Agent-Based Enhanced Workflow in Manufacturing Information Systems: the MAKE-IT Approach

Ernesto Montaldo, Roberto Sacile, Mauro Coccoli, Massimo Paolucci and Antonio Boccalatte

DIST — Department of Communication, Computer, and System Science, University of Genoa, Italy

Manufacturing is one of the main fields of application of software agent technology. In this respect, several current research projects focus on workflow management, with the aim of an integration and coordination among plant and business activities. In section I, the state of the art is presented, relevant to software agents and to the technologies currently adopted in the exploitation of agents in manufacturing information systems. In section II, the possibility of developing real-time agents at the shop floor level is investigated. In section III, the MAKE-IT (Manufacturing Agents in a Knowledge-based Environment driven by Internet Technologies) approach to agent-based enhanced workflow in manufacturing information system is presented. Finally section IV presents our conclusions.

Keywords: workflow management systems and technologies, software agents, intelligent systems, e-commerce systems, manufacturing information systems (MIS).

1. The State of the Art of Software Agents Applied to Manufacturing Information Systems

We are experiencing a software revolution. In the last few years, we have seen a remarkable increment of software technologies dealing with highly distributed information, such as Internet-based technologies. The growth of information sources on the Web has caused many web users to suffer from information overload. Software agents have been developed to solve this kind of problem and others, and they are probably the current fastest growing area of information technology. On the Internet, for example, a software agent (also referred as *intelligent agent)* is a program that can gather information or perform some other services without an immediate user presence. *Software agents* are often used for applications such as personalized information management, electronic commerce or management of complex commercial sites and industrial processes, with special reference to workflow management.

Pattie Maes has given the following definition of an agent: *a computational system which is long-lived, has goals, sensors and effectors, decides autonomously which actions to take in the current situation to maximize progress towards its (time-varying) goals* (Maes, 1997). The same author has also specified a definition of software agents: *particular type of agent, inhabiting computers & networks, assisting users with computers-based tasks* (Maes, 1997). Jennings and Wooldridge have identified three different classes of agents (Jennings & Wooldridge, 1995):

- Agents that execute straightforward tasks based on pre-specified rules and assumptions.
- Agents that execute a well-defined task at the request of a user.
- Agents that volunteer information or services to a user, without being explicitly asked, whenever it is deemed appropriate.

The main characteristics of these agents are (Jennings & Wooldridge, 1995):

- Autonomy: agents should be able to perform most of their tasks without the direct inter-

vention of humans, and they should have a degree of control over their own actions and their own internal state.

- Social ability: agents should be able to interact with other software agents and humans.
- Responsiveness: agents should perceive their environment, and respond in timely fashion to changes that occur in it.
- Proactiveness: while responding to their environment, agents should exhibit opportunistic, goal-directed behavior and take the initiative where appropriate.

In addition to these main attributes, other desirable characteristics have been proposed (Jennings & Wooldridge, 1995):

- Adaptability: the ability to modify its behavior over time in response to changing environmental conditions or an increase in knowledge about its problem-solving role.
- Mobility: the ability to change its physical location to enhance its problem solving.
- Veracity: the assumption that an agent will not knowingly communicate false information.
- Rationality: the assumption that an agent will act in order to achieve its goals and will not act in such a way as to prevent its goals being achieved without good cause.

Nwana has given another point of view about the agent paradigm (Nwana, 1996). The main characteristics an agent should exhibit have been identified in a set of three attributes — *autonomy, cooperation,* and *learning.* While truly smart agents, with all three characteristics, do not yet exist, a more complex agent typology has been defined based on the previous and on other characteristics (Nwana, 1996):

- Collaborative Agents: they emphasize autonomy and cooperation to perform tasks by communicating and possibly negotiating with other agents to reach mutual agreements. These agents are used to solve distributed problems where a large centralized agent is impractical.
- Interface Agents: they are autonomous and utilize learning to perform tasks for their users. The inspiration for this class of agents is a personal assistant that collaborates with the user.
- Mobile Agents: they are computational processes capable of moving over a network (possibly a wide area network), interacting with foreign hosts, gathering information on behalf of the user, and returning to the user after performing their assigned duties.
- Information Agents: they are tools to help manage the tremendous amount of information available through networks such as the World Wide Web and Internet.
- Reactive Agents: they represent a special category of agents which does not possess internal, symbolic models of their environments; instead they act/respond in a stimulus-response manner to the present state of the environments in which they are embedded.
- Hybrid Agents: they are particular agents that combine two or more agent philosophies within a singular agent.
- Heterogeneous Agent System: refers to a collection of two or more agents with different agent architectures.

While the classification from Jennings and Wooldridge, 1995, is oriented to the definition of "how" an agent is stimulated to start its actions, the typology by Nwana seems to be more oriented to possible tasks and applications and to the way agents cooperate to reach their objectives.

On the basis of the previous scientific assumptions, several companies and organizations are developing software based on agent technology, feeling an increasing need to define a standard. In this respect, the Foundation for Intelligent Physical Agents (FIPA) has defined one of the main standards in the field of agent technology (FIPA, 1996).

1.1. Information, Workflow, and Knowledge Management in Manufacturing

Manufacturing is one of the main fields of application of agent technology. Artificial Intelligence (AI) techniques have been widely used in so-called Intelligent Manufacturing for more than twenty years. However, recent developments in agent systems in the domain of Distributed AI have brought new and interesting possibilities.

Global markets and rapidly changing customer requirements are driving changes in the production styles and configuration of manufacturing enterprises. Traditional centralized and sequential manufacturing planning is insufficiently flexible to respond to changing production styles and highly dynamic variations in product requirements. The traditional approaches limit the expandability and reconfiguration capabilities of manufacturing information systems (Shen & Norrie, 1999). Specifically, the main requirements that a manufacturing information system should satisfy are (Shen & Norrie, 1999): enterprise integration, distributed organization, interoperability, open and dynamic structure, cooperation, integration of humans with software and hardware, agility, scalability and fault tolerance. Agent technology provides a natural way to satisfy these kinds of problems, adding intelligence to manufacturing information systems, and satisfying their requirements.

