Automatic Generation of Explanation for Expert Systems implemented with Different Knowledge Representations

AHMED FOUAD SAID[+], AHMED RAFEA[*], SAMHAA R. EL-BELTAGY[Ω],
HESHAM HASSAN[Ω]
[+]Central Lab for Agricultural Expert Systems
Ministry of Agriculture and Land Reclamation.
El-Nour St., P.O. Box 438, Dokki, Giza
[*]Computer Science Department
American University in Cairo
P.O. Box 74 New Cairo 11835
[Ω] Computer Science Department
Cairo University
5 Tharwat Street, Orman, Giza 12613
EGYPT
ahmed@claes.sci.eg, rafea@aucegypt.edu, samhaa@acm.org, hesham@claes.sci.eg

Abstract: - Humans seem to have a natural instinct for wanting to understand and make sense of their environment and things in it. In expert systems (ES), explanation can be used to clarify the reasoning process to users such that they can gain a better understanding of how the system functions. With the help of good explanation facilities, a user can know why an ES is asking a particular question, how the expert system will act if given a certain input, and how the ES reaches a particular conclusion. This is especially important when an ES application is used as a high level advisor to professionals who must retain responsibility for the decisions which are made. However, most ES explanation components require acquiring additional knowledge for explanation, thus increasing the effort of implementing an ES with explanation capabilities. The primary goal of this work is to present a methodology for automatically generating explanations during and at the end of the reasoning process. The developed explanation components can deal with different knowledge representation schemes that are used by problem solving methods namely, the "generate and confirm hypotheses" that is based on the CommonKADS methodology[1], and the routine design generic task[2]. As a proof of concept the explanation components were developed and integrated into the agricultural expert system generic tool (AESGT)[3] .The developed explanation components can be easily reused with expert systems developed by the tool to automatically generate explanation for the reasoning process.

Key-Words: - Expert Systems, Explanation, Knowledge Engineering, Plant Protection

1 Introduction

Explanation is an important part of the human-machine interface in expert systems. In fact, the ability to provide an explanation for its reasoning is a distinguishing, important and powerful feature of expert systems. Explanations are most important in domains where the systems are used as high level advisors to professionals who must retain responsibility for the decisions which are made (e.g. medicine). Explanations are also important to users in a number of other circumstances. For example, when the user perceives an anomaly, when they want to learn, or when they need a specific piece of knowledge to participate properly in problem solving. Explanations, when suitably designed, have been shown to improve performance and learning and result in more positive user perceptions of a system. Because of its importance, explanation has been addressed and implemented in a number of systems notable examples of which are NEOMYCIN [4], XPLAIN [5], MPA [6], and WEBEXPLAIN [7]

However, most of the work in this area either completely ignored the problem solving method (PSM) used in developing the ES, or focused on a single PSM. These limitations are the motivation behind this work which proposes an explanation framework which aims to deal with different PSMs that are used in the most popular ES methodologies like CommonKADS[1] and Generic Tasks[2]. As a proof of concept, proposed explanation components were developed and integrated into the

agricultural expert system generic tool (AESGT)[3]. The developed explanation components can be easily reused with expert systems developed by the tool to automatically generate explanation for the reasoning process.

This paper is organized as follows. Section 2: presents related work, section 3 briefly describes the AESGT architecture with the added explanation facilities, section 4 outlines knowledge representation and reasoning in AESGT, section 5 describes the added explanation facilities in details, section 6 presents a case study and finally, section 7 concludes this paper and presents future directions.

2 Related Work

Explanation facilities have generally been added to knowledge-based systems to explain the inference that occurs as the system reasons. "Why" questions allow users to ask the system why a given question is being asked, while "How" questions allow users to understand how a system has reached a certain conclusion. There are different ways in which these explanations can be provided. NEOMYCIN [4] for example, follows the trace of rules to answer "Why" and "How" questions. NEOMYCIN uses templates and canned text as a method for explanation generation. XPLAIN [5] is a digitalis therapy advisor which advises physicians on the level of the digitalis drug administrated to congestive heart failure patients. It organizes its knowledge in the form of a semantic network. It also has an explanation generator to answer "Why", and "How" questions based on automatic tracing of goal hierarchy that is generated during the reasoning process. A similar approach has been used in WEBEXPLAIN [7] to provide a "How" question facility in a web-based infrastructure using an extended UPML framework [8].

A mission planning assistant (MPA) [6] is a Generic Task program capable of explaining its problem-solving steps and its strategy. MPA was developed based on the Routine Design problem solver. MPA uses frames for knowledge representation where additional slots are added to agent definitions to hold text strings for describing the agents' goals. MPA puts "canned" text together to generate explanation taking into consideration the roles of the various agents specialists, plans, steps, etc.

