An autonomous data collection system for virtual manufacturing systems

Reggie Davidrajuh
Narvik Institute of Technology, Narvik, Norway
Ziqiong Deng
Narvik Institute of Technology, Narvik, Norway

Keywords
Manufacturing,
Systems management, Internet,
Database management

Abstract
First this paper introduces the concepts of virtual manufacturing system (VMS). The host enterprise and the multiple numbers of supply and distribution enterprises that make up a VMS, and the hierarchical and horizontal relationship that exists between these enterprises are explained. The steps involved in formation and operation of a VMS are then analyzed in detail. Second, we present a three view based methodological approach to make a multi-agent model of VMS. Finally, with the help of a testing prototype, we show how to develop an autonomous Internet based data collection system for operation of VMS in accordance with the proposed methodological approach.

International Journal of Agile Management Systems
2/1 [2000] 7–15
© MCB University Press
[ISSN 1465-4652]

Introduction

Virtual manufacturing system (VMS) means a group of enterprises, dynamically organized, cooperating with each other to develop, design, and produce a class of products to respond to the market with attractive and qualitative products, flexible and fast responsiveness, and lowest price.

There are four fundamental flows among the collaborating enterprises in VMS environment. They are material, work, information, and fund flow. This paper concentrates on information flow in VMS, especially on "data collection". As explained in the next section, a VMS consists of a host enterprise and multiple supply enterprises and distribution enterprises. Host enterprise needs information from the other supply and distribution enterprises so that it can select its collaborators for any production. By data collection, we refer to the procedures for providing the host enterprise with the necessary information about the other enterprises.

With the necessary information provided by the data collection system, the host enterprise may use any program or algorithm on the information for the selection process.

In this paper we propose a methodological approach to design a data collection system for operation of VMS, where the following paradigms are used:

- *Distributed event* triggering for control and coordination of the whole system ("control activity");
- *Mobile agents* for communication between enterprises ("communication channels");
- *Wrapper agents* tackle semantic mismatches between data sources, legacy sources etc. ("components")

In the following sections, first we introduce VMS, and then we describe a methodological approach for a data collection system. Finally a testing prototype, which is developed by the authors, will be illustrated.

Virtual manufacturing system

The computerized enterprise integration (CEI) concept was raised in the late 1960s. Conventionally, CEI is mostly concerned with intra-enterprise integration, i.e. integration of computer-automated islands such as computer-aided design (CAD), computer-aided process planning (CAPP), production planning and control (PP&C), computer-aided quality control (CAQ), and computer-aided manufacturing (CAM) within a given enterprise.

Today, manufacturing enterprises are faced with challenges globally. Increasingly, these challenges cannot be effectively met by isolated effort within a single enterprise. Therefore, CEI is nowadays concerned with not only what happens within a given enterprise (*intra*-enterprise integration), but also what happens among a group of enterprises (*inter*-enterprise integration).

To aid the plan and implementation of new inter-enterprise integration, or to aid the analysis and improvement of the existing inter-enterprise integration operation, naturally we need a powerful modeling methodology to realize such purposes.

Inter-enterprise integration can be categorized into product-alliance-centered integration (midcro integration) and finance-alliance and risk reduction centered integration (macro integration) as shown in Figure 1 (Deng, 1997; Deng *et al.*, 1998). This paper focuses on modeling methodology for product-alliance-centered inter-enterprise integration (midcro integration).

Product-alliance-centered inter-enterprise integration involves a host enterprise integrated with enterprises in the supply-chain and distribution-chain for their common products' development, manufacturing, and distribution. The host enterprise is the organizer of the alliance.

However, when market requirements have changed, new products should be turned out to market. In turn, a new combination of enterprises (new supply chain and distribution chain) should be re-organized (or reconfigured, see Figure 1). This dynamic organization of manufacturing can be called virtual manufacturing. Then, the dynamically organized manufacturing system is called a virtual manufacturing system (VMS).

The life cycle of VMS includes phases such as business opportunity identification, partner selection, VMS formation, VMS operation, and VMS reconfiguration (Enator, 1998).

No matter which enterprise in the "Host/ Supply-chain/Distribution-chain" combination, each of them is of autonomous feature and they are geographically distributed. To model such a system, a multi-agent-modeling concept is a natural option, owing to the fact that an agent in multi-agent environment characterizes properties of autonomy and distribution (Nwana and Ndumu, 1998; Laufmann, 1998).

Formation and operation phases of VMS

In a VMS, the host enterprise must search and form the collaborating partners first; this is the formation phase. Then the collaboration is put to use for producing a class of products, which is the operation phase. During the operation phase or after, the host enterprise may look for a new collaboration for producing the same class of product or a new class of product; this phase is called the reconfiguration phase. Clearly, the formation and reconfiguration phases are similar.

Formation phase of VMS

When the host enterprise looks for the collaborating partners, it looks for data about the price, quantity, delivery time, etc. from the potential suppliers and distributors. The host enterprise may launch mobile agents to hundreds of enterprises to fetch these data from their Web pages. With the current HTML (hypertext markup language) based Web sites, it is not possible to fetch product data, as HTML only deals with the appearance of the Web page and does not support automatic retrieval of data by any visiting mobile agents. Another problem is searching potential collaborators' Web sites among thousands of Web sites. We propose an efficient data collection system for formation phase of VMS based on: extensible markup language (XML); and resource description framework (RDF) (see Figure 2).

