INTELLIGENT INTERFACE AGENT FOR AGRICULTURAL EXPERT SYSTEMS

Yasser Abdelhamid[1] , Ahmed Rafea[2] , Nevien Galal[3]
{Yasser, Rafea, Nevien}@mail.claes.sci.eg

ABSTRACT

The acceptance of an expert system by the end user has been regarded as one of the major criteria of expert systems success. Expert systems are characterized by its requirement for heavy and complex interaction with the end user. This paper introduces an approach for interacting with multiple expert system applications through a unified domain-specific intelligent interface agent. The proposed intelligent interface agent communicates with different expert system applications transparently from the end user, and makes the necessary actions when needed. This approach increases the usability of expert system applications and introduces a new methodology for expert systems development using multi-agent systems (MAS).

The proposed approach has been applied by the Central Laboratory for Agricultural Expert Systems (CLAES) where two expert system applications – diagnosis and irrigation – have been interfaced by an intelligent interface agent. According to our proposed approach a number of advantages have been accomplished at both practical and theoretical levels.

KEYWORDS : Intelligent agents, Knowledge-based systems, Human-computer interaction, Distributed AI.

INTRODUCTION

Expert systems in a certain domain are not completely isolated from each other. It is a fact that one expert system may require the output of another to complete its process. For example in the domain of agriculture a treatment expert system requires the output of diagnosis expert system in the form of hypothesized or confirmed disorders. Also, fertilization expert system requires the output of irrigation expert system in the form of irrigation schedule in addition to other information to complete its reasoning process.

To overcome this difficulty, we have two alternatives. The first alternative is to merge dependant expert systems so they become one large system, and this violates the requirement of narrow scope expert systems; The more narrow scope we have, the more easily manageable knowledge we can handle. The second alternative is to ask the user for the required information by conducting separate sessions with other expert systems, and this alternative increases the complexity of interaction with the user.

We believe that we can utilize intelligent agent technology for solving this problem by developing an environment that makes the user interacts with multiple expert systems through a unified intelligent interface agent.

User interfaces are the most expensive component of software applications especially expert systems. Our approach provides the means for building a standard, and unified domain-specific interface for all expert system applications of the domain. This dramatically reduces the overall cost of expert system applications.

As mentioned before, domain-specific expert system applications are usually dependent on each other. The proposed intelligent interface coordinates between these expert system applications transparently from the user, and relieves him from the burden of communicating this information between different expert system applications.

The proposed approach simplifies the design of the expert system by detaching the user interface component from other functions of the expert system application.

Keeping different expert system applications in a standalone form supports modularity and easy

[1] Institute of Statistical Studies & Research (ISSR) – Cairo University, Giza, Egypt

[2] Faculty of Computer Science & Information – Cairo University Giza, Egypt

[3] Central Laboratory for Agricultural Expert systems (CLAES) Giza, Egypt

management of knowledge.

Eventually, we can add new expert system applications, that the proposed intelligent interface agent can provide its services without any change in the manner of interaction between the user and the system.

INTELLIGENT AGENTS

An intelligent agent is one that is capable of flexible autonomous action in order to meet its design objectives [1]. According to this definition, flexible means three things:
Reactivity: intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives;
Pro-activeness: intelligent agents are able to exhibit goal directed behavior by taking the initiative in order to satisfy their design objectives;
Social ability: intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

Intelligent agents are directed towards a single goal, but they possess more knowledge about reasoning within the space of their activity. Knowing when to use other resources (other agents), the preferences of the user or client, constructs for negotiation deals, and other abilities are the marks of an intelligent agent.

MULTIAGENT EXPERT SYSTEMS

In recent years there has been considerable interest in the possibility of building complex problem solving systems as a groups of cooperating experts [2]. A cooperating expert system is composed of a group of agents, each of which contains an autonomous knowledge based system. Typically, agents will have expertise in distinct but related domains. Agents co-operate together to solve a given problem and achieve the goals of individuals and of the system as a whole. An important research project in this direction was transforming expert systems into a community of cooperating agents [3] where the aim of this work was to construct a community of cooperating agents from two standalone and pre-existing expert systems. This was achieved by using the GRATE system, which is a general framework for constructing communities of cooperating agents for industrial applications. The cooperating community worked together to diagnose faults that occurred in the real particle accelerator process.

INTERFACE AGENTS

Intelligent interfaces are one of the applications of intelligent agents, through which the software agent works intimately with the user, functioning as a personal assistant. An intelligent interface can be defined as an intelligent entity mediating between two or more interacting agents who possess an incomplete understanding of each others' knowledge and form of communication [4]. A good interface will lead to better user/expert system interaction and task performance [5].

Interface agents employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular application [6]. Interface agents perform different kinds of tasks as they communicate with their local user or other agents, such as, information filtering and retrieval, scheduling of meetings, mail management, etc. There is a distinction between collaborating with the user and collaborating with other agents as in the case with collaborative agents. Collaborating with a user may not require an explicit agent communication language as the one required when collaborating with other agents. Interface agents support and provide assistance, typically to a user learning to use a particular application such as a spreadsheet or an operating system. The user's agent acts as autonomous personal assistant, which cooperates with the user in accomplishing some tasks in the application.

The objective of interface agents' research is to provide indirect management for human-computer interfaces. Current computer user interfaces only respond to direct manipulation, i.e. the computer is passive and always waits to execute highly specified instructions from the user. It provides little or no proactive help for complex tasks or for carrying out actions such as searching for information that may take an indefinite time. The goal is to migrate from the direct manipulation metaphor to one that delegates some of the tasks to software interface agents in order to accommodate novice users. The hypothesis is that these agents can be trusted to perform competently some tasks delegated to them by their users.

The main functions of an interface agent include:

1. Collecting relevant information from the user to initiate a task.
2. Presenting relevant information including results and explanations.
3. Asking the user for additional information during problem solving.

4. Asking for user confirmation, when necessary.

The knowledge held by interface agents:

1. A model of the user's goals and preferences pertaining to a task.
2. Knowledge of the relevant task assistants that can perform the task.
3. Knowledge of what must be displayed to the user and in what way.
4. Protocols for interacting with relevant task assistants.

AN EXPERT SYSTEM ENVIRONMENT BASED ON COLLABORATIVE AGENTS

The basic idea of our work is to convert the pattern of interaction between the user and a number of knowledge-based systems from that where the user interacts with each individual knowledge-based system separately, to another pattern where the user has only one interface, through which he can interact with multiple knowledge-based systems without having to worry about the requirements of each one.

Our approach is to model each KBS in the current environment as an agent with its own knowledge, and services. Two other special agents are introduced: Interface agent, and Coordinator agent. The interface agent is responsible for managing the interaction between the user and proposed environment, through handling request messages asking the user for a required input, and response messages displaying the reply. The coordinator is an agent that is intelligent enough to determine the user requirements by analyzing the users inputs, and to specify which KBS agent to contact for achieving these requirements, including calling intermediate agents for other services.

ARCHITECTURE OF THE PROPOSED ENVIRONMENT

As shown in Figure [1] the proposed architecture consists of a number of agents. The following is a brief description of the components that comprise the proposed architecture.

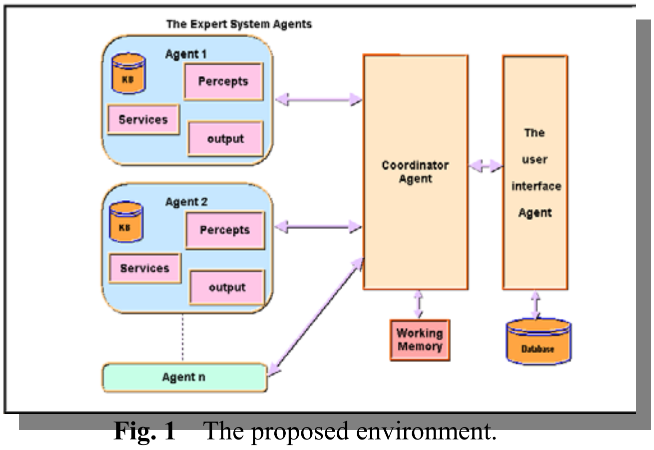

Fig. 1 The proposed environment.

User Interface Agent:

The internal structure of the proposed environment is completely transparent to the user. The only component that is visible is the user interface agent. The user interface agent has two main services: the first is to collect user input in the form of observations, events, data, or requirements, and sending these inputs to the coordinator agent. The second service is to receive and display the replies that could come back from the coordinator agent.

The Coordinator Agent:

The coordinator has its own knowledge about the KBS agents in its environment, and the services offered by each of them. It is stimulated by any message received from the user interface agent, or any of the KBS agents. So, any input introduced by the user interface agent, or a reply from a KBS agent will activate its inference process. The action of the coordinator is a message sent to one or more KBS agents, asking for their services, or a message to the user interface agent as a result of processing. This cycle will stop when no more services are required, or there are no more requests from the user.

KBS Agents:

The KBS agents are the components that are doing the real work in the proposed environment. In other words, they are achieving the requirements of the user in the form of services.

The Database:

The database component is used as a common store of static data that is often required by KBS agents. For example, in the domain of agriculture, the database is used to store data about farms like plantation data, climate data, soil data, and water data.

AGENT INFRASTRUCTURE

The proposed agent environment presumes an infrastructure within which the agents operate and interact. Two major components comprise this infrastructure: Common Ontology, and Communication Protocol.

Common Ontology:

Ontology has been a popular research topic and has been investigated by several AI research communities. An ontology is a formal explicit specification of a shared conceptualization.[7]. The main role of ontology in the knowledge engineering process is to provide the vocabulary of terms and relations used by the knowledge based system.

Making agents understand each other is a major challenge. Agents may have different meanings for the same concept, or they may have different concepts with the same meaning. To overcome this difficulty, there must be some sort of shared knowledge about the domain of discourse that made available to all components in the proposed environment.

In our proposed environment, this shared knowledge is provided through a common Ontology that is a set of concepts and attributes, as well as relations between these concepts and attributes. We used this ontology as the underlying background knowledge that is common for all participating agents.

Communication Protocol:

KQML is one of the pioneer research projects in the field of agent communication languages[8]. KQML provides a large set of primitives through which agents may tell facts to other agents, evaluate expressions for other agents or subscribe to services provided by other agents. KQML suffers from poorly defined semantics. As a result, many implementations have been introduced, but each seems unique. This makes communication difficult, and KQML agent may not be understood.

Arcol is another ACL based on speech acts[9], [10]. Arcol was the basis for the first version of the proposed standard of the Foundation for Intelligent Physical Agents (FIPA), and many of its components survive in the second version as well. Agents conforming to the FIPA specification can deal explicitly with actions. They make requests, and they can nest the speech acts.

In our proposed environment, we use HTTP as the underlying transport protocol, and we use XML language to formulate messages passing between different agents. According to this approach, messages are not tied to a particular implementation, and basically we are not in need to implement a standard communication language like KQML.

DESIGN OF THE PROPOSED ENVIRONMENT

After we have had a general idea bout the proposed approach, and the overall architecture of the proposed agent environment. The following is a detailed discussion of the components constituting the environment.

User Interface Agent:

The user interface agent has two sub-components; request model, and response model. The request model formulates the list of inputs that the user can provide during the session. The expected inputs are derived from the common ontology used by the knowledge-based system agents in the form of properties related to some concepts in the domain of discourse, and the acceptable values for each of these properties.

At any time during the session, the user can use this unified interface to convey some inputs to the system. These inputs can be events like rain or wind, observations on the plant like leaf color or root shape, current date, etc. The user doesn't know which knowledge-based system will reply to these inputs, or which will provide its services. Finally, the request model submits the user inputs to the coordinator agent in a request message.

The response model collects and displays the results or replies coming from the coordinator agent as a reply from one or more knowledge-based system agents. These results and replies can be classified into one of the following categories:

- Predicted disorder name(s).
- Confirmed disorder name(s).
- Treatment materials, quantities, dates and methods of using these materials.
- Irrigation schedule or part of it.
- Fertilization schedule or part of it.
- A plant care operation to be done.

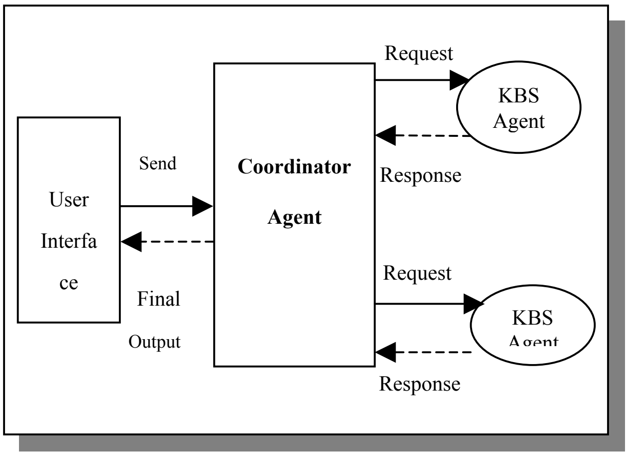

Fig. 2 The coordinator agent.

The Coordinator:

As shown in Figure [2], the coordinator manages the interaction between the user through the user interface agent, and the different knowledge-based system agents in the environment. This is done through messages sent to the various agents requesting for some service, or supplying some information back to the user interface agent.

The coordinator has the knowledge that makes it able to decide which service to be requested for the current session. This knowledge is represented as rules that work on the current property values stored in the working memory, and maps them to one or more services provided by other agents. Figure [3] displays a sample of this knowledge represented in XML format.


```
<Agent Name="diagnosis">
        <Tuple Cpt="leaves_observations" Prop="color/”>
        <Tuple Cpt="leaves_observations" Prop="shape/">
        <Tuple Cpt="stem_spot" Prop="exist/">
        <Tuple Cpt="stem_observations" Prop="shape/">
        <Tuple Cpt="fruit_observations" Prop="color/">
        <Tuple Cpt="fruit_observations" Prop="shape/">
</Agent>
```


Fig. 3 Sample of coordinator knowledge.

The coordinator interacts with the knowledge-based system agents in the following manner:

1. First, the coordinator formulates the input data existing in the working memory into an appropriate request message and sends it to the related agent(s).
2. Second, it receives the reply messages from participating agent(s), updates the working memory and starts a new cycle of sending new messages to the same or other agents if needed, by redirecting the results to them.
3. Third, it sends the final results to the user through the user interface agent in the form of response message.

Figure [4] displays a sample message sent to diagnosis agent. As we can see, the message has three parts, Header, Body, and Fault. The Header contains information about the sender, receiver agent, and the requested service. The Body contains working memory contents related to the designated agent. The Fault element is used to provide information about errors that occurred while processing the message. By nature this element can only appear in answers (response messages).

KBS Agents:

In our proposed environment, we imposed a certain structure on KBSs to achieve our goal for supporting intelligent communication between them. The main architecture of the KBS agents is:

The knowledge base: Which in its turn consists of Domain concepts, and Domain relations:

Domain concepts are structured collections of domain terms (e.g. concept and its properties and values). Concepts can have properties that are defined through their names and the values that they can take.

Domain relations represent the relation between concepts/properties defined in the domain of discourse, these relations are represented in the form of: Rules, tables, or functions.

Services: Which are the functions that the knowledge-based system agent provides. Each service has a unique name, percepts which are the inputs needed to stimulate the service (indicate when that service should be activated), and output of that service.

```
<Message>
<Header>
        <From>Coordinator</From>
        <To>Diagnosis_Agent</To>
        <Service>Generate_Hypothesis</Service>
</Header>

<Body>

<Tuple Cpt="Leaves_observations"Prop="color" Val= "Brown"/>
<Tuple Cpt="Stem_Observations" Prop="Shape" Val= "etiolated"/>
<Tuple Cpt="Stem_Spot" Prop="Exist" Val= "yes"/>

</Body>

<Fault>

</Fault>

</Message>
```


Fig. 4 A sample diagnosis agent message.

KBS agents are implemented as COM (Component Object Model) objects, so they can be accessed as web services. A number of methods have been added to these components so that knowledge base can be manipulated easily.

CASE STUDY

As we have mentioned early in this paper, the proposed architecture has been tested in the domain of agricultural expert systems. Two expert system agents have been implemented: Irrigation agent, and Diagnosis agent for cucumber cultivation under tunnels.

The irrigation agent calculates water requirement for the plant, and generates partial and full irrigation schedules. It is worth mentioning that some circumstances may drive the irrigation agent to revise its previously suggested irrigation schedule, like changes in the weather or having certain disorders.

The diagnosis agent analyses the input observations and farm data, and provides the user with information about the possibility of having a certain disorder(s), and according to extra information provided by the user, the diagnosis agent may confirm the existence of one or more of these disorders. Other agents (treatment agents) can be developed for providing suitable treatment schedules for the confirmed disorders specified by the diagnosis agent.

As shown in Figure [5], the user provides basic information about the case, in addition to some observations that according to the diagnosis agent's knowledge are manifestation for some disorders. At this point, only diagnosis agent is involved, and replies with the names of suspected disorders.

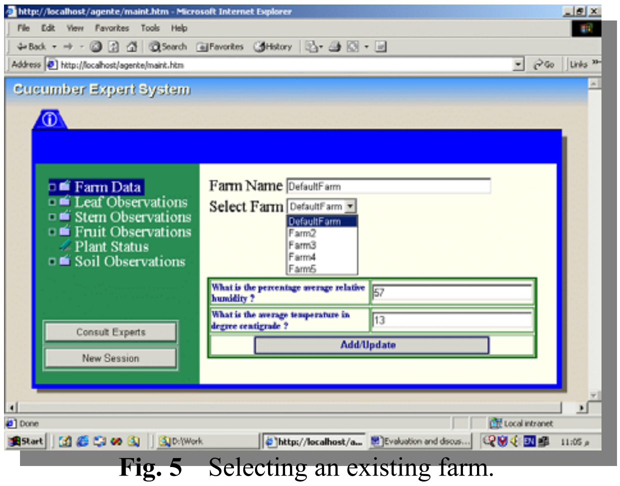

Fig. 5 Selecting an existing farm.

As shown in Figure [6], with extra observations provided by the user, the diagnosis agent will have the possibility of confirming one or more of these suspected disorders.

The coordinator agent receives the initial results of the diagnosis agent, and passes the results to other agents, in our case, the irrigation agent.

According to the irrigation agent's knowledge, the confirmed disorder may affect the quantity or schedule of irrigation. So, it revises the proposed schedule, and passes it back to the coordinator agent.

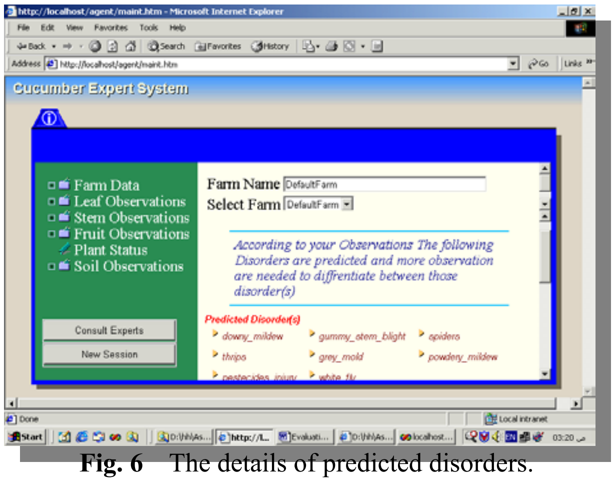

Fig. 6 The details of predicted disorders.

The coordinator agent tries to route the new information to expert system agents again, but this time no one replies.

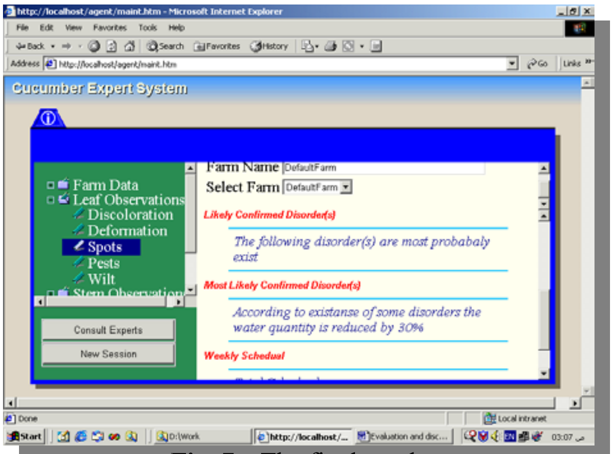

Fig. 7 The final results.

The coordinator agent considers this as a final reply, and reports it back to the user interace agent.

As shown in Figure [7] The user interface agent displays the results to the user with a fully detailed report coming from all agents involved in the session.

CONCLUSION

In this paper we have presented an agent-based architecture that enables knowledge-based system users to interact with a number of knowledge-based system applications through an intelligent interface agent.

Through the analysis and evaluation of the implemented prototype, we have come up with the following conclusions:

Using the presented architecture, the user deals with multiple knowledge-based systems as if they were only one. This approach helped in reducing the interaction between the user and the knowledge-based systems, by making the coordinator agent take over this interaction on behalf of the knowledge-based system user.

Building this standard and unified interface simplifies the design of knowledge-based systems by separating between the interface design and other functions of the system and provides a unified model of interaction between the user and KBSs without having to merge these KBSs into one large, hard to manage KBS.

The proposed architecture is open, in the sense that we can add new knowledge-based system agents without any change in the manner of interaction between the user and the proposed intelligent interface agent.

The proposed architecture is reusable, since we can apply the same environment on different domains of applications.

REFERENCES

[1] Wooldridge M. (1999). "Intelligent Agents", In G. Weiss, editor: Multi-agent Systems, The MIT Press.

[2] Jennings N. R., and Wooldridge M. (2001). “Agent-Oriented Software Engineering”, Handbook of agent technology, ed. J. Bradshaw, AAAI/MIT Press.

[3] Jennings N. R., Varga L. Z., Aarnts R. P., Fuchs J., and Skarek P. (1993). “Transforming Standalone Expert Systems into a Community of Cooperating Agents ”, Engineering Applications of Artificial Intelligence, 6 (4) 1993, 317-331.

[4] Chen C., and Roy R. (1998). “Knowledge-based system Technology: Knowledge-based system Interface”, in The Handbook of Applied Knowledge-based systems, Chapter 6, edited by Liebowitz, J.

[5] Kuo-Wei Su, hu-Hua Liu, and Sheue-Ling Hwang (2001). “A developed model of expert system interface (DMESI)”, Expert systems with Applications (20), pp. 337-346.

[6] Maes P. (1994). " Agent That Reduce Work and Information Overload", Communications of the ACM, 37(7), pp. 31-40.

[7] Gruber T.R. “A Translation Approach to Portable Ontology Specifications”, Knowledge Acquisition 5 (2), 1993, pp. 199-221.

[8] Finin T., Weber J., Wiederhold G., Genesereth M., Fritzson R., Mckay M., McGuir J., Pelavin R., Shapiro S., and Beck C. (1993). " Specification of the KQML Agent-Communication Language", By The DARPA Knowledge Sharing Initiative External Interfaces Working Group.

[9] Breiter P., and Sadek M. D. (1996). “A rational agent as a kernel of a cooperative dialogue system: Implementing a logical theory of interaction”. In ECAI Workshop on Agent Theories, Architectures, and Languages. Springer-Verlag, Heidelberg, Germany, pp.261-276.

[10] Sadek M. D. (1991). “Dialouge acts are rational plans”, In Proceedings of the ESCA/ETRW Workshop on the Structure of Multimodal Dialogue, Maratea, Italy, pp. 1-29.