Contract-Net-Based Learning in a User-Adaptive Interface Agency

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 a multi-agent learning approach to adaptation to users' preferences realized by an interface agency. Using a contract-net-based negotiation technique, agents as contractors as well as managers negotiate with each other to pursue the overall goal of dynamic user adaptation. By learning from indirect user feedback, the adjustment of internal credit vectors and the assignment of contractors that gained maximal credit with respect to the user's current preferences, the preceding session, and current situational circumstances 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 [Lau90]. On the one hand, they provide assistance to users by acting on his/her behalf and automating his/her actions [Nor94]. On the other hand, they allow more human-like communication forms by translating qualitative input of the human user to precise commands which can be interpreted by the application system [WC95].

To assist the user in performing 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 [Mae94]. In such approaches, a single personal interface agent is used which customizes to an individual user by acquiring explicit user data.

Since acquiring user-specific data and building explicit user models has found critique with respect to privacy of personal information [Nor94], 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 a contract-net learning process where multiple agents in the role of

contractors as well as agents in the role of managers negotiate with each other to pursue the common goal of dynamic user adaptation.

The paper is structured as follows: We start by distinguishing our approach from related work in the field of user adaptation and learning interface agents. We proceed by characterizing our learning technique in terms of the learning category and the form of learning. Having described the basic ideas and requirements, we focus on the realization of agents in the role of contractors as well as agents in the role of managers. The learning technique which has been integrated in the interactive 3D-graphics system VIENA is then illustrated by an example of adaptation to users' preferences for different spatial reference frames.

2 Related Work

To realize an automatic adaptation to the varying preferences different users could have for the possible solutions produced by an application system, a computer system must have knowledge about its users and the application domain. Conventional approaches in the fields of Artificial Intelligence (AI) and Human-Computer Interaction (HCI) are commonly based on the modelling of users where information about the users is gathered, represented, and used to adapt to users' requirements [McT93]. When acquired explicitly, a user model demands a large amount of work from the application designer (or the user in the case of end-user programming), and it is relatively static. In the case of automatic aquisition, such disadvantages do not occur. Nevertheless, both methods have the draw-back that knowledge about the user is kept in a explicit knowledge base; this fact has found critique with respect to privacy of personal information [Nor94].

Interface agents are considered as metaphor of indirect management [Kay90] instead of interacting via commands or direct manipulation. For example, [Chi90] realized a learning interface agent that helps a user solve problems in using the UNIX operating system. Such approaches to user adaptation are based on endowing the interface agent with a huge amount of background knowledge about the user and the application [MK93]. Similiar to the methods described above, they have the disadvantage that a huge amount of fixed knowledge has to be acquired.

More elaborated approaches to learning interface agents rely on the automatic aquisition of knowledge about tasks and preferences of the user by applying machine learning techniques. For example, [Mae94] uses the techniques of as learning from observations, learning from feedback, learning from examples, or learning from other agents to build learning interface agents for electronic mail handling and information filtering. [MCFMZ94] use decision trees to realize a learning personal assistant for meeting scheduling. [Sel94] has built a user-adaptive teaching agent that supports users in writing Lisp programs by watching the users actions and using a production system and a blackboard mechanism. The main difference to our approach, presented in this paper, is that a single personal interface agent is used which customizes to an individual

user by acquiring user data. Thus, the problem of keeping personal information of users still consists.

Although a number of multi-agent learning techniques exists, relatively few ones focus on the aspect of user adaptation. In [LMM94], agents collaborate with each other to steepen the agents' learning curves and to handle unencountered situations. With respect to contract-net negotiation, the integration of learning facilities has been studied by some researchers. In [Dow95], for example, agents of a multi-agent system learn about other agents' abilities and task load to switch from broadcasting task announcements to direct task assignment. [OHA96] has realized the multi-robot learning system LEMMING that learns to determine the most suitable robot for task execution by using case-based reasoning within the contract-net negotiation. Similiar to our approach, useful information of previous negotiation messages is extracted. In contrast to our approach, their work is limited to the aspect of optimizing communication overhead. In summary, using contract-net-based learning to achieve user adaptation without building explicit user models seems to be an innovative approach.

3 Characterizing the Learning Method

Before describing the contract-net-based learning method in detail, we will characterize our approach in terms of the learning category and the form of learning with emphasis on the learning strategy and the learning feedback as proposed in [Wei96].

3.1 Learning Category

Allowing adaptation to users' preferences without building explicit user models relies on the design of a multi-agent system that consists of numerous sub-agencies realizing different functionality types. Each sub-agency joins agents of the same type but with slightly different internal functionalities together and corresponds to a class of users' preferences with respect to a functionality type. Each agent of a sub-agency realizes a specialized preference of its sub-agency's preference class.

By exchanging information via a contract-net negotiation mechanism, the agency, as a unit, learns by organizing itself in the way that those agents of each sub-agency can be activated which correspond best to the actual preferences of the current user in the given situation. This means that learning is aimed at a dynamic adaptation with respect to

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

In this way, the approach can be categoriezed as multi-agent learning insofar as it requires the presence of multiple agents which negotiate with each other to pursue the common goal of dynamic user adaptation.