By referencing RDF, the host enterprise gets the Web addresses or uniform resource locators (URL) of the Web sites of the supply and distribution enterprises that are dealing with manufacture of a specific product class. Then the host enterprise or a unit that is central in forming collaboration – let us call it the host coordinator agent (HCA) – sends mobile agents to all the Web sites in order to

Figure 1
Formation and reconfiguration of inter-enterprise integration

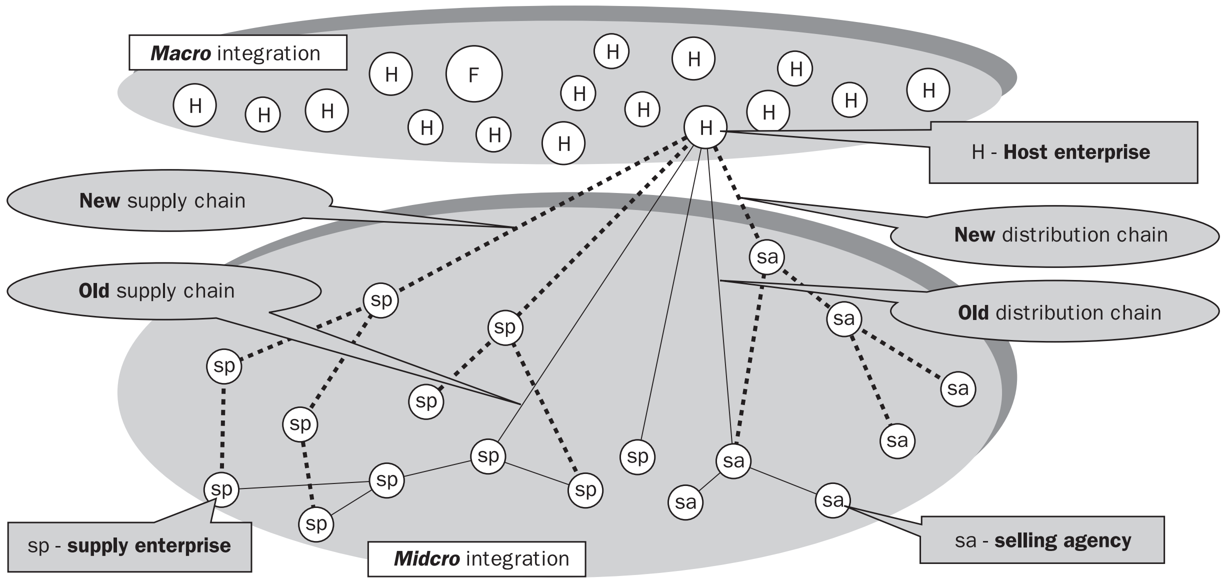
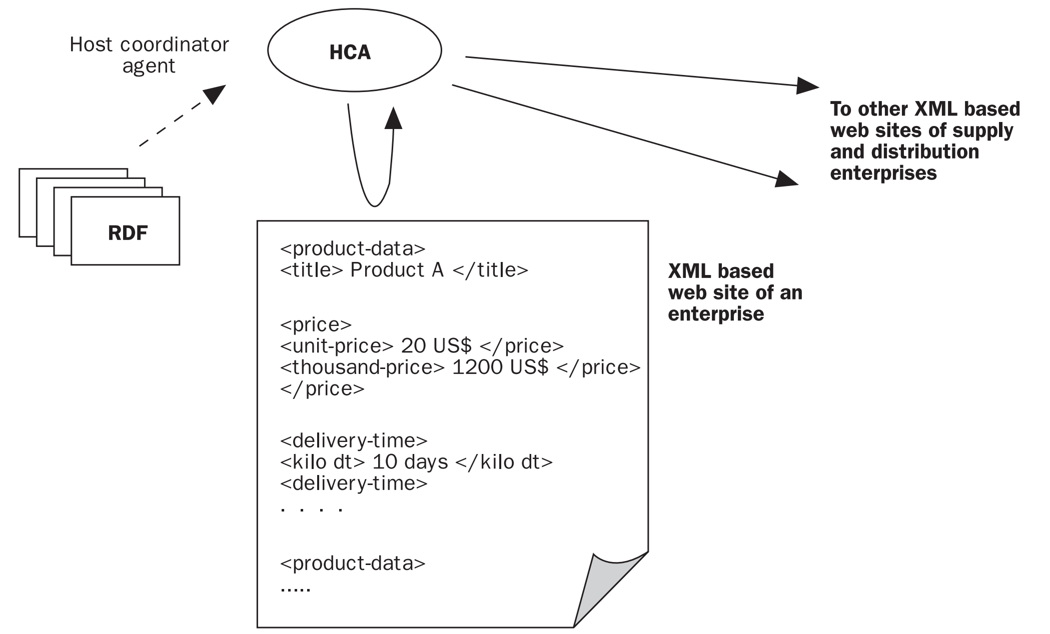


