Using Mobile Agent Technology to Develop a Collaborative Product Lifecycle Oriented Architecture

Khaled Bahloul, Nesrine Darragi, Yacine Ouzrout, Abdelaziz Bouras

Using Mobile Agent Technology to Develop a Collaborative Product Lifecycle Oriented Architecture

Khaled Bahloul[1],Nesrine Darragi[1], Yacine Ouzrout[2]and Abdelaziz Bouras[2]

[1]Univ. Lille Nord de France, F-59000 Lille,
IFSTTAR, ESTAS, F-59650 Villeneuve d'Ascq, France

[2]DISP Laboratory
University of Lyon 2 (IUT Lumière)
160 Boulevard de l'Université, 69676 Bron Cedex – FRANCE

Abstract

Today product development activities are becoming more and more agile, adaptable, and cost-effective. Multi-agent-based system technology has found wide applications in managing company's information. This paper applies Multi-mobile-agent technology in developing a collaborative architecture for product lifecycle. By adopting the Aglets mobile agent platform, a generic platform has been developed for managing legacy product data and information across the product lifecycle. Different lifecycle stage product data and information and their interaction and administration are encapsulated, represented and operated with different agents. The system aims to integrate mobile agents into different locations so as to fully utilize the product lifecycle information, and make the product lifecycle decision satisfying the global target of the company. The architecture and the working principle, framework and implementation of the system are addressed in detail. The model of the mobile agents in the system and the technique to realize them are also illustrated with a real case example. The contribution of this paper is to develop a generic architecture for product lifecycle based on the multiple mobile agent technology, adopt the Aglets mobile agent platform to develop a series of Aglet templates to encapsulate and manage the different legacy product data and production information and integrate them into the product lifecycle context. It cannot only simulate real product lifecycle activities from design to recycling but also provide an environment for managing lifecycle product information.

Keywords:*Mobile Agent, Product Life Cycle, Collaboration, Multi-Agent Systems.*

1. Introduction

Today, challenges faced by product development teams include globalization, outsourcing, mass customization, fast innovation and product traceability, etc. These challenges enhance the need for collaborating environments and knowledge management along the product lifecycle stages. To meet such new requirements, a generic architecture or framework is required to encourage participants to share new ideas, methodologies, techniques, systems, tools, languages, and best practices. The rapid progress of emerging information and network technologies such as Internet and Intranet is driving a revolution of product design and manufacturing patterns for companies. Networked or collaborative product design and manufacturing as an emerging product development paradigm is becoming the subject of research and practice interests.

Numerous efforts have been made to improve the management of product lifecycle information. A variety of information systems and networks working within and between product lifecycle and its supporting activities have been developed to facilitate the flows of information. However, there is a bottleneck, that is, how to develop the networked interfaces for legacy product data and information and resources including legacy software, CAx/DFx [1] and information sharing/exchange. Some limitations are witnessed, such as, systems lacking flexibility, program code maintenance and system security, are not easily realized. There is also which of coordination and integration between these systems. To solve this problem, there two different schemes currently exist [Zhou and Jiang 2005]:

1) Developing new network-centric or shared information models;

2) Developing network-centric interfaces for legacy product information model and manufacturing/ production resources.

However, there are two reasons hindering the putting on the practice on the first scheme. One is that a great deal of legacy manufacturing resources have to be discarded if we want to use new network-centric manufacturing resources to replace old ones. Another is that such new manufacturing resources need much stronger supports of manpower, material resources, financing and techniques. As a transitional solution, the second scheme has been the subject of broad research and practice in recent years.

[1]The acronym "CAx" is sometimes used as a generic term for the various "computer-aided" systems used in manufacturing industry. DFx means "Design for x"

Through adopting APIs to modify and standardize the networked interfaces, the legacy manufacturing resources can be easily integrated into the networked manufacturing systems. Currently, some middleware technologies, such as CORBA, DCOM and multi-agents, have been applied to implement these kinds of APIs, with great success [Zhou and Jiang 2005].

A Multi-Agent System (MAS) is defined as a loosely coupled network of agents that interact to solve problems beyond the individual capabilities or knowledge of each problem solver [Lesser and Durfee 1989]. MAS can be used to model or actually perform tasks in a collaborative product lifecycle management context or support systems due to the similarities of the nature of these two systems. In fact, to make the product development activities become more and more agile, adaptable, and cost-effective, the MAS technology has now been widely used in managing product lifecycle information for modern companies. A mobile agent is defined as a specialized kind of agent which can move itself from one site to another by keeping its essential properties and can take the current memory status with it during its dislocation [3]. The mobile agent technology is an emerging technology whose concept comes from the area of MAS / distributed artificial intelligence [15] [8]. It has the potential to provide a convenient, efficient and robust programming paradigm for distributed applications, even when partially connected computers are involved. The mobile agent technology now attracts many interests from the fields of distributed systems, mobile computing, and electronic commerce, among others. In these settings, our research in this paper aims to propose a Multi-Mobile-Agent (MMA) based solution (architecture) to benefit from advantages of the MAS that represent a new way to analyze, design and implant systems of complex product lifecycle data/information processing as well as the mobility, communication, autonomy and cognitive capabilities of agents.

This paper aims to develop a framework based on the multiple mobile agent technology for collaborative product lifecycle. The contribution of this paper is to adopt Aglets mobile agent platform to develop a series of Aglet templates to encapsulate the different legacy product data and information and integrate them into the product lifecycle context.

The paper is structured as follows. Sections 2 and 3 provide an overview of the MMA technology and propose its application for collaborative product lifecycle. Section 4 proposes MMA architecture for collaborative product lifecycle (CPLC). Section 5 gives more details on the implementation and validation of the proposed MMA architecture for CPLC. Section 6 provides a case study to illustrate and demonstrate how to apply the proposed architecture to the case of a Wagon. This case study presents a generic multi– mobile agent architecture simulated in the PLM context of Wagon. Section 7 concludes the paper and some perspectives are also given.

2. Basic Concepts on Multi-Mobile-Agent System Technologies

In our research context, an agent is considered as a piece of software with the following basic properties [Christoforos et al., 99]: 1) autonomy- an agent can act and control its actions by itself, 2) ability to interact-an agent can communicate with information sources and other agents to exchange information, and 3) reasoning ability - an agent has some means for decision-making about how to solve a problem. These three essential properties make the difference between an agent and a normal computational process. Agents can be dynamically organized based on a control or connection structure. Different types of agents may represent different objects, with different authority and capability, and perform different functions or tasks.

A MAS, consisting of a agent's group taking specific roles within an organizational structure [Durfee and Lesser 1989], is based on the notion that the system intelligence is distributed among a set of agents which cooperate with each other to coordinate their actions and objectives so that problems can be solved. The MAS technology can provide methodologies and techniques for modelling complex, distributed and cooperative domains. It has been applied in many domains, such as [Rabelo et al. 94] [Jennings 94] [Fischer et al. 96] [Spinosa et al. 97]. Four important characteristics are often mentioned as a rationale for adopting the MAS technology [Bond and Gasser, 1988]:

1) Each agent has incomplete information or capabilities for solving the problem with a limited viewpoint;

2) Data, control, expertise, or resources are inherently distributed;

3) The system is naturally regarded as a society of autonomous cooperating components; and

4) The system contains legacy components interacting with other components.

