2nd Hellenic Conf. on AI, SETN-2002, 11-12 April 2002, Thessaloniki, Greece,
Proceedings, Companion Volume, pp. 413-424

Change Visualization in Multi-Agent Systems

A.Papantoniou[1], R. Curwen[2], E. Hattab[1]

[1] National Technical University of Athens, Electrical & Computer Eng. Dpt.,
Heroon Polytechniou 9, 15773 Zographou, Athens – Greece,
apapadon@ central.ntua.gr, ezz@ softlab.ece.ntua.gr
[2] University of Loughborough, Computer Studies Dpt., Leichestershire, UK, LE11 3TU
raj.curwen@gb.abb.com

Abstract. One of the most important characteristics of Knowledge Intensive Work is its ability to handle ad-hoc or evolutionary changes successfully. These changes involve Knowledge and Change Management initiatives within enterprises. A direct consequence of these initiatives is the development of environments supporting knowledge flow and workflow, such as multi-agent systems. Unfortunately, multi-agent systems, that implement both the above-mentioned environments, concentrate more on modeling system's interaction, than on visualizing. This is awaited because generic agent models do not address the visualization issue, as part of the agent architecture, and rely instead on additional user and monitoring agents. Therefore, this generic agent model has been enriched with a visualization module and this external view is presented and commented in the current paper. Furthermore, the adoption of XML/XSL interfaces is proposed to 'bubble' through the hierarchy of implemented agents. Finally an implementation of such a multi-agent system is presented showing how this visualization, Interface Bubbling and interface change through user actions was achieved. This implementation has been partially funded by the European Commission under the ESRPIT project Knowledge Desktop Environment (KDE).

1 Introduction

In the rapidly changing world, where personalization of services for an individual customer forms a key differentiator in today's competitive markets, enterprise employees must be provided with knowledge and information at the right time and in the most suitable format [1], [5], [13]. Additionally, these employees must be provided with the ability to handle changes originating from ad-hoc modifications of the process from a single customer to a complete restructuring of the workflow process, due to re-engineering efforts for efficiency improvement. The first issue is addressed by various Knowledge Management initiatives [1], [5], [13], and the second issue is closely related to Workflow Management, involving implementation workflow and change management fundamentals [9], [11], [20], [25], [30]. From the perspective of Knowledge Management, systems supporting the knowledge worker must provide the ability to locate any available information, a way to retrieve and filter that information and interfaces that help the user

understand the domain of the enterprise and the work processes [1], [5], [13]. From the change management perspective these systems must be able to provide alternative execution paths and functionality and tools to change the workflow type, integrating these changes during runtime [9], [11], [20], [25], [30].
Systems for Knowledge Intensive Work must have embedded within them domain models, or representations, related to the everyday aspect of the knowledge workers environment. These work models must be represented through the use of computer interfaces [18], characterized as intelligent because they address context sensitivity, i.e. they alter their appearance depending upon the information exchange. This is closely related to visualization, which can be seen as the act of transforming representations of this information for the purpose of communicating a more adequate appearance of it to the user. The complexity of the systems mentioned above can be manipulated by the development of various types of agents [4], [6], [7], [8], [10], [14], [15], [21], [22], [29], [31]. Furthermore, the interconnection of these agents leads to the adoption of Multi-Agent Systems (MAS), thus enabling the ensemble to function beyond the capabilities of any singular agent [21].
While there is extensive research in the area of information exchange in MAS [6], [8], [10], [12], information visualization is addressed informally, relying on the use of interface agents [6], [21], [27], intelligent user interfaces in general [16], [18] or Personal Service Assistants [2], [3].
Multi-agent systems are a community of agents which, through re-active behavior, change in state as they adapt to events happening in the environment (real world or from other agents). Furthermore, according to [28], MAS systems must be flexible i.e. new agents can be added, others removed. This leads to visualization of these changes.
Therefore, considering the above issues and addressing the limitations of previous research in the area of agent visualization of the change effects forms the contribution of this paper, described in its following sections, namely: Section 2, where kinds of agent visualization are described and formalized Section 3 presenting the agent-based interaction mechanisms supporting the agent visualization. Section 4 providing an implemented environment whose architecture is based on the agent visualization and relevant mechanisms described in sections 2 and 3, namely the Knowledge Desktop Environment, an ESPRIT project partially funded by the European Commission. Finally, Section 5 provides discussion and related work emerging from the comparison of the presented architecture with existing ones in past and recent research areas, thus revealing the similarities and the innovative aspects of the approach presented in the paper

