5

Merging a knowledge systematisation framework with a mobile agent architecture

J.B.M. Goossenaerts[1], A.T.M. Aerts[2], D.K. Hammer[2]

[1] Eindhoven University of Technology, Information and Technology Division, P.O.Box 513, Paviljoen, 5600 MB Eindhoven, The Netherlands
[2] Eindhoven University of Technology, Department of Mathematics and Computing Science, P.O.Box 513, 5600 MB Eindhoven, The Netherlands

Abstract

The MiViPoRo framework offers a general systematisation of knowledge in domains where work involves the interaction of processes in both the physical domain and the information infrastructure. For the domain of engineering and manufacturing the framework assumes a further division of the cybernetic domain and the physical domain into activity layers for observations and operations (manufacturing), and improvements and innovations (engineering).

Mobile software agents, on the other hand, form a new paradigm for the implementation of flexible communication, computation and coordination services in loosely coupled distributed systems.

This paper will look into the architectural characteristics of both approaches and it will compare and contrast them. Both approaches are fit for complementary purposes, making their merging an attractive option.

Keywords

Enterprise models, product life cycle models, proxy possible flow semantics, mobile agent architectures.

J. J. Mills et al. (eds.), *Information Infrastructure Systems for Manufacturing II*

1 INTRODUCTION

Recently, many developments in the area of business processes have taken place, which were made possible by advances in the field of communication and information technologies. Business processes nowadays rarely are confined to a single production site, but often involve several companies, thus creating extended or virtual enterprises. These developments have led to the emergence of (parts of) business processes which are completely virtual, i.e. they don't have a physical counterpart, such as standard manufacturing processes.

Adaptation (or evolution) of frameworks can be done in a number of ways. One way is to extend the framework to encompass new areas of interest. Another way is to merge complementary frameworks to create a more general one. In this paper we take the second approach and explore the possible merger of two frameworks: the MiViPoRo framework and the Mobile Agent Framework (Dalmeijer, 1998, Hammer, 1998). The MiViPoRo framework (*Modules for innovations, Versions for improvements, Proxies for operations, Records for observations*) has been introduced as a mean to connect work (engineering, manufacturing) to an information infrastructure in (Goossenaerts, 1997a). MiViPoRo conformant system specifications have a so-called proxy possible flow semantics (PPFS). Proxies are unique representations of physical objects or entities which flow through an information infrastructure. These proxies are resident on a network of model execution engines, in "reflection" of the represented physical objects flowing through the physical domain (see Figure 1) (Goossenaerts 1997b). The Mobile Agent Framework (Dalmeijer, 1998, Hammer, 1998) has been introduced to provide an infrastructure for flexible communication and coordination services in loosely coupled distributed systems.

Whereas in MiViPoRo the mobility of proxies in the information infrastructure strictly mirrors the mobility of the physical entities in the physical domain, where the manufacturing processes are being carried out, mobile software agents do not have such a constraint. In fact, the freedom of mobile agents from the physical constraints of place (in the information infrastructure) is akin to two other values that the cybernetic domain adds to the physical domain. These values are the possibility to store data on past events (performance) in support of reporting (memory), and the possibility to work with expected future system states, events or their abstractions in support of planning and scheduling (simulation). Software agents move in the cybernetic domain, free from the constraints in space and time that physical entities have by necessity, and the constraints that proxies have by choice (in the MiViPoRo architecture).

However, the mobile agent framework of (Dalmeijer, 1998, Hammer, 1998) so far has been focussed on providing an infrastructure for agents to move in. It does not provide a systematic framework for capturing knowledge about agents and the objects they interact with. Such a framework is provided by MiViPoRo.