3.2 Form of Learning

The agency learns the goal of user adaptation by offering solutions to the user and observing the user's 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. Explicit negative feedback is given when the user corrects the solution offered by the interface agency.

The feedback provided by the user acts as a critic which is 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. 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 which emerges from consecutive feedback by the user is then achieved by the cooperation of agents.

The learning strategy decribed here can be classified as a form of *learning by discovery* since agents capture knowledge about the interaction process and about other agents by making obeservations and generating solutions on the basis of the observational results. From the perspective of the feedback that is available to the agency the method can be regarded as *reinforcement learning* since the agency has to learn the right actions by not precisely specified feedback.

4 Contract-Net-Based Learning

Considering contract-net negotiation within a multi-agent system, the execution of tasks and sub-tasks is the result of a bidding scheme between agents in the role as a contractor and agents in the role as a manager [DS83]. Once a task has to be executed, a manager agent sends a task announcement to its contractor agents. With respect to the task description, idle contractors generate bids and send them to the manager. The manager evaluates all incoming bids and selects one or more contractors for task execution.

4.1 Extending Standard Contract-Net Negotiation

On the basis of the standard contract-net mechanism, we have integrated the following two steps into the contract-net negotiation to achieve user adaptation:

1. Adjustment of credits in correspondence 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 correspondence to the interaction process

These steps are realized by different agent instances of the interface agency. The first step is realized by agents currently in the contractor role which extract and

store relevant information from incoming negotiation messages, consider actual situation parameters, use the results to adjust credits, and include current credits in their bids. The second step is realized by agents as managers which compare all incoming bids by evaluating credits and the negotiation history, allocate the task to the most promising contractor, and reject the bids of all other contractors. Figure 1 illustrates the negotiation process for the case of two contractors and one manager[1].

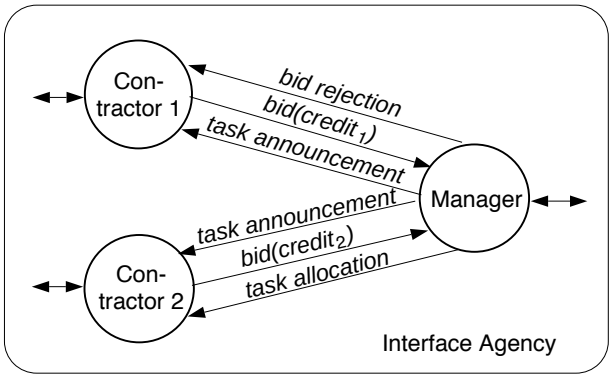

Fig. 1. A detail of the negotiation process: Contractor 2 generates the better bid and gets the task whereas the bid of the potential contractor 1 is rejected.

In this way, the overall adaptation to users' preferences is achieved by a set of negotiation functions that are executed locally between managers and contractors and results are distributed in the agent system. Each of these functions realizes the two steps mentioned above by the concatenation of the functionality of a manager and of a number of contractors. For the case of a manager k and m contractors, the negotiation function at an interaction step t is defined by:

$$f_{neg}(X(t)) = manager_k(contractor_1(X_1(t)), \ldots, contractor_m(X_m(t)))$$

Each contractor function $contractor_i$ realizes the adjustment of credits based on the data $X_i(t)$ extracted from the current and preceding negotiation messages. On the basis of these computations, the manager function $manager_k$ realizes the assignment of the most successful contractor. A schematic view of the negotiation function is shown in Figure 2.

Taking a more abstract point of view, the activation of preferred agents of different sub-agencies gives rise to an implicit, distributed user model where the acquisition of explicit user models is avoided. Besides using the contract-net

[1] We have adjusted terminology used in earlier publications to be consistent with [DS83].

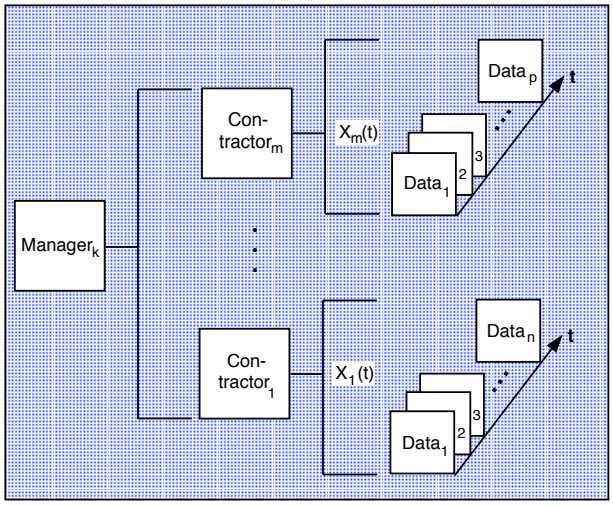

Fig. 2. Functional scheme of the contract-net-based learning approach

mechanism as tool to achieve a user-adaptive multi-agent system, extensions of the standard negotiation process as described at the beginning of this section concern:

- contractors and managers store knowledge about the preceding negotiation processes
- this knowledge is used, by contractors, to adjust credits and, by managers, to assign contractors
- bids include user feedback and credits corresponding to the actual situation

4.2 Requirements of the learning strategy

A prototype version of the adaptation method, where simple heuristics is used concerning the modification of credits and the assignment of contractors, is described in [LW96]. 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 potentially successful contractors and, thus, allow only very short-term adaptation. Moreover, the method is not appropriate for handling any given number of contractors and managers. Finally, no current situation parameters are taken into account when adjusting credits.

