AESGT Extension to support Explanation Facilities That Answers "How", and "Why Asking" Types of Questions

Ahmed Fouad Said  
Central Lab for Agricultural Expert Systems  
Agricultural Research Center  
Ministry of Agriculture and Land Reclamation.  
El-Nour St., P.O. Box 438, Dokki  
Giza, Egypt

Ahmed Rafea  
Department of Computer Science  
The American University in Cairo  
113 Kasr Al Aini Street,  
P.O.Box 2511,  
Cairo, Egypt

Hesham Hassan  
Faculty of Computers and Information  
Cairo University  
5 Tharwat Street, Orman,  
Giza 12613, Egypt

Samhaa R. El-Beltagy  
Faculty of Computers and Information  
Cairo University  
5 Tharwat Street, Orman,  
Giza 12613, Egypt

Abstract: - Plant protection is one of the most important applications in agricultural domain. An agricultural Expert system Generic Tool (AESGT) [1] has been developed for rapid generation of plant protection expert system. AESGT provides generic knowledge modeling approaches namely: CommonKADS[2] and Generic Tasks[3], but This tool doesn't have an explanation facility, which is an important component in any expert system (ES), as is used to clarify the reasoning process to the users and hence improve the understandability and acceptability of the expert system. This paper describes our experience in augmenting the AESGT tool with "How" and "Why Asking" explanation components for different kind of knowledge representation schemes that the AESGT tool is using. The developed explanation components can deal with different knowledge representation schemes that are commonly used by the modeling approaches mentioned above (e.g., tables, structured tables, hierarchy, constraints and rules), and handle the task level explanation. As well, our explanation components have been successfully implemented and used to augment a real agricultural expert system with different kind of explanations.

Keywords: Expert System, Explanation, Knowledge Engineering, Plant Protection.
1 Introduction
Providing explanation in ESs applications is an important issue in man-machine interface. Human seems to have a natural instinct for wanting to understand and make sense of their environment and things in it. As well, explanation is used to clarify the reasoning process to users to improve the understandability and with the help of good explanation user can know what the ES does, how the ES works, why the ES do these specially when this application is used as a high level advisors to professionals who must retain responsibility for the decisions which are made. Therefore, recognition of the importance of explanation components for reasoning systems has existed in number of fields for many years, for example NEOMYCIN [4], XPLAIN [5], MPA [6], and WEBEXPLAIN [7]

Most of the work in this area either completely ignored the problem solver method (PSM) used in developing ES, or focused on a single PSM. These limitations motivated us to propose an explanation framework aim to deal with different PSM that are used in the most popular ES methodologies like CommonKADS, and Generic Tasks.

This paper is organized as follows. Section 2: Related Work. Section 3: AESGT Architecture with Explanation Facility Added. Section 4 Knowledge Representation and Reasoning. Section: 5 Explanation Facilities. Section 6: case study. Section 7: conclusion

2 Related Work
Explanation facilities have generally been added to knowledge-based system to explain the inference that occurs as the system reasons. NEOMYCIN [4] follows the trace of rules to answer the "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 as semantic network. It 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 Routine Design. MPA uses frame as a knowledge representation where additional slots are added to the agent definitions to hold text strings for describing the agents' goals. MPA puts the "canned" text together to generate explanation taking into consideration the roles of the various agents specialists, plans, steps, .. etc.

3 AESGT Architecture with Explanation Facility Added
This section overviews the overall structure of the agricultural Expert system Generic Tool (AESGT) after adding the explanation components. As shown in figure (1) describes this overall structure. The tool consists of the following components: Problem Solving Knowledge, Problem Solver, Working Memory (WM), Explanation facility, Ontology and User Interface.

Fig.1: Overall Structure of the AESGT tool with Explanation Facility
This tool supports four types of knowledge representations: rules, structured tables, frames hierarchy, and agent hierarchy. XML[10] is used as a representation language.

- **Problem Solver:** Two PMSs based on two development methodologies are supported by this tool. These PSMs are:
  1. Generate and Confirm hypotheses PSM: which is on CommonKADS methodology [9].
  2. Routine Design PSM: which is based on Generic Task methodology [3].

- **Working Memory:** contains the user input, and the derived results. The PSM used to derive these results are associated with each piece of these results.

- **Explanation Facility:** responsible of replying explanation coming from problem solving method which is asked by the end user.
4 Knowledge Representation and Reasoning

4.1 Generate and Confirm Hypothesis

Problem Solver Method

This PSM is based on commonKADS Methodology, which differentiate three types of knowledge stacked as three layers namely: domain, inference, and task layers.

4.1.1 Domain Layer

This layer contains two knowledge bases (KB) namely: primary observation KB and classification KB.

