A Negotiating Interface Agency for Adaptation to Users' Preferences

Britta Lenzmann and Ipke Wachsmuth
University of Bielefeld
Faculty of Technology, AG Knowledge-Based Systems
D-33501 Bielefeld, Germany
{britta,ipke}@techfak.uni-bielefeld.de

Abstract

This paper describes an approach to adaptation to users' preferences realized by an interface agency. Using an informed negotiation technique, agents as bidders as well as contractors compete with each other to meet users' preferences. By learning from indirect user feedback, the adjustment of internal credit vectors and the assignment of bidders that gained maximal credit in respect to the user's actual preferences and the preceding session can be realized. In this way, user adaptation is achieved without accumulating explicit user models but by the use of implicit, distributed user models.

1 Introduction

Interface agents are computer programs that enhance the human-computer interaction by mediating a relationship between technical systems and users [Laurel, 1990]. On the one hand, they provide assistance to users by acting on his/her behalf and automating his/her actions [Norman, 1994]. On the other hand, they allow more human-like communication forms by translating qualitative human input to precise commands which can be interpreted by the application system [Wachsmuth & Cao, 1995].

To assist the user in transforming tasks, interface agents need to have knowledge about the user and the application. A prominent approach is to build learning interface agents that automatically acquire knowledge about tasks and preferences of the user by applying machine learning techniques [Maes, 1994]. In this way, personal intelligent assistants, e.g., for electronic mail handling and information filtering, have been built which use techniques such as learning from observations, learning from feedback, learning from examples or learning from other agents [Mitchell, 1994]; [Maes, 1994]. In these approaches, a single personal interface agent is used which customizes to an individual user by acquiring user data and changing its internal functionality.

Since acquiring user-specific data and building explicit user models has found critique with respect to privacy of personal information [Norman, 1994], we pursue a different approach where an interface agency – consisting of multiple sub-agencies – customizes to users' preferences by building an implicit, distributed model of the user. We use learning from indirect user feedback that allows to determine which agents of different sub-agencies are preferred by the individual user and in the actual situation. Internally, the dynamic activation of single agents is realized by using an informed negotiation process.

Our approach is realized in the VIENA multiagent interface system for interaction with a 3D-graphical system. We start with explaining the basic ideas of the learning interface agency in the context of VIENA. Section 3 describes the adaptation process in detail which will then be illustrated by example applications (Section 4). We conclude with a discussion of our approach.

2 Learning Interface Agents in VIENA

In VIENA, we consider the interaction with a 3D-graphical system by way of verbal and gestural input [Wachsmuth & Cao, 1995]. As an example application, a virtual office room can be manipulated by the user. A multiagent interface system translates the qualitative instructions to internal commands which can be interpreted by the graphical system. To enhance interaction comfort, we have realized an anthropomorphic agent, named Hamilton, that is visualized in the scene (Figure 1) and can respond to users' instructions.

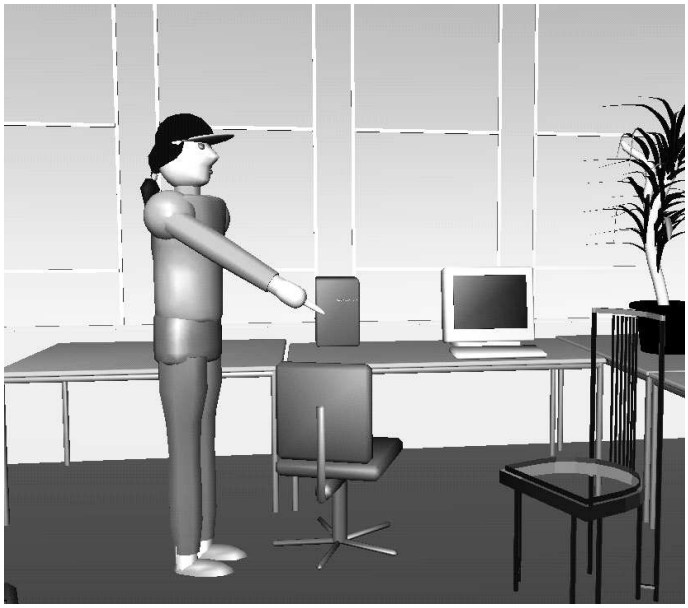

Figure 1: Snapshot of a VIENA example scene

The interface agency consists of different sub-agencies, each of them realizing different functionalities; e.g., a space agency determines spatial transformations, a colour agency changes the appearance of scene objects, a hamilton agency determines actions of the anthropomorphic figure. Agents communicate and cooperate by using a variation of the contract-net negotiation protocol [Davis & Smith, 1983] in which each agent can take on the role of a contractor as well as a bidder [Lenzmann *et al.*, 1995].

