Enhancing the Interaction between Agents and Users*

Marcelo Armentano, Silvia Schiaffino, and Analía Amandi

ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA
Campus Universitario, Paraje Arroyo Seco, Tandil, 7000, Argentina
CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
{marmenta,sschia,amandi}@exa.unicen.edu.ar

Abstract. A key aspect when interface agents provide personalized assistance to users, is knowing not only a user's preferences and interests with respect to a software application but also when and how the user prefers to be assisted. To achieve this goal, interface agents have to detect the user's intention to determine when to assist the user, and the user's interaction and interruption preferences to provide the right type of assistance at the right time. In this work we describe a user profiling approach that considers these issues within a user profile, which enables the agent to choose the best type of assistance for a given user in a given situation. We also describe the results obtained when evaluating our proposal in a calendar application.

Keywords: intelligent agents, user profiling, human-computer etiquette.

1 Introduction

In the last years, there has been an increasing interest in the area of human-computer etiquette [9]. Particularly, when talking about interface agents, researchers and developers are directing efforts towards learning how to best assist the user, that is providing the right help, at the right time and in the right way, without hindering the user's work. Learning users' habits, preferences and interests to provide them personalized assistance with a software application is not the only goal.

When working with a software application, a user can perform different tasks. Consequently, it is very important for an interface agent to know exactly the task the user is carrying out because it gives the context in which the user is working. By taking this context into account, the agent may infer the user's intention and try to collaborate with him. In addition, if the agent knows the user's intention, it will avoid interrupting him at an improper time. Users generally do not want to be interrupted while working on a specific task, unless this interruption is strongly related to the task they are performing, or it has a high priority. Also, users differ in their preferences about how and when they want to be assisted, and even a single user may differ in the type of assistance he prefers for different contexts [11,12]. For example, if the agent observes that the user is scheduling a work meeting for the following day, the agent can offer to automatically complete the information required and send an email on the

* This work was also supported by ANPCyT, Argentina, through PICT Project 20178.

G. Zaverucha and A. Loureiro da Costa (Eds.): SBIA 2008, LNAI 5249, pp. 53–62, 2008.

user's behalf to each participant of the meeting, provided that it knows the user's preferences about the kind of meeting he is scheduling. The user might accept this type of assistance for a certain type of meeting, but he might prefer only a suggestion or no assistance at all in a different context.

In this work, we propose a new definition for a user profile considering these issues. We also propose a profiling approach to acquire the different components of the proposed profile. This enhanced profile enables interface agents to decide how to best assist a user. Our profiling approach uses, first, plan recognition to detect a user's intentions. Plan recognition aims at identifying the goal of a subject based on the actions he performs [2]. Then, we use two user profiling algorithms we developed, namely *WATSON* and *IONWI*, to learn a user's interaction and interruption preferences [12]. Finally, we combine the different components of the user profile in a decision making algorithm that enables an interface agent to decide how to best assist a user in a given situation.

The rest of the work is organized as follows. Section 2 presents an overview of our proposed approach. Section 3 describes how to detect a user's intention using plan recognition. Section 4 describes how to learn a user's interaction and interruption preferences. Section 5 presents the results obtained when evaluating our approach. Then, Section 6 analyzes some related works. Finally, we present our conclusions.

2 Overview of Our Proposed Approach

A user profile typically contains information about a user's interests, preferences, behavioral patterns, knowledge, and priorities, regarding a particular domain. However, such information is not enough to personalize the interaction with a user. The user's intentions with a software application and his interaction preferences play a relevant role in user-agent interactions.