Decisions relevant to the planning of the activities involved in manufacturing production should rely on information about the status of the processes, from supply to production and distribution. Manufacturing information systems that integrate resource management and planning activities, such as ERP (Enterprise Resource Planning) systems, are frequently integrated with MES (Manufacturing Execution Systems) systems that are devoted to the monitoring and control of production activities at the shop floor level.

Associations of manufacturing companies have naturally started to define process reference models aiming to integrate well-known concepts of business process reengineering, benchmarking, and process measurement into cross-functional frameworks. These reference models are the result of a traditional knowledge engineering approach, where communication, knowledge and enterprise modeling are the main objectives. For example, the recent Supply Chain Operations Reference (SCOR) model defined by the Supply Chain Council (Supply Chain Council, 1996) has had a deep impact on business system planning. According to the SCOR model, ERP products should address the requirements of the PLAN, SOURCE and DELIVER processes, while MES are the primary component of the MAKE process.

Most large and mid-size companies are currently implementing ERP systems but, as they attempt to enhance the integration of their manufacturing information systems with the production plant, it becomes evident that a gap in operation functionality exists. This leads manufacturing organizations to look for a new approach to plant operation management. The ability to create an effective link between plant resources and production management, together with tight closed-loop integration among plant, suppliers, and customers, has become more and more necessary.

Recently, in order to obtain software environments for the management of the modern manufacturing enterprise, an innovative REPAC (Ready, Execute, Process, Analyze, and Coordinate) model (AMR Research, 1998) has been proposed. In this new REPAC model, manufacturing is not a "static" process. It is the result of a continuous flow of actions and information inside the enterprise. Data is as much an enterprise resource as raw materials, and in this respect, the information is also manufactured according to a proper workflow.

Workflow can be defined as *the computerized facilitation or automation of a business process in whole or part,* and the workflow management system as *a system that completely defines, manages, and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic* (Workflow Management Coalition, 1995).

Due to their ability to integrate different information systems, such as plant and business information, workflow management is the major application of software agents in manufacturing. Figure 1 shows two basic ideas of agent-based architecture applied to workflow management (Odgers et al., 1999).

The first is from the ADEPT (Advanced Decision Environment for Process Tasks) project by British Telecom labs (Jennings et al., 1996). The ADEPT system consists of multiple software agents that concurrently negotiate an agreement on how resources should be assigned to support a business process. The software agents take full responsibility for business process provisioning, execution, and compensation, with each agent managing and controlling a given task or a set of tasks.

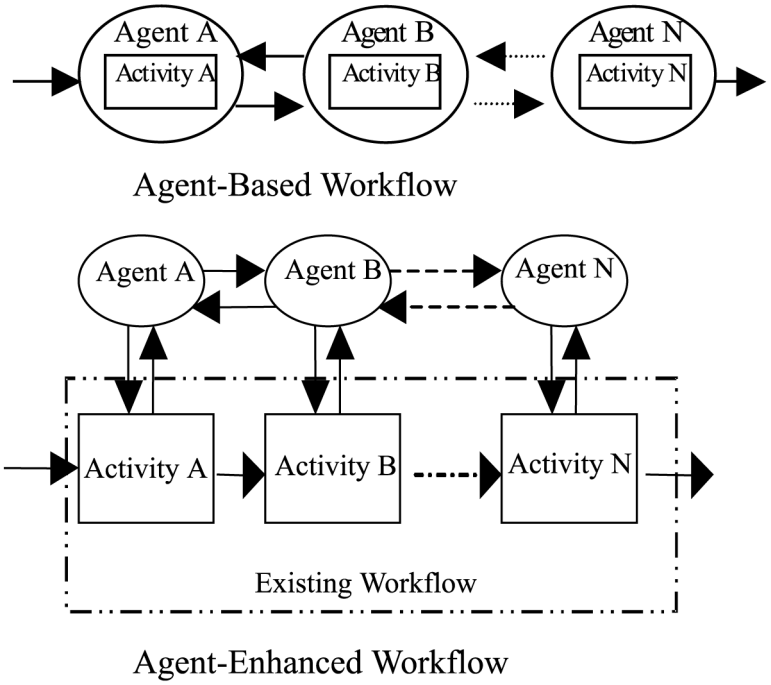

Fig. 1. Agent-based workflow vs. agent-enhanced workflow (redrawn from Odgers et al., 1999).

The second agent-based architecture, named *agent-enhanced workflow,* is shown in the lower part of Figure 1 (Odgers, et al. 1999). Here, agents provide an additional layer to the existing manufacturing information system thereby enhancing or introducing workflow management. Judge et al. (Judge et al., 1998) have recognized the dependency on legacy workflow management systems as well as the cleaner interface that software agents would provide to underlying process tasks. In their work, the integration of agent-based process management with existing commercial workflow management systems has been also investigated.

In the editorial by M. Aparicio (Aparicio, 1999), some recent important examples of agent-based architectures applied in the manufacturing domain have been introduced. The "Simulation Based Design" program combines agents with distributed object-oriented techniques (Dabke, 1999). The "Manufacturing Agent-Based Emulation System" represents an open framework for design and analysis of discrete manufacturing systems (Ivezic et al. 1999). A multi-agent approach, where user and resource agents collaborate in a virtual enterprise to enable knowledge exchange, adding agent functionality to an existing framework, called "Product Realization Environment", has been described by Pancerella & Berry, 1999. Finally, the "Autonomous Agents at Rock Island Arsenal" (AARIA) provides a demonstration of how manufacturing can move towards mass customization by using the Internet as a natural platform for managing distributed operations and by using autonomous agents as the tools for efficiently reconfiguring available productive resources (Baker et al., 1999). However, as assessed in (Aparicio, 1999), while simple agent technologies for notification and personalization already exist, there is much work yet to be done to make Internet connectivity in the field of manufacturing more communicative, adaptive, and goal-oriented — in a word, more intelligent.