2 Visualizing Agent to Human Interaction

An agent's social behavior, the ability of "agents to interact with other agents and humans via some kind of agent communication language" [33], gives rise to the need for visualization. Two kinds of interaction are defined in the following section: - agent to agent and agent to human.
Agent to Agent interaction requires Interfaces rather than explicit visualization. The agent needs to represent its goals. Firstly, these reflect the agent's autonomous and pro-active behavior [33] and secondly it's internal planning [26]. These required functions of the

agent entity are translated to roles during the analysis stage, which are further translated to services during the design stage, as proposed in [34]. Agent to Agent interaction is characterized by the Service interface to the agent. A second interface for agent-to agent interaction comes out of the work of [19], in which three requirements are proposed for the successful application of a MAS, namely, a dynamic environment, flexible interaction and natural distribution. While the first two are addressed by the weak notion of agent [32], as being re-activity and social behavior respectively, the third one implies that the distribution must be maintained when mapping a distributed domain to a model. As MAS are ideal environments for modeling such distributed domains an adequate interface is required for their interaction, maintaining the agent's visual interface, through augmentation, in cases of its mobility.

In the second interaction type, agent to human, context has always been recognized as a critical factor to its effectiveness [16], [17]. In an agent to human interaction, the required context visualization can be seen from two different perspectives, the agent's context perspective, described through its goals [17] and the agent level information perspective [6]. Therefore, the external agent state, i.e. what the agent presents to its environment can be summarized to the following interfaces, as shown in figures 1 and 2.

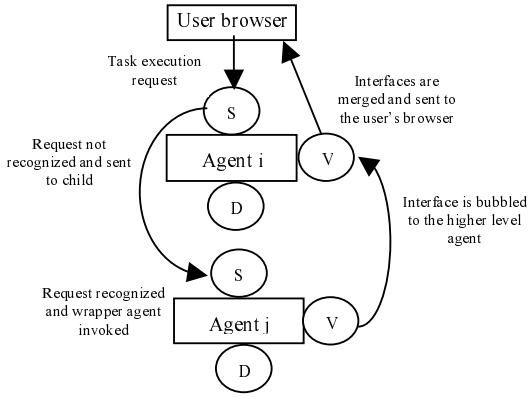

Fig. 1. Agent Visualization mechanism

a. Services interface, a visualization of the agent's goals, in order to be registered in the acquaintance model [14], [22], [23], [34] and further to the yellow pages model [21] of every other agent in the MAS environment.
b. Visualization interface, through which an agent shows an interface to the user, if requested. One can identify three request types :
    i. Pick Interface, a simple representation of the agent to show it actually exists, and the kinds of things the agents can do. This is a visual version of the agent's services. It might be called to ask the agent to present itself in a hierarchy of agents, so the user can choose the appropriate service.

ii. Input Interface, the way an agent returns interfaces when requested to gather information. This is the result of calling a service.
iii. Results Interface, through which the agent can be asked to display the results of any action. This is the consequence of calling a service.

c. The distribution interface which provides the ability to use peer-to-peer communications once two agents have been put in touch with each other.

3 Agent-based Interaction Mechanisms

Agents in MAS often belong to a specific hierarchy, depending on the roles and further categorization [6], [17], [34]. Therefore agents in upper hierarchical levels are considered to be the "parents" of their "children" i.e. the agents that reside in lower levels. The two perspective interaction between these agents and the human users, as described in section 2, is supported by corresponding mechanisms, namely the service registration mechanism and the visual interface streaming mechanism or interface "bubbling" as it is called throughout the paper.