Consider for example the following situation. A user of a calendar application has the intention of arranging a work meeting with John Smith, his project manager. To achieve this, he has to perform a set of tasks, such as selecting John Smith from his contact list, creating a new event, entering the subject, date, place and all the information required about the meeting, and sending an email to John Smith. The sooner the agent detects the user's intention, the better it will assist him in accomplishing his intention. We propose to use Plan Recognition to detect the user's intention. Plan recognition aims at identifying the goal of a subject based on the actions he performs [2]. Once the agent has detected that the user wants to arrange a work meeting with John Smith, it can use the information contained in the classic user profile to assist him. However, different users may have different preferences about the type of assistance they welcome from an interface agent. For example, some users may prefer the agent to automatically complete all the tasks it can, while others just prefer to receive suggestions. Moreover, this information is strongly dependent on the situation in which the agent is about to assist the user. In our approach, the information needed to determine what type of assistance a user wants to receive in a given situation is contained in the user interaction profile. This profile also comprises the expected modality of the assistance. In a certain context the user might want just a notification

containing the suggested meeting date or place, while in a different context the user might prefer an interruption.

To obtain the proposed components of a user profile, we developed two profiling algorithms: *WATSON* and *IONWI*. *WATSON* learns a user's assistance preferences, that is, when a user wants a suggestion, a warning, an automated action or no assistance. *IONWI* learns a user's interruption preferences, that is, when a user prefers an interruption and when a notification. To achieve their goals, these user profiling algorithms analyze the user's interactions with the agent recorded when observing the user's behavior, and they consider the feedback provided by the user after the agent assisted him. An overview of our proposal is shown in Figure 1.

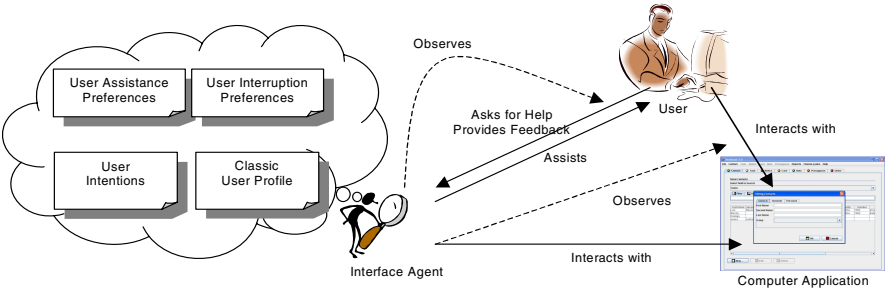

Fig. 1. Proposed user profiling approach

We define a user profile as: *User Profile = Classic User Profile + User Intentions + User Interaction Profile*, and *User Interaction Profile = User Assistance Preferences + User Interruption Preferences.* The user intentions are the set of all the possible intentions the user can be trying to achieve, each of them with a degree of certainty: *User Intentions = {<User intention, Certainty>}.* The assistance preferences are a set of problem situations or contexts with the required assistance action and a parameter (certainty) indicating how sure the agent is about the user wanting that assistance action in that particular situation: *User Assistance Preferences = {<Situation, Assistance Action, Certainty>}.*

We define the interruption preferences as a set of situations with the preferred assistance modality (interruption or no interruption). They might also contain the type of assistance action to execute. A parameter indicates how certain the agent is about this user preference: *User Interruption Preferences ={<Situation, [Assistance Action], Assistance Modality, Certainty>}.* The following sections explain how we obtain these components.

3 Detecting a User's Intentions

Plan recognition can be used to infer the user's intentions based on the observation of the tasks the user performs in the application. Plan recognizers take as inputs a set of goals the agent expects the user to carry out in the domain, a plan library describing the way in which the user can reach each goal, and an action observed by the agent.

The plan recognition process itself, consists in foretelling the user's goal, and determining how the observed action contributes to reach it. There are two main aspects that make classical approaches [2,10] to plan recognition unsuitable for being used by interface agents. First, the agent should deal with transitions and changes in the user intentions. Second, we have to take into account the influence of a user's preferences in the plan recognition process.

