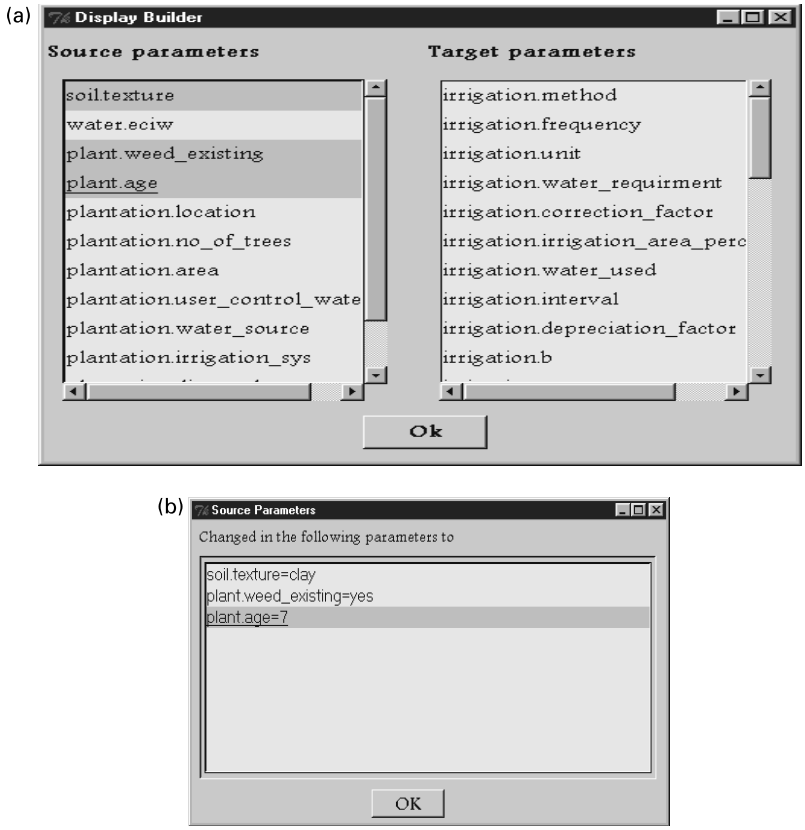

Fig. 11. The verification screens.

depending on the items selected from the other set (see Fig. 11(a)).

After selection, the tool runs the scheduling knowledge base system and displays the result required, in additional to save that result in text file. The user can rerun the knowledge base system after changing the source parameter using the screen shown in Fig. 11(b). The tool displays a complete history about the source and target parameter after each run.

The second mode differs from the first one in that the user selects a specific domain model to test only the relations included in that domain. Once the user selected one of the domain model, the two sets source and target parameters will be displayed in a way similar to the first mode with a difference that the source parameters, in this case, include the input properties used by the selected domain model. The target parameters include the output properties of the selected domain model.

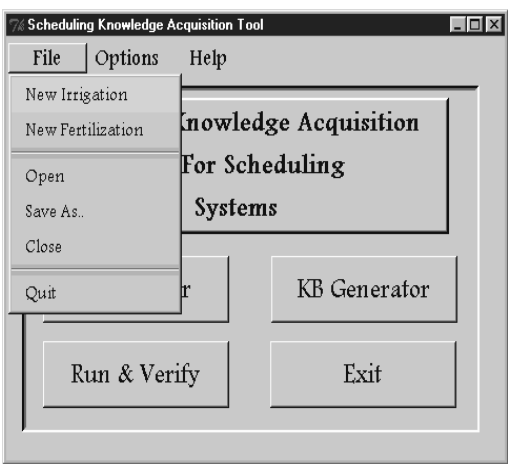

Fig. 12. The tool main screen.

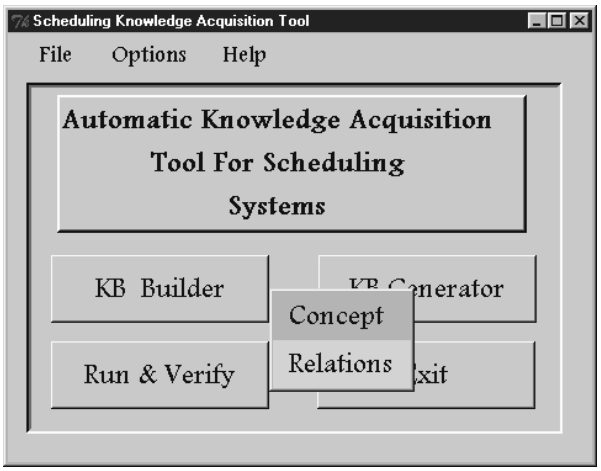

Fig. 13. The tool main screen.