To allow a more flexible adaptation, our first version has been further elaborated with respect to the manager and contractor functions. 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. To achieve

an ongoing adaptation by a negotiation technique, a simple representation of credits or a simple selection process of contractors is not suitable. 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 contractors to allow an intelligent assignment of the best contractor.
2. Current situation parameters constrain the adjustment of credits to model the influence of situational circumstances on users' preferences.
3. Contractors store different credits for different contractor-manager relationships to distinguish between managers.
4. On negative feedback, the direct activation of a contractor that has not caused the undesired solution is possible.
5. Time-varying preferences can be handled.
6. Assignment of dominant contractors from any number of contractors and by any number of managers can be realized.
7. To reduce communication overhead, tasks are allocated to a contractor directly if the contractor was successful in the preceding session.
8. Conflicts caused by the fact that one contractor is equally successful as other contractors can be resolved.

The first three aspects concern the contractor function; the last three aspects concern the manager function; aspects four and five pertain to both functions. Since agents of the entire multi-agent system can take on the role of a contractor as well as the role of a manager in different situations and at different time steps, each agent has to perform both functions. The execution of these functions is 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. In the following two sections, we describe in detail the heuristics to define both functions, the contractor and the manager function.

5 Agents as Contractors

To adapt to users' preferences, the contractor learning functions realize the adjustment of credits based on data extracted from negotiation messages and the manager learning functions realize the assignment of most promising contractors. In detail, each contractor function performs the following steps: extract and store relevant information from incoming negotiation messages, compute actual situation parameters, and use the results to adjust credits which are included in their bids.

5.1 Representation of Credits

The basic idea to decide which information is relevant for the adaptation task and, thus, has to be captured is motivated by the way tasks would normally

be assigned in a company. Applied to our context, a contractor is considered successful when (1) a high number of tasks was sent to the contractor's sub-agency, (2) a high number of tasks was performed by that contractor, and (3) a high number of tasks was successfully performed by that contractor.

Realizing this idea demands that each contractor acquires knowledge at any negotiation step and represents the knowledge in some kind of a time-dependent vector of which it keeps track during the ongoing session. In fact, two kinds of vectors have to be defined. First, a data vector is used which represents acquired message data to determine the three above numbers. At any negotiation process t each contractor stores the following data vector[2]:

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

where *type* refers to the negotiation technique used (task announcement or direct task allocation) and *content* represents the kind of feedback the user has given, i.e. positive or negative.

Second, a vector of credit values, in short, credit vector, is used which represents the agents' strengths at discrete negotiation steps including the three numbers mentioned above. At any interaction step t the credit vector of a contractor is defined as follows:

$$credit(t) \;=\; (conf(t), sit(t), suc\_task(t), perf\_task(t), task(t))$$

where *task* is the number of tasks sent to the contractor's sub-agency, *perf_task* is the number of tasks performed by that contractor, and *suc_task* is the number of tasks successfully performed by that contractor. *sit* represents the conformity of the current situation with the preference embodied by the contractor. *conf* results from combining the other vector components in a special way and represents the overall confidence the contractor has doing the task successfully.

In general, contractors cooperate with several managers. Since it is possible that a contractor has successfully performed tasks allocated by one manager but was unsuccessful performing tasks allocated by another manager, each contractor stores a credit vector for each manager and manipulates each vector depending on the current manager.

5.2 Computation of Situation Parameters

Naturally given users' preferences may depend on situational circumstances what can result in a variation on their preferences. For example, in our application scenario (cf. Section 7) the orientation of reference objects to the user's view is a relevant situation parameter. To handle this situational aspect, current situations have to be taken into consideration to realize user adaptation. In our approach, this is achieved by integrating the computation of situation parameters in the

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

adjustment of credits. That means that the current situation has to influence the adjustment of credits positively whenever this situation seems to be in conformity with the user's preference that the contractor is realizing.

To analyze the situations and integrate the results in the adaptation process, a set of independent features

$$f_1, f_2, \ldots, f_n$$

is used. Each feature has a value f_i^{opt} that is given when a situation confirms this feature optimally with respect to the preference the agent realizes and a current value $f_i(t)$ that determines the value of that feature at the negotiation step t. The kind of features and the number of features is defined locally by each agent and can be considered as part of its local knowledge. Features vary over sub-agencies but are equal within one sub-agency since each sub-agency models the same functionality type. Optimal feature values also vary within a sub-agency since each member of this sub-agency realizes another preference with other basic requirements of the modelled preference class.