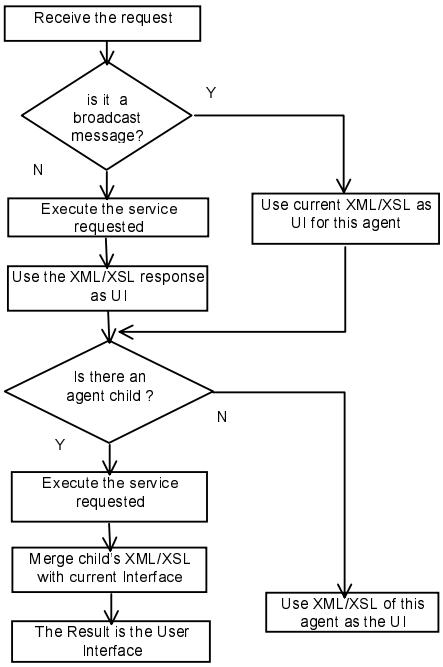

Fig. 2. Agent Visualization and interaction mechanisms

Description of KDE Agent to Agent Interaction
Agents must register their services before they can be used. A service may call upon other services within the agent, or make calls to services registered by other agents. Agents remember who provided services in the past in their Acquaintance Model. High level agents, that represent workflow, call sub ordinate, or child level agents, to interact with legacy systems, for example. If the agent does not have an entry in its acquaintance model for a service it requires, or if that service is no longer available, the agent will ask the Yellow Pages server for a list of services that provide the functionality required. If more than one service, meeting the requirements, is returned then some form of service negotiation is required. This topic is beyond the scope of this paper. Once an agent has called a service it adds that service to its acquaintance model. These types of interaction are defined purely as Agent to Agent.
Visual interface streaming mechanism
Agents can be asked to visualize themselves to allow users to Pick a Service, to ask for Input, and to display Results. These Agent-to-Human forms of interaction are described in section 2.
The whole process starts with a gateway page that asks the top level agent (usually an agent that represents the user), for a pick interface and in its turn, this agent will ask for a similar interface from all the federating child agents it has been connected to. These sub-agents represent the Tasks a user has been authorized to perform, by a Knowledge Manager. The separate interfaces, coming from all sub-ordinate agent levels, are aggregated and returned to the top level agent, i.e. they bubble up through the layers to the browser, as presented in figures 1 and 2. In this way each agent can have an interface that is context-dependent with the interfaces of other agents.
This mechanism is implement through the use of XML [35] and XSL [36] files, due to their extensive information aggregation capabilities as shown in the next section of the paper.

4 An Implementation : Knowledge Desktop Environment

A project that implemented the above visualization techniques is The Knowledge Desktop Environment (KDE), a European Commission partially funded ESPRIT project. This project, achieved its aim, this being the development of a Knowledge Management (KM) Method, specific computer based KM tools and a Knowledge Worker Desktop (KWD) environment, which were validated on two end user applications in different industry sectors, namely Bureau Veritas and ABB Eutech. KDE adopted in its architecture the agent model visualization and relevant interaction mechanisms, described in sections 2 and 3 of this paper. In the following the basic issues of KDE implementation are presented. In order to deliver the desired functionality, the KDE architecture consisted of a MAS, involving relevant agencies, agent levels and interactions, these being described in the following.
The User Agent handles functions and interfaces that are specific to each user, thus providing the possibility of accessing all the system functionality. The Process Agent encapsulates the concept of process, this being the highest level of abstraction of the Knowledge Intensive Work for a specific user in an enterprise. A Knowledge Worker can