3 AESGT Architecture with the Added Explanation Facilities

This section overviews the overall structure of the Agricultural Expert System Generic Tool (AESGT) after adding the explanation components. This tool was developed with the aim of accelerating the development time of Agricultural Expert Systems through the adoption of task specific templates. Figure (1) shows the overall structure of the tool.

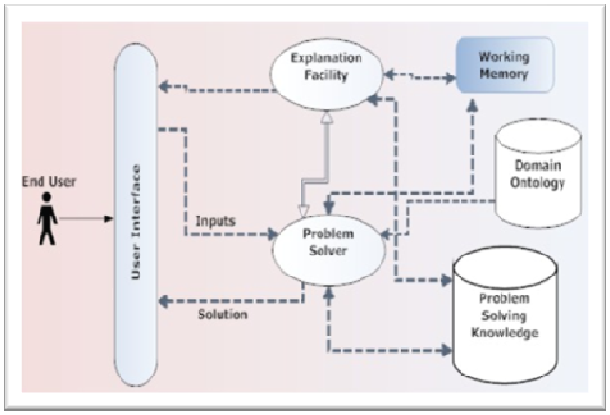

Fig.1: Overall Structure of the AESGT tool with the Explanation Facility

The tool supports four types of knowledge representations: rules, structured tables, frames hierarchy, and agent hierarchy. XML[10] is used as a representation language. The main components of the tool are as follows:

- **Problem Solvers**: Two PMSs based on two development methodologies are supported by this tool. These PSMs are:
  1. The Generate and Confirm hypotheses PSM: which is based on the CommonKADs methodology [9].
  2. The Routine Design PSM: which is based on the Generic Task methodology [2].
- **The Working Memory**: the working memory is the component that stores user inputs, and the results derived based on these inputs. Each result in the working memory is associated with the PSM used to derive it.
- **Explanation Facility**: the explanation facility is responsible for presenting the user with explanations obtained from the problem solving invoked by him/her, upon request.
- **The Domain Ontology**: Ontologies play an important role in many applications examples of which can be found in [11] [12] and [13]. In knowledge based systems, an ontology can offer a way for sharing knowledge. So this component, contains concepts, properties, values,

and relations used and shared by the two implemented PSMs.

- **The User Interface(UI)**: The UI is the means through which users can communicate with the tool.

4 Knowledge Representation and Reasoning

4.1 The Generate and Confirm Hypothesis Problem Solving Method

This PSM is based on the commonKADS Methodology[1], which differentiates three types of knowledge stacked as three layers namely: domain, inference, and task. Each of these is described in the following subsections.

4.1.1 The domain layer

This layer contains two knowledge bases (KB) which are the primary observations KB and the classification KB. The primary observations KB describes the knowledge needed to generate primary observations. In adapting this tool for agriculture this knowledge base contains the relationship between the growth stage and observations that appear on a specific plant part during this growth stage. This knowledge is represented as an XML tree. "GrowthStages" is the root node and each growth stage is a child node of the root. Plant parts that appear during this growth stage are children of this growth stage, and observations for each plant part are the leaves. Figure (2) shows a snapshot for the primary observations knowledge base.

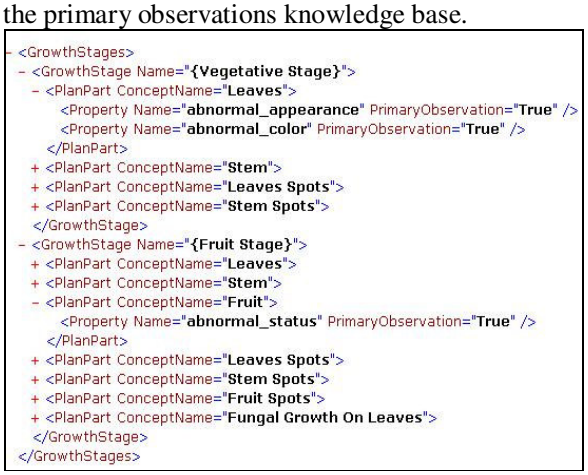

Fig.2: Snapshot of the Primary Observations Knowledge Base

The classification KB describes the relationship between observations and disorder causes (see figure (3)). This KB is used by three inference steps namely: generate primary observations, generate hypothesis and confirm hypothesis. Figure (4) describes the inference structure for generate and confirm hypothesis PSM. The knowledge representation used in this knowledge base is rule-representation where the premises are the observations and the conclusion is the cause of these observations. For each cause there is a set of rules to diagnose this cause. The rules are represented using XML where the cause is the element node, a rule is a child element, and the premises of this child node are the leaves.