We propose Bayesian Networks (BN) [6] as a knowledge representation capable of capturing and modeling dynamically the uncertainty of user-agent interactions. We represent the set of intentions the user can pursue in the application domain as an Intention Graph (IG). An IG is materialized as a BN and represents a context of execution of tasks. BN are directed acyclic graphs representing probabilistic relationships between elements in the domain. Knowledge is represented by nodes called random variables and arcs representing causal relationships between variables. Each variable has a finite set of mutually exclusive states. Nodes without a parent node have an associated prior probability table. On the other hand, the strengths of the relationships are described using parameters encoded in conditional probability tables.

BN are used for calculating new probabilities when some evidence becomes available. By making use of BN probabilistic inference we will be able to know, having as evidence the set of tasks performed by the user, the probability that the user is pursuing any given intention modeled by the IG. Moreover, if the user explicitly declares his intentions, we will be able to probabilistically assess the tasks he has to perform to achieve his goal.

In our IG variables correspond to goals that the user can pursue and to tasks the user can perform in the application to achieve those goals. The two possible states of these variables are true, indicating that the user is pursuing that goal or that the user performed that task, and false. We call certainty of an intention to the probability of a variable being in a true state. Evidence on a task node will be set when the user interacts with a widget in the application GUI that is associated to the execution of that task. Our IG includes a third kind of variable: context variables. This kind of variables will be used to personalize the intention detection process by learning new relations that may arise between the attributes of the tasks performed by the user and the intention nodes in the IG.

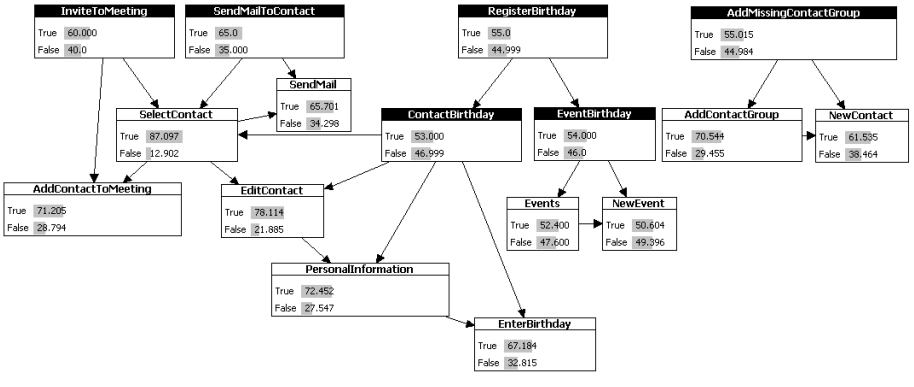

Fig. 2. Example of an intention graph

For example, in a calendar application, the user can select a contact from the address book with the objective of sending this contact an email or with the objective of scheduling a meeting with this contact, as shown in the IG presented in Figure 2. The IG constructed manually by a domain expert will allow the agent to rank which of the two goals is more probable, given that the user selected a contact in his address book. However, the information of the selected contact can be relevant in discerning which goal the user is pursuing. To consider this information, we introduce into the definition of our IG, the concept of traceable nodes. A traceable node is a node in which we want to register the values taken by some attributes of the corresponding task performed by the user, with the aim of adding new variables that represent the context in which the user performs that task and to find new relations between these variables and the nodes in the IG. In the example above, the task corresponding to the selection of a contact in the address book is a traceable node. The designer of the IG should decide which attributes of this task are of interest (for example, the city in which the contact lives or the group the contact belongs to) for which set of intentions (sending a mail to the selected contact or scheduling a meeting with him or her).

