Mediating Negotiations in a Virtual Enterprise via Mobile Agents

Nick Szirbik, Hans Wortmann, Dieter Hammer, Jan Goossenaerts, Ad Aerts
{n.b.szirbik ; j.c.wortman ; j.b.m.goossenaerts}@tm.tue.nl, {hammer ; wsinatma}@win.tue.nl
tel.: +31-40-247-4370, fax: +31-40-243-2612
Eindhoven University of Technology, Postbus 513, 5600 MB, The Netherlands

Abstract

The manufacturing process in a Virtual Enterprise (VE) is a more complex task than in the single integrated enterprise case. We are proposing an architectural systematisation of the manufacturing and control structures in a VE, in the context of business processes and high-level planning. For the Information Technology (IT) integration of the VE, we propose the mobile agent paradigm, defining the concept of a Mobile Agent Web (MA-web). The role of the agents in this environment is to track and monitor the manufacturing processes and to mediate negotiations between the parties of the VE when it is necessary. We are making some assumptions about the new behaviour and code of conduct in the envisaged MA-web, such as the willingness to share data and knowledge, and the inclination to cooperative interaction, both induced by the usage of collaborative agents. We are presenting some of the basic principles of architecturing such systems and show a way of how mobile agents can be used in practice.

1 Introduction

A new and revolutionary tool can change its user [Bradshaw99]. In a nutshell, is what he hope we, along with the rest of the software agent research community can achieve in the future: a change for the better in applying the agent paradigm in real-life systems. Good tools can also benefit those who are developing agent theory. Creating powerful tools allow us to explore specific decisions about the design of such systems or various conversation policies in the context of a given application.

The goal of this paper is to present the main design decisions made for the PROVE system, developed by our research group. For the application, we are investigating the use of the mobile agent paradigm for integration of IT support systems in a Virtual Enterprise (VE), especially the use of the agents in negotiation processes. What distinguishes our research from previous work in the field is the exploitation of the *closeness* concept [Szirbik&al99a]: we are linking the agent to the processes it has to interact. The classical approach is to use the agent as a wrapper of functionality of legacy systems, in order to achieve uniformity (see [Kjenstad99]). Our approach is completely different: the agents are in charge with certain tasks, and they interact with the legacy systems via a specialised infrastructure, the *mobile agent dock* [Goossenaerts&al98].

The paper is structured in the following way: in the next section we explain our perception about what is a VE, and the used terminology, together with a short presentation about the agent paradigm and the mobile agent technologies and how these can be applied to VE integration. Section 3 explores aspects of the architecturing decisions for the PROVE system. Section 4 presents some details about the negotiation processes which are carried with agent support. We discuss the advantages of our approach, point some future developments and conclude the paper in section 5.

2 The Virtual Enterprise Model for our research

There are many conflicting views about what is a Virtual Enterprise. In the next subsections, we present an example of Virtual Enterprising and introduce definitions according to our view. Because the software agent technology is new and known mainly to a specialised circle of researchers, we present the standard concepts and the key challenges in the development of agent-based systems. The whole idea of our research is based on the idea that the concept of the Virtual Enterprise and the concept of Mobile Agent-based Systems (MAS) are complementary, and we consider that the combination of the novel organisational concept of the VE and the new MA technology is worth to study and try to apply in the integrated information infrastructures of the future.

2.1 An example of Virtual Enterprising

Consider the case when a commercial jet needs a high-value replacement part like an undercarriage, because the old one was damaged in a rough landing. International regulations prohibit the airline to repair this part of the plane by changing only the severely damaged parts and tinker the slightly damaged ones. They ask for a full replacement of the landing gear. Because the airline companies usually do not keep reserve undercarriages in their spare parts stocks, the airline encountering such an event has to order a new undercarriage. The problem is that 80% of the global commercial fleet consists of aircraft which are not in current production. Worse still, many of the companies who built the planes are not any more on the market.

However, the market of high-value replacement parts is very active. There are some reasons for this situation. First, the demand is high, because a grounded plane is not making money. Second, for such orders, the needed manufacturing resources and material resources are taken not from a single company, like Boeing, who concentrates to deliver new aircraft and not spare parts. A typical case is when a company like Fokker for example, is executing the final assembly and installation, but the parts are made elsewhere or are purchased from distant stocks, where these are available. For an undercarriage, the number of companies involved is typically around a dozen, and more important than the number is the fact that these companies could be located on different continents. In our view, this is one of the most characteristic examples of Virtual Enterprising. Of course, it has a limiting nature, because is characterised by an one-of-a-kind ordering style, and contains no development and engineering of the manufactured product. But in our research approach, we considered that studying this type of cases is a easier way to start. We show later in the paper that we can extend our proposed architecture to more general cases.

2.2. The terminology

In the example presented above, the group of enterprises which participate in the manufacturing, assembly, purchase, and delivery processes leading to the installation of the new undercarriage, IS NOT in our view (or better to say in our terminology) a Virtual Enterprise.

Our definition of a Virtual Enterprise is related to the concept of *virtual capability* [Wortmann97],[Kornelius99]. By taking a global market perspective, many enterprises, from different continents, *can* participate in such a process. We assume that in this group, enterprises with similar types of resources can be grouped in smaller subsets. This implies that a certain degree of redundancy exists. On the other hand, to execute an order for a single undercarriage for example, only a small number of enterprises from the big group will be actually involved. The selection of the members of this smaller set depends on how their resources can be used, in terms of available time-slots, stocks, prices and a possible pre-selection made by the customer. Our definition for a Virtual Enterprise is: "*A group of enterprises, which has the **capability** of realising in a collaborative manner, items and/or services from a limited portfolio of products, is a Virtual Enterprise.*"
This definition is not related to the temporal dimension. Only if new enterprises join or leave the group. But it is not capturing the status at a certain moment in time. We introduced a new concept for that: "*A group of enterprises, members of a bigger Virtual Enterprise, which are currently involved in the execution of an order from a customer, is a Cluster*." Many Clusters can be active simultaneously in a Virtual Enterprise.

We make the assumption that these enterprises will dedicate only a small fraction of their resources to participate in Clusters (a single enterprise can be at one moment participating in many Clusters). This is happening due to the fact that orders leading to Virtual Enterprising are typically opportunistic and very hard to forecast. For example, is impossible to forecast when a landing gear will be broken and needs replacement.

The resources used in a Cluster must be free at the moment when they are needed in the execution of the order. It was observed in a study [Kornelius99], that there are specific sources of free resources:

- overcapacity policies, very popular now, because they increase flexibility and agility
- delayed or cancelled orders
- order, process, and operation scheduling.

The latter is the most frequent source of free time-slots. Even the most advanced schedulers cannot fill completely the available time of the resources, due to batching, sequencing and especially dependency constraints. In terms of material resources, there is always a chance, that in a huge Virtual Enterprise, we can find somewhere a slack stock, which can be viewed as an equivalent of a free manufacturing resource.