Since the user interacts with the system by way of qualitative verbal and gestural instructions (which are often situated), their precise meanings can usually not be resolved unambiguously. Rather different solutions are possible. The practical experience with the VIENA system has shown that significant variations of users' preferences exist with respect to possible solutions. Therefore, we have built agents of the same type but with

slightly different internal functionalities – corresponding to different users' preferences – and joined them together in a sub-agency.

What to learn (from)?

The objective of the learning method is to determine those agents of each agency which correspond to the actual preferences of the current user. This means that dynamic adaptation has to be realized with respect to

1. preferences of different users, as well as
2. time-varying preferences of an individual user during a session.

Agents learn from getting user feedback, i.e., implicit positive and explicit negative feedback. Implicit positive feedback is given when a user's instruction is followed by any instruction which does not decline the previous one (instruction 2). Explicit negative feedback is given when the user corrects the visualized solution offered by the interface agency (instruction 3).

1. *Hamilton, look at the chair.*
2. *Go left.*
3. *wrong.*

Until the user's instructions are evaluated entirely with respect to his/her preferences, a number of sub-tasks have to be solved by the interface agency. Therefore, a number of communication and cooperation processes are carried out between different agents within and across sub-agencies. The overall adaptation is then achieved by the cooperation of agents which emerges from consecutive feedback by the user.

How to learn?

The learning method described here can be classified as a form of reinforcement problem since the interface agency has to adapt to users' preferences by not precisely specified feedback. User feedback represents reinforcement signals which are interpreted and encoded by the interface agency in the form of credit values. Credits are stored locally by each agent and correspond to agents' strengths at discrete interaction steps. Learning is achieved by the following two steps:

1. Adjustment of credits in correspondance to the user feedback and the actual situation parameters
2. Assignment of those agents that are eligible for the task and have maximal credits in correspondance to the interaction process

These steps are realized by different agent instances of the interface agency. Whereas the first step is realized by agents as bidders, the second step is realized by agents as contractors. In more detail, the process consists of the following steps: a contractor agent sends a task posting to each bidder agent that is eligible for the task; with respect to the task description, each bidder generates a bid including its actual credits and sends it to the contractor; the contractor evaluates and compares all incoming bids; the bidder

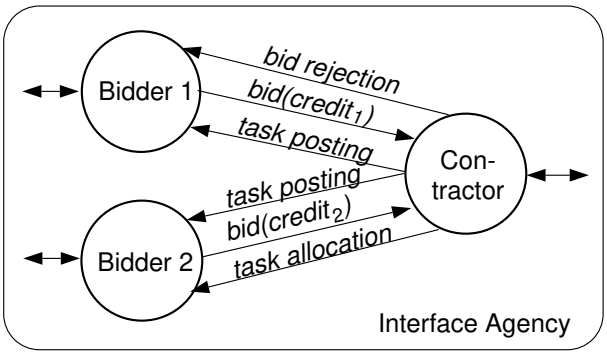

Figure 2: A detail of the negotiation process: bidder 2 generates the better bid and gets the task whereas the bid of bidder 1 is rejected.

with the best bid will get the task, whereas the bids of the other bidders will be rejected. In this way, the adaptation to users' preferences is achieved by a negotiation process (cp. Figure 2). The activation of preferred agents of different sub-agencies describes an implicit, distributed user model.

A prototype version of this adaptation method, where simple heuristics is used concerning the initialization and modification of credits and the assignment of bidders, is described in [Lenzmann & Wachsmuth, 1996]. With this version, our system was already able to adapt to changing preferences of users. A disadvantage of this first prototype system is that credits are simply represented by scalar values that loose information about the history about possibly successful bidders and, thus, allow only very short-term adaptation. Moreover, the method is not appropriate for handling any given number of bidders and contractors. To allow a more flexible adaptation, our first version has been further elaborated. Requirements and methods of a more advanced version are described in the following section.