participate in many instances of each process type. Each process is realized by one or more "procedures". The process agent therefore must display his child procedures and activate them as required by the user.
The Procedure Agent represents a "procedure", this being a container that includes various tasks. Each procedure has a start date, an end date, a list of participants and a type, described in the workflow ontology[1] developed within the project [37], [38]. This agent must handle the procedure information for each user and give him the possibility to work on its child tasks. The Task Agents represent an individual activity (i.e. a task), like writing a report or filling a database form. Tasks are enterprise dependent; they are the particular instantiation of KDE on a specific case. Therefore a Task Agent typically has to interface itself with some legacy system, like for example a database, this being achieved through Interface Building, with agents acting as "wrappers". Task agents are then grouped into procedures. The Workflow Agent handles all information concerning the hierarchy in KDE, like the available processes for each user, which procedures realize a process, through which tasks a procedure is carried out and all the instantiations of them. Finally the Search Agent provides search, indexing and querying capabilities to the system.
Using this architecture two different problems had to be addressed. In the first place how to propagate down the agent chain the commands originating from the knowledge worker's browser, in order to locate the agent capable of processing this command. Secondly, the way to exchange specific user-interfaces that can be combined to provide the user-interface presented in the browser. These two problems where solved by using a generic agent class, embedding the agent-to-agent and agent-to-human fundamental functionality described previously in sections 2 and 3. The generic agent class had as prime role to facilitate the development of agents within the KDE MAS as specified in [37]. According to the specification, every agent had five models forming its internal state. These models were embedded in the generic agent class, which was implemented through the Enterprise Java Beans (EJB) specification [39] as follows:

1. the Self Model,
2. the Acquaintance Model,
3. the Yellow Pages Server Model,
4. the Communication Model supporting the required communicative acts in agent-based interactions
5. the Knowledge Model, this being the actual implementation of the services making up the business rule the agent encapsulates according to the enterprise needs.

Using Enterprise Java Beans a generic agent class was developed that provided two forms of agent communication: - interface streaming and one-to-one. The first method was

[1] The activity (processes, procedures and tasks) models are expressed using the terms and relations defined in the workflow ontology. This allows for communicating knowledge about the workflow to other parts of the system, which in its turn creates the possibility to link workflow-knowledge to information and enterprise-knowledge. Changes in activities occurring in workflow execution are automatically updated in this ontology, its initial development being hand cored.

implemented according to the specification described in section 2 of the paper, while the second involved the development of a fixed syntax common to all agents, as for example "get interface", an issue which is not covered thoroughly in the paper. XML and XSL files were exchanged between agents in both communicative acts. The way to exchange such messages was described in the generic class's built-in components, namely the "processCommand" method, used to send a command (XML format) to another agent and the "agentXML" and "agentXSL" parameters, representing the XML and XSL data respectively, that the agent must show at his parents.

To aid understanding of the system we consider a typical example of a user performing his everyday activities through his Knowledge Desktop Environment (KDE), involving processes that can be carried out through tasks that represent legacy systems or databases.