MAS and mobile agents have been widely used in Supply Chain Management (SCM). Long et al. published a study with focus on modeling and distributed simulation in Supply Chain, assisted by MAS [Long et al., 2011]. Cheng et al. presented a system of mobile agents ensuring the order tracking to customers in a supply chain. The aim of the mobile agent's use is to provide a data exchange platform [Cheng and Wang, 2008]. Mobile network agents are programs that can be dispatched from one computer and transported to a remote computer for execution. Arriving at the remote computer, they present their credentials and obtain access to local services and data. The remote computer may also serve as a broker by

bringing together agents with similar interests and compatible goals, thus providing a meeting place at which agents can interact. Specifically, a mobile agent is a process that can migrate from one computer to another during its execution. It possesses the facility to move while carrying away its code, its data, as well as its state of execution. Mobile agents can also communicate with each other, clone, merge, and coordinate their computations. Mobile agents are autonomous agents in the sense that they control their relocation behaviour in pursuit of the goals with which they are tasked. These properties are additional reasons that make mobile agents a good candidate to be used in a loosely coupled network environment. The programming for mobile agents is a complementary paradigm for the programming of distributed applications. There are seven good reasons for using the mobile-agent technology [Harrison et al. 00]:
1)Reduction of the network load: The agent can work locally and also transports its program and its information (state and properties).
2)Surmounts the latency of the network: While executing itself locally, the agent avoids latency.
3)Encapsulation protocols: The agent can establish a channel to implant a protocol ownership in a distant execution environment.
4)A synchronous and autonomous execution: While the network is breaking down or becoming slow, the agent does not have to wait.
5)Dynamic adaptation: The agent is reactive and can reconfigure itself on a network
6)Heterogeneous: The agent is independent on the platform and it only depends on the environment of execution.
7)Robust and tolerant to failings: At the time of closing of the plots, when warned before, agents can move

A mobile agent platform is a development environment that offers APIs permitting the development and management some mobile agents capable of evolving in a distributed environment. Generally, a development platform offers:
1)Some primitives permitting the managing the mobility of agents (i.e., transfer from one node to other);
2) The primitive bound to relative environment releases to the transfer of agents (arrived on a node, departure of a node);
3)The primitive of communication adapted to the distributed environments: synchronous and asynchronous communication between agents.

The most advanced features include technologies such as Java permitting the development of agents in the heterogeneous environments. Two mobile agent types exist: portable and importable agents. The portable agents are the flat mobile agents that are Java based, in which there is no problem of portability of one operating system to the other, and the importable agents are the flat mobile agent that are not Java based in which there are therefore problems of portability. Here, we present a set of features that are applicable in the choice of the platform. General features of the flat mobile agents including programming language of implementation, protocol network are [Papastavrou et al., 99]:
1)Lifecycle of the agent
2)Mobility of the agent (e.g., control of the migration)
3)Communication of the agent (e.g., communication from afar, synchronous, asynchronous….)
4)Functionalities of security (e.g., internal security mechanism…)
5)Nature of the license (e.g., free, commercial software...)

Here are some flat mobile agent platforms that are widely used and / or the best known (the list is not exhaustive):
1)Aglets Software Development Kit (IBM Japan),
2)Grasshopper of IKV++ (Germany, Java based) [Magedanz et al., 00],
3)Odyssey of General Magic (USA, Java based),
4)Voyager of ObjectSpace (USA, Java based),
5)Agents of the Darmouth College (USA, non-Java based),
6)OpenCybele of Intelligent Automation Incorporated (USA, Java based).

3. Overview of the Multi-Mobile-Agent Based Collaborative Product Lifecycle

In the literature reviews, a product consists of three separate dimensions that when combined, form the finalized product. In order to actively explore the nature of a product further, those three different views should be considered: the core product, the actual product, and finally, the augmented product. These are known as the three levels of a product [Kotler, 88] [Khosrowpour, 99].

- The core product is not the physical product; it represents the benefit of the product that makes it valuable to the customer.
- The actual product is the tangible, physical product.
- The augmented or increased product is the non-physical part of the product. It may contain information and detailed representations of the actual product.

In this context, the consideration of an intelligent product capable of embedding and managing all or part of its information, in order to launch and organize its own manufacturing or simply to enable access to its informational items, is probably one of the most promising approaches of the last decade. The paradigm of an improved product able to embed information is a first step towards the evolution of core products.

During the last decade, the concept of «Intelligent Product» has been widely used in manufacturing systems.

This concept is often used to indicate a product equipped with a specific technology that enables it to develop certain capacities. Indeed, according to [Wong et al., 04] an intelligent product possesses all or some of the following features:

1. A unique identity
2. The capacity of communication with its environment
3. The capacity to retain or store data about itself
4. The capacity to use a language to display its features, requirements etc.
5. The capacity to participate in or make decisions relevant to its own destiny

Using these features, [Wong et al., 04] categorizes product intelligence in two levels:

Level 1: information oriented intelligence that enables the product to be conscious of its status, and to be able to communicate about it. This level of intelligence is ensured by the first three features of the previous list.

Level 2: In this case the product is supposed to be able to influence and control operations related to its manufacturing. This level of intelligence is said to be decision-oriented. To achieve this level of intelligence the product should ensure all features of the list below.

Jiao et al. proposed a system paradigm of mobile agent collaboration, including the implementation of a negotiation system for global manufacturing Supply Chain [Jiao et al., 2005]. Mahdjoub et al. suggest a knowledge management approach based on an expert knowledge agent to facilitate the design process [Mahdjoub et al. 2009]. On the other hand, Xu provides a cloud manufacturing system inspired from cloud computing to ensure scalability of resources and the complexity of manufacturing processes [Xu, 2011]. The use of mobile agent is to support the inter-connexion and the intercommunication with CAx interfaces.

Through Product Lifecycle Management (PLM) we gain access to managing and acting upon the gathered data throughout the subsequent lifecycle phases. However, the data flow must not remain unidirectional; it must also flow from the design phase and production phase downstream through the usage phase to the end of life phase, closing the information loops across the entire product lifecycle. This allows for access to relevant information at later stages of the product lifecycle, enabling the optimisation of downstream processes such as operation, maintenance or repair. By this means, the total cost of product lifecycle and the ownership of legacy data and systems can be reduced.

Product lifecycle management or support systems (PLMS/PLCS) inherently have domain characteristics of the MAS technology. The virtual enterprise (VE) concepts can even impose changes in the way of a company normally operates, i.e. from an isolated, non-cooperative and selfish to a largely open, cooperative and democratic way of working. Suitable information technology (IT) with true partnership is one of key aspects to establish a successful VE and, hence, to facilitate the collaborative product lifecycle activities. To better exploit the potentials of an intelligent agent approach or the MAS technology within a product lifecycle context, the following requirements need to be satisfied [Hardwick et al. 1996, Rabelo et al. 1997, Zweben 1996]:

1. Information and company integration: A multi-agent-approach requires an efficient information management mechanism and flow to support a very intensive information exchanging process. The more an enterprise has its information suitably modelled and integrated in all the involved product lifecycle activities - the more efficient and reliable that process tends to be.
2. Standards for communication: An efficient PLM or PLCS system requires companies to exchange information with each other, with much more relevant aspects when the mobile-agent-based approaches are involved. However, different companiescan use different information technologies. Thus, the use of standards is an essential requirement to make possible a more "direct" communication among them, both in terms of code and semantics.