Figure 2
Formation phase of VMS with mobile agent, XML and RDF


Host coordinator agent
HCA
To other XML based web sites of supply and distribution enterprises
RDF
XML based web site of an enterprise
<product-data>
<title> Product A </title>
<price>
<unit-price> 20 US$ </price>
<thousand-price> 1200 US$ </price>
</price>
<delivery-time>
<kilo dt> 10 days </kilo dt>
<delivery-time>
. . . .
<product-data>
.....

retrieve data from these Web sites. To allow this information exchange, the product data on these Web pages are tagged with XML tags (for example, <product-data> <price> ... </price> </product-data>, see Figure 2) so that the visiting mobile agents can recognize these data. However, before such a page is constructed, there must be agreement on:

1 the tags that are allowed;
2 how tags are nested within one another;
3 how tags should be processed (Bosak and Bray, 1999).

There is no agreement for product data description based on XML up to now; until such an agreement is made, our proposed data collection system for formation phase of VMS will only be a conceptual one. With the data collected from diverse enterprises, the host enterprise may choose the best collaborators for good operational properties and go on to the operation phase of VMS. Herein, the very good properties mean that the VMS can supply customers with a sufficient quantity of products with customer-satisfied quality, price, and response time, and at the same time it can get higher profit.

Operation phase of VMS

During the operation phase of VMS, the host enterprise coordinates the supply chain and distribution chain as shown in Figure 1. Therefore, as shown in Figure 3, a VMS coordinator agent (VCA) is needed for coordinating tasks among different enterprise coordinator agents, which are host enterprise coordinator agent (HCA), supply enterprise coordinator agents (SCA), and distribution enterprise coordinator agents (DCA).

Host coordinator agent HCA is the agent that requests VCA to fetch data from the other supply and distribution enterprises. These data are then kept in the host enterprise local database for its local agents' consumption. Supply or distribution enterprise coordinator agent (SCA or DCA) has tasks such as: coordinating with VCA to provide the host enterprise with new data, to notify change of information, etc.; coordinating internally with its local agents to get the data from them and store it in the local database.

We may define sub-agents inside an enterprise as product design agent (PDA), product manufacturing agent (PMA), just-in-time procurement and distribution agent (JPDA), and accounting agent (ACA), which are under the coordination of enterprise coordination agent as shown in Figure 3.

In reality, other than vertically hierarchical relationship of agents as shown in Figure 3, there exists also horizontal relationship among agents in different enterprises. For example, when the host enterprise product design agent PDA is designing products, it may coordinate and negotiate, not only with the product manufacturing agent (PMA), just-in-time procurement and distribution agent (JPDA), and accounting agent (ACA) inside the same enterprise, but also with PDAs in supply and distribution enterprises. Thus, the horizontal coordination among agents can be logically shown in Figure 3.

A methodological approach to model operation phase of VMS

A detailed description of formation and operation phases of VMS is given in the previous section. It is also stated that the XML based data collection system presented for the formation phase is only in the conceptual stage. In this section we present a methodological approach to model the information flow activities in the operation phase of VMS.

The three views of the methodological approach

In this subsection we first present a methodological approach for modeling information integration in VMS.

The methodological approach that is presented here starts with a simple overview of the information infrastructure. There are three major actors dealing with the information infrastructure:

1 The information resources and applications.
2 Networks for mobility of information, and
3 Control or supervisory mechanisms for routing and control of information.

Therefore our methodology too is divided into three interrelated (and overlapping) views (see Figure 4). This division is to reduce complexity and to allow independent and parallel development of components under different views.

The three views are: control; mobility; and component-communication. Figure 4 shows the formation of views of the architecture. Figures 5, 6 and 7 show the detailed exposition of the views.

As shown in Figure 5, in the control view, the middleware is partitioned into two layers of abstraction. The application interface layer represents the connection of the applications and data sources to the integrating infrastructure through agents. In the control layer, there are two central controllers that take basic control actions on agents. These two controllers are the agent manager and the event manager.

The agent manager controls the life cycle of the agents and their actions, and offers relevant services relating to this area. Agent startup, execution context, inter-communication, and persistency are under the agent manager control. The event manager controls the distributed event triggering. The event manager registers the agents that wish to be notified about an event. The event manager notifies these agents whenever an event happens for which the agents are registered.

The underlying communication infrastructure refers to the hardware and the net operating system.

From Figure 5, it is obvious that distributed events paradigm is useful for this situation where applications and data sources are distributed and autonomous. For example, the simpler way for an application to understand the change of state of a data source on the other side is by notification of the change by an event whenever the change occurs.