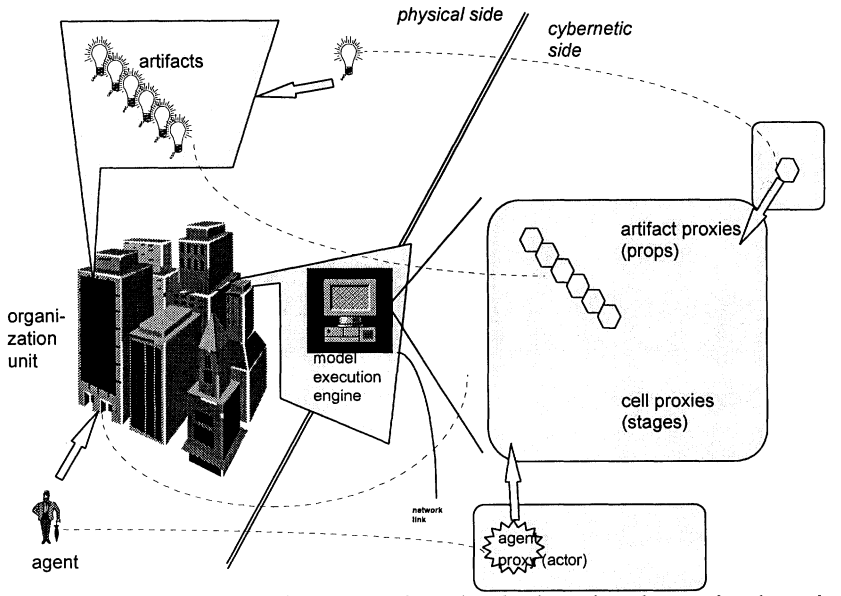

Figure 1 The relation between the physical and cybernetic domains in MiViPoRo.

Both frameworks appear to be complementary, making their merging an attractive option. In this paper we will explore how the two frameworks can be merged and why it is advantageous to do so.

Overview

The MiViPoRo framework and the mobile agent architecture will be presented in sections 2 and 3 respectively.

In section 4 we will examine the two alternative architectures by looking into the following issues:

1. Interaction model for mobile software agents and proxies;
2. Migration protocols for software agents and proxies;
3. Directory and trading services;
4. Inference services.

To illustrate the various concepts involved we will use a small case from a manufacturing context.

The PC order process case

To illustrate the application of the MiViPoRo and mobile software agent models we present a case called the 'PC order process' (van Stiphout, 1998a, 1998b), featuring the assembly to order of PCs. The 'PC order process' consists of the following process steps (Figure 2):

1. An order for a PC is entered into the system. The order specifies what CPU, memory and hard disk the customer wants.
2. After the order has been received, the PC is assembled.
3. Parallel to the assembly process, the customer is sent a bill.
4. When the payment of the bill is received, this is registered.
5. After the PC has been assembled and payment has been received, the PC is shipped.

If we look at the 'Assemble PC' activity we see that it in turn consists of a number of smaller activities:

1. Insert CPU.
2. Insert memory.
3. Insert hard disk.

The process consists of two streams: the production stream dealing with the actual assembly and shipping of the PC, and the administrative stream dealing with the financial aspects of the ordering process.

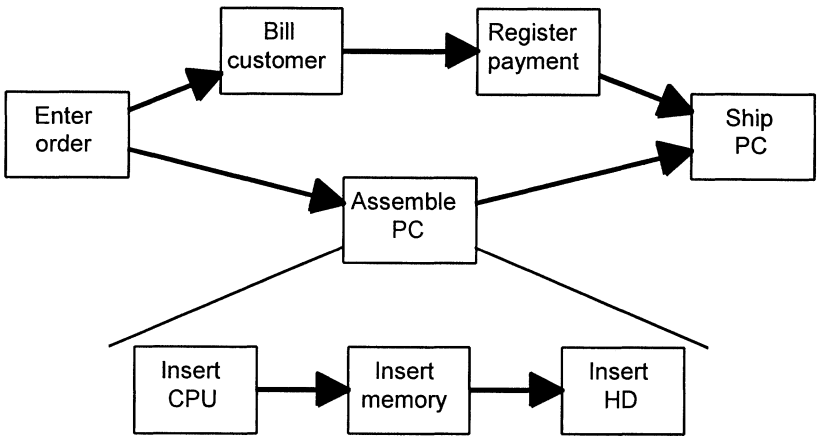

Figure 2 PC order process.

2 THE MiViPoRo FRAMEWORK

The MiViPoRo framework (*Modules for innovations, Versions for improvements, Proxies for operations, Records for observations*) (Goossenaerts, 1997a) divides the problem domain of enterprise integration into two sub-domains and four activity layers (Figure 3). The sub-domains are: the *physical domain* comprising the physical space, time and matter, with artifacts, goods, agents, and cells (spatial units) having their lives in it; and the *cybernetic domain* which adds memory,

communication, monitoring (including knowledge-capture) and control (computations) services to the physical domain. The activity layers cover observations, operations (manufacturing), improvements and innovations (engineering).