To achieve these issues, a product-centric information management approach is employed in this research. A Product Avatar [Cassina, 2008] is an abstraction of a product. It is the virtual representation of a physical product. The Product Avatar concept is proposed to provide a suitable mechanism for the establishment of architecture management information, supporting the product-centric perspective described above, closing the information loops and enabling a bidirectional information flow. The Product Avatar's ability to participate and make decisions or to operate upon information about [Cassina et al. 2009] ensures itself functionalities. The Product Avatar denotes a concept where individual products are viewed as physical entities which are exclusively coupled with digital entities representing them throughout their entire life-cycles. To develop such a product-centric information management system, a multiple mobile-agent technology is proposed to solve the problems identified above and to develop a collaborative product lifecycle architecture and framework.

The basic working scenario of the proposed mobile agents approach/technology for product lifecycle is as follows: Once the product lifecycle (from its concept to disposal) and its mission are formed, the administrator of the product lifecycle creates the lifecycle stage mobile agents according to its submissions and subtasks, such as design agent, production agent, maintenance agent, and recycling agent (see Section 4 below), imposes and instructs them about the «mission» and dispatch them to the product lifecycle stages in order to perform or end their

missions or tasks in the lifecycle. The modeling for a generic mobile agent is described below. Details of modeling and implementation for product lifecycle mobile agents will be described in Sections 4 and 5.

Normally, a mobile agent life cycle is composed of five general phases [Rabelo et al. 1997]: creation, mission programming, dislocation, mission execution and mission ending. The mission ending in turn involves two actions which can be performed after the execution of its mission: the mobile-agent (self-) killing or its return to the site which has originally launched it. The modeling of a mobile-agent can use the object-oriented modeling and programming paradigm. This means that the mobile-agent knowledge can be represented in terms of its attributes and methods. The attributes of the mobile agent object indicate its characteristics, including the identification, the site of origin, the restrictions of time to perform its mission, the specification of the access rights at the explored site, the high level communication protocol ontology. The methods of the mobile agent object describe the functionality by the agent the mission programming of the mobile agent can be carried out regarding the type of its mission, which means that different classes of missions can be configurable or can be developed along time. In the proposed approach, creating mobile agents means instantiating the agent model according to the coordinator's objectives or the missions of the mobile-agents and their related characteristics. Once an agent finishes its mission it can either kill itself at its own site or it can return to the administrator/coordinator site. This behaviour also depends on the agent's mission. Many mobile-agents may be located temporarily at a certain site. Depending on its mission, the mobile agent may move itself to other companieswith the information it had in the previous site. This is especially useful when more complex decisions in which a global view upon the product lifecycle is necessary.

4. Proposed Multi-Mobile-Agent Architecture for Collaborative Product Lifecycle

In this section, we discuss the proposed mobileagent-based collaborative product lifecycle architecture. In the product lifecycle context, many "services" can take advantage of mobile-agents, especially when they are programmed to play as information providers (push technology). These agents are listed below:

1) Administrator agent;
2) Product agent;
3) Concept/design agent;
4) Production agent;
5) Maintenance agent; and
6) Recycling agent.

The proposed architecture focuses on the product life-cycle and its decomposed phases which are considered the most interesting, such as design, production, maintenance and recycling phases.

This architecture allows reducing the interactions between these different phases and centralizes data on same products. In this context, it is necessary to put forward a methodology and framework for shaping the product and its development during these phases. Our objective is to put the product in the focus of the architecture and to take into account the necessary resources. The notion of the Product Avatar in Section 3 above can provide a suitable mechanism for the foundation of the information management architecture. We shape each enterprise as an agent; and each agent takes the information concerning a phase in the product lifecycle. Equally, we define the product agent in terms of information concerning its structure from one phase to another. This agent is the focus of the architecture; it embeds the information and defines even the notion of the Product Avatar. Finally, we use the administrator agent to create of all other agents, each of which is in its machine following well defined URL addresses. This agent does not play any role after the creation of the other agents. During its life-cycle, the product passes through different phases: from the design to the recycling. The necessity to present a collaboration environment encourages exchange and communication between different actors concerned by those phases. The proposed architecture guarantees a communication level that assures the exchange between the different agents and the agent producers. Interactions between these agents are represented in the collaboration diagram. Figure 1 shows an overall MMA architecture and framework for collaborative product lifecycle. It can be divided into four functional units: client sites, mobile agent servers or runtime, mobile agent templates and legacy product information and manufacturing resources.

To help us understand the usage of the architecture in the product lifecycle, a UML use case diagram, as shown in Figure 2, is used to analyze the MMA architecture and possible scenarios occurring for the collaborative product lifecycle. The icon of a man is a symbol as the actor for an agent, and the ellipse diagram represents a use case. From the use case diagram, five agents as actors are mainly involved in collaboration of the product lifecycle architecture. The system can save a lot of necessary product information and system states, which are available for product lifecycle management or support systems.

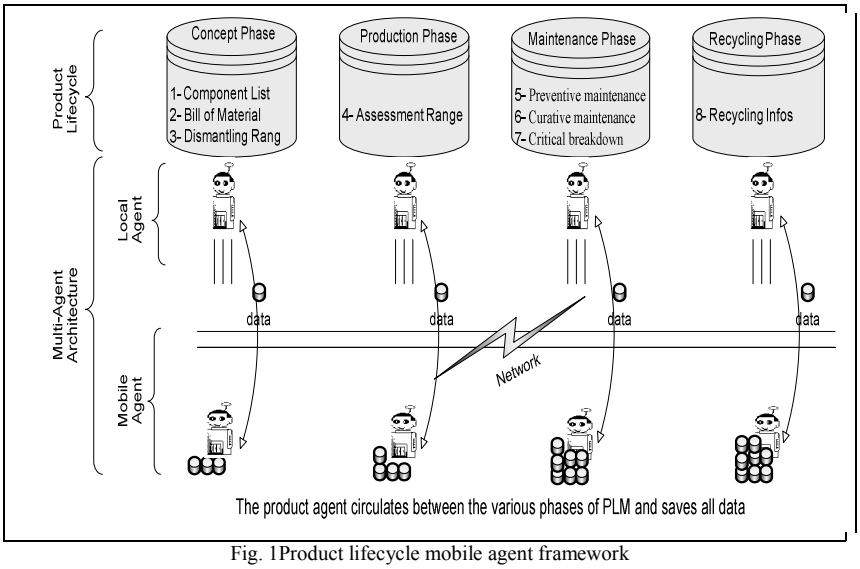

Fig. 1Product lifecycle mobile agent framework

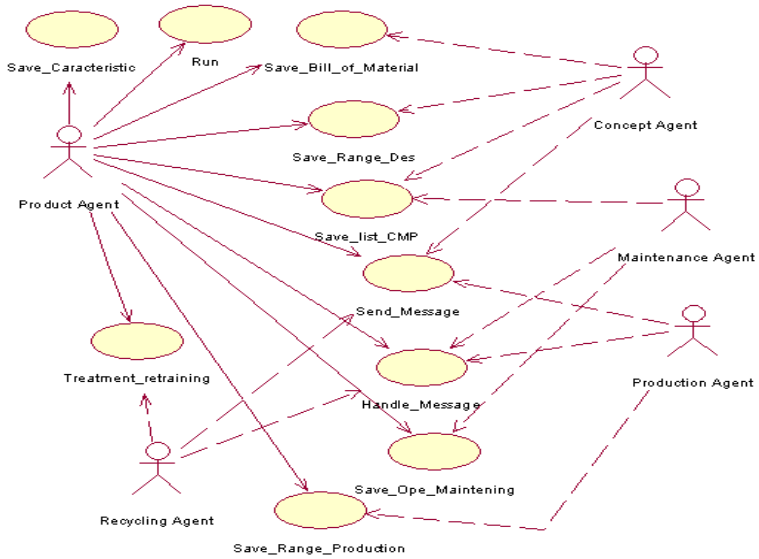

Fig. 2 Use case diagram for the global architecture

5. Implementation

The implementation of a prototype MMA system is based on the IBM *Aglet* mobile agent platform as a collaborative environment. In this section, we first give an overview of the *Aglet* platform, and then provide more details on the implementation of the proposed architecture in the *Aglet* platform.