An important observation is that the enterprises participating in Clusters are actually making their main part of the turnover from long term contracts, representing their principal commitments. The Virtual Enterprising is viewed only as a collateral activity, but it is helping them to:

- fill the gaps in the usage of their resources and subsequently increase their overall performance
- respond quickly to new and unexpected market demands, that is, increasing their agility
- participate actively in a globalised market, by collaborating with remote enterprises and by delivering to far situated customers
- extend virtually their resource types, without actually doing so, that helps to concentrate and improve their core competency.

Forms of Virtual Enterprising existed in the past (a striking example is presented in [Preiss&al96], about the XIX century's whaling industry). Today, there are certain enablers, which favorise the expansion of Virtual Enterprising, such as:

- the eradication of tax-barriers and market globalisation
- the scale and ease-of-use of efficient and fast transportation resources
- increased human mobility and ease of human interaction
- ease of communication, with a special emphasis on mobile devices and the Internet.

In [Preiss&al96], a group of enterprises linked to the Internet, is called an E-web (Electronic web). However, by merely interconnecting computers sitting in different enterprises is not at all a guarantee for following successful integration in Virtual Enterprising. To enable collaborative processes, we need a structured approach, to identify the architecture, the requirements, and the problems of IT support for Virtual Enterprising.

In our approach, we model the single enterprise as a layered and hierarchical control structure, vertically integrated from the IT point of view.

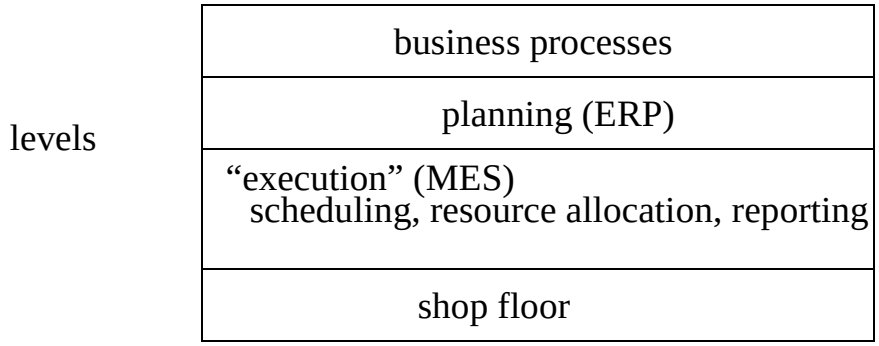

figure 1: The levelled model for interaction in a VE

At the highest level, the business processes are taking place, laying down a strategic course of action for the enterprise. The actors here are humans (executives, managers) and even if they are heavily relaying on computer use, their activity cannot be automated (or replaced) by software components, due to the inherent complexity and human nature of the processes. At a lower level, the so-called "*planning level*", automated ERP systems (like Baan and SAP solutions) are playing nowadays a major role. The lowest level is the shop floor control level. Computerised systems are used here in CAM, machine control and operation monitoring. From the Virtual Enterprising point of view, this is not yet an interesting level.

The important level for interconnecting enterprises in a Virtual Enterprise is between the shop floor level and the ERP level, namely the "*execution*" level. At this level, there are many processes which are current targets for benefiting from IT support and even fully automation:

- detail planning and production scheduling
- resource allocation
- product and equipment tracking
- production flow monitoring
- inventory and quality management
- reporting to higher levels

There is a lot of interaction between enterprises at the business level, but this is realised by humans, even if they are using Internet enabled communication, like e-mail, Group Support Systems or teleconferencing software. Also,

at the ERP level, the automated systems can be interconnected. For example, Baan is providing a specialised module for that, the Enterprise Interconnectivity Component, for linking two different ERP systems. But this kind of connection is characteristic for long term interaction, when the involved enterprises are participating for example in long lived supply chains or they have an indefinite time ending contract.

As we argued before, Virtual Enterprising is based mainly in identifying fast the available resources for non-forecasted opportunities. Many enterprises today are installing Manufacturing Execution Systems (MES), which are basically integrated software solutions for the execution-level processes listed above. Of course, this trend is a new enabler of Virtual Enterprising, because we can link the MES components via Internet. Moreover, our approach is to automate the tasks arisen by the *needs of interaction* at this level, with a special emphasis on negotiation processes.

2.3 Agent based systems and Virtual Enterprising

The paradigm we use for the IT integration across the VE is the *software agent framework*. The typical approach by using agents is to wrap the functionality of the local IT components [Kjenstad98], in order to provide an open and uniform communication and coordination interface between the modules sitting at different sites in the Virtual Enterprise.

To have uniform communication, all the agents must "speak" the same *language*, and for uniform coordination, all the agents must have the same *behavioural model*. Currently, for these two, the most used standards are KQML (an Agent Communication Language - ACL, see [KQML97]), and the BDI behavioural model [Wooldridge99]. This model states that an agent must have Goals ("Desires"), Knowledge ("Beliefs"), and an Agenda ("Intentions"). A reasoning mechanism (for example, an inference engine, like a rule-based production system) is needed to trigger *agent-actions* from the Agenda, in order to achieve finally the Goals. Knowledge (for example "if-then" rules) are previously coded, or acquired in time, to regulate correctly the reasoning process. To insert and sequence actions in the agenda, a planning capability has to built within the agent. Sometimes, to simplify the internal structure of the agenda, the reasoning mechanism is used as a heuristic-based planner. Sequences of actions can be structured in *tasks*.

The actions which an agent can perform, are *functions* of the wrapped modules. If there are more agents, with different capabilities, they can perform *coordinated* actions (in parallel, or sequential), and execute *group tasks*. More than that, sets of agents can have common Goals, and *cooperate* (by exchanging knowledge and/or performing tasks for each other) to achieve these. The most complex form of interaction is when agents are planning their agenda together, and even identify common Goals in the environment. We call this *collaboration*; in the agent literature terms like "team-work", "long-term cooperation", or "joint intentions" are used.

If a group of enterprises from the E-web decide to adopt an agent-based solution for the integration of the MES-level functionality, they must agree in a very precise manner about the *common specifications* for the communication language to be used, and the common behavioural model of the agents. Each enterprise has to implement locally its wrapper software, according with the specifications. Finally, the overall agent system must be tested, validated and updated when necessary. This scenario for the development of an agent-based system illustrates clearly that such a process is an extremely difficult undertaking [Wooldridge98]. This is because:

- for a large group of enterprises, it is difficult to reach in a short time a common agreement about the specifications
- it is hard to believe that all the enterprises have the necessary skilled people to implement the wrapping, or they are willing to pay for specialised help
- testing such a huge system, without a centralised view, is almost impossible
- each update needs again the common agreement of all the involved parties.

These reasons are explanatory for the lack of uptake in the fielded agent-based systems. The agent field of research enjoys today a lot of progress and energy, but there are still only a few real world everyday applications [Jennings99], and none of these is for Virtual Enterprising. Even if the idea of using the agents in VE is not new (see [Fischer&al96]), the and research done in this field produced a lot of novel ideas (for surveys, see [VanDykeParunak98], and [Hendler99]).