On this basis, the current situation is then measured by computing the weighted sum of feature deviations:

$$\sum_{i=1}^{n} w_i \triangle f_i(t) = \sum_{i=1}^{n} w_i |f_i^{opt} - f_i(t)|$$

where the weight w_i determines the importance of the feature f_i as measurement for a situation. The result of this equation is a measurement for the deviation between the current situation and a situation where the preference is confirmed optimally. Since the component *sit* of the credit vector is a measurement for the conformity of the current situation with the preference embodied by a contractor, at any negotiation step t *sit* is computed as:

$$sit(t) \;=\; 1 - \sum_{i=1}^{n} w_i \triangle f_i(t)$$

The value of *sit* will be calculated whenever a contractor wants to generate a bid in response to a task announcement. Especially at the beginning of a session (when no or few information is available), the consideration of how well the current situation confirms a given preference is useful to make more promising predictions. with respect to adaptation efficiency. In the following section, we describe how *sit* is further used and integrated in the adjustment of credits.

5.3 Adjustment of Credits

So far, we described what kind of information is represented and how the situational influence on users' preferences is measured. For a given credit vector $credit(t) = (conf(t), sit(t), suc\_task(t), perf\_task(t), task(t))$, we now explain how the last three credit components and the confidence value are computed. The values of the last three vector components are computed on the basis of the

information stored in the data vector and acquired at each interaction step t. Using this information, *task*, *perf_task*, and *suc_task* are updated by the following rules where c could be any contractor of a sub-agency sa. 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 contractor c; otherwise it remains the same.

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

The value of *perf_task* is determined in a similar but a bit more restricted way such that it will be incremented whenever a contractor c has actually performed the corresponding task; otherwise it remains the same.

$$perf\_task(t) := \begin{cases} 0, \text{if} & t = 0 \\ perf\_task(t-1)+1, \text{if} & (type(t) = \text{task allocation} \wedge \\ & recipient(t) = c) \\ perf\_task(t-1), \text{else} & \end{cases}$$

The number of tasks successfully performed is incremented by 1 whenever a task was successfully performed by a contractor c, that is, the user feedback is positive; otherwise it remains the same.

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

Confidence. Confidence refers to the trust a contractor has doing the task successfully. Having computed the last three components and the *sit* component of the credit vector as described above, the value of *conf* can be updated by using these results in a combined way. The confidence a contractor has that it will execute the posted task successfully is influenced by three factors:

- P: Performance with respect to the previous task announced to the contractor's sub-agency
- D: Degree of the contractor's dominance in the preceding session
- S: Degree of situational conformity

Speaking metaphorically, these three factors describe factors that influence and constrain the selection of users' preferences in the way that they reflect the user's satisfaction with the offered solution, the interaction history, and the situational dependency.

The performance of a contractor is the result of the user feedback and the contractor's involvement in the previous task. In more detail, the performance P at any negotiation step t is computed as:

$$P(t) := \begin{cases} 0, \text{ if } (task(t) = 0 \ \vee \ perf\_task(t) = 0) \vee \\ \qquad (perf\_task(t) - perf\_task(t-1) = 0) \\ 1, \text{ if } (perf\_task(t) - perf\_task(t-1) = 1) \wedge \\ \qquad (suc\_task(t) - suc\_task(t-1) = 1) \\ -1, \text{ if } (perf\_task(t) - perf\_task(t-1) = 1) \wedge \\ \qquad (suc\_task(t) - suc\_task(t-1) = 0) \end{cases}$$

This means that the performance is high when a contractor has performed the previous task successfully; the performance is low when the solution which the contractor has offered was corrected by the user; the performance is neither high nor low when the contractor has either never performed before or when the previous task has not been performed by this contractor.

To determine the degree of the contractor's dominance, a test $dominant(t)$ is carried out which checks (1) if the number of successfully performed task in relation to the number of performed tasks (d_1) is greater than a threshold $\alpha(t)$ and (2) if the number of performed tasks in relation to the number of announced tasks (d_2) is greater than a threshold $\beta(t)$. The dominance degree D is based on the result of the *dominant* predicate at any negotiation step t:

$$D(t) := \begin{cases} 0, \text{ if } (task(t) = 0 \ \vee \ perf\_task(t) = 0) \\ (d_1 + d_2)/2, \text{ if } dominant(t) \\ (d_1 + d_2)/2 - 1, \text{ if } \neg dominant(t) \end{cases}$$

The last influence factor S is based on the credit component *sit* in the way that a test *sit_conform* is performed which checks again if the value of *sit* is greater than a threshold $\gamma(t)$. The result of this test is used to compute the value of S at any negotiation step t in the following way:

$$S(t) := \begin{cases} sit(t), \text{ if } sit\_conform(t) \\ sit(t) - 1, \text{ if } \neg sit\_conform(t) \end{cases}$$

The thresholds α, β, γ are updated in correspondence to the contractors' performance P. When the performance is positive the thresholds are decreased; in the other case, the thresholds are increased. For example, the value of α is computed at any negotiation step t by the following rule:

$$\alpha(t) := \begin{cases} \alpha_{min}, \text{ if } \quad t = 0 \\ \alpha(t-1) - 1/n \ \triangle\alpha(t) P(t), \text{ else} \end{cases}$$

where $\triangle\alpha(t)$ determines the intervall remaining for manipulation. Thus, contractors need a lower dominance and a lower situational conformity to be considered as successful in future interactions when they have performed task positively in comparison to executing tasks negatively.