5.1. Overview of the Aglet Platform

*Aglet*s Software Development Kit is a free platform from (IBM), It offers an environment for programming mobile agents in JavaTM over the internet. An A*glet* represents the next development forward in the evolution of executable content on the Internet, introducing program code that can be transported along with the information state. *Aglet*s are Java objects that can move on the Internet from one host to another, i.e. an *Aglet* that executes on one host can suddenly halt execution, dispatch itself to a remote host, and resume execution there. When the A*glet* moves, it takes along its program code as well as its data and the states of all the needed objects. A built-in security

mechanism makes it safe to host entrusted A*glets*. The goals of the system are:

1) To rovid an easily comprehensive model for programming mobile agents without requiring modifications of the Java VM or native code.
2) To support dynamic and powerful communication that enables agents to communicate with unknown agents as well as well-known agents.
3) To design a reusable and extensible architecture.
4) To design a harmonious architecture with existing Web/Java technology.

The *Aglet*s architecture consists of two APIs (*Aglet* API and *Aglet*s Runtime Layer - The implementation of *Aglet* API) and two implementation layers (Agent Transport and Communication Interface and Transport Layer)
The *Aglet*s runtime layer is the implementation of *Aglet* API, which provides the fundamental functionality such as creation, management or transfer of *Aglet*s. This layer defines the behaviour of APIs such as *Aglet* and *AgletContext*, and can serve multiple *AgletContext* objects.
The transport layer is responsible for transporting an agent to the destination in the form of a byte stream that contains class definitions as well as the serialized state of the agent. This layer is also defined as an *API*, called Agent Transfer and Communication Interface (ATCI), which allows the *Aglet*s runtime to use the transport layer in a protocol-independent manner. The implementation of *ATCI* is responsible for sending and receiving an agent and establishing a communication between agents. The current Aglets implementation uses the Agent Transfer Protocol (*ATP*), which is an application-level protocol for transmission of mobile agents. ATP is modeled on the *HTTP* protocol, and can be used to transfer the content of an agent in an agent-system-independent manner. To enable communication between agents, ATP also supports message-passing.
When an aglet issues a request to dispatch itself to a destination, the request travels down to the Aglets runtime layer, which converts the *Aglet* into the form of a byte array consisting of its state data and its code. If the request is successful, the *Aglet* is terminated, and the byte array is passed to the *ATP* layer through the *ATCI*. The *ATP*, which is the implementation of *ATCI*, then constructs a bit stream that contains general information such as the agent system name and agent identifier, as well as the byte array from the Aglets runtime.

5.2. System Implementation Architecture

We use the **Aglet** Java-Based internet agent to deploy our architecture. The proposed architecture is based fundamentally on four phases: the design phase, the production phase, the maintenance phase and the recycling phase. Relations between classes of the architecture are presented in the class diagram, as shown in Figure 3. Interactions between different objects involved in our system are represented in UML sequence diagrams.

(1) Administrative Agent

The administrative agent (Figure 4) is able to:
a) Create each agent in its machine,
b) Capture features of the product as the name, category, mark, identifier and date of creation. Users must have already chosen the address URL among an existing list. Such as:

- The Aglet is composed of several attributes and methods.
- The attributes ensure the utility of Aglet in the platform and characterize it while specifying:
- its identifier (myID: AgletID),
- its proxy (Proxy_Ag_Admin: Agletproxy)
- its context (Contexte_Ag_Admin: AgletContext)

The principal methods defining the events of Aglet:
run ():for the execution of Aglet
HandleMessage (): for the receiving messages starting from other Aglets
SendMessage ():for sending messages towards other Aglets
Dispatch_Recycling_Agent():to dispatch the Recycling Aglet
Dispatch_Concept_Agent():to dispatch the design Aglet
Dispatch_Production_Agent(): to dispatch the Production Aglet
Dispatch_MaintenanceAgent(): to dispatch the Maintenance Aglet
Dispatch_Product_Agent(): to dispatch the Product Aglet

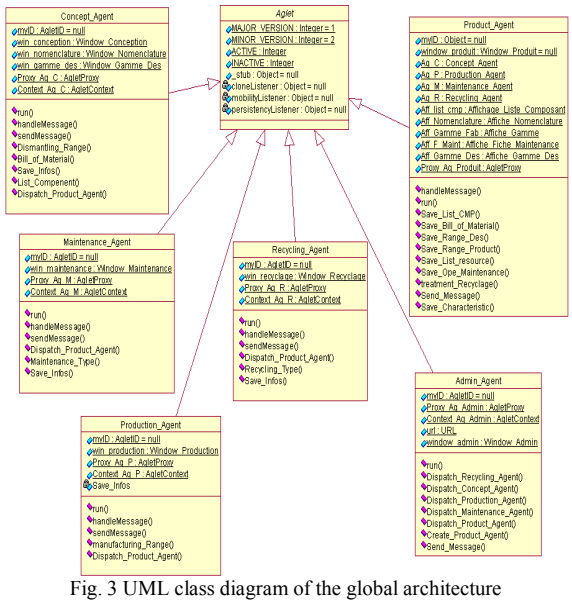

Fig. 3 UML class diagram of the global architecture

(1) Administrative Agent

The administrative agent (Figure 4) is able to:
a) Create each agent in its machine,

b) Capture features of the product as the name, category, mark, identifier and date of creation. Users must have already chosen the address URL among an existing list. Such as:

- The Aglet is composed of several attributes and methods.
- The attributes ensure the utility of Aglet in the platform and characterize it while specifying:
- its identifier (myID: AgletID),
- its proxy (Proxy_Ag_Admin: Agletproxy)
- its context (Contexte_Ag_Admin: AgletContext)

The principal methods defining the events of Aglet:
run ():for the execution of Aglet
HandleMessage (): for the receiving messages starting from other Aglets
SendMessage ():for sending messages towards other Aglets
Dispatch_Recycling_Agent():to dispatch the Recycling Aglet
Dispatch_Concept_Agent():to dispatch the design Aglet
Dispatch_Production_Agent(): to dispatch the Production Aglet
Dispatch_MaintenanceAgent(): to dispatch the Maintenance Aglet
Dispatch_Product_Agent(): to dispatch the Product Aglet