2.4 Software mobility is perfectly suited for VE

Our view about how to develop an agent based system for IT integration in VEs is different. We believe that the solution is to use *mobile* agents [Wong&al99]. These are basically the same in their internal structure with the

above presented static agents, but they have a special feature, they can migrate from a computer to another and execute their code on different locations, by their own "will". Because this implies a high degree of portability, mobile agents are usually coded in Java. The simplest example of a Java mobile agent can be the Java applet, a piece of software which is downloaded to the invoking computer and executed there. But a fully mobile agent must be able to migrate when *he* is taking this decision, not when it is invoked. If for running an applet, we need a Java enabled Internet browser, for allowing the mobile agents to run on our computers, we need a specialised software, which in literature is called "mobile agent server", "mobile agent dock" or "mobile agent platform".

On the software market, there are now many mobile and static agent development systems, which provide the functionality of the dock and a common basic structure for the agents. The first mobile agent development tool was General Magic's Odyssey [odys], followed by IBM's Aglets [agle] and ObjectSpace's Voyager [voya] (which is also CORBA compliant). More mobile agent commercial products and academic tools appear on the market every year (for an good survey, see [list]). Almost all these tools are using Java as the implementation language.

An alternate scenario for the development for a VE agent-based MES integrator is to *install docks at all the involved enterprises*. The decision makers of the enterprises have to reach only the agreement about the type of the dock which will be adopted. To develop the agents, we believe that the best approach is to rely on a *third trusted party*, specialised in software development. We call this party the Software Agent Common Provider (SACP). The agents can be migrated (deployed) to the enterprises' computers when they are needed, this model having a very easy installation process. The testing and validation of the system is made by a single party (the SACP), who is in charge also with the collection of the specifications and the execution of the upgrades.

This approach is also different from the classical one, because the goal is not to obtain the wrapping of the local MES functionality. Using mobile agents, we must make them able to identify and *invoke* local functionality and execute them as agent actions when necessary. The communication in the system is realised by the agents, who rely on a common ACL. The role of the local people involved in the agent system development process is to provide access for the agents to local functionality. Our development tool for example, the Mitsubishi Concordia Mobile Agent Framework [conc], has specialised skeleton modules (Java Classes named Service Bridges) to help making these interfaces between the agents and the local software. These Service Bridges are part of the dock, and in the simplest case, they can use Java applets which put questions to the human users and ask them to act on the behalf of the software agent which is initiating the interaction with the local system (this is the case when the enterprise has not an automated MES tool).

However, in this approach, the emphasis is not on the local functionality. By trying to link the paradigm of Virtual Enterprising and the paradigm of Mobile Agents, we decided that we have to concentrate on *order executions*, namely the Clusters. To simplify the case, we considered that the Virtual Enterprise is an E-web, and more than that, all the member enterprises has mobile agent docks installed inside their MES. We call such a group of enterprises a Mobile A-web (Agent web). From the product ordering point of view, an MA-web of enterprises is able to deliver a specific portfolio of products. When an order is issued, a small fraction of the MA-web is committed to execute the order (they have a "target"), and we called this group a *Target Cluster*. Probably, more enterprises can participate to execute a specific order at a specific moment, due to the assumed redundancy, but a selection process is needed to establish the Target Cluster. We call the bigger group a Virtual Cluster.

2.5 The process of mobile agent dispatching

In a Target Cluster, we have pairs of buyers and suppliers, and how this pairs are structured in supply chains depends on the configuration of the ordered product. Basically, there are three models of mobile agent usage between the pairs:

- the push model,
- the broker model
- the pull model

In the *push model*, the agents are sent by the suppliers to the potential buyers, announcing the available resources (one agent for each resource for example). When a customer order is issued, the enterprises can decide (by browsing and selecting the docked agents received from suppliers) who will participate in the Target Cluster. There are some problems with this model, first because the make-to-order policy used by the Virtual Enterprise in relation with its customers is in principle a pull process, and second, the information kept by the agents docked at the buyers site must be continuously updated and kept consistent.

In the *broker model*, the agents of the suppliers are not sent directly to the buyers, but to a virtual meeting place (a Mobile Agent Market). When a resource is needed by a buyer, this is sending a mobile agent to the market (the market can be maintained by the SACP) and by an auction process for example, the buyer's agent is selecting a resource and announcing the buyer. The main disadvantage of this model is that implies a high level of centralisation and a high traffic of mobile agents to a single point of reunion, and this may lead to a lack of robustness of the system and low speed performance. But it has the big advantage of having up-to-date information about the resources.

Both the push and the broker model concentrate on the process of selection of the enterprises which enter the Target Cluster for a specific order. We discovered that the need for interaction in a Virtual Enterprise does not end here. The real problems appear *after* the execution of the order begins. There are three basic classes of problems:

- changes in the due-dates between suppliers and buyers
- changes in the configuration of the product
- de-commitments of enterprises during the order execution process and selection of new ones on-the-fly (from the bigger Virtual Cluster, which acts as a reserve pool of resources).

Related to these problems, the result of our analysis was that *the most appropriate model is the pull model*. In this case, the agents are sent from the buyers to suppliers. A similar approach is used in MAgNET [Dasgupta&al99], where mobile agents (the used platform is IBM's Aglets) are sent to suppliers when a customer is issuing an order. In this case, the customers are part of the MA-web (called by the authors a Networked Electronic Trading System) and their requests can go down at many levels of suppliers. For an request of a buyer, an agent is sent to a list of potential suppliers, ask for quotations, make reservations for the resources and come back to the buyer, which will select the most advantageous supplier (in terms of price and due-date). The system permits also a monitoring of the payments between buyers and suppliers, but is not monitoring the whole execution of the initial order of the customer.

3 The architecture for a VE mobile agent system

Our research group have started to develop a Concordia platform-based prototype named PROVE [Szirbik&al99a]. It is part of a interdisciplinary research project, named ROVE (**R**easoning about **O**perations in **V**irtual **E**nterprises) at the Eindhoven University of Technology. The role of the prototype is to identify the problems which appear in the Virtual Enterprising process, and to find the best architectural patterns for IT VE integration oriented systems.

3.1 The central repository

The agents in PROVE will use a sub-set of KQML for communication and a BDI behavioural model. They have three roles:

- the selection of the enterprises for a Target Cluster
- the tracking and the monitoring of the ordered products
- supporting negotiation processes between the enterprises when problems appear.

The selection process is somewhat different from the MAgNET model. We are using a semi-broker model, by keeping a *centralised repository* of the available resources at the SACP site. This repository is continuously updated by the so-called *roaming mobile agents*, which have the role to visit as often as possible all the enterprises in the MA-web and report to the repository all the appearing changes. An alternative is to use a specialised static agent, for each enterprise, which reports the changes to the repository (like an alert-server). But we realised that the reporting process, which is normally time-triggered, leads to overflows of information to the repository site, that is, all the enterprises tend to report at the same moment. A roaming agent reports sequentially the situation of the sites he visits, and the reporting process is initiated not by a time-trigger but by the event of his visit, a method which we realised that is more reliable.