Each time the user performs a task corresponding to a traceable node, the agent will observe the values taken by the attributes of the task (for example, the selected contact is from New York and belongs to the group of friends). Then, the agent will continue observing the user until it can infer which his intention(s) are and will record the experience in an interaction history. Each experience will be of the form: $<attribute_1,,...,attribute_n, intention_1, ..., intention_k>$, where $attribute_i$ is the value taken by the $attribute_i$ and $intention_j$ is true if the agent infers that the user was pursuing $intention_j$, or false otherwise. This database of experiences is then used by the agent to run a batch learning algorithm and a parametric learning algorithm to find relations between the attributes and between the attributes and the intentions [6]. To adapt the probabilities (set by the domain expert who constructed the network) to a particular user's behavior, we take an statistical on-line learning approach [6].

Finally, as stated in Section 2, most of previous plan recognition approaches do not consider the uncertainty related to the moment in which the user starts a new plan to achieve a new goal. Those which consider this issue limit the memory of the plan recognizer by making evidence to be present in a fixed interval of time and then completely disregarding it. We take a different approach in which evidence is gradually forgotten. We adopt the concept of soft evidence to fade the evidence we entered to the BN as the user performs further tasks [6]. To do this, we use a fading function to gradually forget the tasks performed by the user. Evidence is faded according to this function until it reaches its original value, that is until the probability of a given node becomes less than the value that it would have if we would not have observed the execution of the corresponding task in the application. Fading functions can be any function that, given the IG and the evidence on tasks performed so far, decrements the certainty of the evidence gradually, according to some heuristic. For example, we can decrement current evidence by a fixed factor $0 \leq \nabla \leq 1$ every time the user performs a task in the application.

4 Learning a User's Interaction and Interruption Preferences

To learn a user's interaction and interruption preferences, the information obtained by observing a user's behavior is recorded as a set of user-agent interaction experiences. An interaction experience *Ex=<Sit, Act, Mod, UF, E, date>* is described by six arguments: a situation *Sit* that originates an interaction; the assistance action *Act* the agent executed to deal with the situation (warning, suggestion, action on the user's behalf); the modality *Mod* that indicates whether the agent interrupted the user or not to provide him/her assistance; the user feedback *UF* obtained after assisting the user; an evaluation *E* of the assistance experience (success, failure or undefined); and the *date* when the interaction took place. For example, if we consider an agent assisting a user of a calendar management system, an assistance experience could be the following. John Smith is scheduling a new event: a meeting to discuss the evolution of project A with his employees Johnson, Taylor and Dean. The event is being scheduled for Friday at 5 p.m. at the user's office. The agent has learned by observing the user's actions and schedules that Mr. Dean will probably disagree about the meeting date and time because he never schedules meetings on Friday evenings. Thus, it decides to warn the user about this problem. In reply to this warning, the user asks the agent to suggest him another date for the event.

To obtain a user's interruption and assistance preferences, we developed two algorithms: *WATSON* and *IONWI*. These algorithms use association rules to discover the existing relationships between situations or contexts and the assistance actions a user requires to deal with them, as well as the relationships between a situation, a user task, and the assistance modality required.

Association rules imply a relationship among a set of items in a given domain. As defined by [1], association rule mining is commonly stated as: Let $I = i_1,\ldots, i_n$ be a set of items and D be a set of transactions, each consisting of a subset X of items in I. An association rule is an implication of the form $X \rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \varnothing$. X is the rule's antecedent and Y is the consequent. The rule has support s in D if s percent of D's transactions contains $X \cup Y$. The rule $X \rightarrow Y$ holds in D with confidence c if c percent of D's transactions that contain X also contain Y. Given a transaction database D, the problem of mining association rules is to find all association rules that satisfy minimum support and minimum confidence.