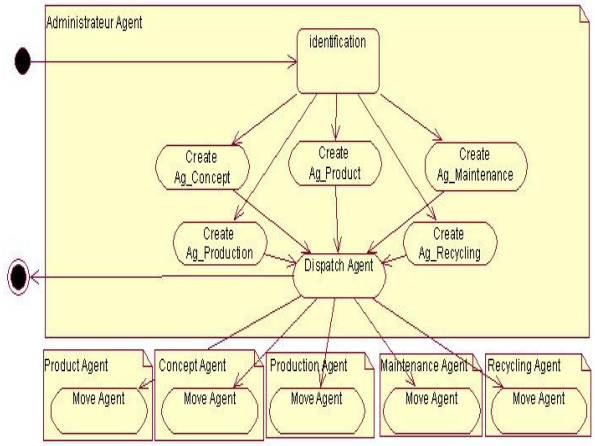

Fig. 4UML State/Activity Diagram: Administrator Agent

(2) Conception Agents

All the activities performed by the design agent are presented in the activity diagram (Figure 5). It allows us:
a) To enter, initially, the component list while specifying features (name, identifying, type, mark) of every component,
b) To provide the definition of the component nomenclature while choosing the component among the list, assigning a design, and then defining the list to be composed,
c) To define the dismantling range which enables the capture of the data that will be useful in the phase of retaining the information about the dismantling range of a component through a very definite operation following a precise type,
d) To register of information,
e) To send information toward to product agent,
f) To allowed the product agent towards to migrate the next phase (e.g., the production phase) while carrying all information.
The Aglet of the design agent is composed of several attributes and methods. The attributes ensure the utility of the Aglet in the platform and characterize it while specifying:

- its identifier (myID: AgletID),
- its proxy (Proxy_Ag_C: Agletproxy)
- its context (Context_Ag_C: AgletContext)

The principal methods defining the events of the Aglet include:
Dismantling_Rang (): to save the range of dismantling of the product
Bill_of_Material (): to save Bill of Material of the product
List_Component (): to save the list of the components of the product
Dispatch_Product_Agent (): to dispatch the product agent towards the next phase in the product lifecycle.

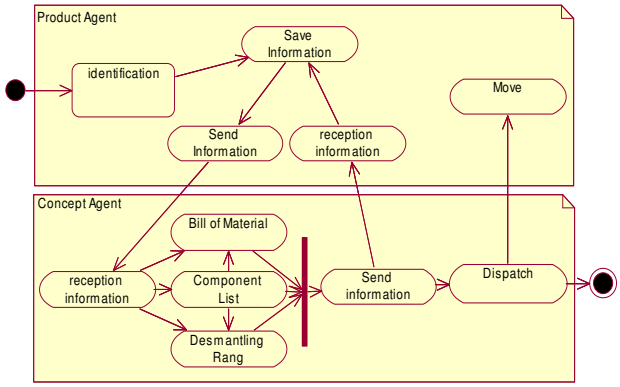

Fig. 5 UML Activity Diagram: design agent

(3) Production Agent

Based on information saved in the product agent, the production agent allows us:
a) To define the range of manufacture of the product while choosing components treated among the list of components in every operation, in addition to the machine and the period of treatment,
b) To register these operations,
c) To send information towards the production agent, and it also allows
d) The agent's migration towards the next service (the maintenance service) while carrying all the information.
The Aglet is composed of several attributes and methods. The attributes ensure the utility of the Aglet in the platform and characterize it while specifying:

- its identifier (myID: AgletID),
- its proxy (Proxy_Ag_P: Agletproxy)
- its context (Contexte_Ag_P: AgletContext)

The principal method defining the events of Aglet:
Manufacturing_Range(): to define and save the range of manufacturing

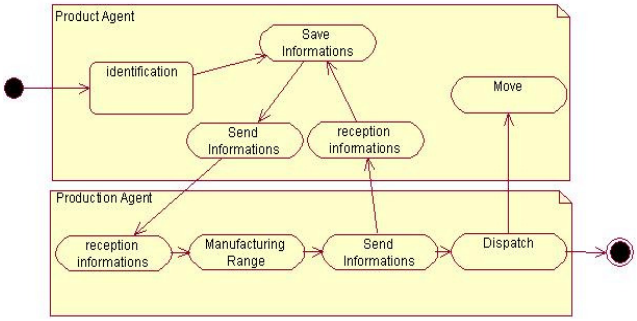

Fig 6 UML Activity diagram: Production Agent

(4) Maintenance Agent

The maintenance agent (Figure 7) creates the document of maintenance while taking information saved on the product as a basis. This agent allows: a) the capture of the date of the maintenance operation, b) the definition of the operation type: preventive maintenance, curative maintenance or critical breakdown, c) the selection of components treated in every operation , d) the capture of a commentary for the description of the operation , e) the safeguard of this information, f) Sending this information towards the agent producers, and g) The agent's migration towards the next service, while carrying all the information. Note: in case of breakdown the maintenance agent must send a message of information toward the production agent.

The Aglet is composed of several attributes and methods. The attributes ensure the utility of Aglet in the platform and characterize it while specifying:

- its identifier (myID: AgletID),
- its proxy (Proxy_Ag_R: Agletproxy)
- its context (Contexte_Ag_R: AgletContext)

The principal method defining the events of Aglet:
Maintenance_Type(): to define and save the range of maintenance; in this operation the Aglet needs to have information in Aglet Product.

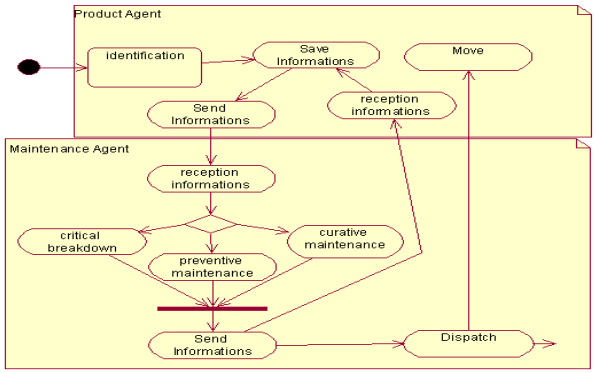

Fig. 7 UML Activity Diagram: Maintenance Agent

(5) Recycling Agent

In the last phase (recycling phase), the recycling agent consult all information (Figure 8); in addition it allows: the capture of a commentary on the history of the product allows us to define the definite ranges of disassembling.
The Aglet is composed of several attributes and methods. The attributes ensure the utility of Aglet in the platform and characterize it while specifying:

- its identifier (myID: AgletID),
- its proxy (Proxy_Ag_R: Agletproxy)
- its context (Contexte_Ag_R: AgletContext)

The principal method defining the events of Aglet:
Recycling_Type(): to define and save the type of recycling, in this operation the aglet needs to have information for the Aglet Product.

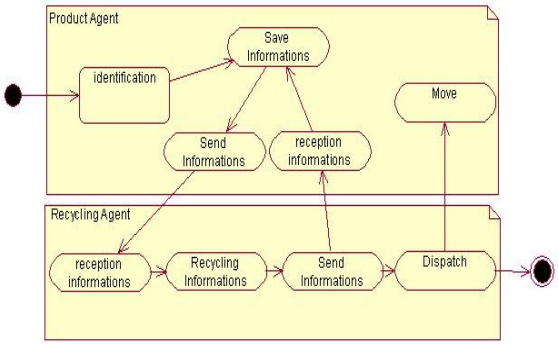

Fig 8UML Activity Diagram: Recycling Agent

(6) Product Agent:

The product agent is the most important agent in the chain, which is a carrier of information and moves according to the other demanding agent. The collaboration diagram (Figure 9) shows that the agent's displacement produces localization to another phase (service) while following events (Dispatch (URL)) of the other agents. It also shows the exchange of messages between the agents; these messages are essential information and data collected constitute the product itself.