The repository can be viewed as a Gantt chart (see figure 2), where all the foreseeable available time-slots for resources are stored, for a considered horizon in time. The extension of the horizon is depending on the time necessary to executing an order. For example, if the VE is about replacement landing-gears, and it is known that the time from ordering to delivery for such a product is one month, the horizon must be slightly more than one month. This data structure is appropriate for manufacturing resources, but we can use it also for material resources. This Gantt chart is used for a Global Scheduling (GS) activity. When a customer (we are not including yet the customer in the MA-web like MAgNET does) is issuing an order, the enterprise who is taking the order (we call it a VE "gate") is creating an agent, with the help of the SACP. This agent is dedicated for the execution of the order. The final product to be delivered is a Target, and it is linked to this agent, who has a Goal, to realise the Target.

The product, for example the undercarriage, is to be assembled using components from suppliers, which have to manufacture their parts or simply deliver these from existing stocks. Each component is expressed as a resource type. Also, the assembly (and installation) process is viewed as a resource type. The agent at the gate knows the structure of the resource types needed, and invokes the Global Scheduler from the SACP site, sending a message there, containing all the needed resource types and the dependency constraints between them.

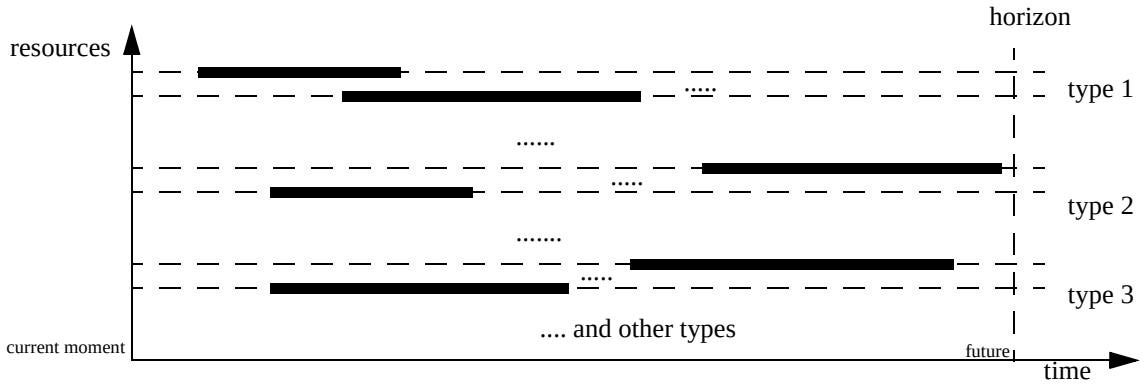

figure 2: The central repository data model

The Global Scheduler, using the data from the resource availability repository, is generating a schedule to purchase, manufacture, assemble and install the product. For each component and process, a free resource of that type is assigned. Knowing the resource, it means to know the enterprise who is offering it. This information is sent back to the agent.

3.2 Dispatching the mobile agents in the Target Cluster

The agent at the gate is creating a mobile agent for each resource, and sends these agents to the related enterprises. When the agents arrive there, they are asking if these resources can be committed for the current target, and for what price. If the response is yes, the agents are sending confirmatory messages to their creator agent, and if this one agrees with the selection, it sends a message to the central repository, where the assigned resources are deleted from the repository. At this moment, a contract is signed with the customer, and a delivery date (resulted from the GS) is established. As stated before, we consider that time is more important than price, and we are not making any optimization in terms of costs. It is possible to keep in the central repository price information about the resources, and a more advanced scheduler can generate a GS according with financial criteria also.

This scenario has only one level of product decomposition. The usual case is multi-level. If so, each supplier becomes a new gate, and the process is repeated. The main problem here is that the delivery date for components can exceed the delivery date initially stated in the first GS. In this case, a re-schedule of the previous levels is needed and this could lead to many invocations of the GS, process which can cascade out of control. Also, the contract is not to be signed before the decomposition reached the lowest levels, and all the enterprises in the Target Cluster are committed with their resources (it is necessary to have a full Available To Promise check). We are currently analysing the case of dispatching the mobile agents to the lowest level, in a single move, based on a single GS, which takes into account from start, all the components in a product, to the lowest detail. That means that the gate has to know the configuration of the product (the augmented BOM - bill of material, together with the processes linked to it) to last bolt, if this bolt is purchased or manufactured within the MA-web.

3.3 The monitoring phase

After all the agents are dispatched in the Target Cluster, and all the enterprises are committed to execute that order, the work-effort can begin, following the overall plan of the GS. In the product execution phase, the role of the agents is to monitor the processes, by invoking monitoring functionality in the local MES. The agents must be able to detect any problems which can occur at a local site, with respect to the normal execution of the target. When a component is finished at a supplier's site, the agent is reporting that, and after the component leaves the enterprise to the buyer's site, the agent is also migrating to the buyer's site, tracking the component. His creator agent, located currently at the buyer's site, is taking the component's agent data which is relevant for further use, and deletes the agent. This process is a reversal of the dispatching process in the selection process; finally, a single agent remains at the initial gate, and when he is reporting that the target is finished, the product can be delivered to the customer.

In order to be able to monitor the execution of the local component, the agent must have access to various data in the MES, such as daily schedules and confirmatory reports. Another policy is to directly ask human users, in a pro-actively manner, about the status of the component execution. The agent must be "intelligent" enough to detect when a problem occurs.

3.4 Problem management

Our research emphasis is not on the dispatching and target execution monitoring, (which are quite simple and straightforward) but on using the agents to support processes which are taking place when problems appear during the target execution. Analysing different cases of undesired events, we discovered that the worst case is when an enterprise is not able to fulfil its commitment and decides to leave the Target Cluster. Usually, this is happening before the work-effort has to start at that site, and one of the key roles of the monitoring agent is to identify early this problem. We call this the RENDER case.

If an enterprise has to render, another enterprise has to be found from the current Virtual Cluster. Because the repository of the resource availability is continually updated, the configuration of the Virtual Cluster at the rendering moment is different from the one when the initial GS was established. Some resources have been committed, and other new ones have been reported by the roaming agents. The ideal case is when we can find a new resource, replacing the lost one, which fits in the same time slot. The agent located at the rendering enterprise has the task to invoke the central repository to find such a resource. If the resource was found, the agent is migrating to the enterprise which is offering this resource, ask for its commitment, and if successful, informs the eventually agents below him about the necessary re-routing in transportation.