Physical and cybernetic domain and their interflow

On the physical side, the term 'entity' refers to an artifact, agent (person, machine) or place that has a definite physical existence in itself. At any point in time, an entity is present at exactly one place. It is characterized by an existence in matter and a facility to move in space (with the possible exception of immobile places such as assembly stations). A (physical) agent has powers that determine the actions it can execute. Entities are grouped in classes on the basis of similarities in their time, space and matter situated possible lives (the PC as a specification, the PC in various states of assembly, the PC in its box on the way to the customer). At a point in time, the place (cell, organization unit) where an entity is located, will accommodate certain possible or required life phases of the entity. This set of possible life phases may change as the entity progresses from one place to another and determines the events in which the entity may involve next. An entity has a life span, a succession of space-time situated events in which other entities may be involved as well (e.g. when the PC is assembled, the tools at the assembly station are involved in its life span).

In the example (see Figures 1 and 2) artifacts are, for example, PC's being assembled, or the (paper) bill sent to the customer. Examples of agents are technicians doing the assembly and sales representatives entering the order, billing the customer, or registering the payment. There are also cells (places) such as the desk where the order is entered, and the assembly stations, where the PC's are assembled from parts.

In the cybernetic domain the agents are represented by actors (agent proxy) in the form of, for example, user profiles with authorization to use various software applications for supporting order entry and billing, or for checking out (and locking) the work order for the PC to be assembled next. An artifact is represented by a prop (artifact proxy) such as the bill of material for the custom made PC and its assembly status. The possible life phases of an artifact are situated at cells or organization units. The requirements phase for the PC (the custom order) is supported by the sales department, the production phase (assembly) is situated in the manufacturing department and the transportation phase is supported by the shipping department (the PC is being shipped). Each step in Figure 2 coincides (in this case) with a life cycle phase.

The modeling primitives on the *cybernetic side* are modules. Modules are created and transformed in the innovation layer. One module typically captures time-space-matter independent aspects (knowledge) on one possible or required phase of the life span of a class of entities. For example a module may specify a generic bill of material for a PC, detailing which type of components are present in

a certain (commercial) model. Modules are combined into a *possible lives model* or *enterprise formula* (EF) (Goossenaerts 1997b).

Artifact possible lives models (APLM) model the possible life cycles of products. An APLM is a set of modules - each of which describes a possible life phase of the artifact. The modules in an APLM describe concurrent, successive, alternative or mutually exclusive phases in the possible lives of entities, as illustrated in the PC order case. One can view an APLM as a product centered workflow specification, with each module comprising the specification of a range of activities in which the product may or must be involved.

An *enterprise formula* (EF) (Goossenaerts 1993) offers a modular description of the enterprise processes. It comprehensively specifies the operational roles and the coordination of persons, machines, parts and products making up an enterprise.

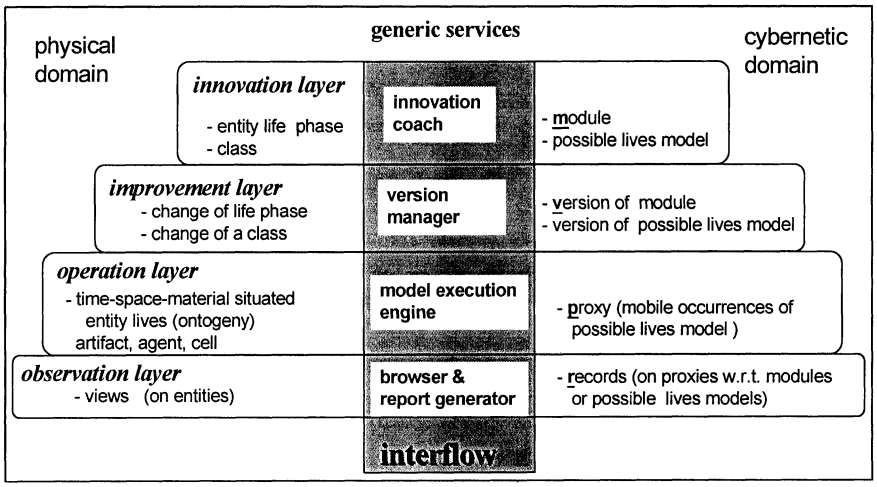

Figure 3 MiViPoRo's activity layers, generic services, and modeling primitives.