Resulting from the above computations, the confidence value at any negotiation step t is determined by:

$$conf(t) := \begin{cases} P(t), \text{if} \quad P(t) = -1 \\ (w_p P(t) + w_d D(t) + w_s S(t), \text{else} \end{cases}$$

The rationale behind the definition above is as follows. A contractor which has a low performance as result of having received negative feedback gets the least minimal confidence value possible. In the other case, its confidence is given by computing the weighted sum of the three influence factors P, D, and S[3].

Combining these three factors allows to efficiently assess contractors just by considering one value. In a similiar way, [LMM94] use a trust value to select appropriate agents for collaboration. Nevertheless, the credit vector contains additional components that allow a more detailed discrimination between contractors in conflicting cases without solely relying on contractors' evaluations.

Defining credits in this way, the requirements described in Section 4 can be satisfied as far as they concern agents as contractors. (1) The credit vector, especially the *conf* value, represents a kind of strength and captures information that can be used for a more stable adaptation; (2) the current situation is integrated in the adjustment of credits; (3) different contractor-manager relationships are modelled; (4) the direct activation of another contractor on negative feedback as well as (5) the adaptation to time-varying preferences is prepared by the definition of *conf*.

6 Agents as Managers

Based on the contractor function, the manager function of each agent has to perform the following steps to achieve the overall adaptation to users' preferences: decompose and announce tasks, pool incoming bids corresponding to the task announcement, and assign the most promising contractor based on the results of the comparison of bids.

6.1 Announcement of Tasks

Each user's instruction requires the solution of a number of sub-tasks by the agent system as a unit. Therefore, a number of negotiation processes are carried out between agents within and across sub-agencies. To solve a sub-task, a manager usually gets other agents or sub-agencies involved which supply the manager with sub-results. Thus, the manager decomposes a sub-task in further sub-tasks and generates a task announcement for each task to be executed.

Besides general information including the sender, the send time, and the task description, a task announcement contains the user feedback of the current instruction. Negative user feedback is included explicitly whereas positive feedback

[3] In our current implementation, the weights each have the fixed value of 1/3.

is included implicitly. More precisely, negative feedback is indicated by appending a negative instruction in front of the task description which is left out in the case of positive feedback.

The question remaining is to which agents credit has to be assigned (commonly known as *credit-assignment problem*). In multi-agent learning, this problem decomposes into the problems of inter-agent and intra-agent credit-assignment [Wei96]. Concerning the inter-agent problem, the agent system has to decide what actions by what agent contributed to the performance change. In our approach, this decision can easily be made since users' preferences are considered as independent. This means that each user's instruction concerns one preference, i.e., a special sub-agency. Thus, the involved sub-agency can easily be determined and credited. Nevertheless, our approach can also handle preferences which depend on each other by assigning credit or blame to any sub-agency that contributed to the offered solution.

With regard to the intra-agent problem, an agent has to decide which facts led to the contributing action. In our case, the activation of a special agent of the contributing sub-agency relies on the credits included in bids. This means that this kind of problem is solved by adjusting internal credit values in correspondence to the preceding negotiation processes and the current feedback included in the task announcement. More generally, the inter-agent credit-assignment problem is solved by agents as managers whereas the intra-agent problem is solved by agents as contractors.

The execution of tasks by the way of announcing tasks requires a number of communication and cooperation processes. To reduce this interaction overhead, managers allocate tasks directly if contractors have executed tasks successfully over a period of contracts. Therefore, each manager acquires knowledge about contractors it is collaborating with and about their success or failure in the preceding session. This information is captured in a time-dependent history vector for each sub-agency[4]:

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

where *task* represents the number of tasks allocated to a sub-agency, *success* stores whether the last task was performed positively or negatively, *recipient* refers to a specific contractor of the considered sub-agency, and δ corresponds to a time-varying threshold which is used for bid evaluation tests.

Using this knowledge, a manager can decide if a contractor has performed tasks successfully over a period of contracts and in the positive case allocate the next task to this contractor directly. A similiar idea was presented by [Dow95] where a learning contract-net algorithm is used to reduce communication overhead.

[4] For the ease of reading, the index *sa* indicating a special sub-agency is dropped in the formula; cf. footnote 1.

6.2 Pooling Bids

After having sent the task announcement, a manager waits for corresponding bids of the contractors of the collaborating sub-agency. Each incoming bid is preprocessed by

1. extracting credits included in the bid,
2. pre-computing factors which are relevant for comparison of bids, and
3. pooling the results in an internal credit data structure.