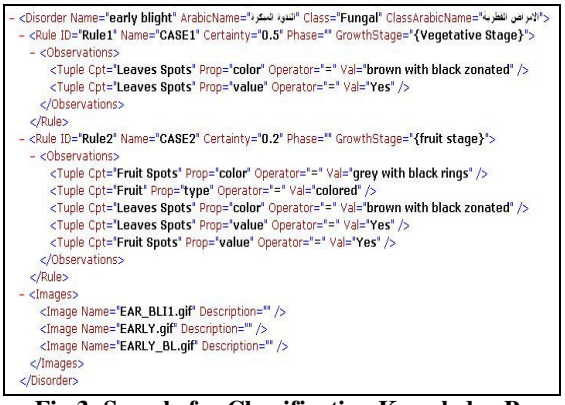

Fig.3: Sample for Classification Knowledge Base

4.1.2 The inference layer

This layer contains three inference steps namely: generate primary observation, generate hypothesis and confirm hypothesis. The following subsections describe each one of them:

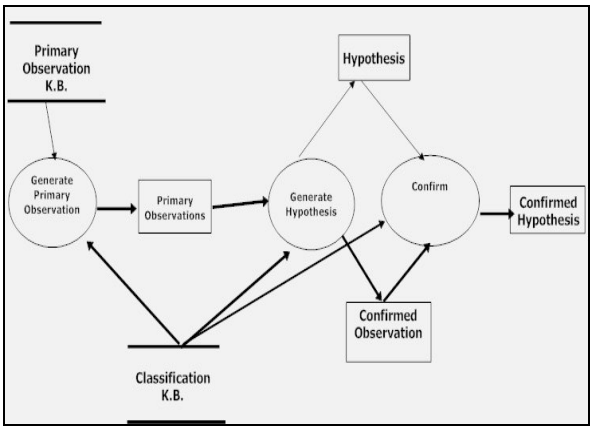

Fig.4: Inference Structure for Generate and Confirm Hypothesis

4.1.2.1 Generate primary observations

This inference step determines the primary observations based on the knowledge provided in the primary observation KB. The classification KB is used to generate the values of these observations used in the rules. It is possible to generate these values from the Ontology as well. But we preferred to generate them from the classification KB so as to

reuse the ontology in other applications that may use more values of the observations that do not appear in the underlying knowledge base.

4.1.2.2 Generate hypotheses

This inference step uses both the selected primary observations, and the classification KB, represented as rules where observations are the premises and disorders are the conclusions of the rules, to generate the suspected hypothesis. It also generates the observations required to confirm the suspected hypotheses. This inference method can be described as follows:

- Select all the disorders existing in the conclusion parts that have the primary observations provided by the user, in the premises of the rules.(These disorders are the generated hypotheses). Let us call the set of rules that have the primary observations, the **selected rules**.

- Identify the observations' attributes to confirm these hypotheses by selecting all observation attributes (not including the primary observations provided by the user as described in the previous step) that appear in the selected rules. Let us call these observations' attributes, **the identified observations attributes.**

- Generate a data entry screen containing the **identified observations attributes** and their possible values

4.1.2.3 Confirm hypotheses

This inference step uses the **identified observation attributes** and the classification KB to confirm the suspected hypothesis by using the following steps:

- Select all the suspected hypotheses appearing in the conclusion parts of the **selected rules** that have the **identified observations attributes**, appearing in the premises parts of these rules, whose values are provided by the user

- For the partially matched rules that have the selected hypotheses (from now on we will refer to those as **confirmed hypotheses**) as their conclusions, calculate the matching degree of each hypothesis using one of the partial matched rules by applying the following equation:

  Confirmed Hypothesis matching degree = number of matched observation premises / total number of observation premises

- The final matching degree for each confirmed hypothesis is then set to equal the maximum (the confirmed hypothesis matching degrees using each related rule)

4.1.3 The Task Layer

This layer describes how the system diagnoses the causes for the abnormal observations. Figure (5) shows the diagnosis task procedure.

4.2 The Routine Design Generic Task Problem Solving Method

This PSM is one of the generic tasks that can be used only when substantial experience is available (i.e. it can not be applied for totally novel situations).

```
TASK-METHOD diagnosis-through-generate-and-confirm;
 DECOMPOSITION:
INFERENCES: GeneratePrimaryObservation, GenerateHypothesis, ConfirmHypothesis;
ROLES:
        STATIC:
                PrimaryObservation knowledge;
                Classification Knowledge;
        INTERMEDIATE:
                PrimaryObservations;
                Suspected Hypothesis;
                ConfirmedObservations;
                ConfirmedHypothesis ;
CONTROL-STRUCTURE:
  GeneratePrimaryObservation (Primary Observation knowledge+ Classification
                                        Knowledge -> Primary Observation);
  GenerateHypothesis (Primary Observation + Classification Knowledge -> Suspected
                                Hypothesis);
  REPEAT
  ConfirmHypothesis (Confirmed Observation + Classification Knowledge+ Suspected
                                Hypothesis -> ConfirmedHypothesis);
  UNTIL "result = equal or no more Confirmed Observation ";
  END REPEAT
  IF result == equal THEN causes := hypothesis;
    ELSE "no solution found";
  END IF
```