Figure 6 shows the mobility view. In this view multiple resources on heterogeneous platforms are pooled together as a homogeneous system to serve the mobile

Figure 3
Vertical and horizontal relationship of agents in virtual manufacturing system

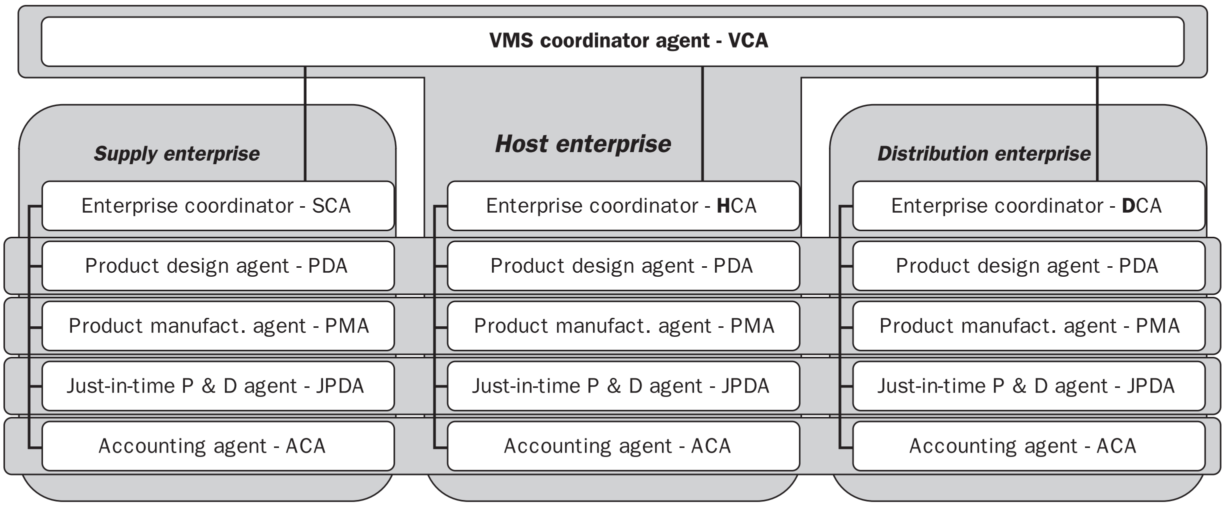
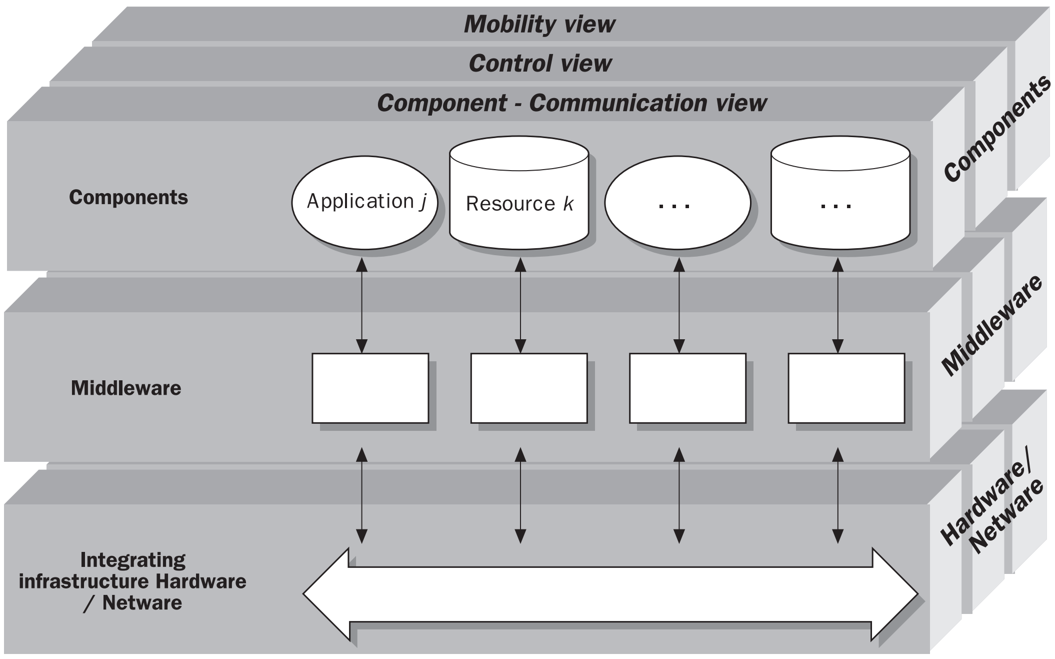
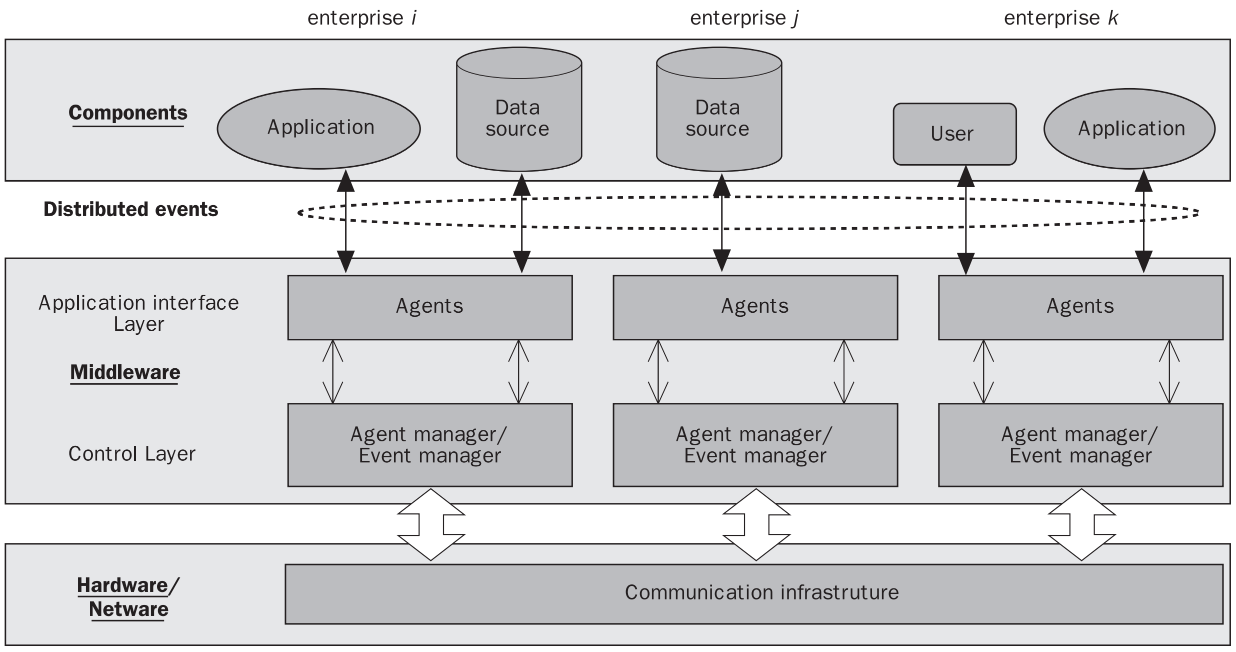