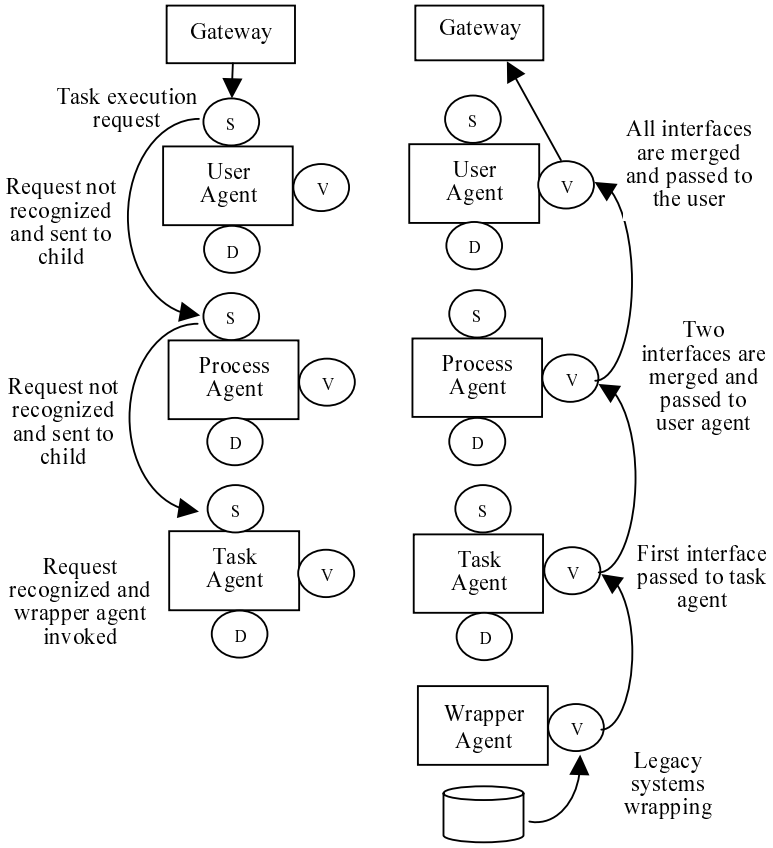

Fig. 3. Interaction mechanisms in KDE project

As shown in figure 3, a normal HTTP request arrives at the system's gateway page, generated from a user request to perform a specific task, like for example "write to database…" The request is processed and transformed in an XML stream, this being sent to the top agent, in our example the user agent.
The agent scans the XML stream to check the command tag. If the agent doesn't recognize the command, it passes the same stream to its child. This process continues until an agent that recognizes the command can be found. In this case the agent executes the command and returns to the caller an
XML / XSL interface. In the previous example, the activity agent will receive a "create task" command; then the activity agent will instantiate a task agent, will receive the task agent initial interface and return the XML/XSL as shown before.

5 Visualizing Change in KDE

The KDE project addressed the popular notion of flexibility, required in the context of modern workflow systems [9], [11], [20], [24], [25], [30]. In their work [11] two aspects of flexibility can be distinguished that need to be supported by a WfS, namely flexibility by selection and flexibility by adaptation. The first situation provides the user a certain degree of freedom to execute a workflow by offering multiple alternative execution paths while flexibility by adaptation provides the user functionality and tools to change the workflow type and integrate these changes during runtime.
Furthermore in the work of [9], [30], a classification of change is provided, revealing the causes (ad-hoc or evolutionary) and the actual implementations of it, namely workflow extension, replacement and re-order. Due to its agent role decomposition, generic agent class development and interface bubbling mechanism, as described in the previous sections, the Knowledge Desktop Environment can perfectly handle the required workflow flexibility and visualize respective change.
In order to understand the way KDE systems handles workflow change, visualizing its effects to the knowledge worker, an example is provided, describing the way a surveyor of Bureau Veritas (KDE consortium end user partner) handles the Marine Inspection process.
This process consists of three procedures, namely "Prepare the survey of a ship in service", "Realize the survey of a ship in service" and "Close the survey of a ship in service". These procedures are further decomposed into the following tasks, thus forming the complete workflow of the surveyor work when realizing the Marine Inspection process:

a. Prepare the survey of a ship in service
    i. Capture the request for survey
    ii. Send the Request For Job Number to the head office
    iii. Generate and print the survey check list
    iv. Consult the relevant working instructions
b. Realize the survey of a ship in service
    i. Search data about the potential damage
    ii. Sign the Request for Survey
    iii. Realize the check list itens

iv. Control and update the documents on board the ship
v. Fill the paper report and update the ship certificates

c. Close the survey of a ship in service
i. Report the survey
ii. Establish the survey report

The tasks that the KDE system provides in order to help the surveyor handle the procedures mentioned above are:

a. Consult and print the existing damage and repair reports
b. Retrieve useful and adequate information
c. Identify a specialist in case of exceptional consulting
d. Send a Request for Repair advice
e. Propose best practices and similar cases
f. Update the Damages Databases with the new damage and repair reports

KDE system provides process and procedure templates, through which a "knowledge architect" i.e. a user responsible in defining processes, procedures and tasks, can easily either provide the system with new workflow components or modify the existing ones.
The knowledge worker, on the other hand, is automatically informed of these new workflow components due to the interface bubbling mechanism. Additionally he is provided the ability to change the execution path of a process, performing procedures and relevant tasks in an order guided from the notion of his work and not from the system itself.