This scenario is eloquent about our perception about the relations which must exist between the constituent parties of a Virtual Enterprise. In such an organisation, de-committing is not leading to legal action and deteriorating the relations between the enterprises, but a mutual understanding exists, together with a common willingness to solve the situation on-the-fly. Because the parties are willing to give information to agents updating the central repository, there is a reward, in case of undesired events the flexibility of the manufacturing process permits easy adaptation to the new situation. Thus, an enterprise who enters the MA-web, by installing a dock at its site, is also taking a more deep commitment, that is to treat its partners in a collaborative manner. It is very interesting to foresee how a new technology can help to improve enterprise to enterprise relations and subsequently enable better Virtual Enterprising.

If there is no similar resource available in time to replace the rendered one, the agent is invoking a new GS, which is not taking into account the committed resources which are already in use in the work-effort for that Target. There is a slight possibility that the final due date to the customer still could be met. Somebody has to decide if this course of action will be taken, because this implies that the new resources in the new GS has to be committed. And that means that some of the previously committed resources have to be de-committed, and in an extreme case, it is possible that all the enterprises which have been committed, but yet not started to work for the target have to de-commit.

3.4 Local negotiation vs. overall re-scheduling

If the delivery to the customer cannot make the due date, the product will have a lower price, and the difference is to be paid as a penalty by the enterprise which rendered the resource. If this enterprise is seen as a supplier, the buyer linked to it can solve the situation in a simpler and more efficient manner, resulting in a win-win-win (supplier, buyer, final customer) situation. The agent at the buyer's site can look into the repository for a resource which has the same type, but a slightly later time-slot, and send an agent there, to check commitment. Between the buyer and the new supplier, a negotiation to fix a *new due date* can be arranged. The buyer is trying to make its time-slot smaller, by moving its work effort start to a later moment, and the new supplier is trying to deliver earlier, to a date which is before the buyers starting date. Of course, these mean that the work-effort has to be more intensional (by hiring temporary staff and machines, working more hours, etc.) and the rendering enterprise has to pay to the buyer and the new supplier the supplementary effort. But it is possible that this sum is less than the penalty to be paid to the customer, and more than that, the initial GS can be followed, without de-committing enterprises which are not "guilty". And more important, the customer delivery date is kept.

We identified many more possible scenarios and the main role of the PROVE system will be to check various solutions to solve these scenarios. Close to the RENDER case, we have:

- the POSTPONE case, when an agent is detecting that a resource's time-slot is moved later in an enterprise which has not started yet the work effort for the target,
- the EARLIER case, when the time-slot is moved earlier
- the DELAY case, which is close to the POSTPONE case, but the enterprise has started the work effort
- the CRASH case, when an enterprise which already started, has to de-commit.

Each of these cases has many possible scenarios and ways to solve the problem. We called this process the Delay Management. In each situation, the main common goal of the agents is to find a solution which leads to the delivery of the product to the customer as close as possible to the initial due date. As a secondary goal, it is to find the less harmful solution, the one which implies the smaller penalties and disturbances for the enterprises involved. Following the behaviour of such an agent-based system means to share risks, profits, and improve customer satisfaction. Also, it is leading a better cooperation within the Virtual Enterprise. By allowing the agents to interfere in the negotiation processes, we empower them with decision making capabilities. They will act as mediators of the negotiation process (see figure 3). And this process will be executed according with the rules coded in the agents. All the enterprises must be willing to obey these rules, which enact some sort of generic code of conduct during the negotiations within the Virtual Enterprise.

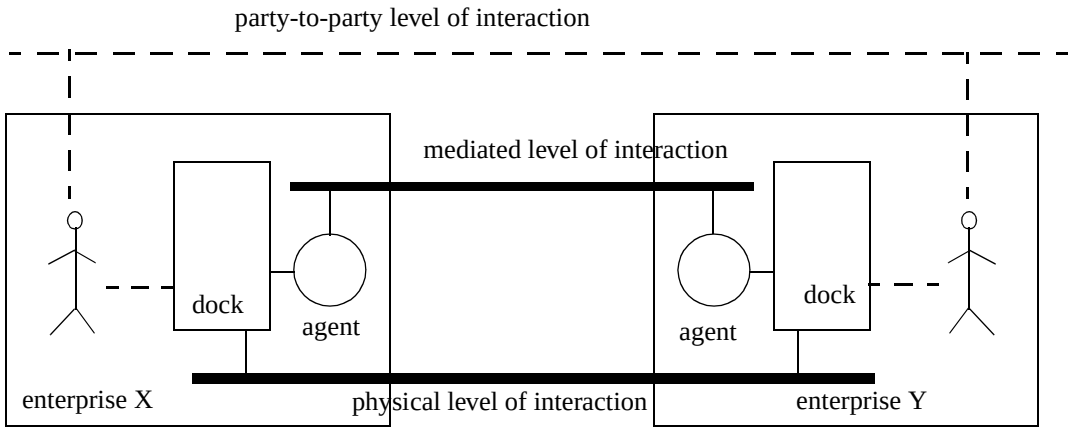

figure 3: The agent-mediated model of interaction

3.5 Change management

Another problem may appear if an enterprise is making small changes in the configuration of the product (not changing its functionality and overall form, but only by changing fit characteristics). These changes can affect the following steps of manufacturing, and the agents must be able to detect, report and solve eventual disagreements. It is possible that a change at a site is not accepted by the others, and the enterprise has to de-commit, or the non-accepting parties have to de-commit. For a more detailed discussion about the management of configuration change using agents, see [Szirbik&al99b].

4 Agent mediated negotiation

By trying to identify *common patterns* for solutions in the presented cases of undesired events, we realised that the solving process is always finalising in a negotiation between a pair of enterprises, and the issue of the negotiation is a *price*. In such a negotiation, we have a seller (supplier) and a buyer. The seller is trying to sell at the highest price and the buyer is trying to buy at the lowest price. Before the negotiation is starting, each party has a *reservation price* (the "walkaway" price) and the buyer will not pay more and that (we refer to this price as *b*) and the seller will not accept less that his (a value referred as *s*).

4.1 The negotiation basic scenario

Using mediator agents, we can have the following scenario for interaction:

1. Both parties are sending their reservation price to the locally docked agent
2. The agents compare the prices, and if $b>s$, then a *zone of agreement* exists and a fair price can be the simple average of b and s (like in figure 4). Negotiation ends.
3. If $b<s$, there is no possibility for agreement, and the agents have to ask the parties to give in a little bit and propose new reservation prices. We go back to step 2.

