An Interaction Agent in a Brokerage Environment : Methodology, Design and Application

Author Names :
Koutsabasis P., Darzentas J. S., Spyrou T., Darzentas J.,
Research Laboratory of Samos
Department of Mathematics
University of the Aegean
GR 83200 Karlovassi, Samos
Greece
email:kgp@aegean.gr

Abstract

This paper presents a framework for capturing the interaction requirements and provides the design and implementation of an interaction agent, focusing on the use of HCI techniques. The term 'interaction agent' is used to describe an interface agent that facilitates the interaction between the user and the underlying system. The purpose of an interaction agent is to offer personalised services at the interface level of the underlying subsystem. It is intended that by using the interaction agent users will perform their tasks more effectively. The agent provides assistance with the input to the system, and adapts the output to the user. It can offer alternative ways of carrying out tasks, and can guide the sequence of tasks or suggest ways of performing them. The interaction agent has been partially implemented in the context of ACTS (AC221) project GAIA at the time of writing, according to the project work plan.

Introduction

The intelligent agent paradigm has gained much popularity in the last few years, although the concept was first discussed almost three decades ago [1]. From the application point of view, there are agents that are invisible to the user. These could be broadly described as autonomous processes that perform well-defined tasks, such as network monitoring (SNMP MIB agents), alerting services [40], autonomous searching (search engines' spiders), etc. Such autonomous processes qualify as agents because they possess many of the characteristics that define agents such as autonomy, adaptivity, reactivity, pro-activity, social ability, robustness [20,35,29,25] etc.

Other agents, while still possessing agent features, are, on the other hand, highly visible to the user. They take on the metaphor of agent in the real-world meaning of the term agent, as in travel agent, or estate agent, or personal assistant. These agents are sited at the interface of a computer system and «intervene between the human user and the computer system» [21]. It is claimed that user interface agents act as a bridge between a user's goals and the computer. Such agents can make the interface more intuitive and can encourage types of interactions that might be difficult to evoke with a conventional interface [38]. A variety of user interfaces that employ some sort of agency attributes have been named as interface agents, from Web search engines, to lifelike, animated characters.

In this paper the term 'interaction agent' is used to describe an interface agent that facilitates the interaction between the user and the underlying system. The interaction agent is naturally the visible part of the computer based system. It interacts with both ends of communication: user and underlying subsystem, and is capable of reasoning about inputs it gets from both sides of interaction. In this context, a search engine could not be named as an interaction agent as it is not interactive. On the other hand, a lifelike animated character could be characterised as an interaction agent if it is capable of reasoning about its actions during interaction.

The purpose of an interaction agent is to offer personalised services at the interface level of the underlying subsystem. Its is intended that by using the interaction agent users will perform their tasks more effectively. The agent provides assistance with the input to the system, and adapts the output to the user. It can offer alternative ways of carrying out tasks, and can guide the sequence of tasks or suggest ways of performing them.

This paper gives an account on a methodology, based on HCI methods and techniques, that has been employed to capture the interaction requirements and to provide the design of an interaction agent applicable in a brokerage environment. Through the description of this methodology the problems encountered in designing agent software are identified from a variety of perspectives. Additionally the paper presents the technical approach taken, presenting its advantages and limitations with regard to a number of Web agent, interface design and

technology issues. The implementation of the user interface is discussed as well as future work. The interface agent discussed in this paper, is sited at the interface of GAIA [40]. The GAIA project is developing a sector and supplier independent Generic Architecture for Information Availability to support multilateral information trading. In the context of GAIA, the purpose of the interaction agent is to offer personalised services at the interface level of this brokerage system (and as GAIA defines a generic architecture, this interaction agent could be employed to offer brokerage services in other implementations of brokerage systems as well). The user tasks, during interaction with the brokerage system, have been defined and analysed according to the framework introduced in this paper. The interaction agent has been partially implemented at the time of writing, according to the project work plan.

Previous and Related Work

Considerable work has been done in the last few years with regard to interface agents. They have been mostly employed to assist users with their personal everyday tasks such as managing electronic mail [22], meeting scheduling, [22, 30], personalised information filtering [22,33], recommendation systems [11,23,37,41] and others [27,35]. Such systems, while they emphasise the capability of the agent to utilise user preferences, do not claim to be based on HCI techniques and usability principles.