6 Related Work

The term agent model is found in numerous research attempts addressing methodologies for agent-oriented analysis and design [4], [12], [33], [34]. Furthermore, agent models are proposed to be generic [4], structuring in a way the design process, thus addressing limitations of common agent appearances, multiple task performances and variation of abilities as expressed in [21]. These methodologies are concentrated with the way agent societies cooperate to realize the system level goals and what is required of each individual agent to do this. How agents realize their services and how they visualize their interaction results is beyond the scope of these methodologies, depending on the particularities of the application domain. [34].
Therefore, while these agent models deal successfully with the reactivity behavior, as expressed through the weak agent notion [32], they do not present its effects, in a standard way, in the agent physical environment and especially to the users of the multi-agent system. This is mainly due to fact that agent models lack a standard way of visualizing themselves.
Agent visualization can be found in recent research activities in the form of intelligent user interfaces [16], [18], personal service assistants [2], [3], mediators [10], monitoring agents [28], interface agents [6], [21] and multi-agent visualization seen from the perspective of the internal workings of the system [27]. In the work of [6], agent visualization is handled by providing a visualization agent that provides an application

view, as well as individual state views of the agents themselves. We have taken the approach of incorporating this system view into the agents directly, as described in section 2. The approach presented in this paper can be considered a generic one, as it does not require the specification and development of special kinds of agents for visualization, according to previously mentioned research attempts, relying instead on simple interface building agents, acting as wrappers to legacy systems and to the development of domain dependent task agents.

7 Conclusions

In this paper we have described the use of Agents in the European Research Project KDE. These agents were implemented using a lightweight agent abstraction layer, coded using Enterprise Java Beans. We have show how the social aspects of agents can be broken down into Agent-to-Agent and Agent-to-Human interaction. The latter was further decomposed into Visualization of Pick, Input and Results interfaces. We make the case for visualization as part of the agent, and show how such a feature can be exploited, through Interface Bubbling, to provide interfaces to the end user. Finally, we have shown how the current research fits with the ideas presented in the KDE project, and how KDE can address issues of Visualizing Change in the interface.

8 Acknowledgements

We would like to acknowledge the help of the consortium partners of the KDE project these being, ABB Eutech, Bureau Veritas, Intrasoft International, TXT Ingeneria, University of Amsterdam and Salustro Reydel Management.

References

1. Abecker A., Bernardi A., and Sintek M., Proactive Knowledge Delivery for Enterprise Knowledge Management, in Learning Software Organizations – Methodology and Applications, Springer-Verlag, LNCS (1999)
2. Arafa Y. and Mamdani A., Virtual Personal Service Assistants : Towards Real-time Characters with Artificial Hearts, In IUI New Orleans LA USA, ACM Press (2000)
3. Bickmore T. W., Cook L. K., Churchill E. F. and Sullivan J. W., Animated Autonomous Personal Representatives, International Conference on Autonomous Agents, ACM Press (1998)
4. Brazier F. M. T., Jonker C. M. and Treur J., Compositional Design and Reuse of a Generic Agent Model, Journal of Applied Artificial Intelligence Vol 14 (2000)
5. Breuker J., Jansweijer W. , Van de Stadt E. and Van Lieshout J., Manageable and meaningful Information Broking, EKAW 2000 (2000)
6. Cui Z., Odgers B. and Schroeder M., An In-Service Monitoring and Analysis System, Proceedings of the 11th International Conference on Tools with Artificial Intelligence – ICTAI 99, IEEE Press (1999)
7. Debenham J., Supporting Strategic Process, 5th International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology, PAAM 2000 (2000)