It is important to observe that the information and the knowledge managed in manufacturing information systems are subject to different requirements and have different features according to whether the communication flows inside or outside the enterprise. Inside the enterprise, the knowledge networking mainly focuses on workflow management and corporate memory. Outside the enterprise, different capabilities are requested. One example is the communication need of international manufacturing networks to rapidly face changes in global market opportunities and to jointly develop global competitive capabilities (Shi & Gregory, 1998). Another

example is the need for developing electronic commerce techniques (Kodama, 1999) that include knowledge-based technologies such as data mining. A wide range of assessed computational techniques (e.g. artificial neural networks, fuzzy logic, machine learning, statistics, expert systems, and data visualization) have provided new intelligent tools for automated data mining and knowledge discovery, with a deep impact on current practices used in manufacturing and directly affecting the production processes. In conclusion, knowledge will be more and more a part of a manufacturing information system design, and, due to the distributed nature of modern manufacturing, intelligent software agent architectures are likely to be more and more used to reach this goal.

2. Real-time Agents for Manufacturing

An effective alternative to the model of a centralized workflow management can be obtained by exploiting an agent-based architecture. In this case, manufacturing agents may need to interact with the external world, or with the plant and related human operator-driven activities. For these reasons, to be able to reach to asynchronous signals or to respect time constraints or rendez-vous, they should show some kind of real-time behavior.

In most of the applications in which users may need time determinism within their software applications, the possibility of achieving real-time tasks becomes a pre-requisite of the software architecture. To cope with such problems, many specific real-time operating systems exist, guaranteeing real-time reliability but causing the program to be executed in a non-standard runtime environment. Often, time constraints are not so rigid, and the differences can be made between hard and soft real-time applications, always keeping time determinism as a mandatory requirement. This is a typical scenario encountered in manufacturing applications, in which real-time is strongly needed and different layers are managed by different software and hardware configurations: in particular the low-level plant control tasks are performed by dedicated programmable logic controllers in a hard real-time environment (e.g., level control in a tank, speed or axis control of DC motors, etc.) while supervision and monitoring operations are performed at higher production levels and with different needs; in this case, due to the presence of graphical user interfaces, to the interaction with humans, and to slow network connections (anyway, slower with respect to some field-busses), the real-time frame can be much wider. Moreover, in a manufacturing context, the introduction of software agents is a very promising research area: they can be very helpful in monitoring and scheduling production processes as well as in integrating the plant with other business activities, within a virtual manufacturing vision. Manufacturing software agents may also need to be real-time agents, that is, they may need to have the ability of acting timed communications, respecting some time constraints, managing and scheduling their activities according to priority-based or time-based policies.

In order to guarantee that the proposed real-time agents can deal with the variety of hardware and software that can be encountered in a manufacturing plant, they should be developed in such a way to be platform independent. An immediate answer to this last problem can come from the use of the Java platform whose claim is “write once run everywhere”. It can be observed that Java could be not the most appropriate language for real-time programming; on the other hand, it offers all of the other features needed for the development of software agents, including an international standard for agents implementation and communication (FIPA, 1996).

Given general characteristics that a real-time system should offer (Nissanke, 1995), Java seems not to be adequate at all for real-time programming; the main motivation can be found in the fact that Java is a platform-independent language, and a real-time scheduling of Java threads cannot be guaranteed if there is no a priori knowledge about the operating system-scheduling characteristics. Moreover, Java has an automatic garbage collector whose influence should be taken in account (Johnson, 1992; Baker, 1991). However, since Java naturally supports threads (Sun Microsystems, 1995a), it appears to be suited for the same kind of real-time programming. In fact, real-time applications have to be written as a series of separate component programs that can execute concurrently in a multi-threading organization; indeed

every Java thread is a complete program capable of independent execution, sharing its memory with others, always keeping separate addresses, so that context switching can be very fast. Moreover, since its first appearance, in May 1995, Java was presented as a language for facilitating the development of embedded systems software (Sun Microsystems, 1995b), and most embedded computer systems have to deal with real-time constraints. In order to provide Java with real-time characteristics, the first solution is being developed by a consortium comprehending IBM, Hewlett Packard, and Newmonics: they have already developed a modified Java Virtual Machine, called Perc (that is a real-time dialect of Java). It is important to notice that such a consortium, however supported by major software houses, is not sustained by Sun that has made up an Expert Group on its own. Whoever it can derive from, it is evident that such a solution would have a great impact on the platform independence that any Java applications should keep on claiming and guaranteeing. Another possible alternative solution should go towards the development of specific real-time programming facilities to handle process scheduling and real-time features, allowing to respect time constraints that (soft) real-time applications always have to deal with. It can be seen that it is possible to have some degree of real-time reliability with Java too, provided that Java programming is performed in a special environment. Moreover, many embedded systems' applications more and more need a Web connection (Perrier, 1999) and Java is the best language for Web programming because it has a special ability for the development of distributed applications.

A package for real-time Java programming (Boccalatte & Coccoli, 2000) has recently been developed. This package allows to respect time constraints that (soft) real-time applications always have to deal with, providing a standard Java environment with real-time capabilities of scheduling threads according to time-based policies or by priority and events. In this way, the order of the messages to an agent, which sensibly affects the agent decision-making process, is respected. For this purpose, a set of methods for threads to communicate through queues, messages, and mailboxes has been developed.

3. Agent-based Enhanced Workflow in Manufacturing Information Systems: the MAKE-IT Approach

One of the urgent needs of manufacturing information systems is to integrate different information systems enhancing functionalities such as workflow management. Software agents can solve the problem as a new software layer, to be added to an existing manufacturing information system.