There is keen interest from the HCI community in interface agents, although there is no work known to the authors which specifically discusses the design of interface agents. However, this paper shows how HCI techniques such as task analysis[5,6,18], user modelling [15,14] and usability engineering [12,13,36] can be used to help design the interaction requirements of this type of agent.

Problem Space

Currently, information brokerage systems are capable of searching enormous quantities of data, often from heterogeneous information sources. In this they can be extremely efficient, producing vast result sets, drawn from repositories sited world wide. However, it seems they lack an important ingredient for success- namely, the absence of a mediator capable of personalising customer services. In contrast, human brokers, such as travel agents, or librarians, before commencing a search, spend time in discussion with their clients in order to better ascertain their needs. After a broker collects information for a client, he spends time discussing the significance of the information with the client, weighing it up and assessing whether or not it is the information required by the client.

This mediation phase is non-existent or very rudimentary in most information brokerage systems. They rely on the input to the system being a fairly precise expression of the user's requirements. This gap between information availability and the relevance of information to the user, often results in "information overload" in various guises: the sheer volume of returned results is overwhelming, much of the information is not relevant to the user, some of the material is not accessible to the user because he does not possess the necessary computer infrastructure or is not sufficiently expert in computer literacy to be able to "decode" various formats, etc. The result is frustration and exasperation, and loss of precious time.

On the other hand information providers know that the quantity of information they provide is increasing and that unless they provide their customers with useful and usable ways of accessing that information customers will go elsewhere for that information. The prevailing subscription business model, combined with increase in the number of content service providers, will result in customers being more discriminating. The loss of revenue to information providers, or indeed information brokers, who do not attempt to deal with the problem of information overload could be decisive to their survival.

Information system developers attempt to tackle the problems of both user and information providers and combat the negative effects of information overload with well known solutions of search engines, personal agents, and other software that are offered commercially. However these are often piecemeal solutions which offer limited help. Even after using them, users may still have a vast set of results returned and information providers still cannot be sure that users will access their information when it's hidden in huge result sets.

In this paper interaction agents are proposed as a new concept and tool that visualises the metaphor of the agent – personal assistant to computer – based applications that tend by their very nature to produce information overload to

users. The research in interaction agents involves a variety of issues from requirements capture to technology selection and deployment. This paper presents a methodology for capturing interaction requirements and presents this methodology in practice.

The use of interaction agents should not be treated as a solution applicable to all types of interaction problems. As each application has it own user interface requirements, the decision of designing a directly manipulated user interface, an interface agent, or an interaction agent must be based on a generic approach (a methodology) that will reveal the specific needs of users that are going to use the system. It is commonly accepted that these users should be the starting point of the study of such requirements. It is most usual however that the application to be developed is intended to be used by a variety of users who differ in a number of aspects (e.g. familiarity with using computer systems, knowledge of the tasks, etc.). In this case there is obviously need for an adaptive and interactive user interface. Generally speaking, if it is possible for the system to undertake tasks on behalf of the user, or to perform tasks collaboratively with him, an interaction agent can be employed to perform (autonomously or collaboratively) such tasks.

Methodology for Designing the Interaction Agent

The common practice for agent design is to base them on the reflection of the problem into architectures and the deployment of those architectures into software. There has been a variety of agent architectures that attempt to present and explain agent behaviour. These architectures focus either on the description of an agent as a autonomous reasoner/planner [8,17,9,10] and are thus are characterised by a degree of generality, or are based on the domain of application [26,35,27,19].

In the first case, agent architectures cannot provide a generic model that would describe human-agent interaction, because their focus is on the internal structure of agent modules and functions. In the second case, agent architectures seem to regard the agent as a type of software that performs well-defined computer tasks, and they focus on describing the performance of those tasks. In both cases agent architectures do not provide a clear design methodology with regard to how the user interface would be developed.

Agent architectures are of course necessary in order to describe the agent inference mechanism and the knowledge the agent needs to employ, and in this context they are used in this paper. It is considered essential that the description of an agent architecture at least provides some hints to the developers of the system relatively to system development issues and programming tools to be used.