3 Learning by informed negotiation

The basic requirement is that the interface agency as a whole should be able to organize itself in the way users' preferences and actual situation parameters call for. A simple representation of credits or a simple selection process of bidders cannot satisfy the requirements of ongoing adaptation by the use of a negotiation technique. To this end, general heuristics has to be defined which concerns the adjustment of credits and the assignment of agents. More concrete, the following requirements should be satisfied:

1. Credits represent the strengths of bidders to allow an intelligent assignment of the best bidder.

2. Bidders store different credits for different bidder-contractor relationships to distinguish between contractors.

3. On negative feedback, the direct activation of a bidder that has not caused the undesired solution is possible.

4. Time-varying preferences can be handled.

5. Assignment of dominant bidders from any number of bidders and by any number of contractors can be realized.

6. To reduce communication overhead, tasks are allocated to a bidder directly if the bidder was dominant in the preceding session.

7. Conflicts caused by the fact that one bidder is equally successful as other bidders can be resolved.

The first two aspects concern the functionality of the bidders; the last three aspects concerns the functionality of the contractors; aspects three and four belong to both functionalities.

Agents as bidders

The basic idea which is motivated by the way tasks would normally be assigned in a company is that a bidder is successful when (1) the number of tasks sent to the bidder's sub-agency in relation to (2) the number of tasks performed by that bidder in relation to (3) the number of tasks successfully performed by that bidder is high. Realizing this idea demands that each bidder keeps track of these numbers during the ongoing session and captures this knowledge in a time-dependent vector of credit values, in short, credit vector. At any interaction step t the credit vector of a bidder is defined as follows[1]:

$$credit(t) \;=\; (conf(t), success_task(t), perf_task(t), task(t))$$

where *task* is the number of tasks sent to the bidder's sub-agency, *perf_task* is the number of tasks performed by that bidder, *success_task* is the number of tasks successfully performed by that bidder, and *conf* represents the confidence the bidder has doing the task successfully.

The values of the first three vector components are computed on the basis of message data acquired at each interaction step t. In detail, each bidder stores the following data vector:

$$data(t) \;=\; (message_id(t), sender(t), recipient(t), type(t), content(t))$$

where *type* refers to the negotiation technique used (task posting or direct task allocation) and *content* represents the kind of feedback (positive or negative). Using this information, *task*, *perf_task*, and *success_task* are updated by the following rules where b could be any bidder of a sub-agency sa:

$$task(t) \;:=\; \begin{cases} 0, & \text{if} \quad t = 0 \\ task(t-1) + 1, & \text{if} \quad (type(t) = \text{task posting} \wedge recipient(t) = sa) \vee \\ & \qquad\quad (type(t) = \text{task allocation} \wedge recipient(t) = b) \\ task(t-1), & \text{else} \end{cases}$$

[1] Precisely, a credit vector $credit^b_{sa}(t)$ is computed for each bidder b in any sub-agency sa involved. For the ease of reading, we have dropped indices b and sa in the formula. The same holds true for $data(t)$, $task(t)$, etc.

The value of *task* is incremented by 1 whenever a task is posted to a sub-agency *sa* or a task is directly allocated to a bidder b. The value of *perf_task* is determined in a similar but a bit more restricted way such that it will be incremented whenever a bidder b has performed the corresponding task.

$$perf_task(t) := \begin{cases} 0, & \text{if} \quad t = 0 \\ perf_task(t-1) + 1, & \text{if} \quad (type(t) = \text{task allocation} \wedge \\ & \qquad\quad recipient(t) = b) \\ perf_task(t-1), & \text{else} \end{cases}$$

The number of tasks successfully performed is incremented by 1 whenever a task was successfully performed by a bidder b, that is, the user feedback is positive.

$$success_task(t) := \begin{cases} 0, & \text{if} \quad t = 0 \\ success_task(t-1) + 1, & \text{if} \quad (type(t-1) = \text{task allocation} \wedge \\ & \qquad\quad recipient(t-1) = b \wedge \\ & \qquad\quad content(t-1) = \text{positive}) \\ success_task(t-1), & \text{else} \end{cases}$$

Having computed the first three components of the credit vector in this way, the value of *conf* is updated by using these results as follows:

$$conf(t) := \begin{cases} 1, & \text{if} \quad (success_task(t) - success_task(t-1) = 1) \vee \\ & \qquad [(perf_task(t) - perf_task(t-1) = 0) \wedge \\ & \qquad (success_task(t) \div perf_task(t) > \alpha) \wedge \\ & \qquad (perf_task(t) \div task(t) > \beta)] \\ -1, & \text{if} \quad (success_task(t) - success_task(t-1) = 0) \wedge \\ & \qquad (perf_task(t) - perf_task(t-1) = 1) \\ 0, & \text{else} \end{cases}$$

The rationale behind the definition above is as follows. The confidence is high when a bidder has performed a task successfully or when the actual task has not been performed but the bidder has been dominant in the preceding session; the confidence is low when the solution which the bidder offered was corrected by the user; the confidence is neither high nor low in any other case. In a similiar way [Lashkari *et al.*, 1994] use a trust value to select appropriate agents for collaboration.