7. Case study

The objective of this experiment is to illustrate how the domain experts can use our proposed knowledge acquisition tool for building an irrigation schedule knowledge base system from scratch. The following steps show the building process sequence of actions

(1) The user selects 'New Irrigation' from the 'File' pull down menu item in the tool main screen (see Fig. 12).

(2) The user can customize the irrigation task according to: problem solver type (simple, or complex[3]), plantation type (open field, or under tunnel), and crop type (seasonal or permanent). This customize is done by selecting the appropriate choices under the menu item 'options'. Our scenario supposes that the user selects the simple problem solver, open field for plantation type, and permanent crop type.

(3) To fill the domain models, the user clicks 'KB Builder' menu button then clicks on 'Relations' option (see

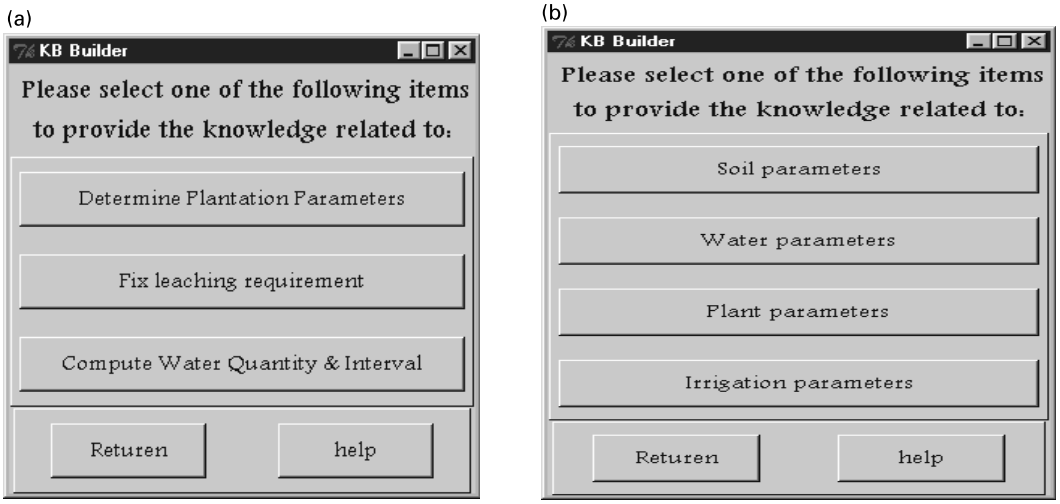

Fig. 14. The relations editor main screens.

Fig. 13). The relation editor main screen shown in Fig. 14(a) will be invoked. For instance, suppose the user wants to instantiated the domain model of the inference step 'Determine Plantation Parameters' from the screen in Fig. 14(a), he must select this inference step. Once he selects it, the four domain models of this inference step are displayed as shown in Fig. 14(b). When he selects the domain model 'Soil parameters', its relations schema are displayed as shown in Fig. 15. The user can instantiate the 'determine_soil_avaliable_water' relation schema by clicking the 'Provide knowledge' button in Fig. 15. Since this relation is of type table (i.e. its intermediate structure represented as table), the tool automatically activates the table editor shown in Fig. 16. Through this screen the user can add values either by selecting from a displayed list (in

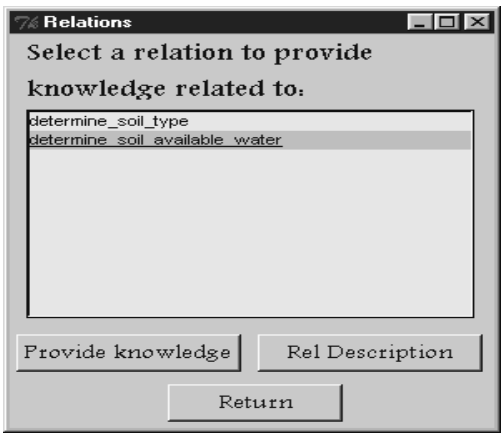

Fig. 15. The relation screen.