The approach that was taken was to develop for the interaction agent a design methodology directed toward the following aims:

1. *To define the user tasks to be performed;*
2. *To capture the Interaction Knowledge required;*
3. *To define usability goals;*
4. *To prototype the user interface and agent behaviour;*

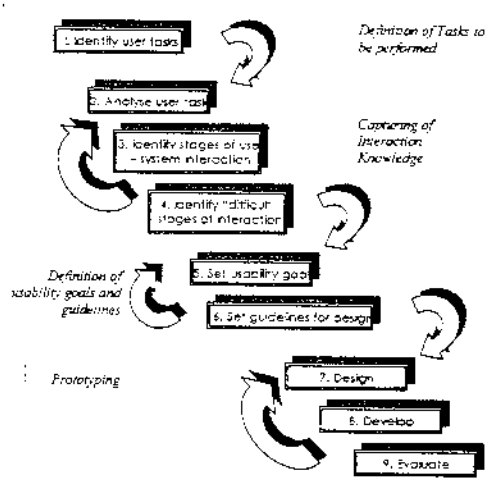

Figure 1 A Generic Approach for Interaction Requirements Capture and Design.

The remainder of this section presents the application of the above approach and the techniques used to tackle the problem of user interaction with a brokerage system as well as discusses the application of this approach into the GAIA project.

Identification of User Tasks

As part of the design of the overall brokerage system architecture represented by GAIA an extensive document capturing the GAIA domain and user requirements had been produced, and this was used as a starting point for identifying the user tasks within a brokerage system. This analysis resulted in a set of user tasks which were divided into three categories:

A. Tasks that consist of the *actual* user interaction with the brokerage system: (1) Searching for desired items, (2) Evaluating search results, (3) Ordering preferred items under the best terms.
B. Management of Personal Information: (1) Shopping Basket, (2) Preferences Profile, (3) Ordering Profile, (4) Alerting Profile, (5) Session History.
C. Other Tasks: (1) Generic Information about GAIA, (2) Information about GAIA Tariffing Policy, (3) Information about GAIA Privacy Policy.

Further analysis of these tasks needed to be undertaken to determine the stages of the user-system interaction, and the subset of those stages which represents difficulties for the user (and thus justifies the use of an interaction agent) as well as the knowledge each interacting entity (user and agent) should possess in order to interact successfully with one another. The next section describes the process used to capture this requirement.

Capture of Interaction Knowledge

A task analysis was performed with regard to the above described tasks in order to capture the interaction requirements. From among a variety of known and applied techniques [18,16], the Task Knowledge Structures (TKS) technique [5] was selected and used to identify the knowledge for the performance of user and agent tasks. TKS identifies the knowledge about the following components for a given task: roles, goals, sub-goals, sub-tasks, procedures, strategies, actions and objects. It can assist the analyst to capture interaction requirements by identifying the stages of interaction and the knowledge required in each stage of it, in order to perform a task. It does not however describe the «particular form of presentation that the user interface might have».

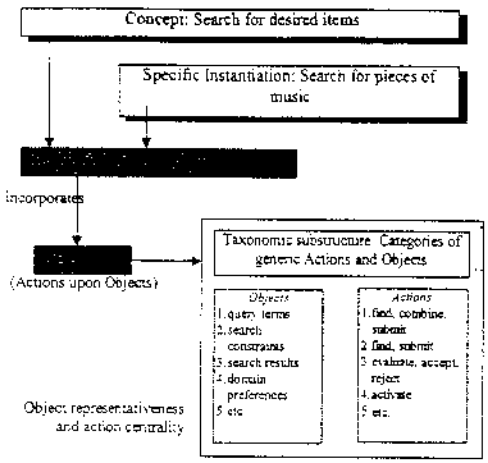

Figure 2 "Search for Desired Items" TKS.

Figure 2 provides an approximation to describe a user's TKS when the task of searching for desired items is carried out. The actions the user needs to perform in this TKS can be separated into actions that may put heavy cognitive load on users (find, evaluate, activate), and actions that are actual instructions to the system lie checking boxes and do not require any deep mental processing. *It is suggested that the sub-tasks / actions that are likely to put cognitive load on the users, as these are identified by task analysis, are those stages of interaction when an interaction agent can best assist the user-system interaction.* A plan-goal oriented substructure that the user would need to follow in order to perform the task is shown in figure 3.