VMS coordinator agent - VCA
Supply enterprise
Host enterprise
Distribution enterprise
Enterprise coordinator - SCA
Enterprise coordinator - HCA
Enterprise coordinator - DCA
Product design agent - PDA
Product manufact. agent - PMA
Just-in-time P & D agent - JPDA
Accounting agent - ACA

Figure 4
Formation of the three views of the methodological approach



agents seeking data from these resources to satisfy the applications which started them. To enable this, resources have to be encapsulated, and the request to them and response from them to the agents has to be transparent of where the resources are located. With CORBA technology the kind of common bus for mobile agent transportation as shown in Figure 6 can be created. To do this, different nodes where diverse applications and resources are attached, should use CORBA object request broker (ORB). Local resources attached to the individual nodes are encapsulated ("wrap-up") by CORBA interface definition language (IDL), and their presence is let known to all the nodes by registering in the local interface repository.

Figure 5
Control view



The coordinator agents (HCA, DCA, and SCA) are classified as wrapper agent too, providing a level of abstraction between a data source (or simply database) and requesting mobile agent. The wrapper agent understands how to access the data source and the permission structures associated with it and wrap-up legacy data for transparent cross-platform access. The host coordinator agent HCA has an additional function too, which is launching (or dispatching) the mobile agent VCA.

Figure 7 shows the component-communication view. This view identifies the components and their communication relationship. There are several kinds of agents, and this kind of differentiation of agents strongly influences the architecture (in VMS, coordinator agents such as SCA, HCA, and DCA, and functional agents such as PDA, PMA, JPDA, and ACA). Also, there are static agents in addition to the mobile agent. In Figure 7, only VCA is mobile.