Fig.5: Diagnosis Task Procedure

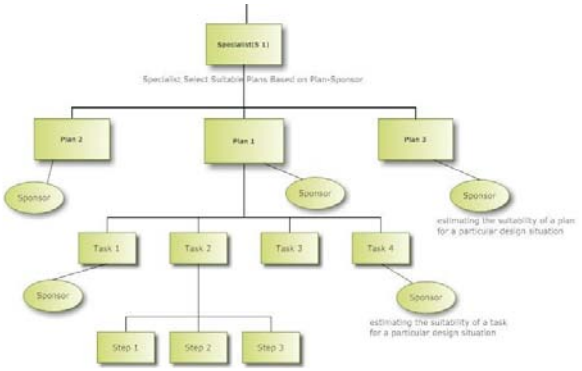

Fig.6: General Structure of a Routine Design Problem Solver

In the Routine Design PSM, a problem is divided into a collection of specialists, and each specialist is responsible for accomplishing a small part of the overall design. Following the Generic Task view, the specialist decides which of his plans should be carried out based on the plan constrains (Plan selector). Generally each specialist selects one of its plans. Likewise each plan selected by the specialist decides which tasks should be carried out based on the task constrains (Task selector), and each task has

a number of steps that determine the detailed description for it. Figure (6) shows the general structure of a proposed routine design problem solver. The following subsection describes the knowledge representation and the routine design task.

4.2.1 Knowledge Representation

Routine Design makes use of hierarchical structures of design specialists to perform design. The knowledge representation uses both the hierarchical frames, and structured tables for representing the routine design knowledge where each node in the hierarchical structures is represented as a frame, and the plan selector, task selector, and step knowledge is represented as a structured table.

The hierarchical structures are represented using XML where the specialist is the element node, and other nodes in the hierarchical structures are represented by XML element node in the XML tree. The plan and task frames contain two slots: the name of the plan or task, and the sponsor result. The values of these slots are filled by the methods, attached to these slots, which apply the structured table inference method on the plan or task selector knowledge. The step frame contains two slots: name, and the location where the result of applying this step, will be stored in the working memory. The slots are represented as attributes in the XML nodes. Figure (7) shows a sample of the routine design problem solver knowledge represented as XML for the land preparation expert system developed using the proposed tool.

Fig.7: Sample of the Routine Design Problem Solving Knowledge represented as XML

4.2.2 Routine Design Task

In routine design, top down control is typically used where a specialist selects the suitable plan based on the plan-selector. Figure (8) presents pseudo-code for the routine design task.

```
For each plan in specialist Plan-List
        // estimating the suitability of a plan for a particular design situation.
        Plan-SponsorValue =EvalautePlanSponsor();
        If Plan-SponsorValue is "Perfect" then
                For each Task in Plan Task-List
                // estimating the suitability of a task for a particular design situation
                        Task-SponsorValue =EvalauteTaskSponsor();
                        If Task-SponsorValue is "Perfect" then
                         For each Step in  Task Step-List
                                StepValue=EvalauteStepValue();
                                Add StepValue to working Memory;
                          End For each
                        End If
                End For each
        End;//we reach the suitable plan end the inference
        End If
End For each
```

Fig.8: pseudo-code for the routine design task

5 Explanation Facilities

The developed explanation components include the following explanation primitives: why asking, how, why-not-solution, why-solution, and what-if. In this paper we are going to concentrate on why-asking and how explanation primitives.

Before describing these two primitives, we like to state some development issues:

- The tool's working memory was modified to allow storing user inputs in the phase in which this input is acquired. This modification allows the tool to be sensitive to the state of the PSM.
- In order to integrate the explanation facility with the expert system, links are dynamically added to conclusions and questions. Each link calls a JavaScript method with parameters that indicates the current PSM, the phase, the explanation primitive, and other optional data depending on the explanation primitive and PSM (see figure 9) being used. The JavaScript method encodes the parameter as an XML string (Explanation Query), sends the explanation query to the server side using AJAX technology and receives the generated explanation.

```
<ExplanationQuery>
  <Sender>Pest Identfcation</Sender>
  <Lang>AR</Lang>
  <Respresenation>Rule-Based</Respresenation>
  <Question>Why-Asking</Question>
  <ConsultationPahse>During</ConsultationPahse>
  <OptionalData>
    <![CDATA[ <OpVar>
    <Item Concept="Plant" Property="Appearance"></Item>  </OpVar>
  </OptionalData>
</ExplanationQuery>
```