We use the Apriori algorithm [1] to generate association rules from a set of user-agent interaction experiences. Then, we automatically post-process the rules Apriori generates so that we can derive useful information about the user's preferences from them. Post-processing includes detecting the most interesting rules according to our goals, eliminating redundant rules, eliminating contradictory rules, and summarizing the information obtained. To filter rules, we use templates or constraints [7] that select those rules that are relevant to our goals. For example, we are interested in those association rules of the forms: *situation, assistance action* $\rightarrow$ *user feedback, evaluation,* in the *WATSON* algorithm, and s*ituation, modality, [assistance action]* $\rightarrow$ *user feedback, evaluation*, in the *IONWI* algorithm, where brackets mean that the attributes are optional. Rules containing other combinations of attributes are not considered. To eliminate redundant rules, we use a subset of the pruning rules proposed in [13]. Basically, these pruning rules state that given the rules $A,B \rightarrow C$ and $A \rightarrow C$, the first rule is redundant because it gives little extra information. Thus, it can be deleted if the two

rules have similar confidence values. Similarly, given the rules A→B and A→B,C, the first rule is redundant since the second consequent is more specific. Thus, the redundant can be deleted provided that both rules have similar confidence values. Then, we eliminate contradictory rules. We define a contradictory rule in *WATSON* as one indicating a different assistance action for the same situation and having a small confidence value with respect to the rule being compared. Similarly, in *IONWI*, a contradictory rule is one that indicates a different assistance modality for the same context. After pruning, we group rules by similarity and generate a hypothesis that considers a main rule, positive evidence (redundant rules that could not be eliminated), and negative evidence (contradictory rules that could not be eliminated). The main rule is the rule in the group with the greatest support value. Once we have a hypothesis, the algorithm computes its certainty degree by taking into account the main rule's support values and the positive and negative evidence. To compute certainty degrees, we use equation 1:

$$Cer(H) = \alpha Sup(AR) + \beta \frac{\sum_{k=1}^{r} Sup(E+)}{\sum_{k=1}^{r+t} Sup(E)} - \gamma \frac{\sum_{k=1}^{t} Sup(E-)}{\sum_{k=1}^{r+t} Sup(E)} \quad (1)$$

where α, β, and γ are the weights of the terms in the equation (we use α=0.7, β=0.15 and γ=0.15), $Sup(AR)$ is the main rule support, $Sup(E^+)$ is the positive evidence support, $Sup(E^-)$ is the negative evidence support, $Sup(E)$ is the support of a rule taken as evidence (positive or negative), r is the amount of positive evidence, and t is the amount of negative evidence. If the certainty degree of the hypothesis is greater than a given threshold value δ, it becomes part of the user profile. Otherwise, it is discarded.

5 Experimental Results

We carried out two types of experiments. First, we studied the influence of users' preferences on the detection of users' intentions with plan recognition. Then, we analyzed the precision of our profiling approach at assisting users.

5.1 Evaluation of Our Plan Recognition Approach

To analyze the influence that the user's preferences have on the detection of the user's intention, we selected an scenario in which a user used a calendar application to organize a meeting with some contact in his address book, and then selected another contact to register his or her birthday. To achieve these goals, the user performed the following sequence of tasks: Select Contact, Add Contact To Meeting, Select Contact, Edit Contact, Personal Information, Enter Birthday. We recorded the certainty values for all the intentions both in the IG containing the user's preferences information and in the same IG without context nodes (the one owned originally by the agent). Figure 3 shows the evolution of the certainty values of related intentions. Intentions in the original IG are indicated with completed lines. In the first time slice, we show the a priori probabilities of each intention when the user has not perform any tasks in the application yet. "Send Mail To Contact" is the most probable intention, while "Contact Birthday" is the least probable one. When the user performed the first task,

"Select Contact", the ranking remained unchanged, although there was a small increment in those intentions that contained this task. Then the user performed "Add Contact To Meeting", and "Invite Contact To Meeting" became the most probable intention. With the next set of tasks performed by the user, "Contact Birthday" increased its certainty while the other intentions decreased them. The agent considered a threshold value of 0.7 to believe in the intention pursued by the user. It needed both tasks to be performed to detect "Invite To Meeting", and three tasks out of four to detect "Contact Birthday".