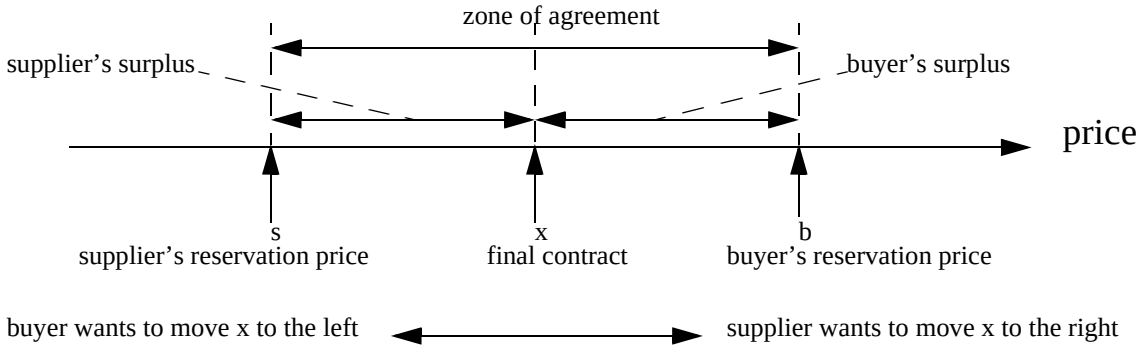

figure 4: The geometry of bargaining ([Raiffa82])

This mediation method can be enhanced in many ways. The problem is because the parties in a mediated price negotiation tend to exaggerate the reservation prices [Raiffa82]. A solution is to use instead of fixed value reservation prices, uniform probability distributions, in terms of price intervals. The buyer and the supplier are sending to the agents price zones $[b_l, b_h]$, $[s_l, s_h]$. In this case (the "canonical case", see [Harsanyi77]) the agents can compute the probability to reach an agreement. For example, if the price zones are [100,200] for the supplier and [150,300] for the buyer, the probability to reach an agreement is very high, *p(a)*=11/12 (see figure 5). The agents can decide if the probability is high enough and asks for the disclosure of the real prices. Or if the probability is too low, the agents can ask for a an adjustment of the distributions.

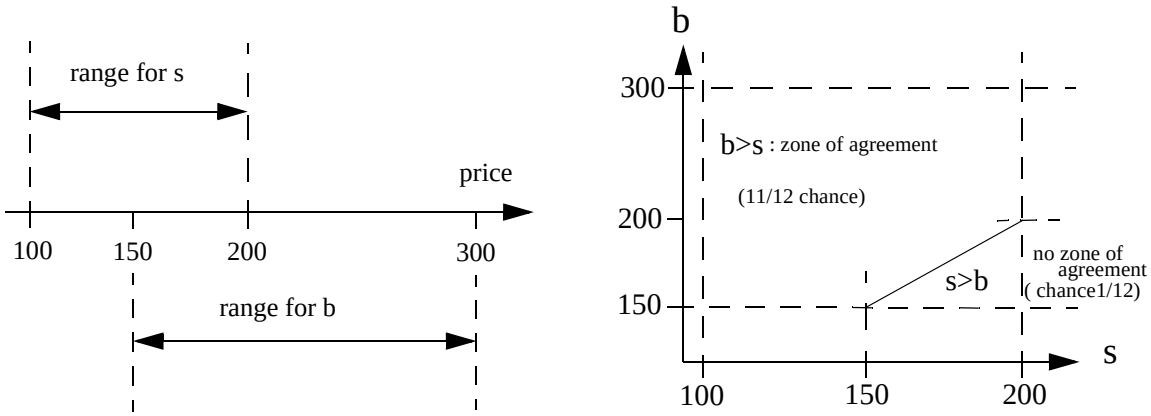

figure 5: The case of uniform distribution of reservation prices ([Harsanyi77])

Also, the agents can have previous knowledge about the price zones (from previous negotiations between the same parties, or other pairs) and try to keep the current values within "normal" boundaries, by not letting a party to exaggerate the price zone too much. Another possibility is to ask neutral parties in a similar context about the prices they are willing to pay and compare to the current situation. The final price can be influenced by this supplementary knowledge, because sometimes is not fair to apply the average formula (*1/2*(b+s)*), if one of the party exaggerated and the another not. This is named *asymmetric mediated price negotiation* [Raiffa82] and empowers the mediators with more power than in the normal situation. It is up to the mediator to consider which of the parties is exaggerating and which is not.

The negotiation process can be viewed as a agent group task, composed by actions taken by the agents who are driving it. In PROVE, we adopted a Java written inference engine, JESS (for Java Expert System Shell), which is built with the similar approach as the classical CLIPS inference engine [jess]. The agenda for the negotiation and the triggering of the actions are a result of the rules written by the agent designer. The rules are looking like these examples:

IF buyer(myself) *and* price(xxx) **THEN** send_price(pair_agent)
IF supplier(myself) *and* price(yyy) **THEN** send_price(pair_agent)

The effect of applying these rules in both agents is that the prices are sent to each other at the moment when they are known from the human user. We are still in the phase for experimenting with various bodies of rules, and the length of the agenda is limited to only one action. But the point is that the "programming" of the agents is made not directly in Java, but in much more expressive language (if-then) rules, which can be understood by different decision people who are adopting the common body of rules for the entire Virtual Enterprise. We consider that it is very important to illustrate clearly the behaviour of the agents to the people who will use them as mediators. This is because these people must be willing to rely on a system towards they are delegating part of their power. From the programming point of view, the rules can be more easily changed than programming code, or for different products in the same VE, we can have similar PROVE agents, with different rule-bases, adapted for specific interaction and behaviour.

4.2 An example

As an illustration, we show the following case: a party S1 with the due date T1 is rendering, and in the worst case scenario, implying a totally new re-schedule, it has to pay a big penalty *p* to the customer. The agent at S1 is messaging to the agent at the buyer B. This agent is looking for a replacement resource at S2, but the due date T2 is slightly later than T1. The buyer is compressing its time slot for its work effort to start later at T3, but this is still earlier that T2. The cost of the compressing is *b*. The buyer asks S2 to compress its time to be able to deliver before T3 (see figure 6). The cost of the compressing at S2 must be less than *p-b*, which is the reservation price for B. The agent at S2 can seek for similar resources (in terms of type and duration), and ask about the price of compression. From these data, it is possible to construct a probability distribution and compute the probability for the agreement. If there is no zone of agreement with any possible S, the agent at B can contact the next buyer, B2, and negotiate with him a possible compression, leaving T2 as the starting moment at B, but delaying the delivery to B2 to T4.

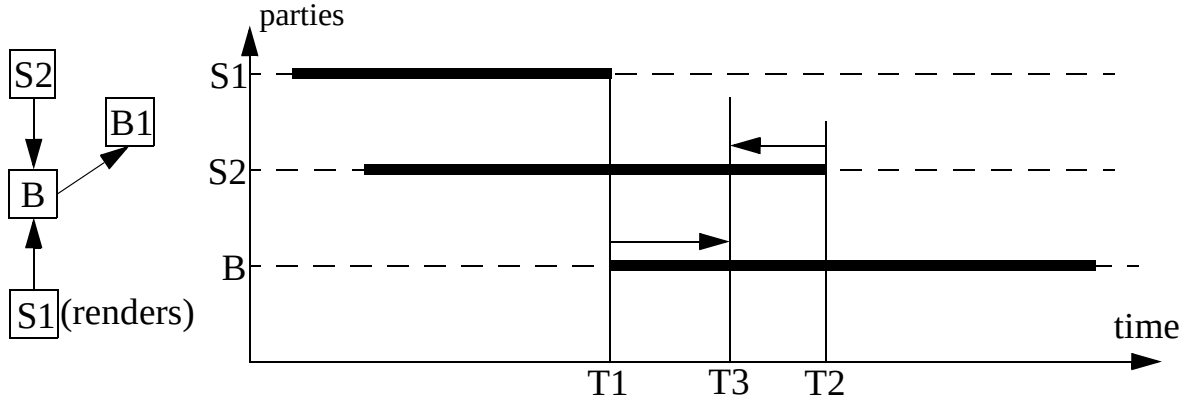

figure 6: The compression of time-slots made by the buyer and the supplier S2

The agents are viewed in these negotiations as third trusted parties, and they are not disclosing the reservation prices of the other party. They are only announcing if the agreement can be reached, and in the "yes" case, together with the final price, which is always above the reservation price (see the "surplus" in figure 4).