The term *interflow* (short for *inter*active *flow*) denotes the co-ordination and monitoring of the physical processes by means of computational processes. Obviously, when a hard disk is inserted in the PC, this has to be communicated to its cybernetic counterpart in order to keep the physical and cybernetic states synchronized. The concept of a computational process covers the transformation and storage of digital representations by computer programs. Models of artifacts and enterprises exist in the cybernetic domain.

The four activity layers

The four activity layers span the two sub-domains and cover observations, operations, improvements and innovations. The discussion of the example case so far has been set at the operation layer. At this layer, the model execution engine connects repetitive work, action in the physical domain, to computations and communications in the cybernetic domain.

At the improvement layer the version manager supports the evolution of versions of modules and the corresponding life phases. For instance, in the PC order process, the transformation of the order entry process into one that also accepts orders sent by email.

At the innovation layer the innovation coach supports the concept of entirely new entity life phases (through the development of new modules and artifact possible lives models). Examples of entirely new life phases in the case of the PC order process are the introduction of laptop assembly, or multi-processor PC assembly. There is an assumption that design in the cybernetic domain precedes realization in the physical domain.

At the observation layer the browser and report generators are used to create views of entities based on the records of their proxies.

After their creation, modules and possible lives models can be employed in the operation layer and observation layer. *Versions* of modules and possible lives models capture modifications as they are performed in the improvement layer. At this level, for example, minor variations with respect to the generic module may be specified, such as various options on hard disks, CPU's and memory. Major variations, such as laptop and desktop models, correspond to innovations and are dealt with at the generic level. At the operation layer, a *proxy* exists as a unique representative of a physical entity. It captures knowledge on the entity and the modules (each belonging to the possible lives model of the class of the entity) for which it has been earmarked. At this layer a particular PC's components are set. For observation purposes, a *record* of a proxy is created in memory of the proxy's past or current flow through a network of cells. This flow reflects the time-space-matter situated life span of the corresponding entity. Records are recorded during operation layer processes.

MiViPoRo thus allows us to systematically capture knowledge about manufacturing systems, products and their representations in an information infrastructure. At present it has an emphasis on synchronizing between the real world system and the information system supporting and controlling it. This leaves unexplored possibilities such as dealing with processes that exist purely in the cybernetic domain.

3 MOBILE AGENT ARCHITECTURE

The MiViPoRo framework is well suited for describing processes with hard physical constraints such as manufacturing processes, where physical objects (raw materials) are transformed into products by the components (milling machines, lathes, measuring devices) of the production system. In such systems, static agents are sometimes used (Zwegers 1997) to provide the flexibility needed to deal with disruptions of the process by unpredictable delays and other unanticipated events. These agents provide the intelligence for solving problems, but are immobile since they encapsulate the fixed components of the production system.

In mobile agent architectures (Dalmeijer 1998a, Hammer 1998), the agents can move through the distributed system. Mobile agents are software components that perform a task for a client, which may be a human, or another software component. An example of such a task is the execution of a workflow, such as the PC order process. The agent receives a specification of the task to be performed, detailing the subtasks and their dependencies. The agent then deals with each eligible subtask in its turn, by locating the nodes in the network, where the services and other resources are provided that are needed for the subtask, selecting an appropriate one and then migrating there to have the work done. In some cases the task specification may involve subtasks that can be carried out in parallel such as the billing and registering tasks that can be done in parallel to the assembly task, and the agent then may generate clones for each subtask. These clones will, after carrying out their subtask, merge again if the task specification requires such, as Figure 2 illustrates at the point where the customer has paid the bill and the PC has been assembled and shipping can be started.

As already suggested in the previous paragraph, mobile agents operate in a distributed environment, consisting of nodes, connected by a communication network. Such environments may have various topologies. For instance, a number of nodes may be connected by a very reliable and fast local area network. Such a collection of nodes is called a site. Sites may participate in a wide area network, in which the communication links between sites are much less reliable and slower than in a local area network. One can imagine that in a large company, the sales department is located in one place, the assembly facilities in another and the financial services in yet another place, each place corresponding to a site in a wide area network. Mobile agents are able to move reliably through such a distributed environment. In particular, migration is failure atomic: it either succeeds or the agent stays at its current place (Dalmeijer, 1998b; Hammer, 1998). Mobile agents therefore will experience little hindrance from unreliable services or communication links. They therefore can operate in environments where some links or even nodes may not be available from time to time. This can occur either by accident or on purpose as in the case of hired lines or mobile computers. A mobile agent is also able to wait for a resource or service to come up again or for

the connection to the site it wants to migrate to, to become available again. In addition, it has the ability to look for alternative sites, where it can carry out its task. For instance, when an agent is assigned the task of fulfilling a certain order, it must find a place where the PC can be assembled. When all technicians are busy, the agent must wait until a technician becomes available and then migrate to the place (assembly station) of this technician.