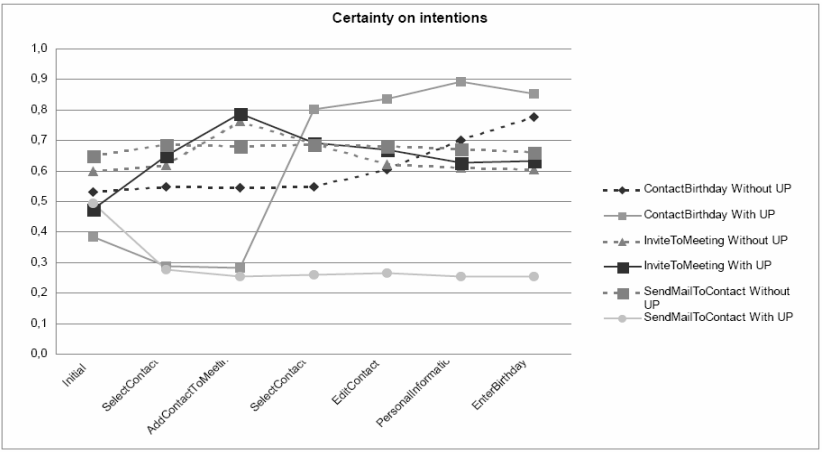

Fig. 3. Experimental results obtained with plan recognition

Dotted lines in Figure 3 show the same scenario but using the IG with context nodes merged. The first "Select Contact Task" was performed when the user selected a contact from the "Friends" group, living in "New York", who already had the birthday registered. We can see that the certainty for "Invite To Meeting" is higher only with the first task performed. We can also see that the other intentions dramatically decreased their certainty values. Obviously the user would not register a birthday in the contact information because the selected contact already had a birthday date.

The second contact selected by the user was also from the "Friends" group and its city was "New York", but the birthday was not registered yet in this case. So, we can see that with the mere selection of the contact, "Contact Birthday" intention could be predicted.

It is worth noting that the curves corresponding to the possible user's intentions are closer when we do not consider the user's preferences than when we do. Another interesting fact that can be observed in Figure 3 is that the certainty of finished intentions gradually decrements to its original value, as happens with "Invite To Meeting" intention. This is due to the fading function used by the IG that gradually decrements by a fixed constant to the strength of the evidence on the performed tasks. Similar experiments were carried out with different scenarios, with similar results.

5.2 Evaluation of Our Profiling Algorithms

To evaluate the precision of our interaction profiling approach at assisting users, we studied the number of correct assistance actions, where the "correctness" is determined by the user through explicit and implicit feedback. To do this, we used a precision metric that measures an agent's ability to accurately assist a user. We used this metric to evaluate the agent's performance in deciding between a warning, a suggestion, or an action on the user's behalf; and between an interruption or a notification. For each problem situation, we compared the number of correct assistance actions against the total number of assistance actions the agent executed. The resulting ratio is the agent's precision. To carry out the experiments, we used 33 data sets containing user-agent interaction experiences in the calendar management domain. Each database record contains attributes that describe the problem situation (or the situation originating the interaction), the assistance action the agent executed, the user feedback, and the user's evaluation of the interaction experience. The data sets contained anywhere from 30 to 125 user-agent interactions[1]. We studied four calendar-management situations: new event, the user is scheduling a new event, and the agent has information about the event's potential time, place, or duration; overlapping, the user is scheduling an event that overlaps with a previously scheduled event; time, not enough time exists to travel between the proposed event and the event scheduled to follow it; holiday, the user is scheduling a business event for a holiday.

We compared the precision values for three approaches: confidence-based (this algorithm is classical in agents [8]), *WATSON*, and *WATSON+IONWI*. The values were obtained by averaging the precision for the different datasets belonging to the different users. We used percentage values because the number of user-agent interactions varied from one user to another. Our approach had a higher overall percentage of correct assistance actions or interactions, 86% for *WATSON* and 91% for *WATSON+IONWI*, than the confidence-based algorithm, which got a 67% of precision. In some situations, although the etiquette-aware agent knew how to automate an action on the user's behalf it only made him a suggestion because it had learnt that the user wanted to control the situation by himself. The agent using the standard algorithm made an autonomous action in that case because it knew what the user would do. However, it did not consider that the user wanted to do the task by himself. When the problem is infrequent and warnings are required, the three algorithms behave similarly.