4.3 The improved method

Using mediator agents is somehow equivalent to the well known method for price bargaining named *simultaneous-revelation resolution procedure* [Pratt79]. This method is the fastest known, but it is inefficient because encourages exaggerations. In [Raiffa82] it is stated that "partners consider that it is culturally acceptable to exaggerate a little bit in your favour". The problem is that if both parties exaggerate a lot, then the chances for an agreement are very poor. A laboratory experiment involving humans (made by Samuelson, presented in [Raiffa82]) compared the ***t**rue **r**eservation **p**rices* (TRP - established by the experimenter) and the ***a**nnounced **r**eservation **p**rices* (ARP) in simulated price negotiations. The experiment revealed that in many cases, especially when the TRPs values were close, the negotiation failed, because the ARPs give no zone of agreement.

Fortunately, the simultaneous-revelation resolution can be altered in such a way as to engender truthfulness [Chaterjee78]. Suppose the there is a supplier S, a buyer B, and a mediator AG (in our case, the pair of agents). Imagine that A can induce B to make honest revelations: his declared price, ARP_B, is the same as his real price TRP_B. How can AG get S to be equally honest? If S real price is TRP_S and if announces ARP_S, while B announces a new ARP_B = TRP_B, the payoff for the supplier will be [(ARP_B+ARP_B)/2]-TRP_S. Of course, if a zone of agreement exists (ARP_B>ARP_S). In order to have truthfulness, an adjustment to this payoff is made.

A supplementary amount is added (paid by the buyer), depending on the ARP_S the supplier announces (see figure 7).
.

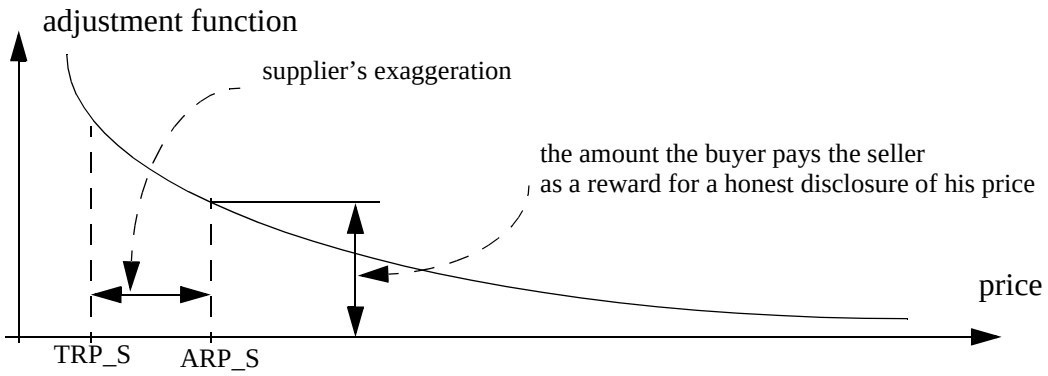

Figure 7: The extra payoff the supplier receives as a function of ARP_S (proposed by Chaterjee).

Notice the higher supplier's ARP_S the lower the adjusted payment he will receive from the buyer. The difficult part of implementing this scheme is to construct the adjustment in such a way so that if the buyer tells the truth (ARP_B = TRP_B), then the supplier's overall response is also to tell the truth (ARP_S = TRP_B). The problem is that the function depends on the value of TRP_S and the interval between this value and TRP_B, because it is important that S has to announce ARP_S = TRP_S for all possible values or TRP_B and TRP_S. If we assume that the supplier is announcing the correct value, how we can determine the adjustment that B will have to pay him? AG will induce B to pay this adjustment by reversing the roles and making the supplier pay to S an adjustment value that depends solely on ARP_B. AG will construct the adjustment function for B so as to make it most profitable for him. The final expectation is that the side adjustments will sum to zero. With suitable adjustments functions, honest revelations are in equilibrium, and each party should tell the truth if the other does.

Can all this be done? The experts [Raiffa82] [Chatterjee78] [Pratt79] say that the buyer and the supplier have to agree to it before the negotiation begins. If they are using the agents as mediators, that is implicit. The mediator has to calculate appropriate adjustment functions, and he needs to know previously the probability distributions that are yielding the drawing of TRP_B and TRP_B. In this case, it is necessary for the agent system to keep a history of the previous negotiations, in order to select the appropriate adjustment function when necessary.

4.4 Experimentation techniques

In designing a mobile agent-based system, the main concerns are: robustness, speed performance, dependability, maintenability, security and other X-abilities. Because we are relying on the Concordia system, which has all the above enumerated characteristics, our emphasis is to find the rules which will define the agent behaviour, especially in the negotiation processes. As stated in the previous subsection, it is also very important to find the appropriate adjustment functions in the simultaneous revelation procedure, and the rules how to use these functions.

This kind of system is not to be fully automated, and humans will have to take decisions and to give inputs. This class of systems is named *mixed-initiative systems* [Horvitz99]. Because it is very hard to predict human behaviour, the development and tuning of such systems can be made only by laboratory experiments involving humans. Our idea is to develop a prototype with a limited body of rules, necessary for dispatching, monitoring, migration, problem detection and rudimentary negotiation. We will simulate negotiations between parties, involving human participation, in certain environments and situations, and conclude which are the most appropriate rules to resolve the problem in a harmless way, improving gradually the system. This is one the reasons why the rule-based inference machine is used in the agents, because it is easy to change rules and not the Java code. To find the good rules is not only a problem of programming but also a problem of human psychology, and psychologists specialised in negotiation and conflict management will participate in our experiments, especially to prepare the experiments' human context. For them, it will much easier to understand the rule format than the procedural format of directly coded agents.

5 Conclusions and future work

We presented in this paper an outline of the architecture of the PROVE system. We argued that the mobile agent paradigm is well suited for the IT VE integration and we give insights about the design decisions. As a basic principle, we decided to use a pull model for the agents, instead of a push, auction-based scheme. Also, we presented the advantages of using a central repository, maintained by mobile roaming agents, to keep the needed information in Virtual Enterprising continuously updated. We presented some of the processes which are taking place in the system, like dispatching, monitoring, delay management and negotiation. Finally, we detailed some aspects of price negotiations and how these can be regulated by using agents as mediators.