- **Primary observation KB** is describing the knowledge needed to generate primary observation. In adapting this tool for agriculture this knowledge contains the relationship between the growth stage and observations that appear on a specific plant part that appear during this growth stage. This knowledge is represented as an XML tree. GrowthStages is the root node, each growth stage is a child node of the root, plant parts that appear on this growth stage are children of this growth stage, and observation for each plant part are the leaves. Figure (2) shows a snapshot for primary observation knowledge base.

- **Classification KB** describes the relationship between observations and disorders causes (see figure (3)). This KB is used by three inference steps namely: generate primary observation, generate hypothesis and confirm hypothesis. 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.

4.1.2 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.2: Snapshot for Primary Observation Knowledge Base

![Fig.2: Snapshot for Primary Observation Knowledge Base](image)

Fig.3: Sample for Classification Knowledge Base

![Fig.3: Sample for Classification Knowledge Base](image)

Fig.4: Inference Structure for Generate and Confirm Hypothesis

![Fig.4: Inference Structure for Generate and Confirm Hypothesis](image)

4.1.2.1 Generate Primary Observation

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 observation used in the rules. It is possible to generate these values from Ontology as well. But we preferred to generate them from the classification KB to reuse the ontology in other applications that may use more values of the observations that does not appear in the underlying knowledge base.

4.1.2.2 Generate Hypothesis
This inference step uses both the selected primary observations, and classification KB to derive the suspected hypothesis, and also generate the confirmed observations required only to confirm the suspected hypothesis. The classification KB is used to generate the suspected hypothesis by using the following steps:

- Select all the suspected hypotheses that have the confirmed observation, provided by the user, in the premises of the rules having these hypotheses as their conclusions.
- To confirm the suspected hypothesis the matched rules for the suspected hypothesis are used to generate the confirmed observation by select all observation that appear on the matched rules in condition that this observation is not a primary observation, then generate the values of these observation that are used in the matched rules.

4.1.2.3 Confirm Hypothesis
This inference step uses the selected confirmed observation and classification KB that contain the suspected hypothesis generated from the previous step to confirm the suspected hypothesis by using the following steps:

- Select all the suspected hypotheses that have the confirmed observation, provided by the user, in the premises of the rules having these hypotheses as their conclusions.
- For the partially matched rules that have the selected hypotheses (we will call them now on, confirmed hypotheses) as their conclusions, calculate the matching degree of each hypothesis using one of the partially matched rules by applying the following equation:
  
  Confirmed Hypothesis matching degree = number of matched Observations premises / total number of observation premises

- The final matching degree for each confirmed hypothesis = max( the confirmed hypothesis matching degrees using each related rule)

4.1.3 Task Layer
This layer describes how the system diagnoses the causes for the abnormal observation. Figure (5) shows diagnosis task procedure.

4.2 Routine Design Generic Task Problem Solver Method
This PSM is one of the generic tasks that can be used only when a substantial experience is available (But can not be applied for totally novel situations).
knowledge representation and the routine design task.

### 4.2.1 Knowledge Representation

Routine Design makes use of hierarchical structures of design specialist to perform design. The knowledge representation used both the hierarchical frames, and structured table for representing the routine design knowledge where each node in the hierarchical structures is represented as frame, and the plan selector, task selector, and step knowledge is represented as 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 routine design problem solver knowledge represented as XML for the land preparation expert system developed using the proposed tool.

![Fig.7: Sample of Routine Design Problem Solver Knowledge represented as XML](image)

### 4.2.2 Routine Design Task

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

```java
// For each plan in specialist Plan-List
// estimating the suitability of a plan for a particular design situation.
Plan-SponsorValue = Evaluate(PlanSponsor());
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 = Evaluate(TaskSponsor());
if (Task-SponsorValue is Perfect) then
for each Step in Task Step-List
StepValue = Evaluate(StepValue);
Add StepValue to working Memory;
end For each
end If
end For each
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 working memory is modified to allow storing with the user input the phase in which this input is acquired. This modification allows the tool to be sensitive to the state of 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 indicate the current PSM, phase, the explanation primitive, and other optional data depend on the Explanation primitive and PSM (see figure 9). The JavaScript method encodes the parameter as an XML string (Explanation Query), send the Explanation Query to the Server side using the AJAX technology and receive the generated explanation.

![Fig.9: Example for explanation query.](image)

### 5.1 "Why Asking" Explanation Primitive
During the consultation phase the user wants 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 ask the "why" question.

In the following sections we are going to describe these scenarios in case of Generate and Confirm problem solving Method and Routine Design Problem solving Method.

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

During the consultation session, the PSM get into one of two states: Generate Hypothesis, and Confirm Hypothesis.

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

```plaintext
Receive Query from User Interface asking why the system needs one or more values of the list of values associated with the current question
Related_Observations=Select all Observation from the list of related hypothesis

While Related_observation is not empty do
{
  Current_Observation = Get One (Related_Observations)
  Accumulated_result = Accumulate_Certainty_result (Current_Inference_Result)
  Remove Current Observation from the list of related observations
}
```

**Confirm Hypothesis State:** In this case the explanation facility will respond by giving the set of suspected hypothesis and the set of hypothesis that will be confirmed for each observation possible value 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.