Fig.9: Example for an explanation query

5.1 The "Why Asking" Explanation Primitive

During the consultation phase the user might want to get an explanation for why the system is asking a certain question. Therefore, we associated the why asking link with each input to be provided by the user. This primitive is sensitive to where it has been invoked during the reasoning process. We have proposed different scenarios for responding to this explanation primitive based on what PSM is being used when the user asks the "why" question. In the following sections we are going to describe these scenarios in case of the Generate and Confirm Problem Solving Method and the Routine Design Problem Solving Method.

Scenario 1: Answering "Why Asking" in Generate and Confirm

During the consultation session, the PSM can be in one of two states: Generate Hypotheses, or Confirm Hypothesis.

- Generate Hypotheses State: In this case the explanation facility will respond by giving the set of hypotheses that will be suspected for each observation's possible values associated with the question for which the why-asking primitive is linked. Figure (10) shows the "Why Asking" Explanation generation algorithm in case of Generate Hypothesis State.

```
Receive Query from User Interface asking why the system needs one or more
values of the listed values associated with the current question
Related_Observations=Select all Observation from Classification K.B. which
contains the observation related to the current question
While Related observation is not empty do
{
        Current_Observation= Get_One (Related Observations)
        Assert Observation in the WM
        Current_Inference_Results = Call the inference
        Accumulated_Results= Acumulate _ResultsU (Current_Inference_Result)
        Clear the working memory
         Remove Current_ Observation from the list of related observation
}
        Generate a proper explanation in a
                suitable sentences
          format (Accumulated_Results)
```

Fig.10: "Why Asking" Explanation generation algorithm in case of Generate Hypotheses state

- Confirm Hypothesis State: In this case the explanation facility will respond by giving the set of suspected hypotheses and the set of hypotheses that will be confirmed for each observation's possible values associated with the question for which why-asking primitive is linked. Figure (11) shows "Why Asking" Explanation generation algorithm in case of Generate Hypothesis State.

Scenario 2: Answering "Why Asking" in Routine Design

During the consultation session, the PSM can be in one of three states: Plan-Selector state, Task-Selector state, and Step-state

```
Receive Query from User Interface asking why the system needs one or more
values of the listed values associated with the current question
Related_Observations=Select all Observation from Suspected Hypothesis
Classification K.B. which contains the observation related to the current question
While Related observation is not empty do
{
        Current_Observation= Get_One(Related Observations)
        Save current state of working memory
        Assert Observation in the WM
        Current_Inference_Results = Call the inference
        Accumulated_Results= Acumulate _ResultsU Current_Inference_Result
        Retrieve the working memory saved state
         Remove Current_ Observation from the list of related observation
}
  Generate a proper explanation in a
          suitable sentences
   format(Accumulated_Results)
```

Fig.11: "Why Asking" Explanation generation algorithm in case of Confirm Hypothesis state

- Plan-Selector state: In this state the explanation facility will respond by giving the consequences of selecting a certain plan if one of the values is provided to the attribute associated with the question for which why-asking primitive is linked.
- Task-Selector state: In this state the explanation facility will respond by giving the consequences of selecting a certain task for the selected Plan if one of the values is provided to the attribute associated with the question for which why-asking primitive is linked.
- Step-state: In this state the explanation facility will respond by giving the consequences of determining all steps attribute values for all the selected Tasks in the selected plan if one of the values is provided to the attribute associated with the question for which the why-asking primitive is linked.

5.2 "How" Explanation Primitive (Trace or Line of reasoning)

At the end of the consultation the user may want to get an explanation for how the system reached a certain decision or conclusion. Therefore, we associated the "How" link with the decision made by the system. In order to generate "How" explanation a reasoning memory was proposed to store key decisions taken by the inference methods. The reasoning memory contents depend on the used PSM. We have proposed different scenarios for responding to this explanation primitive based on the PSMs that are being used. In the following sections we are going to describe these scenarios in case of the Generate and Confirm problem solving method and the Routine Design problem solving method.