The Aglet is composed of several attributes and methods. The attributes ensure the utility of Aglet in the platform and characterize it while specifying:

- its identifier (myID: AgletID),
- its proxy (Proxy_Ag_R: Agletproxy)
- its context (Contexte_Ag_R: AgletContext)

The principal methods defining the events of Aglet:

Save_List_CMP():to save the component list from the Aglet design

Save_Bill_of_Material():to save the component list from the Aglet design

Save_Range_Des(): to save the component list from the Aglet design

Save_Range_Product(): to save the range of production from the Aglet Production

Save_List_resource(): to save the component list from the Aglet design

Save_Ope_Maintenance(): to save the range of maintenance from the Aglet Maintenance

treatment_Recyclage():to save the type of recycling from the Aglet Recycling

6. Case Study

The validation of the proposed architecture was done on the basis of an industrial case study (Figure 10). We begin with the definition of the Administrator Agent. Then, we define other agents which represent the phases of the product lifecycle: Design Agent, Production Agent, Maintenance Agent and Recycling Agent. Each agent is created with a specific IP address.

After creating the four synchronous agents, we create the product agent, on the same IP address as the design agent. At the beginning of the product lifecycle, we define the components, the bill of material, the dismantling range, etc. The product agent saves all data and changes its location to move towards the next phase of the lifecycle regarding the new IP address. In the production phase, we define the range of production based on the saved data in the product agent. Similarly, the product agent saves the data and changes its location from a phase to another for the following phases.

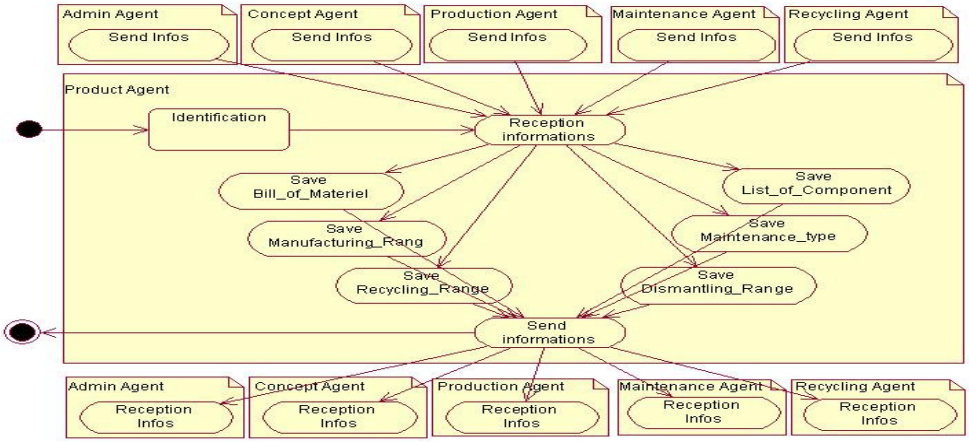

Fig 9UML Activity Diagram: Product Agent

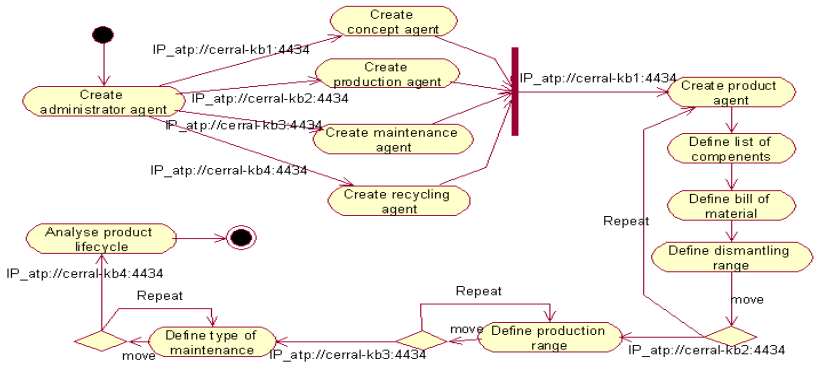

Fig. 10 Case study activity diagram

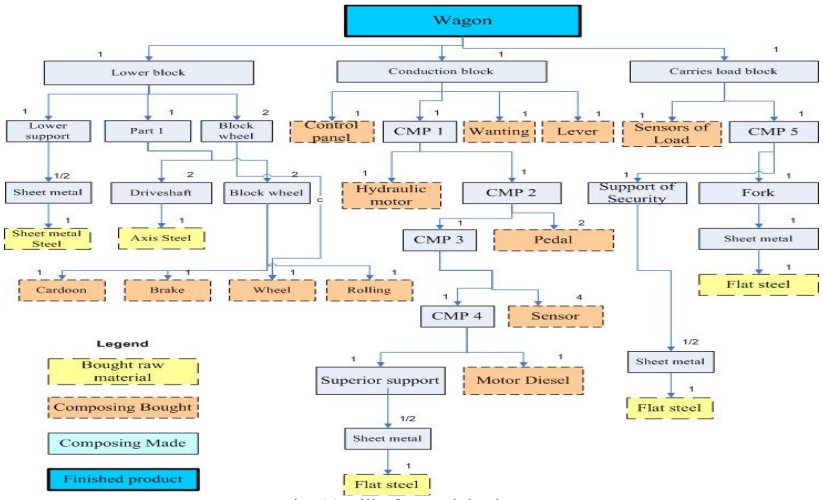

Fig. 11 Bill of materiel schema

Based on the proposed architecture in Section 3 and the choice of the Aglets platform in Section 4, we present in this section the application of the proposed architecture to the of the wagon case. It presents a generic MMA architecture simulated in the PLM context.

6.1. Wagon Lifecycle Analysis

In order to describe the product "Wagon" we present the information defined in the following various phases.

(1) Conception phase: It is the period of time in the product lifecycle during which the conceptual designs and data are created, documented, and verified to satisfy the customer's requirements.

ORTEMS tools [ORTEMS, 2011] have allowed us to develop a unique line of Supply Chain planning, production and scheduling software. Based on these tools, we give the following description of technical data which describe the composition of the product. This configuration is presented in Figure 11.

Dismantling task: it is to define the operations to recycle the parts at the end of the product lifecycle.

(2) Production phase: It is to define the technical data which describe the mode of manufacturing of the product. Table 1 gives a list of the scheduled operations necessary to carry out a part, a finished product or a sub-assembly, working station and make-ready time and unit of execution per operation, management of the validities.

(3)Maintenance Phase: The maintenance is defined as a set of actions to maintain or to re-establish a good condition in a specified state or in measure to assure a determined service. Three types of maintenance are: a) Preventive maintenance, done according to a bill book established according to the time or the number of use units, b) Curative maintenance, a policy of maintenance that corresponds to an attitude of reaction to the more uncertain events and that applies after the breakdown, and c) Critical breakdown, which can cause works of renovation, reconstruction or repair.

(4) Recycling phase: Recycling is a process, in which the materials making up a product at the end of the lifetime are re-used totally or partly. They are thus reintroduced into the cycle of production from which the product results.