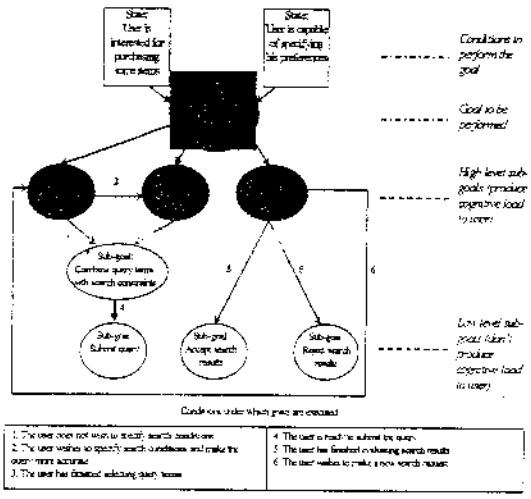

Figure 3 "Search for Desired Items" Plan.

The high level sub-goals imply that mental effort from the user is required. During the performance of actions that may produce

cognitive load to users, an interaction agent can assist the user–system interaction. In such a way the agent becomes a personalised assistant, employing knowledge about user preferences. So in the case of the procedure find(query terms), the interaction agent can propose(query terms) that the agent 'thinks' match the user's preferences. With regard to the procedure evaluate(search results) the agent can perform its own evaluation of search results and present them in an manner appropriate to cognitive considerations. On the other hand in the case of the procedure find(search constraints) the search constraints are imposed by the underlying search mechanism.

The application of TKS results in a definition of user tasks, and the identification of user-system interaction stages. Furthermore it reveals stages of interaction where the actions (find, evaluate) and objects (user preferences) used typically place load cognitive load on the user. The use of an interaction agent, in the role of a personal assistant provides an interactive and user friendly interface. The agent's goals are to assist users perform the actions that represent heavy cognitive load according to TKS. The next section presents the agent inference mechanism employed to deal with the identified problems.

Usability Principles and Paradigms in User Interface Design and interaction agent design

The primary objective of any interactive system would be to provide a usable user interface. In order to design and develop a usable system, and evaluate its usability, the designer should both use a *paradigm* : an example of an already usable (and learnt) system, and base his design on *usability principles*.

A history of usability paradigms include such notions as window systems, the WIMP (Windows, Icons, Mice, Pull down menus) interfaces, the use of metaphors, direct manipulation [16]. But while for instance, direct manipulation had been a solution to many interface problems, in the face of more difficult issues that have emerged, such as information filtering in large information spaces, there has been a notable paradigm shift from passive to active systems, or from direct to indirect manipulation [22]. The understanding is that these active systems will perform some tasks on behalf of the user, and this requires that the users trust the system. In order that the user «trust» the agent the questions of control and confidentiality are probably the two that have emerged as the most important. [22], [32].

Work on usability principles[16] divides them into three categories: those affecting **learnability** such as: consistency, predictability, familiarity; those affecting **robustness** such as task conformance, recoverability, responsiveness; those affecting **flexibility** such as pre-emptiveness and initiative taking.

Although there have been attempts of formal specification of usability principles into user interface design, e.g. via Z notation [18], the deployment of usability principles into user interface design has not been achieved by some HCI design methodology, mostly due to the fact that the users who are going to use the system are the starting point of such methodologies. Designers have to be aware of usability principles though, in order to come up with usable designs. Additionally designers need to take into account of rules as: «the average person can remember 7+/-2 chunks of information», or «providing clear exit points to the user», established by experimental work [16]. In section 4. Implementation, the application of some of these principles and rules in the design of the interaction agent interface are discussed.

Interaction Agent Inference Mechanism

After identifying the stages of user – system interaction, and especially those stages when interaction agent help is required, the designer must decide the interaction agent inference mechanism that will cope with those problems. There is a variety of inference mechanisms – especially with regard to expert systems development- [3, 4,6]. The designer has to decide which of them are most suitable to the given problem as well as to take into account the programming tools that would be most suitable to each alternative.

In this case collaborative information filtering algorithms are being developed in order to provide agent behaviour to the user interface. Collaborative information filtering algorithms [11,23,37,41] seem an efficient solution with regard to both the tasks that this interaction agent will perform, and the tools that have been used for development.