MAKE-IT (Sacile et al, 2000) (Manufacturing Agents in a Knowledge-based Environment driven by Internet Technologies) objective is the definition and the implementation of "small" software architectures to support workflow management in manufacturing information systems of small-medium enterprises. These architectures, called MAKE-IT agents, add functionalities to an existing manufacturing information system, and they can be regarded as an agent-enhanced workflow approach (Odgers, et al. 1999). The MAKE-IT agents can perform simple rule-based actions while performing a quite difficult and complex task through their coordination as a whole. In addition, the MAKE-IT project aims to integrate existing and assessed technologies in the architecture, which are not conceptually distant from the current way of traditional manufacturing information system operation.

MAKE-IT agents are not mobile (Nwana, 1996), and they can be classified either as gopher agents (Jennings & Wooldridge, 1996) or as information collaborative agents (Nwana, 1996). Autonomy, social ability, and responsiveness are the main characteristics of MAKE-IT agents. The environment where MAKE-IT agents live is an information world in which data is stored in and retrieved from relational database management systems (RDBMS), documents are generated according to some events of the manufacturing production process, information can flow inside the enterprise by a distributed MAKE-IT agent network and from/to the external Internet world. MAKE-IT agents are able to produce the data, that is, to transform them, following plant and business processes. Plant data can be retrieved from real-time-RDBMS managed by

proprietary SCADA systems (Supervisory Control and Data Acquisition). In this case, a commercial SCADA system (Windows NT Open System) is used. The other information can be acquired from/stored in RDBMS or Internet.

A simple but great advantage that the MAKE-IT approach offers is the possibility to add the coordination of business and plant processes in the same framework to the existing manufacturing information system. For example, a text document model could be used to automatically produce quality control documents generated by some processes of the shop floor, or e-mail messages could be generated when malfunctioning conditions of the plant can be deduced by the monitoring of some frequent alarms.

Two main aspects characterize the current version of the MAKE-IT agents: *the communication model and the knowledge model.*

The Communication Model

In the MAKE-IT project, the need for an effective and separate workflow of relational data inside and outside the enterprise is stressed. The MAKE-IT communication model is based on XML (extensible Mark-up Language) (W3 Consortium, 1998) tunnelled inside the enterprise boundaries within MSMQ (Microsoft Message Queue) (Microsoft TechNet, 1998) software channels, and outside the enterprise within traditional Internet channels.

XML is an open standard for putting *structured data* in a text file. XML can be considered a set of rules, guidelines, conventions, for designing text formats for such data. Through the use of XML, a computer can handle information, easily, unambiguously, and avoid most common pitfalls, such as lack of extensibility, lack of support for internationalization/localization, and platform-dependency.

When XML language is used in agent communication, it is also important to assert how the data type is represented in order to allow integration with other external information systems. In this respect, the BizTalk group has provided guidelines on schema publishing (BizTalk, 2000). In the BizTalk web site, several schemas concern manufacturing, such as *bill of material (BOM) transactions, product requisition or inventory management transactions, production order transactions, and schema to enable retailers to send purchase orders.*

While XML is a language for structured communication and BizTalk provides guidelines to understand MAKE-IT messages in a defined open context, a reliable channel for the communication is needed.

MAKE-IT agents use MSMQ inside the enterprise boundaries for their asynchronous communication. In programming, message queuing is a method by means of which processes (or program instances) can exchange or pass the data using an interface to a system-managed queue of messages. Specifically, MSMQ can be considered an asynchronous transport protocol based on the IP network protocol. MSMQ enables applications running at different times to communicate across heterogeneous networks with systems that may be temporarily offline. MSMQ provides a guaranteed message delivery, an efficient routing, security, and priority-based messaging.

The MAKE-IT communication outside the enterprise boundaries is designed using traditional Internet applications, due to their greater diffusion among general external users. Some MAKE-IT agents are able to acquire knowledge and information from published web pages, to send information to web pages, or to manage e-mail messages.

The Knowledge Model

MAKE-IT agents are able to perform simple tasks that can generally be expressed as a series of *if condition/s — then action/s* rules. The knowledge base of a MAKE-IT agent is currently modelled in a rule-based system fired by a standard inference engine. In this respect, CLIPS (C Language Integrated Production System) (Riley, 1999) has been adopted. In particular, a set of different facts (the *condition/s* part in a rule) in a CLIPS environment are linked to different variables stored in tables of the RDBMS. If each condition is considered true, the agent will perform a set of actions inside its environment.

3.1. The MAKE-IT Architecture

Three different levels, and three different types of agents characterize the MAKE-IT architecture:

- MAKE-IT agent generator
- MAKE-IT agent network
- MAKE-IT agent workflow manager

Figure 2 shows the interactions among MAKE-IT agent generator, agent network, and agent workflow manager.

A triplet represents each agent:

$$\Omega_i = S_i, CM_i, KM_i \quad i = 1, \ldots, n$$

Where: S_i refers to the state of the $i-th$ agent, that is, whether the agent is running or not; CM_i refers to the communication model of the $i-th$ agent, that is, the set of queues used by the agent to exchange messages; KM_i refers to the knowledge model of the $i-th$ agent, that is, the set of *if-then* rules constituting the knowledge base, and n is the total number of agents. An example of a simple set Ω for a specific agent is defined in the following:

$\Omega = \{ON, ($***input_queue:***
ComputerName|msmq_order_code;
output_queue:
ComputerName|msmq_order_to_production,
ComputerName|msmq_components_request$)$,
(rule1, rule2)$\}$.