Scenario 1: Answering "How" in Generate and Confirm

During the reasoning process the PSM goes through two states: Generate Hypotheses and Confirm

Hypothesis. The reasoning memory keeps track of key decisions made by the PSM as follows:

- During the Generate Hypotheses phase the inference method stores in the reasoning memory the id's of matched rules responsible for generating suspected hypotheses and the corresponding suspected hypotheses.
- During the Confirm Hypothesis phase the inference method stores the id's of matched rules responsible for confirmed hypothesis and the corresponding confirmed observation

Figure 12 shows "How" Explanation generation algorithm in case of Generate and Confirm.

```
Generate HowExplanation()
{
        Display Primary-Observation ( );

/*A "How" button link appears in front of each suspected hypothesis for further
explanation. If the user clicks on the "how" button then the system will call
Genearte_Explanation_Suspected( Suspected Hypothesis) */
        Display Suspected Hypotheses ( );

        Display Confirmed –Observations ();

/*A "How" button link appear in front of each confirmed hypothesis for further
explanation. If the user clicks on the "how" button then the system will call
Genearte_Explanation_Confirmed (Confirmed Hypothesis)*/
        Display Confirmed Hypotheses

}

Genearte_Explanation_Suspected(Suspected Hypothesis)
{
/* Display the related primary observations to the suspected hypothesis together with
  the rules id's in which they appear as premises. If the user clicks on one of the rule id
  containing the displayed observation the rule details will be displayed */
        Display Related Primary Observation (Suspected Hypothesis);

}

Genearte_Explanation_Confirmed(Confirmed_Hypothesis)
{
    Related_Observations= Select all Observations from reasoning memory which are
                          related to the current confirmed hypothesis;
    While Related_Observations is not empty do
    {
        Current_Observation= Get_One(Related_Observations);
        Related_RulesIds= Select all Rules Id's from reasoning memory which are
                          related to the Confirmed_Hypothesis and
                          Current_Observation
        Matching_Degree = Select Matching_Degree from reasoning memory which are
                          related to the Confirmed_Hypothesis and
                          Current_Observation
/* Display current observation related to the Confirmed_Hypothesis together with the
  Rules Id's, and matching degree. If the user clicks on one of the rule id containing the
  displayed observation the rule details will be displayed */
        Display  Current_Observation ();
            Remove Current_ Observation from the list of related observation
        }

}
```

Fig.12: "How" Explanation generation algorithm in case of Generate and Confirm

Scenario 2: Answering "How" in Routine Design

During the reasoning process the PSM goes through three states: Plan-Selector Inference, Task-Selector Inference, and Evaluate Step Inference. The reasoning memory keeps track of key decisions made by the PSM as follows:

- During the Plan-Selector phase the inference method stores in the reasoning memory the index of the matched row in the plan-sponsor, and the selected plan.
- During the Task-Selector phase, the inference method stores in the reasoning memory the index of the matched row in the task-sponsor, selected task, and selected task plan.
- During the Evaluate step phase, the inference method stores in the reasoning memory the index of the matched row in the step-decision table, step name, task name, and the plan name.

Figure (13) shows "How" Explanation generation algorithm in case of Routine Design.

```
Generate HowExplanation()
{
        Display_User_Inputs();   //fed to the system during the Plan-Selector phase

/*A "How" button link appears in front the selected plan for further explanation. If the
user clicks on the "how" button then the system will call Genearte_Explanation_Plan (
SelectedPlan) */
        Display Selected Plan();
        Display_User_Inputs() // fed to the system during the Task-Selector phase

/*A "How" button link appears in front of each selected task for further explanation. If
the user clicks on the "how" button then the system will call
Genearte_Explanation_Task( Selected_Plan,Selected_Task) */
Display Selected Tasks

For each Current_Task in selected-tasks
  Display Name for Current_Task
    For each Current_Step in Current_Task

        /*A "How" button link appears in front of each selected task for further
explanation. If the user clicks on the "how" button then the system will call
Genearte_Explanation_ StepValue ( Selected_Plan,Selected_Task,Current_Step) */
        Display CurrentStep Name, and Value
    End For Each
 End For Each
}
Genearte_Explanation_Plan(Selected_Plan)
{
    Display _User_Inputs() //Related to Selected_Plan acquired during Plan-Selector
    phase
    Display Detail for matched row in the plan-sponsor
}
Genearte_Explanation_Task(Selected_Plan ,Selected_Task)
{
    Display _User_Inputs() //Related to Selected_Plan , and Selected_Task acquired
    during task-Selector phase
    Display Detail for matched row in the task -sponsor
}

Genearte_Explanation_StepValue(Selected_Plan ,Selected_Task,Selected_Step)
{
    Display _User_Inputs() //Related to Selected_Step acquired during step phase
    Display Detail for matched row in the step-decision-table
}
```

Fig.13: "How" Explanation generation algorithm in case of Routine Design.

6 Case Study

As mentioned earlier in this paper, the developed explanation components were implemented as an extension to the AESGT tool, and these components have been successfully implemented and used to augment a real agricultural expert system with different kind of explanations.