The system is designed now to tackle only one-of-a-kind ordering style (like in the landing-gear example). The first logical extension is to extend the model to a lot-size ordering style, and further, to time-phased batches. We still have to take the decision to use only one agent per lot, or to use an agent for each individual item (for a discussion about this, see [Szirbik&al99a]). Another improvement will be the inter-target agent negotiation. In the current model, only agents assigned to the same target can interact. But the VE can have many targets simultaneously, with agents which are competing for resources, and it could be a good improvement to let the agents of different targets to negotiate and resolve resource allocation conflicts, having an overall VE perspective. This is important, because targets have different priorities, depending on the contract with the customer, product complexity etc. Related to the negotiation procedures, we intend to extend them from pair to team negotiations, and use other forms of mediated negotiations, like “divide-and-chose”, asymmetric bargaining, SNT etc. [Raiffa82].

From a higher perspective, it is interesting to note the ethical dimension of this kind of application for the agent technology, such as the domanin of enterprise-to-enterprise interaction. Adopting such a solution, a group of enterprises will agree to adapt part of their behaviour to a common set of norms (simply coded in the agent rule-bases), and to support the enforcement of this behavior by a software system. A psychological problem is that the power delegated to the agents and indirectly to the third trusted party who is manipulating them is making some business people uneasy. But we strongly believe that a system like this will reduce antagonism, threat posturing, distrust and will slide the relations between enterprises towards confidence, honesty and a higher level of cooperation.

References

[agle] http://www.trl.ibm.co.jp/aglets/

[conc] http://www.concordia.mea.com/concordia/

[jess] ***, “The Java Expert System Shell”, http://herzberg.ca.sandia.gov/jess/

[list] ***, “Agent Construction Tools”, http://agentbuilder.com/AgentTools/

[odys] http://www.genmagic.com/technology/odyssey.html.

[voya] http://www.objectspace.com/voyager/

[Bradshaw99] Bradshaw, J.M., (1999), “Everything I Know About System Design I Learned from My Architect”, in *Education and Smart Machines*, (Forbus, Feltovich & Canas, eds.), AAAI Press & MIT Press, Cambridge, MA.

[Chaterjee78] Chaterjee, K., (1978), “A One-Stage Distributive Bargaining Game”, *Report 13-78, Graduate School of Business Administration*, Harvard University, Cambridge, MA.

[Dasgupta&al99] Dasgupta, P., Narasimhan, N., Moser, L.E., Melliar-Smith, P.M., (1999), “Mobile Agents for Networked Electronic Trading”, in *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 4, pp.509-525.

[Fischer&al96] Fischer, K., & al. (1996), “Intelligent Agents in Virtual Enterprises”, in *Proceedings of PAAM’96*, London, pp.205-223.

[Goossenaerts&al98] Goossenaerts, J.B.M., Aerts, A.T.M., Hammer, D.K., (1998), “Merging a Knowledge Systematisation Framework with a Mobile Agent Architecture”, in *Information Infrastructures for Manufacturing II*, (Mills & Kimura, eds.), Kluwer, The Netherlands.

[Harsanyi77] Harsanyi, J.C., (1977), *Rational Behavior and Bargaining Equilibrium in Games and Social Situations*, Cambridge University Press, Cambridge, UK.

[Hendler99] Hendler, J., (1999), “Making Sense out of Agents”, in *IEEE Intelligent Systems and their applications*, March/April 1999, pp.32-37.

[Horvitz99] Horvitz, E., (1999), “Principles of Mixed Initiative Interaction”, in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing System*, ACM Press, pp.159-166; http://research.microsoft.com/~horvitz/uiact.htm

[Jennings99] Jennings, N.R., & al., (1999), “Implementing a Business Process Management System using ADEPT: A Real-World Case Study”, http://www.elec.qmw.ac.uk/dai/pubs.

[Kjenstad98] Kjenstad, D., (1998), “Coordinated supply chain scheduling”, *Phd Thesis (NTNU 1998:24), Department of Production and Quality Engineering*, Norvegian University of Science and Technology, Trondheim, Norway.

[Kornelius99] Kornelius, L., (1999), “Inter-Organisational Infrastructures for Competitive Advantage: Strategic Alignement in Virtual Corporations”, *PhD Thesis, Faculteit Techonlogie Management*, Eindhoven University of Technology, The Netherlands.

[KQML97] Labrou, Y., Finin, T., (1997), “A Proposal for a New KQML Specification”, *Technical Report TR-CS-97-03, Computer Science and Electrical Engineering Dept.*, University of Maryland, Baltimore, MD.

[Pratt97] Pratt, J., Zeckhauser, R., (1979), “Expected Externality Payments: Incentives for Efficient Decentralisation, *Report 26-79, Graduate School of Business Administration*, Harvard University, Cambridge, MA.

[Preiss&al96] Preiss, K., Goldman, S.L., Nagel, R.N., (1996), *Cooperate to compete: building agile business relationships*, Van Nostrand Reinhold, London.

[Raiffa82] Raiffa, H., (1982), *The Art and Science of Negotiation*, The Belknap Press of Harvard University Press, Cambridge, MA.

[Szirbik&al99a] Szirbik, N.B., Hammer, D.K., Goossenaerts, J.B.M., Aerts, A.T.M., (1999), “Mobile Agent Support for Tracking Products in Virtual Enterprises”, in *The working papers of the Workshop on Agent-Based Decision-Support for Managing the Internet Enabled Supply-Chain*, pp.93-100, AGENTS’99 Conference, Seattle.

[Szirbik&al99b] Szirbik, N., Wortmann, H., Hegge, H., Goossenaerts, J., (1999), “Improving manufacturing cooperation in a virtual enterprise by tracker mobile agents”, in *Proceedings of EFTA’99*, pp.1451-8, Barcelona, IEEE Press.

[VanDykeParunak98] Van Dyke Parunak H., (1998), “What Can Agents Do in Industry, and Why?”, in *Proceedings of Cooperative Information Agents II*, Springer, LNAI 1435, pp.1-18.

[Wong&al99] Wong D., Paciorek, N., Moore, D., (1999), “Java-based Mobile Agents”, in *Communications of the ACM*, March 99, pp.92-96.

[Wooldridge98] Wooldridge, M., Jennings N.R., (1999), “Pitfalls of Agent-Oriented Development”, http://www.elec.qmw.ac.uk/dai/pubs/.

[Wooldridge99] Wooldridge, M., Jennings, N.R., (1999), “The Cooperative Problem-Solving Process”, in *Journal of Logic Computation*, vol. 9, no. 4, pp.563-592.

[Wortmann&al97] Wortmann, J.C., Muntslag, J.C., Timmermans, P.J.M., eds., (1997), *Customer Driven Manufacturing*, Chapman & Hall, London, UK.