Agents that can accept and execute tasks on the basis of a specification can communicate at a high semantic or knowledge level and thus are very flexible. Such agents generally use an Agent Communication Language (ACL) such as the Knowledge and Query Manipulation Language KQML that was developed in the context of the ARPA Knowledge Sharing Effort (Genesereth 1994). KQML is a container language for more domain specific languages such as KIF (Finin 1993) or SQL. KIF stands for Knowledge Interchange Format and is based on first order predicate logic. Agents that can operate at a semantic level require considerable intelligence. However, it is not necessary to put all this intelligence inside the agent. This would make the agents very large indeed, whereas the attractiveness of mobile agents stems for a large part from the fact that they can be kept small (by programming them at a high level of abstraction) and therefore do not pose a big burden on the communication infrastructure. Instead, the necessary functionality should be made available by static (immobile) services.

In each node of the network, services are offered to the agents. Every node should offer an interface that agents can use to communicate with the system components present at the node. Such an interface is often called a dock. One of its functions is to encapsulate the local components to hide their heterogeneity and make them accessible to the agents using the ACL. Docks may also support the security of the system, by providing authentication services, and support agent migration by providing reliable, secure transportation services to the agents. The system should also offer generic services, such as directory services, needed to locate agents and components, and trading services needed for an agent to find the nodes where a particular services is offered. The latter services may vary from node to node. One node could offer access to a particular database, another access to a CAD/CAM system or other applications. A third node mightoffer an inference service that an agent could use to make a choice between a number of options, such as which particular service to select given a budget, costs, timing, and other constraints.

Mobile agents provide a generic approach for the integration of heterogeneous distributed systems. They create an infrastructure for doing work in distributed systems and thus for building virtual or extended enterprises. For this to succeed semantic integration of the component systems has to be supported. The agents and the component systems have to share a common context (or ontology) such that the semantics of their interactions is unambiguous. At present, the mobile agent framework of (Dalmeijer 1998a, Hammer 1998) does not provide a systematic way to capture the required knowledge. At this point, MiViPoRo provides the necessary complement.

4 MERGING MiViPoRo AND MOBILE AGENTS

Deployment of mobile agents in a MiViPoRo conformant system can be done in several ways of which we will present two extreme cases. In an actual system both extremes and any variant in between may occur, depending on the scope and the range of the process that needs to be supported and on the available infrastructure. The first example considers a so called lean mobile agent that fully relies on its operating environment. Another example is a fat agent that can operate with only a minimal support from the environment it works in. Both types of agent may act on behalf of or even completely replace a physical agent.

First, consider a *lean mobile agent* that operates in a completely open and homogeneous infrastructure. The software agent can then rely on the required resources (such as knowledge and functionality) needed to execute its task being available locally. It needs to contain only the specific knowledge to carry out its task, such as the proxy of the physical agent it represents, and any modules needed to maintain global knowledge about the task that it is to carry out. This agent would make use of the model execution engines, which play the role of a dock, in the cells it visits to execute the transformations that go with the task it has to do. The application of such agents could result in a more flexible production process, when the agent would, for example, be able to find its own route through the cells. For example, think of an agent that could detect the load on assembly stations and adjust its route accordingly. In our example, it could propose to first have memory inserted and then the CPU into the PC. In such an infrastructure, the mobile software agent is an alternative to having the system components communicate with each other in order to coordinate the creation of a particular product. The agent would provide the (mobile) control and pass the necessary information (such as production parameters) to the system components involved at the right time. Note that a physical agent can start off any number of software agents to perform tasks. These software agents all act on behalf of the physical agent and thus have to carry (a copy of) the proxy of this agent with them to have access rights to information and services at the nodes. Such a parallel execution of tasks can only happen in the cybernetic domain.