Example of the generated explanation for the "Why-Asking" primitive for the tomato disease diagnosis expert system in the generate-hypothesis phase is shown in (Figure 14), and for the confirm-hypothesis phase is shown in (Figure 15). The bold words in all examples highlight words that were inserted dynamically in the explanation template

Figure 16 shows another example of the generated explanation for the "Why-Asking" primitive for the wheat expert system in the plan-selector phase where the “land preparation” specialist is used.

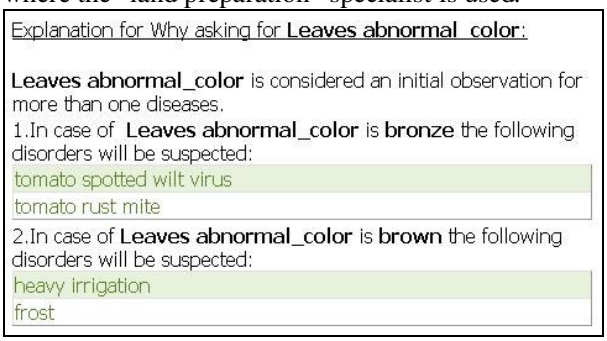

Fig.14: An Example of "Why-Asking" primitive in the Generate-Hypothesis Phase

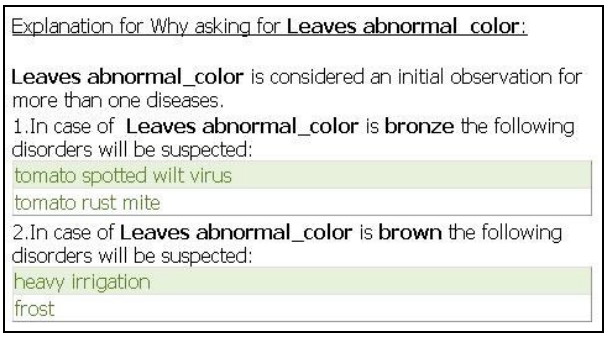

Fig.15: An Example of "Why-Asking" primitive in the Confirm-Hypothesis Phase

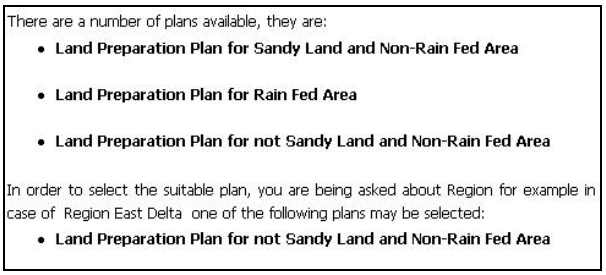

Fig.16: an Example of "Why-Asking" primitive in Plan-Selector Phase.

Figure 17 shows an example of generated explanation for the "How" primitive for tomato disease diagnosis expert system where the Confirm and Generate PSM is used. It is important to note that the user can get further explanation for the generated explanation as shown in the figure.

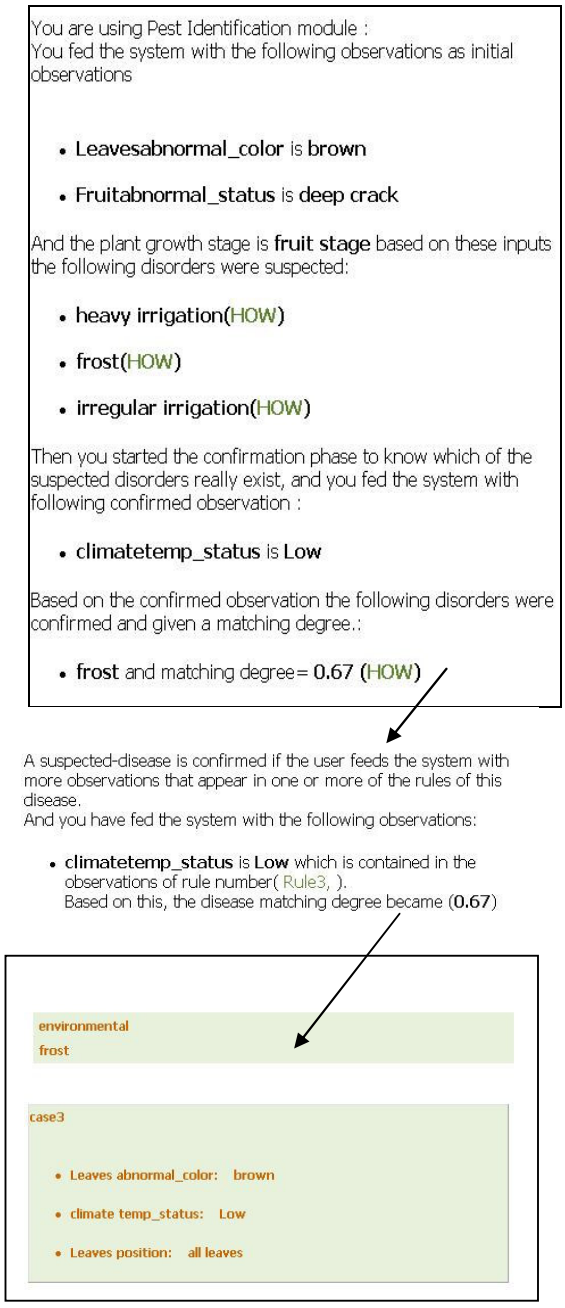

Fig. 17 An example of generated explanation for "How" primitive for Confirm &Generate PSM