```plaintext
Fig.10: "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 get into one of three states: Plan-Selector state, Task-Selector state, and Step-state.

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

At the end of consultation the user wants to get an explanation for how the system reaches a certain decision. 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 keep 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 what PSM are being used. In the following sections we are going to describe these scenarios in case of Generate and Confirm problem solving Method and Routine Design Problem solving Method.

**Scenario 1: Answering "How" in Generate and Confirm**

During the reasoning process the PSM goes through two states: Generate Hypothesis and Confirm Hypothesis. The reasoning memory keeps track of key decisions made by the PSM as follows:
During the Generate Hypothesis 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.

```java
Generate_How_Explination()
{
    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 Generate_Explanation_Suspected_Suspected_Hypothesis */
    Display_Suspected_Hypothesis();
    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 Generate_Explanation_Confirmed_Confirmed_Hypothesis */
    Display_Confirmed_Hypothesis();
}
Generate_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);
}
Generate_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_Rules_M = Select all Rules M’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 matched row in the plan-sponsor, and selected plan.

During the Task-Selector phase, the inference method stores in the reasoning memory the index of 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 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.

```java
Generate_How_Explination()
{
    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 Generate_Explanation_Plan(Selected_Plan) */
    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 Generate_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 Generate_Explanation_Step_Value(Selected_Plan,Selected_Task,Current_Step) */
    Display_Current_Step_Name, and Value
    End For Each
    End For Each
    Generate_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
    }
    Generate_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
    }
    Generate_Explanation_Step_Value(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 we have mentioned early in this paper, the developed components are an extension for 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 generated explanation for "Why-Asking" primitive for tomato disease diagnosis expert system in generate-hypothesis phase (Figure 14), and confirm-hypothesis phase (Figure 15). The bold words in all examples indicate that the words that are inserted dynamically in the explanation template.

**Explanation for Why asking for Leaves abnormal color:**
Leaves abnormal color is considered an initial observation for more than one diseases. 
1. In case of Leaves abnormal color is bronze the following disorders will be suspected:
   - Tomato spotted wilt virus
   - Tomato rust mite
2. In case of Leaves abnormal color is brown the following disorders will be suspected:
   - Heavy irrigation
   - Frost

**Fig. 14: an Example of "Why-Asking" primitive in Generate-Hypothesis Phase.**

**Explanation for Why asking for Leaves abnormal color:**
Leaves abnormal color is considered an initial observation for more than one diseases. 
1. In case of Leaves abnormal color is bronze the following disorders will be suspected:
   - Tomato spotted wilt virus
   - Tomato rust mite
2. In case of Leaves abnormal color is brown the following disorders will be suspected:
   - Heavy irrigation
   - Frost

**Fig. 15: an Example of "Why-Asking" primitive in Confirm-Hypothesis Phase.**

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

- There are a number of plans available, they are:
  - Land Preparation Plan for Sandy Land and Non-Rain Fed Area
  - Land Preparation Plan for Rain Fed Area
  - Land Preparation Plan for not Sandy Land and Non-Rain Fed Area

In order to select the suitable plan, you are being asked about region for example in case of Region East Delta one of the following plans may be selected:
- Land Preparation Plan for not Sandy Land and Non-Rain Fed Area

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

Figure 17 shows an example of generated explanation for "How" primitive for tomato disease diagnosis expert system where the Confirm&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 address the issue that expert system need explanation components. We developed a reusable explanation component that answers "How" and "Why Asking" Types of questions as an extension for AESGT. These components handle the generic knowledge modeling approaches provided by the tool, and the different knowledge representation supported by these approaches, thus reducing the implementation efforts needed to develop an explainable plant protection expert system. For explanation generation we used the reasoning process embedded in the PSM and the same knowledge that was used to solve the problem. We use a web-based infrastructure for explanation which could be useful in the development of problem solvers in general (e.g., web services, agents, etc.).

Future Planned improvements to the AESGT explanation facility include more explanation facilities like why-solution and why-not-solution to be added.

References: