A Model Based Architecture for Intelligent Interface Agents

Yanguo Jing, Keith Brown, Nick Taylor
Intelligent Systems Lab
Department of Computing & Electical Engineering
Heriot-Watt University, Edinburgh, UK
EH14 4AS
{ceyj, keb, nick}@cee.hw.ac.uk

Abstract

This paper presents a model based intelligent interface agent architecture. In this architecture, the domain model contains all the domain information of a target application; the user model contains all users' profiles. We focus on the domain model and user model design and construction. The prototype system and experiments are described using this architecture.

1 Introduction

The field of intelligent interface agents has emerged during the past few years to address the increasing complexity of current software systems. They can give timely, beneficial assistance to users by extracting and analyzing relevant information from the application domain knowledge and a user's profile. In some papers they are called personal assistants. Some of the existing interface agents focus on a user model based on a user's interaction history [Maes, 1994][Lau, 1999][Farrell, 2000]; some focus on user intent ascription [Brown, 1998]; Höök [1996] and Bomsdorf [1996] try to decompose user tasks to make mappings between tasks and interfaces.

We propose a Model based Intelligent Interface Agent (MIFA) architecture (see Figure 1) to support interaction between users, target application domains and interface agents. In this architecture, the Facilitator maintains a knowledge base of the capabilities of a collection of agents. Agents, including the intelligent interface agent, the target application and other agents communicate with each other through this Facilitator. The other agents include a user model management agent, a domain model management agent, etc.

A domain model (DM) contains an explicit representation of the target application. The intelligent interface agent (IIA) gets information from the DM, infers certain operations and sends certain instructions to the target application. The DM consists of a task model (TM), a procedural model (PM) and an associational model (AM). The TM structures tasks in an AND/OR graph. The PM is an executable representation of procedures, which are ordered sets of tasks or actions. The AM links situations to tasks or operations. These three components work together to represent the domain model.

In order to provide different assistance to different users with different kinds of request under different situations, the IIA also need to get the user's personal information. The user model contains all this kind of information, which consists of an Interaction History (IH), an interaction model (IM) and a user profile (UP). The IH records the user's interaction with the target application. The IM represents a user's interaction habits with the target application, which is inferred from the IH. The UP contains the static information about a user such as name, gender, knowledge level, etc. The intelligent interface agent (IIA) looks over a user's shoulder to keep track of the users' interaction with the target application, combine

Figure 1: The MIFA Architecture
the knowledge from the DM and the UM, decide a certain assistance. At the same time, the user model management agent constantly modifies the IH and infers a new IM and modifies the UP when it is necessary.

This paper focuses on the domain model (DM) design method and the Interaction Model (IM) inference method. It is organized as follows. The next section discusses the DM design method. After that, the UM design method including the IM inference method is presented. Then we provide an prototype system and an experiment based on this architecture. Finally, conclusions from this work are detailed.

2 The Domain Model Design Method

As mentioned before, IIA gets all the knowledge of the target application from the DM, which is made up of three parallel models - the TM, the PM and the AM. The diagram in Figure 2 gives an overview of the domain model design method in MfA. First of all, we construct the TM, the PM and the AM using a DM constructor developed to get the original models. Secondly, the original models are translated into rules using a developed DM translator. Finally, these rules are integrated to represent the DM.

2.1 The Task Model Design

The TM structures tasks in an AND/OR graph, every node in the graph being a task. A simple abstract example of a task model is shown in Figure 3. It describes the following relations: before users achieve task A1, they should either achieve task B1 or task B2; before users achieve task B1, they must achieve both task C1 and task C2; before users achieve task B2, they must achieve both task C3 and task C4.

2.2 The Procedural Model Design

A procedure is an ordered set of functions or tasks, possibly consisting of several sub-procedures. A procedural model is an executable representation of a procedure. A simple abstract example of a procedural model is shown in Figure 4. This example describes the following relations: for the task Do-A1, the corresponding procedure sequence is A1-1, A1-2, A1-3. The difference between a node in the procedural model and a node in the task model is that in the task model, order does not matter in a node's pre-condition, but in the procedural model, all the operations and the tasks are linearly ordered.

2.3 The Associational Model Design

An association is a correlation between a state (situation) and an operation (conclusion). The associational model in MfA is an executable representation of the association. A simple abstract example is shown in Figure 5. It states that when the target application is in State 1, the task should be achieved.
2.4 The Domain Model Translation and Integration

As stated earlier, the TM, the PM and the AM are translated into rules.

Each "OR" node in the Task Model is translated into $3+2*n$ rules, where $n$ is the number of tasks in the pre-condition.

1. Rule 1: Select one option from the pre-condition.
2. Rule 2: Get the selection
3. Rule 3: If the selection is invalid, then select again.

For each task node in the pre-condition there are two rules:

4. Rule 2+2*n: If selecting the nth option from the pre-condition, then try to achieve this selected task node.
5. Rule 3+2*n: If the selected task node has been achieved, then try to achieve the current task.

For each "AND" node in the Task Model there are $1+n$ rules where $n$ is the number of tasks in its pre-condition.

1. (1) For each task node in this pre-condition which has not been achieved then achieve it.
2. (2) If all of the task nodes in this pre-condition have been achieved, then achieve this task node.

Each procedure in the Procedural Model is translated into $1+n$ rules, where $n$ is the number of functions/tasks in this procedure.

(1) For each operation/task in this procedure, if it has not been achieved and all the operations or tasks before it in this procedure have been achieved, then achieve it.

(2) If all of the operations/tasks in this procedure have been achieved, then this procedure has been achieved.

The Associational Model translation into rules is quite straightforward [detail see Jing 2001a].

3 The User Model Design Method

The User Model (UM) contains information about the user. It consists of three parts - the Interaction History (IH), the interaction model (IM) and the user profile (UP). The IH records the user's interaction history with the target application. The IM represents a user's interaction habits inferred from the IH. The UP contains the relatively static information about a user such as name, gender, knowledge level, etc. This information is the main knowledge for the user classification, we will mention this in the conclusion section. The user model management agent is supposed to manage the UM including modify the IH, the IM and the UP.

3.1 The Interaction Model Construction

As a user interacts with a system, a lot of events are generated and operations undertaken. All these events and operations constitute this user's interaction history (IH). An interaction history consists of several sequences of operations, which are called action sequences. An action sequence is made up of a user's actions and system events. The IM is inferred from the IH to represents a user's interaction habits. At the moment, the IM inferring method takes into account the order of two actions (or events) in an operation sequence. There are two reasons. Firstly, as the number of the actions whose order is taken into account increases, the computational effort increases exponentially. Secondly, the order between two actions can be used to infer the order among several (more than two) actions, as is shown in the prototype and experiments section.

The method shown in Figure 6 is used to construct a user's IM based on the IH. As previously explained, this method only takes into account two actions’ order. These two actions constitute an ordered action pair. In the first step, we select ordered action pairs from the IH and in the...
second step, the final action pairs are selected, based on a statistically derived threshold from those ordered action pairs. Finally, these final action pairs are added to a graph one by one. The Prune Successor(s) and Prune Predecessor(s) algorithms are used to modify the graph. This graph is the final representation of the Interaction Model [detail see Jing 2001b].

Here is a simple abstract example: Suppose we have the following interaction history (IH), which consists of two action sequences:

Sequence 1: a1 -> a2 -> a3 -> a4.
Sequence 2: a3 -> a1 -> a2 -> a4.

Using the method shown in Figure 6, the IM is inferred by the following steps.

Step 1: We get the following ordered action pairs from Sequence 1: (a3, a4), (a2, a4), (a1, a4), (a2, a3), (a1, a3), (a1, a2) and the following ordered action pairs from Sequence 2: (a2, a4), (a1, a4), (a3, a4), (a3, a2), (a1, a2), (a3, a1).

Step 2: We set the statistical threshold to 100%, then we get the following final action pairs: (a3, a4), (a2, a4), (a1, a4), (a1, a2).

Step 3: The action pairs we get from the final action pairs are added to a graph one by one (we use an arrow to connect two actions in an action pair with the arrow pointing to the second action in the action pair). Step 4 and step 5 are used to prune the graph. For example when action pair (a1, a2) is added to the tree, we get a graph shown in Figure 7, but this is not an efficient graph, the leaf a1 has two paths to a4. Using the step 4 - Prune Successor, we get rid of the arrow between a1 and a4 (the reason why we don’t get rid of arrow between a1 and a2 is that if we do this, we will lost the ordered information between a1 and a2, but if we get rid of a1 and a4, we still can tell that a4 is a successor from a1).

After step 3, step 4 and step 5, we get the following tree shown in Figure 8, which is the IM for this user to represent his/her interaction habits. The detailed algorithms and their implementations can be found in Jing 2001b.
4 Prototype and Experiments

We implemented this method in a system called ARMD for diagnosis of eye disorders [Jing, 2001] using Java and Jess [Watson, 1997]. The main function of this system is to identify indicators of age-related macular degeneration (ARMD). A camera is used to obtain a raw image of a retina. There are two options: to diagnose with a validated artificial neural network (ANN) or to diagnose as a human expert. To diagnose with the validated ANN, a validated ANN must first be selected and then the diagnosis can be performed. If the ANN is not good enough (in term of its accuracy), it should be trained. After the ANN is trained, it should be assessed to evaluate its accuracy. To diagnose with a human expert, an expert must be available first and then the diagnosis can be performed. We develop the models for the ANN case.

4.1 The DM

The DM including the TM, the PM and the AM is designed using the DM constructor, then these models are translated using the DM translator into CLIPS format files, finally these files are integrated into one file to work as an DM inference engine inside IIA using Jess. The IIA looks over the user's shoulder and identifies the user's interaction with the target application, provide the DM inference engine with certain user's intention (in term of tasks and operations), the DM inference engine can provide IIA proper steps to achieve it. For instance, for the task "Get Diagnosis from ANN", we must achieve the task "Get ANN ready" and the task "Get a validated image", which can be represented in an AND/OR graph shown in Figure 9.

![Figure 9 A Part of the DM in ARMD system](image)

This task model is translated into the following three CLIP rules:

\[
\text{(Defrule dm-1 (Get_Diagnosis_from_ANN) (not (Done-Get_a_Validated_Image))) => (assert (Get_a_Validated_image)) )}
\]

\[
\text{(Defrule dm-2 (Get_Diagnosis_from_ANN) (not (Done-Get_ANN_Ready))) => (assert (Get_ANN_Ready)) )}
\]

\[
\text{(Defrule dm-3 ?num1<-(Get_Diagnosis_from_ANN) ?num2<-(Done- Get_a_Validated_image) ?num3<-(Done-Get_ANN_Ready)) => (retract ?num1) (retract ?num2) (retract ?num3) (assert (Do-Get_Diagnosis_from_ANN)) )}
\]

We reserve some keywords:

Do: connects with "-" and a task or an operation node. It means there is no need to consider any other tasks that are included in its pre-condition, just execute this task or operation.

Done: connects with "-" and a task or an operation node. It means this task or operation node has been achieved.

Detailed information about the whole DM content can be found in Jing 2001a.

4.2 The IM

The user model management agent looks over the user's shoulder keep track of his/her actions and the system's events and modify the IH with these actions and events. The IM inference is quite straightforward once the inference algorithms are implemented and the IH is ready. As an experiment, we record 102 action sequences of a user in the IH, using the IM inference engine we get the IM shown in Figure 10. This IM is used to represent the user's interaction habits. We find out that the IM we get is quite well fit to the user's real habit. As we mentioned before, although we only take into account the order of two actions, we still can infer the order of many actions from the IM and we still keep the primary interactions in the IH.

5 Conclusion

In this paper, we have presented an architecture - M² Ato provide beneficial and timely assistance to users. We focussed on the DM and UM design and construction methods. A prototype system and experiments have been presented using the methods. The DM and the UM proved quite functional and helpful enabling the IIA to give
assistance. The methods presented in this paper produce good results according to an experiments. There are still several issues that need to be investigated.

1. How to select the interaction history? If we keep all the user's interactions, it will cost too much calculation time and disk space. One simple method is to keep the relatively later records.

2. How to deal with user's interaction noise? Users sometimes do not mean to do anything, they just are just practicing or playing there. One possible solution is to provide a button for the user model management agent or the users themselves to switch off the IH recorder.

3. How to organize all of the user information? One possible approach is to build up several user classes, each user class representing a particular group of users who shares some fundamental properties and some interaction habits.

Figure 8 The ARMD System using MFA Architecture

Acknowledgements

This work is supported by a Heriot-Watt University Scholarship and Computing & Electrical Engineering departmental funds.

References