7 Conclusion

In this paper, we've presented reusable explanation components that answer "How" and "Why Asking" types of questions. The developed components have the following features:

- Provide generic templates for generating answers to "Why Asking" , and "How" types of questions. The designed templates are suitable for the "generate and confirm" problem solver method (PSM) that is based on the CommonKADS methodology, and the routine design generic task (RD GT).
- Handle different knowledge representations supported by CommonKADS and the routine design generic task approaches
- Achieve re-usability on the knowledge level as the suggested methodology uses the same knowledge that was used to solve a problem for explanation purposes as well.

As a proof of concept, the proposed explanation components were integrated into the agricultural expert system generic tool (AESGT). These components were then capable of handling the generic knowledge modeling approaches provided by the tool, and the different knowledge representation models supported by these approaches, thus reducing the implementation efforts needed to develop an explainable plant protection expert system. For explanation generation we have used the reasoning process embedded in the problem solving method and the same knowledge that was used to solve the problem. We've used a web-based infrastructure for explanation which could be useful in the development of problem solvers in general. Future planned improvements to the AESGT explanation facility include the addition of more explanation facilities such as "what-if" and "why-not" utilities. We also want to enlarge the explanation scope such that it would encompass the needs of knowledge engineers and domain experts. In that case, further research will be conducted to configure explanation presentation according to the type of user and his/her explanation interests. Specifically, means for adapting and personalizing explanations and their presentation depending on the user of the system will need to be investigated.

References:

[1] A.TH. Schreiber B.Wielinga J.M. Akkermans W. Van DE Velde and R. de Hoog, CommonKADS: A comprehensive methodology for KBS development, IEEE Expert, Vol.9, No.6, 1994, pp. 28-37

[2] B. Chandrasekaran, Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design , IEEE Expert, 1986, pp. 23-30

[3] Z. Abdul-Hadi et al., Rapid generation of plant protection expert systems, Proceedings of the 4th World Congress on Computers in Agriculture and Natural Resources, 2006, PP. 282-287

[4] W.J. Clancey, From GUIDON to NEOMYCIN and HERACLES in Twenty Short Lessons: ORN Final Report 1979-1985 , AI Magazine, Vol.7, No.3, 1986, pp. 40-60

[5] R. Neches W.R. Swartout and J.D. moore, Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of Their Development, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol.SE-1l, No.11, 1985, pp. 1337-1351.

[6] B. Chandrasekaran A. Keuneke and M. Tanner, Explanation in Knowledge Systems: The Roles of the Task Structure and Domain Functional Models, Proc. Workshop Task Based Explanation, 1992, pp. 599-626

[7] V. Pinheiro V. Furtado P.P. da Silva and D.L., WebEXPLAIN : A UPML Extension to Support the Development of Explanations on the Web for Knowledge-Based Systems, Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE'06), 2006,

[8] D. Fensel et al., The Unified Problem-Solving Method Development Language UPML, Knowledge and Information Systems An International Journal, , , 2003,pp. 83-127

[9] A. El-Korany E. El-Azhary and M. Yehia, An Approach for building generic diagnosis model in agricultural domain, Fifth International Workshop on Artificial Intelligence in Agriculture, Vol.5 , , 2004, pp. 75-80

[10] http://www.w3c.org/TR/REC-xml

[11] Y-H Kuo, C-S Lee, S-M Guo, and F-T Tu, "Apply FNN Model to Construct Ontology-based Q&A System," WSEAS Transactions on Communications, vol. 3, Issue 1, pp. 328-335, Jan. 2004

[12] C-S Lee, J-C Du, Z-W Jian,Y-H Kuo, and C-K Hung "Ontology-based Measurement and Analysis Web Service for Supporting CMMI Level 2 Assessment," WSEAS Transactions on Information Science and Applications, vol. 1, Issue 6, pp. 1569-1574, Dec. 2004.

[13] SY Yang, How does ontology help web information management processing, WSEAS Transactions on Computers, Vol. 5, issue 9, pp. 1843 -1850, Sept. 2006.