Table 1: The manufacturing range [ORTEMS]

The wording Operation	Machine	Code Operation	Composed	Component	Time
Assembly Wagon	Assembly	ASS WG	Wagon	BI, BC, BPC	5 mn
Assembly Lower block	Assembly	ASS LB	Lower Block (BI)	PA, BR, CI	1 mn
Fixing Part 1	Fixing	FIX P1	Part1 (PA)	BR, AT	1 mn
Fixing Block wheel	Fixing	FIX BW	Block wheel (BR)	CR, FR, RO, RM	2 mn
Turning	Turning	TOUR DS	Driveshaft (AT)	AA	1 mn

Driveshaft					
Machining Lower support	Machining	USIN LS	Lower support (CI)	TC	2 mn
Cutting Steel	Cutting	DECP S	Sheet metal (TC)	TA	1 mn
Fixing Conduction Block	Fixing	FIX CB	Block Conduction (BC)	TB, VL, LV, CMP1	1 mn
Assembly CMP1	Assembly	ASS CMP1	CMP1	MH, CMP2	1 mn
Assembly CMP2	Assembly	ASS CMP2	CMP2	CMP3, CE	1 mn
Fixing Load	Fixing	FIX L	CMP3	CMP4, CT	1 mn
Assembly CMP4	Assembly	ASS CMP4	CMP4	MD, CS	1 mn
Machining Superior support	Manufacturing	USIN SS	Superior Support (CS)	TC	1 mn
Fixing Carries load block	Fixing	FIX CLB	Block Carries load (BPC)	CT, CMP5	2 mn
Usinage CMP5	Machining	USIN CMP5	CMP5	FC, SS	1 mn
Machining Support of Security	Machining	USIN SS	Support of Security (SS)	TC	1 mn
Folding	Folding	FLD	Fork (FR)	TL	2 mn
Smoothing	Smoothing	SMT	Flat steel 1 (TL)	AP	1 mn
_		_	Flat steel 2 (AP)	_	_
_		_	Axis Steel (AA)	_	_
_		_	Flat steel (TA)	_	_

6.2. Prototype Aglet Agents

As discussed in Section 5, the Aglet platform is adapted to our work context. This platform offers some services such as mobility, communication between agents and integration of the graphic interfaces. Aglets are defined for each phase of the product life cycle; each Aglet is responsible for managing various functionalities of each phase based on the information provided by the produced agent. It stores in turn its own information and launches of the dispatching the product's Aglet towards the next phase. To validate the proposed architecture, we present the following prototype and some screenshots as a deployment of the developed concepts.

Administrator Agent:
With the administrator agent (Figure 12), the user can identify the "Wagon" while specifying its features, and then define the location/address of the machine on which agents are going to be created one by one.

Fig 12Administrator agent interface

Design agent
Design agent (Figure 13) is concerned with the following three phases : a) capture of a list of components (for the bill-of-material), b) definition of the bill of material, and c) range of dismantling.

Fig. 13Design agent interface

Production Agent: Elements of the component list in Figure 14 are recovered from the product agent that enters in communication with the production agent.

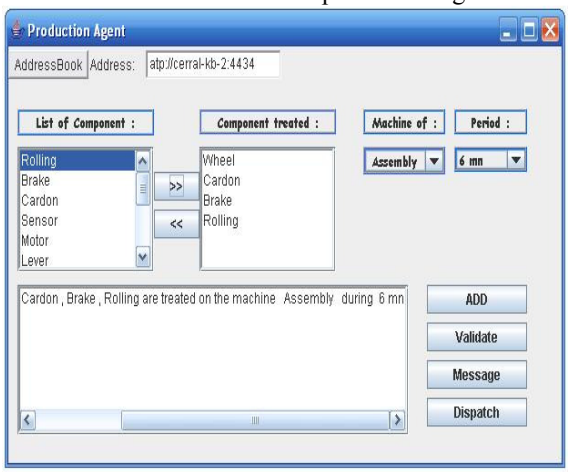

Fig. 14 Production agent interface

Maintenance Agent

The maintenance agent (Figure 15) defines the maintenance of the product as follows, based on the information stocked in the product agent.

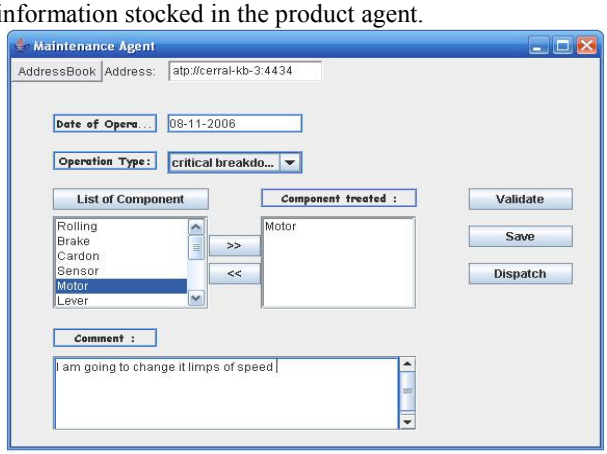

Fig.15 Maintenance agent interface

Recycling Agent:

The recycling agent retains and takes data captured in the Range of Dismantling as a basis; it presents some commentaries on the state of composing them after the operation of retaining.

Product Agent:

The product agent (Figure 16) is the centre of agent services; it allows the exchange of information with all other agents, as well as with pilots of the supply chain. The product agent allows saving product structure data and information; it defines the Product Avatar and ensures a distributed architecture.

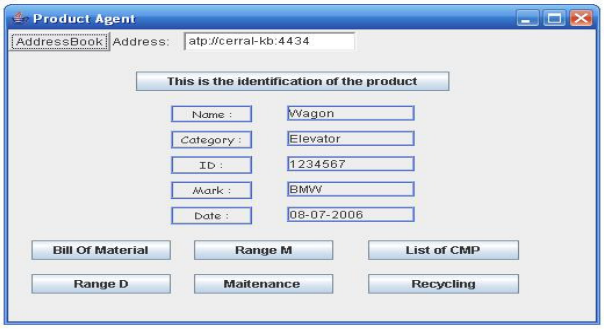

Fig.16 Product agent interface

Finally, all the data are saved in the mobile product agent; this agent makes it possible for other agents to capture necessary information for the effectiveness of the operation. For example, for a critical detection of breakdown the product agent is able to generate information and transmit it to the design agent so as to redesign the product and to define a new version. In this operation, the product agent benefits from its mobility and the multi-agent architecture to communicate with the other agents and to move from one node of the network to another across the phases of the product lifecycle.

7. Conclusion

This paper presented a mobile agent architecture/ framework for collaborative product lifecycle, which is developed by adopting Java and mobile agent technologies. The contribution of this paper is to adopt the Aglets mobile agent platform to develop a series of aglet templates to encapsulate the different legacy product information and manufacturing resources and integrate them into product lifecycle context. It not only simulates real product lifecycle activities from design to recycle but also to provide an environment for managing lifecycle product information. The main advantage of an MMA-based approach over the other approaches is the flexibility, mobility or flow of information, dynamic adaptation, collaboration and comfort for a company and the product lifecycle architecture. One agent placed at one site can take

local decisions, filter unnecessary information, alter its mission advantageously, learn about supplier behaviour, and move to another site, actions which make the product lifecycle management potentially more efficient and intelligent. Beyond that, the application of this approach appears to provide a more rational utilization of the network, which means lower costs. The mobile-agent-based approach can be utilized to provide an enterprise with reliable and in time information about the product lifecycle. With accurate information an enterprise can improve its strategic, tactile and operational plans with support for rapid decision-making, agile reaction and collaboration for product development. One important aspect of the future work is the study of a strong security mechanism for protecting mobile agents.

Disclaimer

No approval or endorsement of any commercial product.

References

[1] Benetti, H., Beneventano, D., Bergamaschi, S., Guerra, F., and Vincini, M. (2005). An Information Integration Framework for E-commerce, Intelligent Systems, 17(1), pp. 18-25,

[2] Browne, J.; Sackett, P.; Wortmann, J., (1995). Future manufacturing systems: Towards the extended enterprise, Computer in Industry, Special Issue on CIM in the Extended Enterprise, Vol. 25 N 3, pp.235-254.

[3] Camarinha-Matos, L. M., Vieira, W., (1997). Mobile Agents and Remote Operation, Proceedings' INES'97, Budapest.