[3] A complex problem solver requires that both the sub-tasks: propose and revise to be include in the application to be developed.

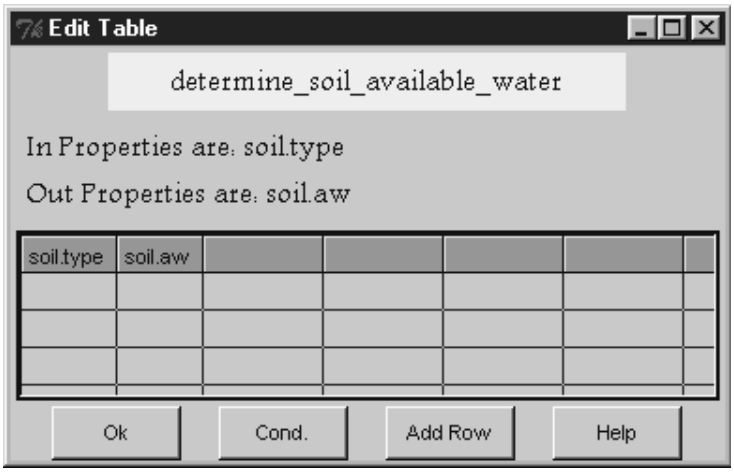

Fig. 16. Table editor screen.

case of nominal type, e.g. 'soil type') or writing it using the keyboard (in case of numeric type, e.g. 'soil aw').

(4) The user can select KB Generator menu button (in the tool main screen in Fig. 12) to generate knowledge base system source code. The tool checks for the complete set of relations. Suppose that the user forgets to fill the relation 'compute water requirement', a warning message asking him to fill the missed relation will be displayed (see Fig. 17). Only when all the relations are provided, the tool will generate the source code.

(5) To generate the appropriate scheduling screen, the tool allows the user to identify the output parameters that will be displayed. For example, if the user selects the plant age and water quantity unit once, while the irrigation frequency, and water quantity properties are selected for each month. The irrigation schedule will be displayed as shown in Fig. 18, when the user clicks on 'Run/Verify' button in Fig. 12.

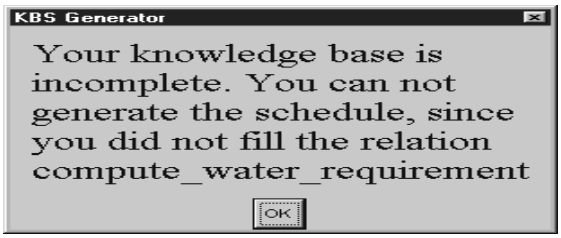

Fig. 17. Incompleteness knowledge message.

8. Conclusion

The objective of our research was to accelerate and improve the knowledge acquisition process by automatically generating the knowledge base system. A task specific knowledge acquisition tool has been built in the agriculture domain to solve the irrigation and fertilization scheduling problems. The final product, which could be obtained from it, is a running knowledge base system that is represented as SICStus Prolog Objects (SICStus, 1995). The problem solving method and domain knowledge schema have been pre-defined in the tool's library. The domain experts are only required to fill this pre-define schema, the knowledge elicitation module is responsible for accepting and storing

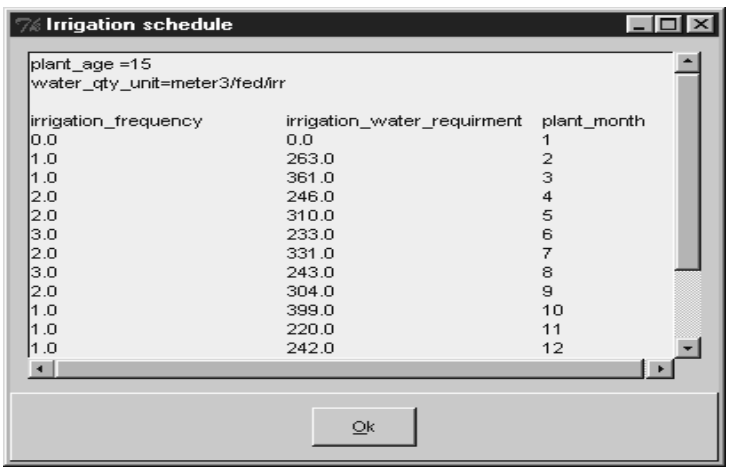

Fig. 18. Irrigation result example.

their knowledge. The tool provides its users the facility to verify their knowledge base.