6 Related Work

Our work is related to those works that study the etiquette of human-computer relationships [9], since learning when to interrupt a user, detecting a user's intentions, and deciding how to best assist him can be considered as part of this etiquette. Considering these issues within interface agent development, and particularly within user profiles, is novel.

Regarding interruptions, they have been widely studied in the Human-Computer Interaction area, but they have not been considered in interface agent development.

[1] The datasets are available at http://www.exa.unicen.edu.ar/~sschia

With respect to agents using plan recognition to detect a user's plans, some works have been done in this direction [4,10]. However, they do not consider the user's preferences within the process. Some algorithms have been proposed to decide which action an agent should execute next. These algorithms adopt mainly one of two approaches: some use decision and utility theory [3,5], and others use confidence values attached to different actions [8]. However, these works do not consider a user's interaction preferences, the possibility of providing different types of assistance, or the particularities of the situation at hand.

7 Conclusions

In this article, we presented an approach to enhance the interaction with users that considers the user's intentions and the user's interaction preferences. To detect a user's intentions we propose a plan recognition approach, which considers the user's preferences to allow an earlier detection. To learn a user's interaction preferences, we proposed two profiling algorithms. We evaluated our proposal with promising results. As a future work, we will evaluate our approach in a different application domain.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. 20th Int. Conf. on Very Large Data Bases (VLDB 1994), pp. 487–499 (1994)
2. Charniak, E., Goldman, R.P.: A bayesian model of plan recognition. Artificial Intelligence 64(1), 53–79 (1993)
3. Fleming, M., Cohen, R.: A user modeling approach to determining system initiative in mixed-initiative AI systems. In: Proc. 18th Int. Conf. On User Modeling, pp. 54–63 (2001)
4. Horvitz, E., Heckerman, D., Hovel, D., Rommelse, K.: The Lumière Project: Bayesian User Modeling For Inferring The Goals And Needs Of Software Users. In: Proc. 14th Conf. on Uncertainty in Artificial Intelligence, pp. 256–265 (1998)
5. Horvitz, E.: Principles of Mixed-Initiative User Interfaces. In: Proc. ACM Conf. Human Factors in Computing Systems (CHI 1999), pp. 159–166 (1999)
6. Jensen, F.: Bayesian Networks and Decision Graphs. Springer, New York (2001)
7. Klementinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A.I.: Finding interesting rules from large sets of discovered association rules. In: 3rd Int. Conf. on Information and Knowledge Management, pp. 401–407 (1994)
8. Maes, P.: Agents That Reduce Work And Information Overload. Communications of the ACM 37(7), 30–40 (1994)
9. Miller, C.: Human-computer etiquette: Managing expectations with intentional agents. Communications of the ACM 47(4), 31–34 (2004)
10. Rich, C., Sidner, C., Leash, N.: Collagen: Applying Collaborative Discourse Theory To Human-Computer Interaction. Artificial Intelligence 22(4), 15–25 (2001)
11. Schiaffino, S., Amandi, A.: User – interface agent interactions: personalization issues. International Journal of Human Computer Studies 60(1), 129–148 (2004)
12. Schiaffino, S., Amandi, A.: Polite Personal Agents. IEEE Intelligent Systems 21(1), 12–19 (2006)
13. Shah, D., Lakshmanan, L., Ramamrithnanm, K., Sudarshan, S.: Interestingness and Pruning of Mined Patterns. In: Proc. Workshop Research Issues in Data Mining and Knowledge Discovery. ACM Press, New York (1999)