Usually, bidders cooperate with several contractors. Since it is possible that a bidder has successfully performed tasks allocated by one contractor but was unsuccessful performing tasks allocated by another contractor, each bidder stores a credit vector for each contractor and manipulates each vector depending on the actual contractor.

Defining credits in this way, the requirements described at the beginning of Section 3 can be satisfied as far as they concern agents as bidders, since (1) the credit vector, especially the *conf* value, represents a kind of strength and captures information of the bidder's history that can be used for a more stable adaptation, (2) different bidder-contractor relationships are modelled, and (3) the direct activation of another bidder on negative feedback as well as (4) the adaptation to time-varying preferences is prepared by the definition of *conf*.

Agents as contractors

Sending a task posting, a contractor receives bids including credit vectors of each bidder that is not occupied by other tasks. To determine the bidder that has worked most successful in the preceding session, the contractor evaluates each incoming bid and compares it with each other bid. The criterion of assigning a bidder can be described by the following rules:

Rule 1:
Choose the bid where *conf* is maximal;
`If` *conf* is equal in two or more bids
`do` *Rule 2*;

Rule 2:
Choose the bid where the relation between *success_task* and *perf_task* is greater than γ and the relation between *perf_task* and *task* is greater than δ;
`If` two or more bids satisfy this test
`do` *Rule 3*;
`else if` no bid satisfies this test
`do` *Rule 4*;

Rule 3:
Choose the bid where the relation between *success_task* and *perf_task* is maximal;
`If` this relation is equal in two or more bids
`do` choose one bid where the relation between *perf_task* and *task* is maximal;

Rule 4:
Choose the bid where the difference between *perf_task* and *success_task* is minimal;
`If` two or more bids satisfy this test
`do` *Rule 3*;

Evaluating the confidence value *conf* first allows that another bidder can be activated when negative feedback occurs (because bids with negative confidence will be regarded last). The rationale behind Rule 2 is that bidders should be preferred if they have worked better than average in the preceding session. By this, time-varying preferences can be handled. Rule 3 describes the final criterion to distinguish between agents and, thus, to determine dominant bidders. So, conflict resolution concerning the assignment of bidders which are equally successful is stopped at this point. When none of the bidders has worked better than average, the contractor assigns that bidder which has caused a minimal number of negative feedbacks during the preceding session (Rule 4).

To enable direct task allocation and, by this, reduce communication overhead, each contractor acquires knowledge about contracting bidders and their success or failure in the preceding session. This information is captured in a time-dependent history vector for each sub-agency *sa*[2]:

$$history(t) \;=\; (task(t), success(t), recipient(t))$$

where *task* represents the number of tasks allocated to a sub-agency, *success* stores information on having performed tasks positively or negatively, and *recipient* refers to a specific bidder of the considered sub-agency.

[2]For the ease of reading, the index *sa* is dropped in the formula; cf. footnote 1.

Using this knowledge, a contractor can decide if a bidder has performed tasks successfully over a period of contracts and in the positive case allocate the next task to this bidder directly. A similiar idea was presented by [Dowell, 1995] where a learning contract-net algorithm is used to reduce communication overhead. Since the approach works for any number of bidders, the requirements stated at the beginning of this section are satisfied completely.

Agents as bidders as well as contractors

Since agents of the entire interface agency can take on the role of a bidder as well as the role of a contractor, each agent has to keep track of the three kinds of vectors described above. The handling and determination of these vectors are part of a communication and cooperation framework, so that internal functionalities can be realized independently, and no supervision by any kind of a globally informed agent is needed.

4 Example Application

The adaptation method described above has been implemented and tested for different examples, primarily for the case of users' preferences for different spatial reference frames. Consider the situation in Figure 3: Two possible solutions can be offered when the user has instructed Hamilton to go left. On the basis of the hamilton-deictic reference frame, Hamilton has to move in the direction indicated by 'H'; on the basis of the user-deictic reference frame, Hamilton has to move in the direction indicated by 'U'. Experiments with the VIENA system have shown that, due to individual differences among users, one spatial reference frame may be preferred over the other one [Jörding & Wachsmuth, 1996]; this is a motivation for the adaptation method presented in this paper.