Software agents may also be used to perform activities that have no direct counterpart in the physical domain, such as exploring possible future or past activities to come up with a new plan or solution in the case of a disruption of the systems operations. Such a plan, if adopted by the physical agent, may have an effect on the physical world. These applications provide an enrichment of the MiViPoRo framework.

Next consider a *fat mobile software agent*, that is, a software agent that has all required functionality on board. In terms of the MiViPoRo framework, such a mobile software agent can be thought of as a compound entity, comprising modules, certain proxies (e.g., of money or information products), and the model execution engine(s) needed to transform proxies of its own, and proxies in the

environment. Such an agent will not depend too much on the presence of model execution engines at the cells it visits or local knowledge to carry out its task. In fact, it would be possible for such an agent to carry out its task with only the minimal amount of interaction with the modules and proxies in the cells it has to visit. This would make it possible for the agent to flow through an infrastructure consisting of systems offering only limited interfaces for communication and limited support for functionality. An example of such an infrastructure is that of an extended or virtual enterprise, where only a limited extension of the local infrastructure can be afforded. It then is the agent's task to coordinate the collaboration between the autonomous companies involved and to constitute the integrating factor. Communication in such a situation would have to be at a high semantic level to bridge all the syntactic and semantic differences present in such a heterogeneous system.

Interaction model for mobile software agents and proxies

In all cases, communication between the software agent and the proxies of the cell the agent is visiting is necessary. Since the proxies in a cell are under the control of the local model execution engine, that is, the model execution engine present in the cell, communication between a mobile software agent and a local proxy would involve this model execution engine. In case the mobile agent carries its own model execution engine, this would have to communicate with the one in the cell. Model execution engines in MiViPoRo posses this property, since it is also needed when a particular proxy flows from one cell to the next and the model execution engine of the cell the proxy is leaving hands control over to the model execution engine of the cell it is flowing to.

Migration Protocols

In the framework of Dalmeijer (Dalmeijer 1998a, 1998b), each node an agent can visit contains a generic piece of infrastructure (sometimes called a dock) that provides the necessary services for migrating agents, and also for creating or destroying agents on the basis of well defined protocols and criteria (such as budgets being available). These services are also relevant for migration of proxies and should be made a generic part of the local model execution engines present in each cell.

Other Services

Other services that need to be part of an infrastructure supporting mobile agents are directory and trading services that tell an agent where a particular entity (cell, agent) may be found and what services are available respectively. Optional services include inference services that an agent needs when it wants to find the quickest route through a network of cells or the fastest way to rush a particular order through the production network.

5 CONCLUSION AND FURTHER WORK

Knowledge systematisation for engineering and manufacturing, and co-ordination, communication and computation in a heterogeneous network are two different dimensions of an information infrastructure. By examining MiViPoRo and the mobile software agent architecture, we found that the ability of software agents to autonomously perform various tasks on behalf of their creators (ultimately physical agents) can be exploited in at least two ways. Software agents can travel between various machines of a distributed system; and they can explore alternative future courses of activities (simulation, scheduling) for the interflow processes. MiViPoRo offers a systematisation of knowledge on interflow processes (interaction of physical agents and cybernetic domain processes). The mobile software agent architecture adds to this support for pure cybernetic domain processes with no necessary involvement of physical agents, and hence a freedom of the space-time-matter constraints that restrict interflow processes. Moreover, a system architecture including mobile agents makes the cybernetic domain system adaptable and easily extendable.

The discussion in the previous section has touched on a number of possible scenarios for merging the MiViPoRo framework and mobile agent architectures. Since they operate in the cybernetic domain, mobile agents can be used to carry out information production processes, in analogy with physical production process already described in MiViPoRo. Further research on how the MiViPoRo framework and the mobile agent architecture merge, has to focus on applications to non-trivial practical cases: (1) illustration of proxy flows and mobile agents in situations where multiple companies are involved such as in a virtual enterprise; and (2) illustration of improved flexibility in the physical system as enabled by the extendable functionality of the agent architectures.

Other research will address questions such as when it is more effective or efficient to use a mobile agent, and when to use a proxy, in case both options are available.

6 REFERENCES

Dalmeijer, M., Hammer, D.K. and Aerts, A.T.M. (1998a) Mobile software agents, submitted to *Int. Journal on Computers in Industry*.