The input and output queues represent the communication model, and rule 1 and rule 2 are the agent's knowledge base, and, in this example, are:

Rule1

IF<Received_Message_From_Order_Code>
AND<I_Can_Produce_The_Order>
THEN<Send_Message_Order_To_Production>
AND<Update_Database>AND<E-mail_To_
Customer>

Rule2

IF<Received_Message_From_Order_Code>
AND<I_Cannot_Produce_The_Order>
THEN<Send_Message_Components_Request>
AND<E-mail_To_Customer>

This agent has to check if a new message about orders has arrived, and if it is possible to produce it. In case it is possible, a message is sent

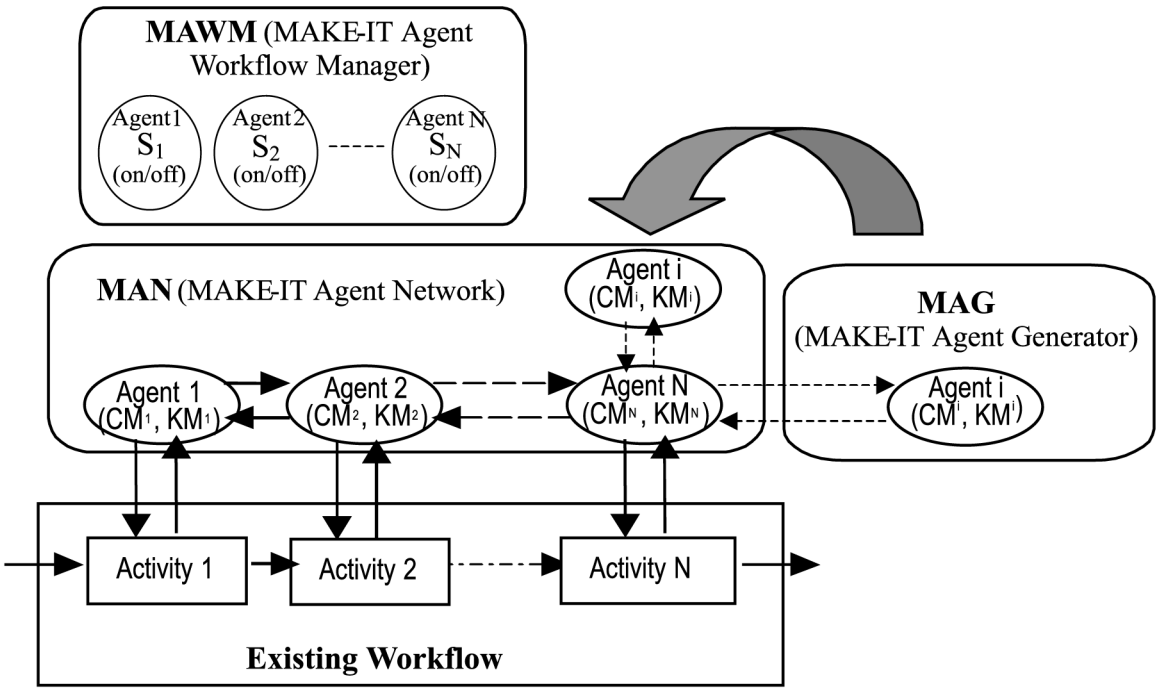

Fig. 2. MAKE-IT system architecture.

to production, the database is updated, and the information is notified to the customer, while, in case it is not possible to produce it, a message to request components is sent and notified to the customer.

The MAKE-IT agent generator supports interpreting an agent and inserting a compiled agent in the MAKE-IT agent network. So the MAKE-IT agent generator can be thought as an agent compiler. The interpreted agent model is created to test the agent during its modelling phase. At the end of the modelling phase, a compiled agent is created and it is integrated with the existing MAKE-IT agent network. The CM_i defines the environment where a MAKE-IT agent lives and the channels on which it can communicate. CM_i includes the MAKE-IT agent identifier, and the name of the IP host the MAKE-IT agent is running on. In addition, it is possible to define the input/output queues (host, name of the queue, properties of the queue according to the MSMQ reference model) on which the MAKE-IT agent can communicate, as well as other Internet communications (Web pages, e-mail etc.) with the external world. Each MAKE-IT agent has also a limited First-In First-Out private MSMQ queue, storing the last task performed as a new message.

The KM_i can be defined graphically, and variables can be defined and linked together with the aim to build some CLIPS rules mentioned before. KM_i has two sections. In the first section, a knowledge engineer can define the variables, which will then be used as graphic objects in the rule-based section to embed knowledge within the agents. The values of this data are initialised by queries on a specified manufacturing information system RDBMS whose XML output value is stored in the variable. This value can be polled and updated at a user-defined frequency. A specific interface allows interaction with real-time RDBMS in order to acquire specific alarms and data from running processes at shop floor level, such as, for example, a control loop for a DC motor: in this respect, a target velocity and a current control signal may be considered as important data whose knowledge is also important at higher decision levels of the enterprise, assuming them to be upper bounded for those systems to work correctly, and generating alarms if this constraint is not satisfied. All of this data, coming from the field, can be arranged in a real-time DBMS. In the second section, the objects defined in the previous section can be linked together to form if-then rules. The current MAKE-IT version has a Windows-based graphical user interface that allows the

Fig. 3. MAWM implemented as an agent web control panel.

system developer to define rules and to display them easily.

From a functional point of view, the MAKE-IT agent network, where many MAKE-IT agents can work cooperatively, represents knowledge networking of the manufacturing information system. MAKE-IT agent workflow manager aims to manage coordination and control of the MAKE-IT agent network, representing the manufacturing information system knowledge networking. The MAKE-IT agents can generally operate autonomously without the intervention of an overall scheduler in their activities. In a manufacturing environment, the agent-based approach can match the complexity of the scheduling problems by adopting a heuristic which decentralizes the decisions made on the basis of the priority of the agents.