Implementation

The user interface paradigm used was the WIMP, as this is a well known paradigm to users

of PCs, even to users of UNIX-based workstations (who are in their vast majority expert users anyway). Furthermore the Java programming language by supporting this paradigm has enabled designers and programmers of user interfaces to provide such interfaces from the Web. According to the WIMP paradigm, one of the most usual ways to initiate the user-system interaction is by using hierarchical drop down menus. Each menu should represent a category of tasks to be performed and collect the basic (sub)tasks within this category. In our case the categories of tasks had been identified (see section 3.1.1. Identification of User Tasks) and provided the organisation of the drop down menus. Thus, there exist three drop down menus, each one offering the functionality for the execution of (sub)tasks that fall into one of the three categories of tasks described in section 3.1.1, plus one drop down menu with only one possible selection: <Exit> to comply with the need for clear exit points.

According to task analysis, the first task a user will need to perform in order to interact with the brokerage system is to search for items of his interest. Task analysis showed that users may wish to perform a simple search (when they are unable to specify their query precisely), or a structured search. Furthermore the agent can propose query terms to users if they choose to ask for its assistance. Those alternatives can be activated from the first menu.

The three basic user tasks (search for items of interest, evaluate search results, order, see section 3.1.1 as well), can be performed from the three subspaces into which the user interface is divided. All subspaces are not activated simultaneously. In order for the search results subspace to be activated, the user has to perform a search request. The same happens with the activation of the ordering subspace.

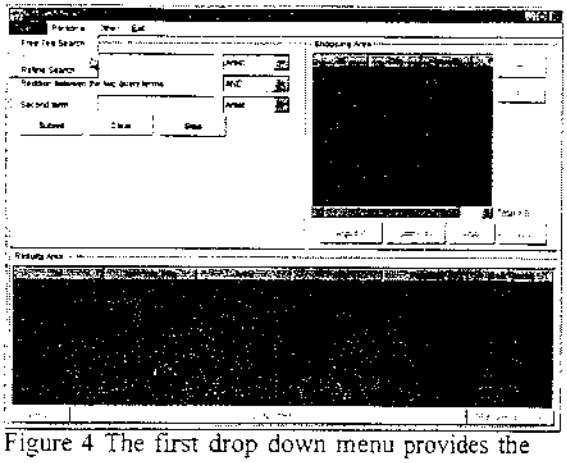

Figure 4 The first drop down menu provides the three alternative ways of searching.

These facts contribute towards three aims: a) the interface is consistent in terms of the way and space into which a (sub)task will be performed, since all options with regard to a (sub)task are available in its corresponding subspace, b) it is difficult for the user to make an error, because other options are not activated and b) in this way the system guides the user during interaction. Consistency is considered to be achieved with regard to another feature as well: tasks that compose a task category are performed in the same way. While actions that belong to the first category of tasks are performed from the main user interface space, actions that belong to the other two categories of tasks are performed from separate pop up windows.

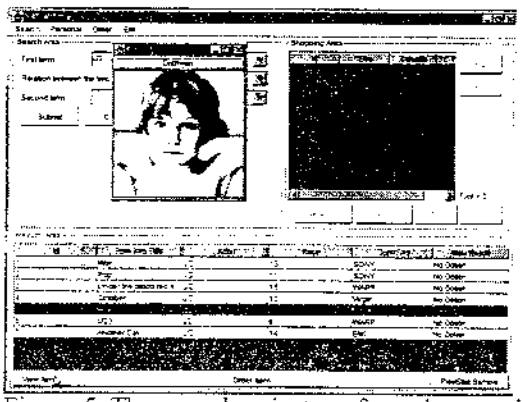

Figure 5 The user has just performed a search request. The basic characteristics of search results are presented in a table, while more information is available in separate pop up windows.

The only case in which pop up windows are presented in the case of performing tasks that fall into the first category of tasks, and exists when the user wishes to be presented with more

information about a search or order item. This happens in order to avoid information load. The user should not be presented with all information in one step since it is considered that this would cause cognitive load. What has been attended though is not to present the user with more than one pop up window at a time (as we often see in Windows based operating systems: Win95 and NT, as well as in other commercial applications). Apart from reducing cognitive load, this also means that a user can access a user interface function at the most by performing two logical steps. It has also been attended not to put more than seven chunks of information in every logical parts of the interface [16].