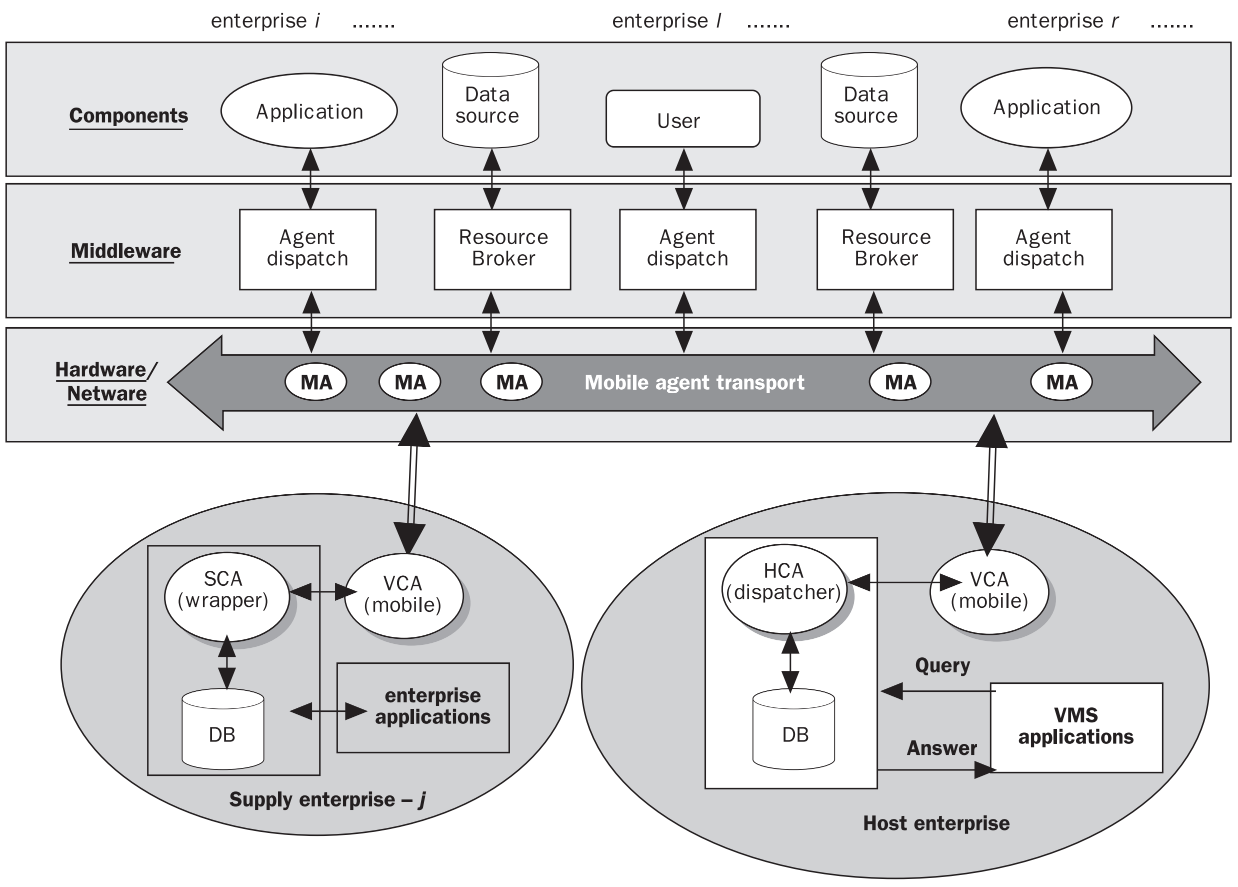
The component-communication view also identifies the agent inter-communication mechanism and their relevance to the communication infrastructure. For example, what are the groups of agents that are collaborating with each other (in both intra- and inter enterprise), what are the vertical and horizontal relationship that exist between these agents, and how to realize these relationships, have to be identified in this view. In Figure 7, the inter-enterprise vertically hierarchical relationship among agents is pictured by HCA on the top of the hierarchy by controlling the launching and directing of VCA, and other coordinator agents (SCA/DCA) in a way, acting as slaves. The horizontal relationship between the functional agents (inter-enterprise) could be realized by exchanging data with the help of mobile VCA.

A testing prototype

Based on the above multi-agent model, a testing prototype was developed in our laboratory. The current version of our prototype under development is implemented with the Concordia agent development software (Concordia, 1998).

In the testing prototype, the whole system's operation is based on the following five asynchronous distributed events:

Figure 6
Mobility view



1 New data (*new_data()*);
2 Change of data (*change_data()*);
3 Privileged update (*privileged()*);
4 Maintenance (*maintenance()*);
5 Refresh data (*refresh()*);

Figure 8 explains the operation of the testing prototype. In Figure 8, the VMS operation phase initialization takes place after execution of the distributed event *new_data()* by an enterprise. After the collaborating enterprises have forwarded their data to the host enterprise by executing *new_data()*, the host enterprise has its local database filled with the data about the collaborating enterprises.

Change of existing data about an enterprise happens similarly; after updating its local database with the new data, the respective functional agent of the enterprise executes the distributed event *change_data()*, which triggers the HCA to start the update cycle similar to that shown in Figure 8.

Privileged update is also similar to change of existing data. The only difference is that, in addition to the host enterprise, the new data are sent to specific enterprises specified by the sending enterprise. Of course, host enterprises always receive any privileged data, as the host enterprise is the master of the collaboration.

During the maintenance cycle, the validity of the data in the host's database is checked. Copy of data about an enterprise is sent to the respective enterprise for verification. This maintenance cycle is triggered by an external real-time driven program or by any applications on the host, after finding invalid data in the database.

Refresh cycle is similar to maintenance; the difference is that this cycle is triggered by an enterprise wanting to make sure that its data in the host's database is valid. In this case, the host sends data to this enterprise only. Refresh is normally triggered after the *new_data()* event.