The current version of the MAKE-IT agent workflow manager only provides the possibility to see the current state of the MAKE-IT agent network architecture, to switch on/off the different agents working in the MAKE-IT agent network in a web based control panel. The switch on/off command is executed sending the highest priority message to a specific MAKE-IT agent. An example of the current MAKE-IT agent workflow manager graphical user interface is shown in Figure 3.

Other controls refer to the agent state. *On-line, off-line, stand by, error, error code, communication from agent* mean, whether an agent is running (on-line) or not (off-line), whether it is waiting to do something (stand-by), whether an error occurred (error) and what kind of error has occurred (error code), and, finally, the name of the agent from which the last message has been received. As previously mentioned, all variations of states are stored, for each agent, in a private MSMQ queue. From the reusability point of view, the MAKE-IT agents are divided into:

- Query Agents (Figure 4): they are able to get a message from a message queue, to query a RDBMS. MAKE-IT Query Agents are also able to translate the records in a XML message and send it to a message queue.
- Dispatcher Agents (Figure 5): they are able to get an XML message from a message queue, to process it and to send it to another queue, according to their knowledge model.
- Clerk Agents (Figure 6): they are able to get a XML message from the Internet, to process it and to send it to another queue, in accordance with their knowledge model. They are also able to get a message from a message queue, and to send internet e-mails.

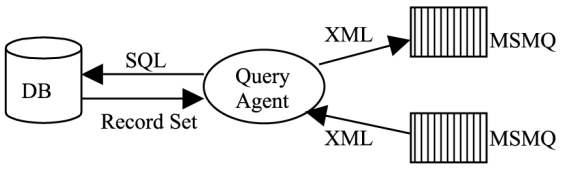

Fig. 4. Query agent.

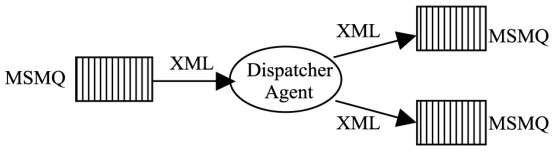

Fig. 5. Dispatcher agent.

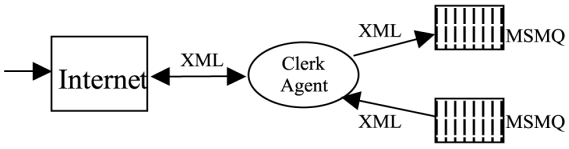

Fig. 6. Clerk agent.

3.2. Adding a MAKE-IT Layer to a Manufacturing Information System of a Small Medium Enterprise

The previously defined MAKE-IT agents can represent the main components of the MAKE-IT workflow management. Defining an *agency* as a set of agents related by their work functions, in a small medium manufacturing enterprise proposed as a demonstrative example, three agencies (Fig. 7) are present, managing purchases, production and sales workflow.

- Sales Agency (SA): receives an order; sends message to the customer that its order is in process; updates the manufacturing information system; sends the order to the Production Agency; E-mails customer about order shippment and when the order will be delivered. In this agency, all three types of agent

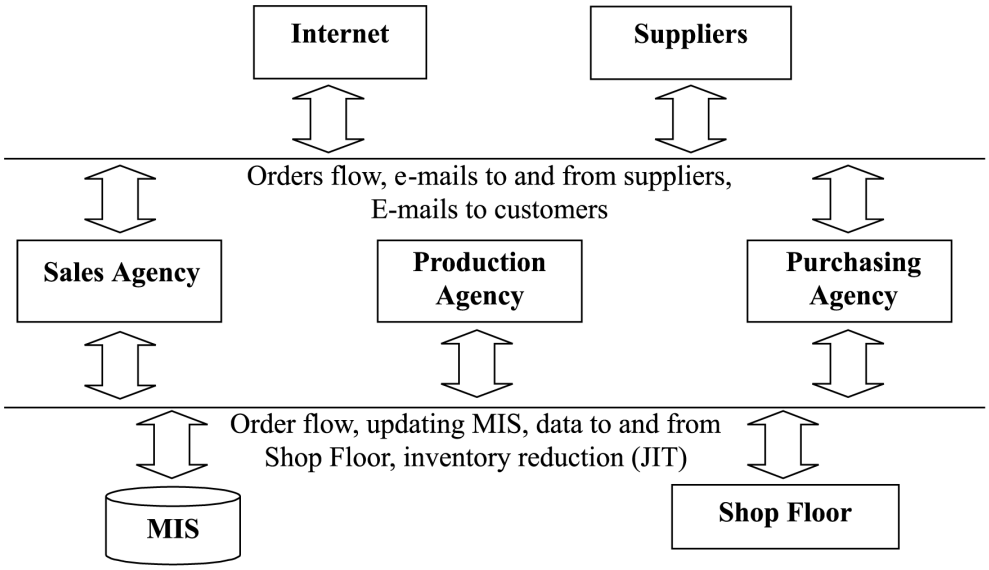

Fig. 7. MIS integration by MAKE-IT agencies.

are implemented: query agents, dispatcher agents, and clerk agents.

- Production Agency (PrA): sends the order to the shop floor; receives information from the shop floor about the state of the production; sends message to SA when the product is ready; updates the manufacturing information system. In this agency, only two types of agents are implemented: dispatcher agents, and query agents.
- Purchasing Agency (PA): applies the Just-in-Time philosophy (Groenvelt, 1993) for the inventory reduction; updates the manufacturing information system; sends orders of raw material. In this agency, only two types of agents are implemented: clerk agents, and query, agents.

These activities are autonomously implemented by a simple MAKE-IT architecture which can be distributed geographically in true remote branches of the enterprise, or physically on different computers, or just on a single PC. However, human users always have the last word in a critical decision: for example, the list of orders of raw material is prepared as an e-mail to be sent, but it is sent by a human operator.