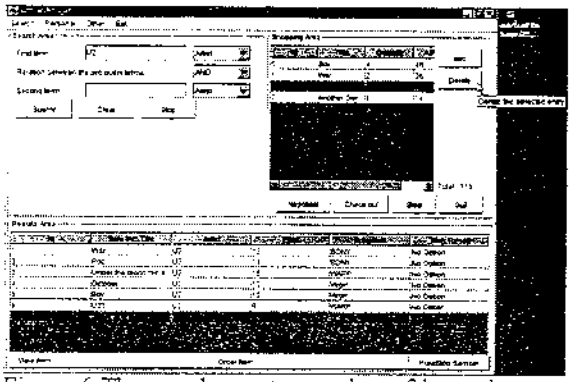

Figure 6 The user has put a number of items into his shopping basket.

With regard to robustness, the user is provided with clear exits or cancel points, in every action that this is required. Additionally the history of information about the latest user actions, as well as the storage of past sessions' shopping baskets will contribute towards this aim.

The use of well known paradigms of real life metaphors is of great importance, since it contributes towards learnability and specifically predictability. This interface employs the use of tables, which is a user interface component commonly used in tasks that are relative to management of information and has been used in logistics applications, e mail management (Eudora), etc. The shopping basket metaphor, used already by many shopping interface applications on the Web, is considered very successful as well, and thus has been employed in this interface too.

Conclusions

This paper presented a framework for capturing the interaction requirements and provided the design of an interaction agent, focusing on the use of HCI techniques. The selection of inference mechanisms used is considered with respect to the tasks that the agent needs to perform. The application of this framework is important that it results to a robust interaction agent design.

References

[1] Negroponte, N. (1970) The Architecture Machine: Towards a More Human Environment, MIT Press, 1970

[2] Salton G., McGill M., (1983) Introduction to Modern Information Retrieval, McGraw-Hill, 1983.

[3] Dubois D., and Prade H., (1988) An introduction to Possibilistic and Fuzzy Logics, Non-standard logics for Automated Reasoning, Academic Press, London, pp. 287-326.

[4] Pearl J., (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA.

[5] Johnson P., Johnson H., Waddington R., Shouls A., (1988) Task-Related Knowledge Structures: Analysis, Modelling and Application, People & Computers: from research to implementation, Cambridge Press pp. 35-62, 1988.

[6] Johnson P., Johnson H., (1988) Task Knowledge Structures: Psychological basis and integration into systems design, Acta Psychologica, 78, pp. 3-26

[7] Riesberg C., K., and Shank R., C., (1989) Inside Case – Based Reasoning, Lawrence Erlbaum Assosiates, Hillsdale, NJ,

[8] Hayes-Roth B. (1991) An Integrated Architecture for Intelligent Agents SIGART Bulletin 2, 1991, 82-84].

[9] Laird J., Hucka M., Huffman S., (1991) An analysis of Soar as an integrated architecture, SIGART Bulletin 2, 85-90, 1991

[10] Vere S., (1991) Organisation of the Basic Agent, SIGART Bulletin 2, 151-155.

[11] Goldberg, David; Oki, Brian; Nichols, David; Terry, Douglas B. (1992) ``Using Collaborative Filtering to Weave an Information Tapestry,'' Communications of the ACM, December 1992, Vol 35, No 12, pp. 61-70.

[12] Nielsen J., (1992) The Usability Engineering Life Circle, IEEE Computer, March 1992

Nielsen J., (1992) Usability Engineering, Academic Press, 1992

[14] Barnard, P. J. (1993) Modelling Users, Systems and Design Spaces (Esprit Basic Research Action 3066). Proceedings of HCI 93, pp.331-342. 19A Elsevier

[15] Byerley, P.F. Barnard, P.J. and May, J., editors (1993) Computers. Communication and Usability: Design issues. research and methods for integrated services. (North Holland Series in Telecommunication) Elsevier: Amsterdam.

[16] Dix A., Finlay J., Abowd G., Beale R., (1993) Human-Computer Interaction, Prentice Hall 1993.

[17] Hayes-Roth B (1993) Opportunistic Control of Action in Intelligent Agents, Knowledge Systems Laboratory, Stanford University, IEEE Transactions on Systems, Man and Cybernetics v. 23, 1993 pp 1575-1587.

[18] Buckingham Shum, S Jorgensen, A. Hammond, N. and Aboulafia, A. (eds.) (1993) Amodeus HCI Modelling and Design Approaches: Executive Summaries and Worked Examples.