Figure 3: Possible solutions of the instruction "Hamilton, go left.": Hamilton can move to the left from the hamilton-deictic (H) or the user-deictic (U) perspective.

Similarly, we have implemented two space agents which compute spatial transformations on the basis of the user-deictic reference frame, or on the basis of the object-intrinsic reference frame, respectively. In addition, we have tested the adaptation method for the case of users' preferences for different colour sensation by implementing two colour agents that offer more drastic or smoother colour transformations. First experiments have shown that the approach described above can realize adaptation to users' preferences effectively and satisfactorily with respect to the requirements stated in Section 3. A fuller evaluation is one of our next goals.

5 Discussion

This paper presented an approach to user adaptation realized by a learning interface agency. The interface agency consists of several sub-agencies which represent different preference classes. Each sub-agency consists of several agents corresponding to the possible preferences. Agents use an informed negotiation process where each agent can take on the role as a bidder as well as a contractor. As bidders, agents acquire knowledge about the user and the preceding session which is captured in internal credit vectors. As contractors, agents acquire knowledge about contracting bidders and their success or failure in the preceding session. By learning from indirect user feedback, bidders compete with each other to meet the users' preferences. Thus, the system's knowledge of the user is expressed in the activation of certain agents of the entire interface agency. In this way, user adaptation is achieved without accumulating explicit user models but by the use of implicit, distributed user models.

In the future, we want to consider, additionally, adaptation to different situation circumstances since experiments with the system have shown that users' preferences may depend on them also. Considering the situation of Figure 3, the preference for using one reference frame over the other one may depend on the orientation and location of the anthropomorphic figure. Therefore, we have to investigate what kinds of situation parameters are appropriate for describing their impact on users' preferences to the end of integrating them in the adaptation process.

References

[Davis & Smith, 1983] Davis, R., Smith, G. Negotiation as a Metaphor for Distributed Problem Solving. In Bond, A.H. and Gasser, L. (eds.): *Readings in Distributed Artificial Intelligence* (pp. 333–356). Morgan Kaufmann, 1983.

[Dowell, 1995] Dowell, M.L. Learning in Multiagent Systems. Ph.D. Thesis at the Department of Electrical and Computer Engineering, University of South Carolina, 1995.

[Jörding & Wachsmuth, 1996] Jörding, T., Wachmsuth, I. An Anthropomorphic Agent for the Use of Spatial Language. Accepted for ECAI-96 Workshop on Representation and Processing of Spatial Expressions.

[Lashkari *et al.*, 1994] Lashkari, Y., Metral, M., Maes, P. Collaborative Interface Agents. In *Proceedings of the National Conference on Artificial Intelligence.* Cambridge (MA): The MIT Press, 1994.

[Laurel, 1990] Laurel, B. Interface agents: Metaphors with character. In Laurel, B. (Ed.): *The art of human-computer interface design* (pp. 355-365). Reading: Addison-Wesley, 1990.

[Lenzmann & Wachsmuth, 1996] Lenzmann, B., Wachsmuth, I. A User-Adaptive Interface Agency for Interaction with a Virtual Environment. In Weiss, G. & Sen, S. (eds.): *Adaption and Learning in Multi-Agent Systems.* Berlin: Springer, 1996.

[Lenzmann *et al.*, 1995] Lenzmann, B., Wachsmuth, I., Cao, Y. *An Intelligent Interface for a Virtual Environment.* KI-NRW Report 95-01, 1995.

[Maes, 1994] Maes, P. Agents that Reduce Work and Information Overload. *Communications of the ACM 37*(7), 1994, 31-40.

[Mitchell, 1994] Mitchell, T., Caruana, R., Freitag, D., McDermott, J., Zabowski, D. Experiences with a learning personal assistent. *Communications of the ACM 37*(7), 1994, 80-91.

[Norman, 1994] Norman, D.A. How Might People Interact with Agents. *Communications of the ACM 37*(7), 1994, 68-71.

[Wachsmuth & Cao, 1995] Wachsmuth, I., Cao, Y. Interactive Graphics Design with Situated Agents. In W. Strasser & F. Wahl (eds.): *Graphics and Robotics* (pp. 73-85), Springer.