The agents of each agency must be modeled according to the requested behavior. Figure 8 shows an example of a MAKE-IT agent while it is being defined in the MAKE-IT agent generator. This agent can manage the quantity stored in the warehouse of three types of components, which are necessary to manufacture a product. Specifically, this agent can check, either cyclically or by a specific request, whether the component types are under a defined threshold, querying the STORES table. When a component type is under the threshold, an order is automatically formulated, and E-mail is copied

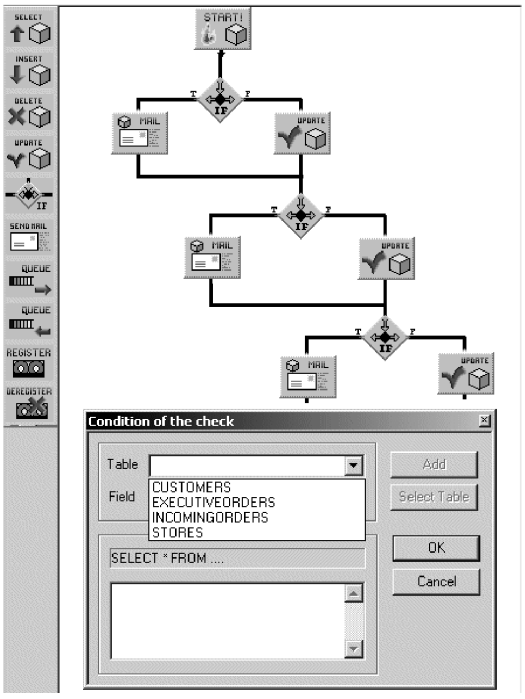

Fig. 8. Defining a MAKE-IT agent in the MAKE-IT agent generator.

in the messanger of the person responsible for the warehouse. He/she can make the final decision whether to send this order or not. When there is a sufficient quantity of a particular component in the warehouse, a checking flag is updated in the STORES table. In Figure 8, the window, which enables one to choose the table and set the query, is also shown.

4. Conclusions and Future Directions

Software agents are a practical and successful example of academic research transferred to industrial applications. They can effectively combine all the advantages of distributed systems such as scalability, reliability, and reusability with AI techniques. Specifically, in manufacturing, software agents can be applied to enhance existing MIS with the aim of a two-folded integration:

- Vertical integration in order to integrate plant and business processes.
- Horizontal integration in order to implement automatic information procedures according to a properly distributed workflow within and outside the enterprise boundaries.

The Java language is probably the most common language used to implement software agents and it is likely to maintain leadership in this field in the next decade, too. With specific reference to manufacturing and with few software "tricks", Java language can be applied even in a real-time context.

On the other hand, many manufacturing enterprises, in particular the small medium ones, generally have simple manufacturing information system, often consisting of relational databases.

The MAKE-IT architecture, presented in section III, combines the main characteristics and advantages of agent architectures distributing the complexity of the management of manufacturing workflow information according to enterprise complexity. In our opinion, MAKE-IT agents can represent an open, effective, inexpensive software architecture, which should satisfy information needs of modern manufacturing, and that can be an "add-on" to an existing manufacturing information system. From a communication point-of-view, the MAKE-IT agent architecture internally combines the benefits of the asynchronous, reliable, transactional-if requested, MSMQ channel with the XML open standard for structured communication in an Internet-oriented information system. With respect to the external world, the MAKE-IT architecture uses all the benefits deriving from an Internet-based communication. From the point-of-view of knowledge modelling, the MAKE-IT agent architecture benefits from the CLIPS reliable inference engine and from the object-oriented user interface. One of the main advantages in this respect is that the human user, who always has the last word in the MAKE-IT approach to workflow management, can be also directly involved in the definition of the agents that he may need to enhance the workflow he is supposed to manage.

Acknowledgements

This work has been carried out at LIDO-Lab (`http://www.lido.dist.unige.it`), an information technology laboratory co-sponsored by Siemens-Orsi Automation S.p.A, Genoa, Italy, (`http://www.orsiweb.com`) and by the Department of Communication, Computer, and System Science (DIST) — University of Genoa, Italy (`http://www.dist.unige.it`).

References

[1] AMR RESEARCH, INC., Do We Need A New Model for Plant Systems? *The AMR Report on Manufacturing* (1998). Available at `http://www.amrresearch.com/repac/default.asp`

[2] M. APARICIO, IV Internet-Scale Network Intelligence. *IEEE Internet Computing,* Vol. 3, No. 5, (1999), 38–40.

[3] A. D. BAKER ET AL., Agents and the Internet: Infrastructure for Mass Customization. *IEEE Internet Computing,* Vol. 3, No. 5, (1999), 62–69.

[4] H.G. JR BAKER, The Treadmill: Real-time Garbage Collection without Motion Sickness. OOPSLA *Workshop on Garbage Collection in Object-oriented Systems.* (1991).

[5] BIZTALK, BizTalk Framework Overview. (2000). Available at `http://www.BizTalk.org`.

[6] A. BOCCALATTE, M. COCCOLI, Java for Real-time Object-oriented Programmino. In *Proceedings of the Workshop Dagli Oggetti agli Agenti: Tendenze Evolutive dei Sistemi Software*. May (2000), 41–46.

[7] P. DABKE, Enterprise Integration via Corba-Based Information Agents. *IEEE Internet Computing,* Vol. 3, No. 5, September/October (1999), 49–57.

[8] FIPA, Foundation for Intelligent Physical Agent. (1996). Available at `http://www.fipa.org/`.

[9] H. GROENVELT, The Just-in-Time System. *Handbooks in OR & MS, Elsevier Science Publisher* Vol. 14 (1993).

[10] INDEXOS, Index of Alternative Operating Systems. (2000). Available at `http://www.indexos.com/`.

[11] N. IVEZIC ET AL., Multi-Agent Framework for Lean Manufacturing. *IEEE Internet Computing,* Vol. 3, No. 5, September/October (1999), 58–59.