[19] P. R. Cohen, A. J. Cheyer, M. Wang, and S. C. Baeg, (1994) "An open agent architecture," in AAAI Spring Symposium, pp. 1--8, March 1994

[20] Foner L., What's an agent anyway?, (1994) online paper: http://foner.www.media.mit.edu/people/foner/Julia/Julia.html

[21] Lashkari Y., Metral M., Maes P., (1994) Collaborative Interface Agents, In proceedings of the National Conference on Artificial Intelligence, 1994

[22] Maes P. (1994) Agents that Reduce Work and Information Overload, Communications of the ACM July 1994, Vol. 37, No. 7.

[23] Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., and Riedl, J.GroupLens: An open architecture for collaborative filtering of netnews. Proceedings of the 1994 Computer Supported Collaborative Work Conference.(1994)

[24] Object Management Group (OMG), (1995) The Common Object Request Broker: Architecture and Specification, Revision 2.0, July 1995, Internet Page: http://www.zgdv.de/www/zgdv-uig/papers/se/standards/CORBA/spec/corba.ps.gz

[25] Petrie, J. C. (1995) Agent-Based Engineering, the Web and Intelligence, IEEE Expert - Intelligent Systems and their Applications, Vol. 11, Issue 6, December 1996, pp. 24-29.

[26] Tate A., (1995) O-Plan Knowledge Source Framework, Artificial Intelligence Applications Institute, University of Edinburgh, Technical Report, March 1995, Internet Page: http://www.aiai.ed.ac.uk/~oplan/oplan/oplan-doc.html

[27] Ygge F., Astor E., (1995) Interacting Intelligent Software Agents in Distribution Managment, in Proceedings of DA/DSM '95.

[28] Wooldridge M., (1995) This is my world: the logic of an Agent-Oriented DAI testbed, in Intelligent Agents - Theories Architectures and Languages, eds. Wooldridge M., Jennings N. R., Springer - Verlag Lecture Notes in Artificial Intelligence Volume 890, pp. 263-274, 1995.

[29] Franklin S., Graesser A., (1996) Is it an agent or just a program? A taxonomy for autonomous agents, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

[30] Haynes T., Sen S., Arora N., Nadela R., An automated meeting scheduling system that utilises user preferences

[31] Jennings N., Wooldridge M., (1996) Software Agents, IEEE Review, pp 17-20, January 1996,

[32] Maes P., (1996) Intelligent Software: Easing the Burdens that Computers put on People, IEEE Expert - Intelligent Systems and their Applications, Volume 11, Issue 6, December 1996, pp. 62-64.

[33] Moukas A., (1996) Amalthea, Information Discovery and Filtering using a Multiagent Evolving Ecosystem, Proceedings of the Conference on Practical Application of Intelligent Agents & Multi-Agent Technology, London, 1996

[34] Hedberg S. (1996), Agents for sale: first wave of intelligent agents go commercial, IEEE Expert - Intelligent Systems and their Applications, Volume 11, Issue 6, December 1996, pp. 16-21.

[35] Jennings N. R., J. Corera, I. Laresgoiti, E. H. Mamdani, F. Perriolat, P. Skarek and L. Z. Varga, (1996) Using ARCHON to develop real-word DAI applications for electricity transportation management and particle accelerator control, IEEE Expert, December 1996.

[36] Diaper D., (1997) HCI and Requirements Engineering - Integrating HCI and Software Engineering Requirements Analysis, SIGHCI Bulletin Vol. 29 No 1, January 1997.

[37] Communications of the ACM (1997), Special Issue : Recommender Systems, Vol. 40, Number 3, March 97

[38] Wang H., Wang C.(1997), A Taxonomy of Intelligent Agents, in Intelligent Agents in Nuclear Industry, IEEE Computer November 1997, pp. 31
[39] ACTS (AC030) KIMSAC project overview: http://www.de.infowin.org/ACTS/RUS/PROJECTS/ac030.htm
[40] ACTS (AC221) GAIA Generic Architecture for Information Availability: http://syspace.co.uk/GAIA
[41] Firefly: http://www.firefly.com/
[42] Sandia Intelligent Agents for Manufacturing (SIAM), http://nittany.ca.sandia.gov:8001/
[43] Mobile AGeNt Architecture, http://www.fokus.gmd.de/ice/projects/magna/description.html