The factors mentioned by step two are results of tests concerning how successful the contractor that has sent the bid has been in the preceding session. Similiar to the way agents as contractors perform tests for computing their confidence, a manager performs tests that are used to determine the contractors' strength. For instance, it checks if the number of successfully performed tasks in relation to the number of performed tasks is greater than a threshold. To perform these computations, the threshold of the *history* vector is used. Before performing the tests, δ is updated by searching for the appropriate entry of the *history* vector and using the entry values in the following way:

$$\delta(t) := \begin{cases} \delta_{min}, \text{ if } t = 0 \\ \delta(t-1) - 1/n\ (\delta(t-1) - \delta_{min}), \text{ if } success(t) \\ \delta(t-1) + 1/n\ (\delta_{max} - \delta(t-1)), \text{ if } \neg success(t) \end{cases}$$

The pooling of bids is constrained by two disjunctive conditions that terminate the waiting process. The manager is waiting for incoming bids until a maximal time interval is reached or all members of the contracting sub-agency have sent their bids. Therefore, the manager determines how many contractors are active before sending the task announcement. As soon as one of these conditions are satisfied, the manager starts to compare bids or, if no bid was sent, stops the current negotiation process.

6.3 Comparison of Bids

To determine the contractor that has worked most successfully in the preceding session, a manager compares each bid with each other bid on the basis of the credit information extracted and pooled in the previous step. For efficiency, this procedure is not executed if just one bid was pooled. In more detail, the manager starts the comparison by considering the entire set of bids and, then, reduces the search space step by step using special criteria, described below, until precisely one contractor is left or all citeria have been matched. In the latter case, one contractor of the remaining set of contractors is chosen.

The criteria for assigning a contractor decompose in four main classes of rules. The first class of rules concerns the consideration and comparison of the confidence values, and includes the following three rules:

Rule 1:
Remove the bids where *conf* has the minimal value;
`If` the resulting set of bids is empty
`do` stop;
`else`
`do` *Rule 2*;

Rule 2:
Determine the maximal *conf* value;
`If` this value is greater than 0
`do` *Rule 3*;
`else` `do` *Rule 4*;

Rule 3:
Choose the bid where *conf* is maximal;
Stop;

Evaluating the confidence value *conf* first allows that another contractor can be activated when negative feedback occurs because bids with minimal confidence value will be removed first. If the maximal confidence value is not positive, the manager does not rely on the contractors' self-evaluation and, thus, proceeds with further evaluations.

The second criterion considers the evaluation of the situational conformity by performing rule four:

Rule 4:
Choose the bids where *sit* is greater than δ;
`If` the resulting set contains exactly one bid
`do` stop;
`else`
`do` *Rule 5*;

Satisfying that the *sit* value is greater than a threshold, a manager allocates a task to a contractor even if its other credit values, especially its confidence value, are low. By this, time-varying preferences based on situational circumstances can be modelled.

The third class of rules similiarly aims at modelling time-varying preferences but concentrates on the degree of dominance which contractors have reached.

Rule 5:
Choose the bids where the relation between *suc_task* and *perf_task* (q_1) and the relation between *perf_task* and *task* (q_2) is greater than δ;
`If` the resulting set contains exactly one bid
`do` stop;
`else if` the resulting set contains more than one bid
`do` *Rule 6*; `else`
`do` *Rule 7*;

The rationale behind *Rule 5* is that contractors should be preferred if they have worked better than average in the preceding session. A manager discriminates between the situations where none of the contractors or more than one contractor satisfies this condition (and/or the condition of *Rule4*) since it makes a difference whether to choose between poor contractors or between better ones.

The last criterion, finally, contains rules to distinguish between contractors in order to determine most promising ones. Thus, these rules can be regarded as conflict resolution rules.

Rule 6:
Determine the bids where $\sum (conf + sit + q_1 + q_2)$ is maximal;
`If` the maximal value is ambiguous within an ε-intervall
    `do` choose the bids where *sit* is maximal;
`else`
    `do` stop;

Rule 7:
Choose the bids where the difference between *perf_task* and *suc_task* is minimal;
`If` the resulting set contains more than one bid
    `do` *Rule 6*;

The first of these rules distinguishes between more promising contractors by taking almost all credit values into account and, in conflicting cases, preferring the one where the situational conformity is maximal. By the latter rule, the manager first chooses that contractor which has caused a minimal number of negative feedbacks during the preceding session (to determine the best one of the poor ones). Subsequently, the manager performs *Rule 6*, described above, to handle conflicting cases.

Defining rules in this way, the direct activation of another contractor on negative feedback, the handling of time-varying preferences, the assignment of successful contractors from any number of contractors, and the discrimination of contrators in conflicting cases can be realized. Moreover, tasks are allocated to contractors directly to reduce communication overhead. Thus, the requirements stated in Section 4 are satisfied completely.

7 Example Application

The adaptation method described in this paper was implemented in a multiagent interface system for interaction with a 3D-virtual environment, carried out in the VIENA project [WC95]. The user can instruct the system by way of verbal and gestural input. These qualitative instructions are translated by the interface agency to internal commands which can be interpreted by the graphical system. As an example application, a virtual office room can be manipulated by the user. To enhance interaction comfort, we have realized an anthropomorphic agent, named Hamilton, that is visualized in the scene (Figure 3) and can respond to and carry out users' instructions [WLJJLF].