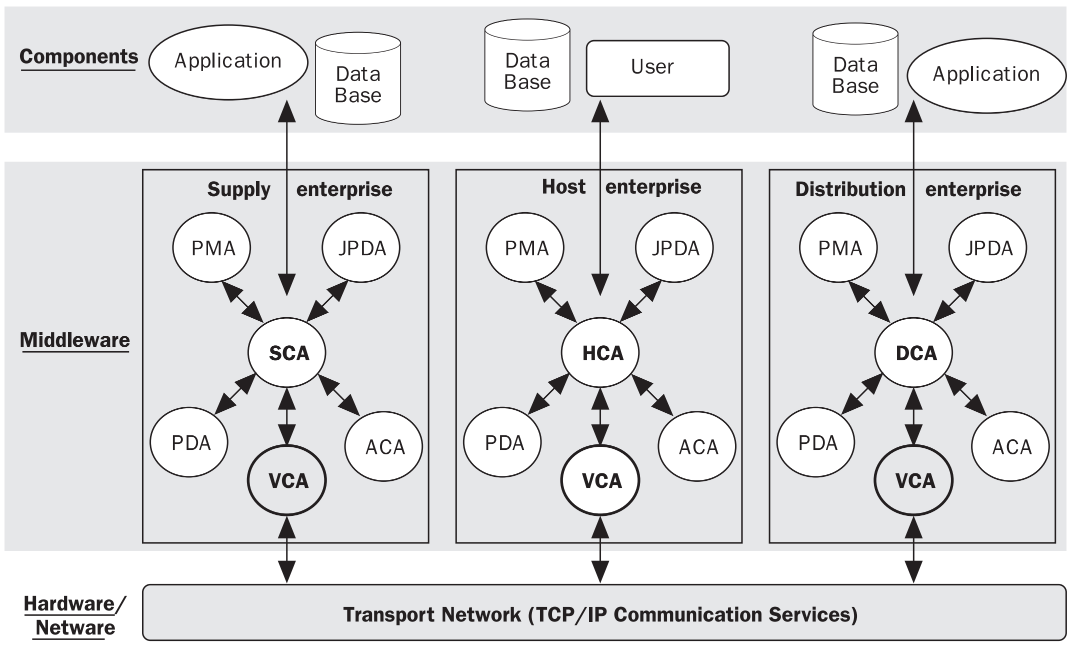
If an enterprise want to distance itself from collaboration for a shorter period, it triggers *new_data()* with empty data. During the next maintenance cycle, the host ignores the enterprise from further collaboration after seeing empty data in its database.

Discussion and further work

In this section, we will discuss some remarks regarding our experiments.

Concordia was chosen to implement the prototype because of its simple programming interface for distributed event management and for its support for agent mobility. Voyager (1999) and Aglets (1999) are the other agent development systems that were considered for implementation of our testing prototype. We considered only the Java based systems because of the well-known pro-Java reasons like platform independency, simplicity and security (Flanagan, 1997); the mobile agent frameworks that are based on other implementation languages (like AgentTcl, which is based on Tcl) were not considered.

Figure 7
Component-communication view

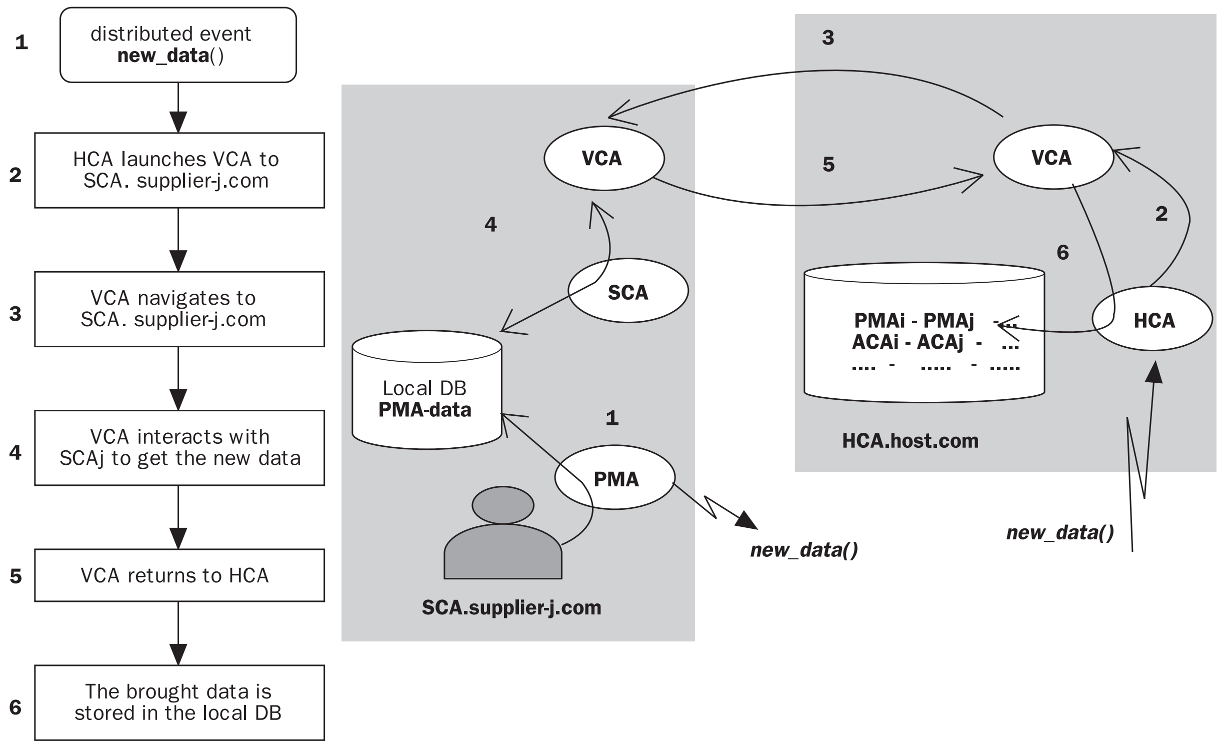


One of the problems that we faced when building our testing prototype was Concordia's weak mobility. The notion "strong mobility" refers to the ability to move the execution state (i.e. program counter, call stack, etc.) in order to give the migrating unit the ability to resume execution right after the migrating instruction. For example, the agent may calculate a part of the computation in one computer and then it may migrate to another computer to calculate the other part of the computation (Fuggetta *et al.*, 1998). This situation may be helpful if the mobile agent (VCA) has to collect (and compute) information from many different sources within a destination (enterprise). In contrast to strong mobility, weak mobility provides only the capability to move code. For our testing prototype, where identical mobile agents are sent to different specific places (destinations) to execute the similar computations or query only on that node, we found that weak mobility was not a serious shortcoming. Voyager and Aglets provide mechanisms for strong mobility.