[12] N.R. JENNINGS, M. WOOLDRIDGE, Software Agents. *IEE Review.* January (1996), 17–20.

[13] N.R. JENNINGS ET AL., ADEPT: Managing Business Processes using Intelligent Agents. *BCS Expert Systems 96 Conference (ISIP Track), Cambridge, UK* (1996), 5–23.

[14] R.E. JOHNSON, Reducing the Latency of a Real-time Garbage Collector. *ACM Letters on Programming Languages and Systems.* March (1992), 46–58.

[15] D.W. JUDGE ET AL., Agent Enhanced Workflow. *BT Technical Journal,* 16:3, (1998), 79–85.

[16] M. KODAMA, Strategic business applications and new virtual knowledge-based businesses through community-based information networks. *Information Management & Computer Security,* Vol. 7, No. 4; (1999), 186–199.

[17] P. MAES, General Tutorial on Software Agents. (1997). Available at `http://pattie.www.media.mit.edu`.

[18] MICROSOFT TECHNET, MS Message Queue Server Overview. (1998). Available at `http://www.microsoft.com/tec|hnet/default.asp`.

[19] NEWMONICS Inc. Making JavaT Applications a Reality in Embedded Systems Today. (2000). Available at `http://newmonics.com`.

[20] N. NISSANKE, Real-time Systems. *Prentice Hall.* (1997).

[21] H.S. NWANA, Software Agents: an overview. *Knowledge Engineering Review,* Vol. 11, No. 3, October/November (1996), 205–244.

[22] B.R. ODGERS ET AL., Technologies for Intelligent Workflows: Experiences and Lessons. *Proceedings of Agent-Based Systems in the Business Context, AAAI 1999 Workshop* (1999).

[23] C.M. PANCERELLA, N.M. BERRY, Adding Intelligent Agents to Existing EI Frameworks. *IEEE Internet Computing,* Vol. 3, No. 5, September/October (1999); 60:61.

[24] V. PERRIER, Can Java Fly? Adapting Java to Embedded Development. *Embedded Developers Journal.* September (1999), 8–18.

[25] G. RILEY, Clips: A Tool for Building Expert Systems. August 28, (1999). Available at `http://www.ghg.net/clips/CLIPS.html`.

[26] D. RIPPS, *An implementation guide to real-time programming*, Yourdon Press Computing Series, Prentice Hall. (1989).

[27] RTJ, The Real-Time for Java^{TM} (2000). Available at `http://www.rtj.org`.

[28] R. SACILE ET AL., The MAKE-IT project: Manufacturing Agents in a Knowledge-based Environment driven by Internet Technologies. *Proc. IEEE-AIWORK-2000: An international working conference and industrial expo on new advances and emerging trends in virtual enterprises and mobile computing.* Buffalo (NY), April 27–29 (2000), 191–196.

[29] W. SHEN, D.H. NORRIE, Agent-Based System for Intelligent Manufacturing: A State of the Art Survey. *Knowledge and Information Systems* (1999), 129–156.

[30] Y. SHI, M. GREGOR, International manufacturing networks to develop global competitive capabilities. *Journal of Operations Management.* Vol. 16, No. 2–3; (1998), 195–214.

[31] SUN MICROSYSTEMS, INC., The Java Language Overview. *Mountain View, CA.* (1995)a.

[32] SUN MICROSYSTEMS, INC., The Java Language Environment: a White Paper. *Mountain View, CA.* (1995)b.

[33] SUPPLY CHAIN COUNCIL, Reference Model (SCOR). (1996). Available at `http://www.supply-c|hain.org/html/scor_overview.cfm`.

[34] N.R. JENNING, M. WOOLDRIDGE, Applying Agent Technology. *Journal of Applied Artificial Intelligence.* Special Issue on Intelligent Agents and Multi-Agent Systems (1995).

[35] WORKFLOW MANAGEMENT COALITION, The Workflow Reference Model. (1995). Available at `http://www.aiim.org/wfmc/mainframe.htm`.

[36] W3 CONSORTIUM, Extensible Markup Language (XML). February 10, (1998). Available at `http://www.w3.org`.

Received: January, 2001
Revised: November, 2001
Accepted: September, 2002

Contact address:
Roberto Sacile,
DIST — Department of Communication,
Computer, and System Science
University of Genoa,
Via Opera Pia 13
Phone: +39-010-353-2284
Fax: +39-010-353-2154
e-mail: `robby@dist.unige.it`

ERNESTO MONTALDO, born in Genoa (Italy) on 1st September 1971, is an Electronic Engineer. Currently, he is a third year Ph.D. student at DIST (Department of Communication, Computer, and System Science) of the University of Genoa. His research focuses on distributed software technologies applied to manufacturing, with special attention to agent technologies.

ROBERTO SACILE, Ph.D., born in Genoa (Italy) on 30th December 1965, is an Electronic Engineer. He is a researcher at DIST, University of Genoa, Italy, working on systems and information management techniques applied to environment, manufacturing and health care, with special attention to decision support technologies.

MAURO COCCOLI, Ph.D., born in Genoa (Italy) on 5th May 1970, is an Electronic Engineer. He is a researcher at DIST, University of Genoa, Italy, working on automation, control and robotic systems.

MASSIMO PAOLUCCI, born in Genoa (Italy) on 15th February 1961, graduated as an Electronic Engineer in 1986 from the University of Genoa. He received the PhD in electronics and computer science at DIST of the University of Genoa in 1990. Currently he is working as an assistant professor at DIST.

ANTONIO BOCCALATTE, born in Genoa (Italy) on 24th November 1951, he took a Degree in Electronic Engineering in 1976 from the University of Genoa. In 1987 won a chair at the University of Genoa, currently he is professor of "Data Base Management Systems" at the Faculty of Engineering. He is author of more than 70 scientific papers presented at international congresses or published in international journals. His research interests include: system architecture and artificial intelligence, decision support systems and database.