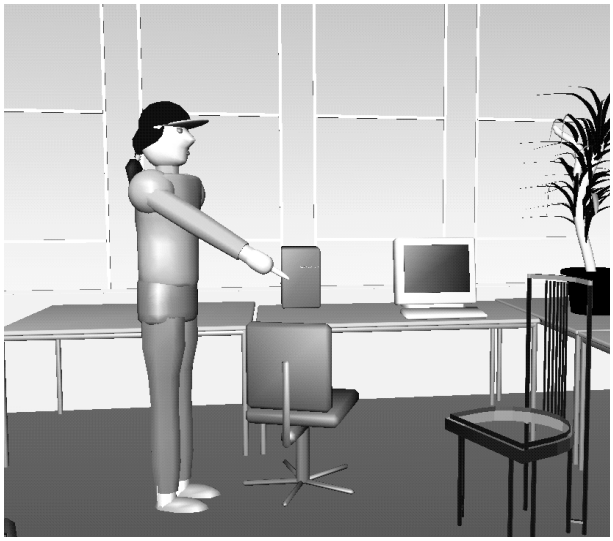

Fig. 3. Snapshot of a VIENA example scene

7.1 VIENA Interface Agency

To reconstruct the overall user's instruction, a number of different tasks have to be solved which are distributed within the multiagent interface system. The entire interface agency consists of a set of communicating and cooperating subagencies, each of them realizing a different functionality (cf. Figure 4). For instance, a space agency determines spatial transformations, a color agency changes the appearance of scene objects, a hamilton agency determines actions

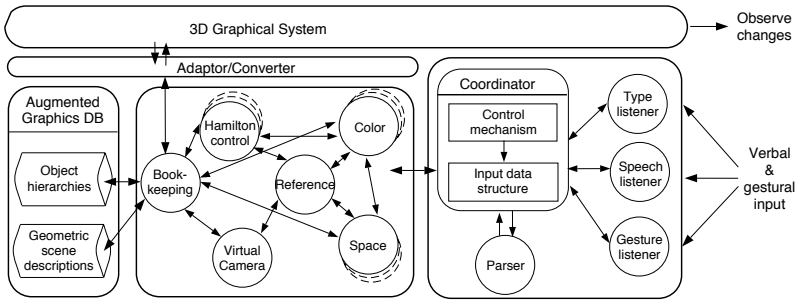

Fig. 4. Architecture of the VIENA multiagent interface system

of the anthropomorphic figure. Agents communicate and cooperate by using a variation of the contract-net negotiation protocol in which each agent can take on the role of a contractor as well as a manager [LWC95].

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. Thus, the VIENA agency proves to be a reasonable testbed for integrating adaptation facilities. 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. To this end, the requirements for using the contract-net learning approach (cp. Section 3) are satisfied.

7.2 Contract-net Learning in VIENA

The adaptation method described in this paper has been implemented and tested for a variety of examples, primarily for the case of users' preferences for different spatial reference frames. The semantics of spatial instructions may depend on different perspectives [Ret88], e.g., they may be interpreted from the user's point of view (deictic perspective) or, alternatively, from the point of view of an object which has a prominent front (intrinsic perspective). Experiments with the VIENA system have shown that, due to individual differences among users,

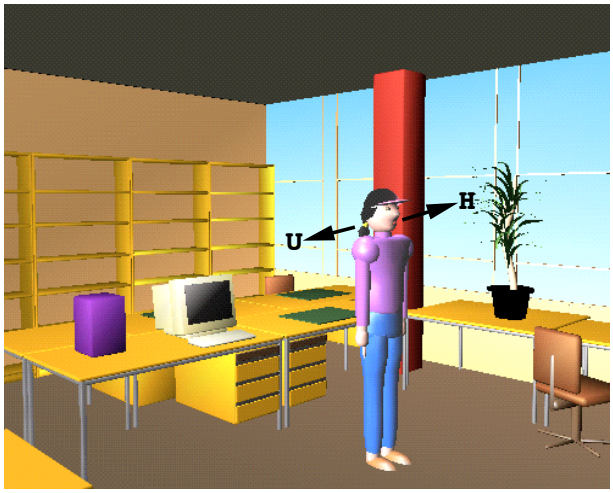

Fig. 5. Possible solutions of the instruction "Hamilton, go left.": Hamilton can move to the left from the hamilton-intrinsic (H) or the user-deictic (U) perspective.

one spatial reference frame may be preferred over the other one [JW96].

As an example, consider the situation in Figure 5. Two possible solutions can be offered by the system when the user has instructed Hamilton to go left. In addition, users' preferences for spatial reference frames may depend on the orientation of the object given in the actual situation. On the basis of the hamilton-intrinsic reference frame, Hamilton would move in the direction indicated by 'H'; on the basis of the user-deictic reference frame, Hamilton would move in the direction indicated by 'U'. Which reference frame is preferred depends on the preference of the current user in the given situation. Therefore, we have implemented two instances of the Hamilton sub-agency and equipped them with the contract-net learning framework. Whenever the user gives an instruction concerning the anthropomorphic figure, both contractors compute situation parameters, adjust their credits, and generate a bid including credits. Situation features concern the orientation of Hamilton and the distance between Hamilton and the user (camera). Any other agent of the interface agency, e.g. the coordinator agent, can act as a manager.