8. Eiter T. and Mascardi V., Comparing Environments for Developing Software Agents, Infsys Research Report 1843-01-02, Technical University of Vienna (2001)
9. Han Y. and Bussler C., A Taxonomy of Adaptive Workflow Management, Conference on Computer supported Co-operative Work, ACM Press (1998)
10. Haustein S. and Lüdecke S., Towards Information Agent Interoperability, in Proceedings of the 4th International Workshop CIA 2000 (2000)
11. Heinl P., Horn S., Jablonski S., Neeb J., Stein K. and Teschke M., A Comprehensive Approach to Flexibility in Workflow Management Systems, in International Conference on Work Activities, Co-ordination and Collaboration, ACM Press (1999)
12. Inglesias C. A., Garijo M. and Gonzalez J. C., A Survey of Agent-Oriented Methodologies, Intelligent Agents V – Proceedings of the fifth International Workshop on Agent Theories, Architectues and Languages (ATAL-98), Springer-Verlag, Heidelberg (1999)
13. Jansweijer W., Van de Stadt E., Van Lieshout J. and Breuker J., Knowledgeable Information Brokering, EKAW 1999 (1999)
14. Jennings N. R., Norman T. J., Faratin P., O' Brien P. and Odgers B., Autonomous Agents for Business Process Management, in Journal of Applied Artificial Intelligence 14 (2) 145-189 (2000)
15. Joeris G., Klauck C. and Herzog O., Dynamical and Distributed Process Management based on Agent Technology, in Proceedings of the Scandinavian Conference on Artificial Intelligence (1997)
16. Maybury M., Intelligent User Interfaces : An Introduction, in IUI Redondo Beach USA, ACM Press (1999)
17. Miles S., Joy M. and Luck M., Designing Agent-Oriented Systems by Analyzing Agent Interactions, in Proceedings of the First International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick, Ireland (2000)
18. Miller C.A., Intelligent User Interfaces for Correspondence Domains : Moving IUIs 'Off the Desktop', in International Conference on Intelligent User Interfaces, ACM Press (2000)
19. Mueller H.J., Towards Agent Systems Engineering, in Data & Knowledge Engineering, 23(3):217-245 (1997)
20. Myers K. L. and Berry P. M., Workflow Management Systems: An Al Perspective, Technical Report, Artificial Intelligence Center, SRI International (1998)
21. Nwana H. S. and Ndumu D.T., A Perspective on Software Agents Research, The Knowledge Engineering Review (1999) 14(2):1-18
22. O' Brien P. D. and Wiegand W. E., Agent based process management: applying intelligent agents to workflow, in The Knowledge Engineering Review (1998) 13(2) 161-174
23. O' Brien P., Wiegand M., Odgers B., Voudouris C. and Judge D., Using Software Agents for Business Process Management, in British Telecommunications Engineering (1997)
24. Odgers B. R., Thompson S.G., Shepherdson J. W., Cui Z., Judge D. and O' Brien P. D., Technologies for Intelligent Workflows: Experiences and Lessons, in AAAI'99 Workshop on AI Technologies in Business Processes (1999)
25. Reichert M. and Dadam P., $ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control, in JIIS Special Issue on Workflow and Process Management (1998)
26. Russel S.J., and Norvig P., Artificial Intelligence: A Modern Approach, Prentice Hall (1994)
27. Schroeder M. and Noy P., Multi-agent visualization based on multivariate data, in International Conference on Autonomous Agents, ACM Press (2001)

28. Timm I. J., Multiagent Architecture for D-Sifter, a modern approach to flexible information filtering in dynamic environments, TZI-Bericht Nr. 21 (2000)
29. Tveit A., A Survey of Agent-Oriented Software Engineering, NTNU Computer Science Graduate Student Conference (2001)
30. W. M. P. van der Aalst, How to Handle Dynamic change and capture management information? An approach based on generic workflow models, in CoopIS 1999 (1999) 115-126
31. Wood M. F. and DeLoach S. A., An Overview of the Multi agent Systems Engineering Methodology, in ICSE 2000 Workshop (2000)
32. Wooldrigde M. and Jennings N.R., Intelligent Agents : Theory and Practice, The Knowledge Engineering Review (1995) 10(2):115-152
33. Wooldridge M. and Ciancarini P., Agent-Oriented Software Engineering : The State of the Art, in the Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing Co. (2001)
34. Wooldridge M., Jennings N. R. and Kinny D., A Methodology for Agent-Oriented Analysis and Design, in Proceedings of the 3rd Annual Conference on Autonomous Agents, ACM Press (1999)
35. w3c XML Working group, available in http://www.w3.org/TR/2000/REC-xml-2000100636 w3c, w3c recommendation (2000)
36. w3c XSL Working group, available in http://www.w3.org/TR/2001/REC-xsl-20011015, w3c recommendation (2000)
37. INTRASOFT *et al*. Specification of the Distributed Architecture, Technical report, The KDE consortium, D2.2 (2001)
38. University of Amsterdam. Development o KM Application Models, Technical report, The KDE consortium, D3.4 (2001)
39. Sun Microsystems Inc., Enterprise Java Beans Specification, available in http: // java. sun.com/ products/ejb/docs.html (1999)