A number of case studies have been developed to demonstrate the tool capabilities for developing irrigation and fertilization scheduling systems. Also, it was tested in developing knowledge base systems. It succeeded in providing valid knowledge base systems and was flexible enough to cover different plantation situations. Also, it can handle the exception situations. One of the lessons that has been learned from our research is that the proposed tool was useful in structuring process of knowledge elicitation by focusing on the relevant domain knowledge, which of course leads to accelerate building knowledge base system in such tasks.

References

Abdelhamid, Y. (1998). *Sharable and reusable Infra-knowledge for large scale knowledge-base systems*. PhD Thesis, ISSR.

Abdelhamid, Y., Hassan, H., & Rafea, A. (1997). An approach to automatic KBS construction from reusable domain-specific components. *Ninth Software Engineering and Knowledge Engineering Conference (SEKE'97)*, Madrid.

Breuker, J., & Van de Velde, W. (1994). *Common KADS library for expertise modeling reusable problem solving components*. Ios Press.

Citex (1997). Detailed Design of the Citrus Production Management in Field Agriculture of Citrus Fruits (CITIX) Version 2.2 (TR/CLAES/11/97.5).

David, J.-M., Krivine, J.-P., & Simmons, R. (1993). Second generation expert systems: A step forward in knowledge engineering. In J.-M. David, J.-P. Krivine, & R. Simmons (Eds.), *Second generation expert system* (pp. 3–23). Berlin: Springer.

Edrees, S., & Rafea, A. (1997). ITAKA: An integrated tool for automatic knowledge acquisition. Research and development in expert systems XIV. *Proceeding of Expert Systems 97, The Seventeenth SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, Cambridge, December.

Eriksson, E. (1991). *Meta-tool support for knowledge acquisition*. PhD Thesis. Linköping University, Sweden.

Eshelman, L. (1987). Mole: A knowledge acquisition tool that buries certainty factors. In J. H. Boos, & B. R. Gaines (Eds.), *The foundations of knowledge acquisition* (pp. 203–217). New York: Academic press.

Fujihara, H., Simmons, D. B., Ellis, N. C., & Shannon, R. S. (1997). Knowledge conceptualisation tool. *IEEE Transactions on Knowledge and Data Engineering*, *9*(2), 209–220.

Gruber, T. R. (1987). Acquiring strategic knowledge from experts. In J. H. Boose, & B. R. Gaines (Eds.), *The foundations of knowledge acquisition* (pp. 115–133). New York: Academic Press.

Hazman, M. (1999). *Automatic knowledge acquisition tool for scheduling systems*. MSc Thesis, ISSR.

KROL (1998). *KROL User Guide*, UM/CLAES/3/98.1.

Krueguer, C. W. (1992). Software reuse. *ACM Computing Surveys 1993*, *24*(2).

Marcus, S., & McDermott, J. (1989). SALT: A knowledge acquisition language for propose-and-revise systems. In B. G. Buchanan, & D. C. Wilkins (Eds.), *Reading in knowledge acquisition and learning automating the construction and improvement of expert systems* (pp. 263–281). Los Altos, CA: Margan Kaufmann Publishers.

Rafea, A. (1994). Agricultural expert systems developed in Egypt. *Proceedings of International Conference on Expert Systems for Development, ICES-94*, Bangkok, Thailand, March.

Rafea, A., El-Azhari, S., Ibrahim, I., Edres, S., & Mahmoud, M. (1995). *Experience with the Development and Deployment of Expert Systems in Agriculture Proceedings of IAAI-95 Conference*. Montreal, 19–25 August, Canada.

Rafea, M., & Rafea, A. (1998). Automatic expert system development tool based on hierarchical classification generic task. *The Fourth World Congress on Expert Systems*, Mexic, March 16–20.

Rafea, M., Shaalan, K., & Rafea, A. (1997). Towards a knowledge representation language based on open architecture model (OAM). *Fifth International Conference on Artificial intelligent applications*, Cairo, 27 February–2 March.

SICStus (1995). SICS Programming Systems Group, *SICStus Prolog User's Manual*, Swedish Institute of Computer Science, June 1995.

Tomatox (1995). Generic Irrigation Design Applied in Tomato Crop (TR/CLAES/72/99.5) May.