Security and safety of the three main actors of the prototype, namely, mobile and static agents, distributed event triggering, and data sources, are of major concern. Security and safety measures against tampering and misuse of these three actors must be guaranteed. Concordia provides secure and safe operation by extending standard security mechanism in Java language; An agent while travelling or not, and the data it carries are protected from tampering by transmission protection using secure Sockets Layer version 3 (SSLv3) and storing protection by encryption (Walsh *et al.*, 1998). Data sources and distributed events are protected by means of the Resource permissions. Resource permissions determine if an agent or object may trigger an event or it may receive an event. By enabling appropriate resource permissions, the untrusted agents' actions are controlled (Walsh *et al.*, 1998). By the use of proxies, queuing, and persistence, robust transmission of agents and distributed events are guaranteed (Walsh *et al.*, 1998).

Many argue that problems like information integration should be solved by combining AI (artificial intelligence) and mobile agent technology. AI community provides languages like agent communication languages ACL (KIF and KQML) combined with application-specific ontology which could be used for the problems like information integration. With ACL alone, it will be much more difficult to program agent collaboration. In addition, ACL does not support mobile agents (Wong, 1997). Some others argue that by combining AI and database technology, information integration

Figure 8
Physical processes of integrating information on host enterprise



problems could be best solved (Levy, 1998). We believe that "information integration" problem lies at the intersection of AI, database technology and mobile agent technology. Though we see many research programs on AI-DB and AI-mobile agent area, so far we have not heard about any research program combining these three fields.

Conclusion

How to model and analyze inter-enterprise integration is one of the major concerns in today's enterprise modeling circles. Owing to the complexity and large scale of virtual manufacturing system, it is difficult to pursue an effective modeling methodology for it. This paper attempts to make use of multi-agent technology for this purpose because the agents possess characteristics of autonomy, coordination, distribution, and mobility. So we can implement independent enterprises and their independent functions as agents to be not only autonomous, but also coordinated. Also, even though they are geographically widely distributed, by means of the mobility of agent they are communicative.

The architecture proposed in this paper is based on the integration among small and medium sized enterprises. For the large enterprise, the vertical structure of the agents needs to be extended to possess more levels of hierarchy. Because of a large workload in agent programming, we have only partially implemented the coordinated agents in our testing prototype. Further work is necessary to develop a complete experimental paradigm for exploring the "Plug and Play" inter-enterprise integration to cope with the natural multi-vendor collaboration environment.

References

Aglets (1999), *Aglets Software Development Kit*, http://www.trl.ibm.co.jp/aglets/

Bosak, J. and Bray, T. (1999), "XML and the second-generation WEB", *Scientific American*, May.

Concordia (1998), *Product Information*, http://www.concordia.mea.com/

Deng, Z. (1997), "A model of methodology need for AQAL production system", *Reengineering for Sustainable Industrial Production*, Chapman & Hall, London.

Deng, Z., Bang, B., Laksa, A. and Nadarajah, S. (1998), "A model of enterprise integration and collaboration tools and communication infrastructure for inter-enterprise collaboration", *Globallization of Manufacturing in the Digital Communications Era of the 21st Century*, Kluwer Academic Publishers, Norwall, MA.

Enator (1998), *Virtual Enterprising*, * http://195.100.12.162

Flanagan, D. (1997), *Java in a Nutshell*, 2nd ed., O'Reilly.

Fuggetta A., Picco G. P. and Vigna G. (1998), "Understanding code mobility", *IEEE Transactions on Software Engineering*, Vol. 24 No. 5.

Laufmann, S.C. (1998), "Agent software for near-term success in distributed applications", *AGENT TECHNOLOGY – Foundations, Applications, and Markets*, Springer-Verlag, New York, NY.

Levy, A.L. (1998), "The information manifold approach to data integration", *IEEE Intelligent Systems and Their Applications*, September-October 1998, Vol. 13 No. 5.

Nwana, H.S. and Ndumu, D.T. (1998), "A brief introduction to software agent technology", *AGENT TECHNOLOGY – Foundations, Applications, and Markets*, Springer-Verlag, New York, NY.

Voyager (1999), *Product Information*, http://www.objectspace.com/Products/voyager1.htm

Walsh, T., Paciorek, N. and Wong, D. (1998), "Security and reliability in concordia", *Proceedings of the 31st Annual Hawaii International Conference on System Sciences (HICSS31)*, Hawaii, January.

Wong, D., Paciorek, N. and Walsh, T. (1997), "Concordia: an infrastructure for collaborating mobile agents", *First International Workshop on Mobile Agents (MA'97)*, April, Berlin, Germany.