Dalmeijer, M., Rietjens, E., Soede, M., Hammer, D.K., and Aerts, A.T.M. (1998b) MAF: A Java-Based Implementation of a Mobile Agent Architecture, *1st IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC'98)*, April 1998, Kyoto, Japan.

Finin, T., Weber, J. et al. (1993) Specification of the KQML Agent-Communication Language (Draft), *The DARPA Knowledge Sharing Initiative.*

Genesereth, M.R. and Fikes, R.E. (1994) Knowledge Interchange Format, Technical Report Logic-92-1, Univ. of Stanford.

Goossenaerts, J. (1993) Enterprise Formulae and Information Infrastructures for Manufacturing. In: Yoshikawa, H., Goossenaerts, J. (eds.): *Information Infrastructure Systems for Manufacturing*, IFIP Transactions B-14, Elsevier Science B.V.(North Holland), Amsterdam.

Goossenaerts, J. (1997a) A Framework for Connecting Work and Information Infrastructure. In: Goossenaerts, J., Kimura, F., Wortmann, J.C. (eds): *Information Infrastructure Systems for Manufacturing*, Chapman&Hall.

Goossenaerts, J. (1997b). Proxy Possible Flow Semantics for Enterprise Formulae and Artifact Possible Lives Models. In: Kosanke, K., and Nell, J.G. (ed.) *Enterprise Engineering and Integration: Building International Consensus - Proceedings of ICEIMT'97, International Conference on Enterprise Integration and Modeling Technology*, Research Reports Esprit, Springer Verlag, Berlin.

Hammer, D.K., Aerts, A.T.M., and Dalmeijer, M. (1998) Mobile Agent Architectures: What are the Design Issues? In: *Proceedings of Engineering of Computer Based Systems (ECBS98).*

Stiphout, R. van (1998a) *Using exceptions to support transactions in workflows.* Eindhoven University of Technology Master thesis.

Stiphout, R. van, Meijler, T.D., Aerts, A.T.M., Hammer, D.K., Comte, R. Le (1998b) "TREX, Workflow TRansactions by means of EXceptions", WFMS'98 EDBT Workshop on Workflow Management Systems, Valencia, 1998, J. Eder, H.-J. Schek, L. Salsa, Eds., pp 21-26.

Zwegers, A.J.R., Beukering, L.H.Th.M. van , Schenk Brill, D. van, and Pels, H.J. (1997) A Comparison of three Agent Based Control Systems, *Proceedings of ASI'79,* Budapest.

7 BIOGRAPHY

Dr. Jan B.M. Goossenaerts is assistant professor at the Eindhoven University of Technology. He holds degrees in mathematics, philosophy and computer science from the University of Leuven. In 1991 he received his Ph.D. degree from the same university with a dissertation on the design of an enterprise modelling language. Prior to joining the University of Eindhoven, he was visiting researcher at the University of Tokyo (1992-93) and at the United Nations University, International Institute of Software Technology in Macau (1994). His research interests include the architecture of an information infrastructure for manufacturing, knowledge systematisation for manufacturing, product life cycles and extended enterprises, and the application of virtual manufacturing.

Dr. A.T.M. Aerts received his Ph.D. from the University of Nijmegen in the Netherlands in 1979. He joined the Department of Mathematics and Computing

Science of the Eindhoven University of Technology in 1985 as an assistant professor. His research interests include methods, tools and techniques for the specification, design and analysis of information systems, and for integration of and collaboration between information systems. Recently he has been working on topics in the area of supporting collaboration on the World Wide Web and mobile agent systems. Homepage: http://wwwis.win.tue.nl/~wsinatma

Prof.Dr.Dipl.Ing. Dieter K. Hammer studied Technical Physics at the Technical University of Vienna where he attained his masters degree in 1966, and was a staff member of the physics department from 1966 to 1976. Between 1976 and 1987 he headed several system software development departments at Philips Vienna and Philips Telecommunication and Datasystems in Hilversum. In Hilversum he worked on a distributed operating system and embedded software for telecommunication switching. Since 1987 he is a full professor at the Computing Science Department of the Technical University of Eindhoven, The Netherlands. His research area is the construction of embedded systems for process control applications, including the networking and software engineering aspects. His special interests are object-oriented techniques, distributed (operating) systems, and the systematic design of (hard) real-time and fault tolerant systems, including requirements engineering.