[4] Cassina, J. (2008). Extended Product Lifecycle Management, PhD thesis at Politecnico di Milano, April 2008

[5] Cassina, J., Cannata, A., Taisch, M., (2009). Development of an Extended Product Lifecycle Management through Service Oriented Architecture.

[6] Cheng, C.B., Wang, C. (2008). Outsourcer selection and order tracking in a supply chain by mobile agents. International Journal Computers and Industrial Engineering archive, Volume 55, Issue 2, Pages 406–422

[7] Christoforos, P., Samaras, Pitoura, G. Evripidou, E. P. (1999). Parallel Computing Using Java Mobile Agents.25th Euromicro Conference Special session on Network Computing.

[8] Demazeau, Müller J.P. (1990) 'Distributed AI', Vol. 1, Elsevier North-Holland.

[9] Fischer, K., Müller, J., Heimig, I., Scheer, A., (1996). Intelligent Agents in Virtual Enterprises.Proceedings PAAM'96, pp.206-223.

[10] Finin, T. Frizson, R. (1994). KQML - A Language and Protocol for Knowledge and Information Exchange. Technical Report CS-94-02, Computer Science Department, University of Maryland.

[11] Genesereth M., Singh N., (1994). A knowledge sharing approach to software interoperability. Technical Report RL-94-6, Logic Group, Department of Computer Science, Stanford University.

[12] Guanhui Zhou and Pingyu Jiang (2005) 'Using mobile agents to encapsulate manufacturing resource over the internet', International Journal of Advanced Manufacturing Technology, 25: 189-197.

[13] Hardwick M., Spooner D., Rando T., Morris C., (1996). Sharing Manufacturing Information in Virtual Enterprises.Communications of the ACM, Vol. 39 N 2, pp.46-54.

[14] Harrison, C. G., Chessm, D. M., Kershenbaum, A. Mobile Agents: are they a good idea? Research Report, IBM Research Division.

[15]Huhns, M.N. (1987. Distributed Artificial Intelligence. Pitman Pub., Morgan Kaufmann,

[16] Hsu, C., Gerhardt, L., and Rubenstein, S., (1994) 'Adaptive Integrated Manufacturing Enterprises: Information Technology for the Next Decade', IEEE Transaction on System, Man and Cybernetics, 24(5), pp. 828-837.

[17] IBM Aglets Software Development Kit – available on http://www.trl.ibm.com/aglets/

[18] IBM Product Lifecycle Management, http://www-1.ibm.com/solutions/plm/

[19] IEEE Standard Computer Dictionary (1990) 'A Compilation of IEEE Standard Computer Glossaries'. New York, NY.

[20] Jennings, N. (1994). Cooperation in Industrial Multi-Agent Systems', World Scientific Publishing Co.

[21] Jiao, R., You, X. and Kumar, A. (2006). An agent-based framework for collaborative negotiation in the global manufacturing supply chain network. International Journal of Robotics and Computer-Integrated Manufacturing, vol. 22, no. 3, pp. 239-255.

[22] Kim, H., Kim H.S., Lee J.H., Jung J.M., Lee J.Y., and Do N.C., (2006). A framework for sharing product information across enterprise. International Journal of Advanced Manufacturing Technology, 27: 610-618.

[23] Kotler, P., (1988). Marketing management: analysis, planning, implementation, and control. Englewood Cliffs: Prentice-Hall International'.

[24] Khosrowpour, M. (1999). Managing information technology resources in organizations in the next millennium.In

Information Resources Management Association International Conference. Hershey, PA, USA.

[25] Kubota, F., Sato, S., and Nakano, M., (1999). Enterprise Modeling and Simulation Platform Integrating Manufacturing System Design and Supply Chain. IEEE International Conference on Systems, Man, and Cybernetics, 4, pp. 511-515.

[26] Long, Q. Lin, J. Sun, Z. (2011). Modeling and distributed simulation of supply chain with a multi-agent platform. International Journal of Advanced Manufacturing Technology, 55:1241–1252.

[27] Magedanz, T. Bäumer, C Breugst, M. and Choy S., Grasshopper-A Universal Agent Platform Based on OMG MASIF and FIPA Standards- IKV++ GmbH Germany, available on http://www.ikv.de/

[28] Mahdjoub, M. Monticolo, D. Gomes, S. Sagot, J.C. (2010). A collaborative Design for Usability approach supported by Virtual Reality and a Multi-Agent System embedded in a PLM environment. International Journal of Computer-Aided Design, Vol 42, Issue 5, pp. 402–413

[29] ORTEMS, 2011, http://www.ortems.com/

[30] Papastavrou, S. Samaras, G. and Pitoura, E. (1999). Mobile Agents for WWW Distributed Database Access. Proc. 15th International Data Engineering Conference, Sydney, Australia.

[31] Rabelo, R. J. Camarinha-Matos, L. M. (1994). Negotiation in Multiagent Based Dynamic Scheduling. International Journal on Robotics and CIM, Vol. 11, No. 4, pp.303-310, Pergamon.

[32] Rabelo, R. J. Spinosa, L.M. (1997). Mobile-agent-based Supervision in Supply Chain Management in the Food Industry.AGROSOFT 97 - Feira e Congresso de InformáticaAplicada à Agropecuária e Agroindústria - BeloHorizonte – BRAZIL.

[33] Spinosa L. M., (1997). For a Decision Support System to Distributed Manufacturing Systems: a multiagent and CIMOSA based approach', to appear in Proceedings IFAC/IFIP Conference MCPL'97 Management and Control of Production and Logistics, Brazil.

[34] UGS PLM Solutions, http://www.eds.com/products/plm/index.shtml

[35] Victor R. Lesser and Edmund H. Durfee 1989. "Negotiationg task decomposition and allocation using partical global planning" Chapter 10 Distributed Artificial Intelligence

[36] Wong, C.Y.; McFarlane, D.; Ahmad Zaharudin, A.; Agarwal, V. (2004). The intelligent product driven supply chain.IEEE International Conference on Systems, Man and Cybernetics. Hague, The Netherlands.

[37] Xu, X. (2012). From cloud computing to cloud manufacturing. International Journal of Robotics and Computer-Integrated Manufacturing. Volume 28, Issue 1, Pages 75–86

[38] Zweben M., (1996). Intelligent Agents.Computer Integrated Manufacturing and Engineering, Vol. 1 N 2, pp.14-15.

Khaled BAHLOUL is currently a researcher of IFSTTAR institute in France. He obtained his PhD in Computer Science from the National Institute of Technology (INSA) of Lyon in France.

Nesrine Darragi is currently a PhD student of IFSTTAR institute in France. She obtained his Master degree in Computer Science from Graduate School of Science and Technology of Tunis in 2008.

Yacine Ouzrout is currently Assist Professor at the University of Lyon (France) where he leads a research team of the DISP laboratory.He obtained his PhD in Computer Science from the National Institute of Technology of Lyon in France.He is also a member of the CERRAL Innovation Center of the Lumière IUT Technology Institute, a unique public center in France.

Abdelaziz Bouras is currently Professor at the University of Lyon (France) where he leads a research team of the DISP laboratory. He has been recently conferred the HONORIS-CAUSA honorary Doctoral Degree in Science of the Chiang Mai University (Thailand) from Her Royal Highness Princess Maha Chakri Sirindhorn of Thailand. He is also leading the CERRAL Innovation Center of theLumière IUT Technology Institute, a unique public center in France.