Similarly, we have implemented two space contractors 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 to the two situation features mentioned above, the space agency uses three further features to analyze the situations: intrinsic character of the reference object and of the possibly underlying object as well as the orientation of the underlying object.

Finally, we have tested the adaptation method for the case of users' preferences for different color sensation by implementing two color agents that offer more drastic or smoother color transformations. Situation features could pertain to the lightings of the scene but are, so far, not integrated in the system.

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 4. A fuller evaluation, to be carried out on the basis of artificial users, is one of our very next goals.

8 Discussion

This paper presented an multi-agent learning approach to user adaptation realized by an interface agency. The interface agency consists of several sub-agencies which represent different pre-established preference classes. Each sub-agency consists of several agents corresponding to the possible preferences. Agents use a contract-net based negotiation process where each agent can take on the role of a contractor as well of a manager. As contractors, agents acquire knowledge about the user, the preceding session, and situation parameters, which is captured in internal credit vectors. As managers, agents acquire knowledge about collaborating contractors and about their success or failure in the preceding session. By learning from indirect user feedback in the ongoing session, contractors and managers compete with each other to meet the users' preferences.

Besides extending and using the contract-net negotiation as an intuitive mechanism to achieve user adaptation, one of our main results is a general framework for adaptation to users' preferences which can easily be integrated in a contract-net based multi-agent system. By this, internal functionalities can be realized independently, and no supervision by any kind of a globally informed agent is needed. Furthermore, 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 by the use of implicit, distributed user models, without accumulating explicit user models.

In the future, our main working topics will concentrate on the evaluation of the learning strategy. In order to carry out replicable experiments, we will define artificial users and compare the results offered by the interface agency with predefined expectations of the artificial users. This evaluation will include the detailed evaluation of the contractor and manager function and its optimization.

References

[Chi90] Chin, D.N. Intelligent Interfaces as Agents. In Sullivan, J.W. & Tyler, S.W. (eds.): *Intelligent User Interfaces* (pp. 177-206). New York: ACM Press, 1990.

[DS83] 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.

[Dow95] Dowell, M.L. Learning in Multiagent Systems. Ph.D. Thesis at the Department of Electrical and Computer Engineering, University of South Carolina, 1995.

[JW96] Jörding, T., Wachsmuth, I. An Anthropomorphic Agent for the Use of Spatial Language. To be published in Olivier, P. & Maass, W. (eds.): *Vision and Language*, Springer, 1997.

[Kay90] Kay, A. User Interface: A Personal View. In Laurel, B. (ed.): *The art of human-computer interface design* (pp. 191-208). Reading: Addison-Wesley, 1990.

[LMM94] Lashkari, Y., Metral, M., Maes, P. Collaborative Interface Agents. In *Proceedings of the National Conference on Artificial Intelligence.* Cambridge (MA): The MIT Press, 1994.

[Lau90] 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.

[LW96] 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* (pp. 140-151). Berlin: Springer, 1996.

[LWC95] Lenzmann, B., Wachsmuth, I., Cao, Y. *An Intelligent Interface for a Virtual Environment.* KI-NRW (Applications of Artificial Intelligence in North-Rine Westphalia) Report 95-01, 1995.

[Mae94] Maes, P. Agents that Reduce Work and Information Overload. *Communications of the ACM 37*(7), 1994, 31-40.

[MK93] Maes,P., Kozierok, R. Learning interface agents. In *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 459-465). AAAI Press/The MIT Press, 1993.

[McT93] McTear, M.F. User modelling for adaptive computer systems: a survey of recent developments. *Artificial Intelligence Review 7*, 1993, 157-184.

[MCFMZ94] 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.

[Nor94] Norman, D.A. How Might People Interact with Agents. *Communications of the ACM 37*(7), 1994, 68-71.

[OHA96] Ohko, T., Hiraki, K., Anzai, Y. Learning to Reduce Communication Cost on Task Negotiation among Multiple Autonomous Mobile Robots. In Weiss, G. & Sen, S. (eds.): *Adaption and Learning in Multi-Agent Systems* (pp. 177-190). Berlin: Springer, 1996.

[Ret88] Retz-Schmidt, G. Various views on spatial prepositions. *AI magazine 9*(2), 1988, 95-105.

[Sel94] Selker, T. Coach: A Teaching Agent that Learns. *Communications of the ACM 37*(7), 1994, 92-99.

[WC95] Wachsmuth, I., Cao, Y. Interactive Graphics Design with Situated Agents. In W. Strasser & F. Wahl (eds.): *Graphics and Robotics* (pp. 73-85). Berlin: Springer, 1995.

[WLJJLF] Wachsmuth, I., Lenzmann, B., Jörding, T., Jung, B., Latoschik, M., Fröhlich, M. A Virtual Interface Agent und its Agency. Poster contribution to the *First International Conference on Autonomous Agents, Agents-97.*

[Wei96] Weiß, G. Adaptation and Learning in Multi-Agent Systems: Some Remarks and a Bibliography. In Weiß, G. & Sen, S. (eds.): *Adaption and Learning in Multi-Agent Systems.* Berlin: Springer, 1996.