INDUSTRIAL MULTI-AGENT SYSTEMS
INTEGRATION: AN APPROACH BASED ON
INFORMATION EXCHANGE AMONG
STANDARDS

Leandro Loss, Ricardo J. Rabelo
(loss@gsigma-grucon.ufsc.br), (rabelo@das.ufsc.br)
UFSC - Federal University of Santa Catarina, DAS - Department of Automation and Systems
Brazil

Abstract: This work proposes an approach to facilitate the integration in the communications among industrial systems and shop floor devices by means of a global and integrated environment. This environment is open, makes use of a number of standards, providing a transparent way to integrate the standards CORBA, KQML, MMS and XML in all the required levels. This proposal focuses on how the communication of Industrial Multi-agent Systems can be harmonized and encapsulated, comprising all the necessary actors and, at the same time, maintaining the essential properties of each protocol within the borders for which they have been developed/proposed. It means to provide a leaner interface to any external application that needs to have access to shop floor data, without requiring the knowledge of the syntax details of both protocols used and the particularities of each machine's controller that it is necessary to communicate with.

Key words: MMS, XML, KQML, CORBA, Multi-agent Systems.

1. INTRODUCTION

Huge efforts have been made by the researches in the last years to design supporting platforms for Virtual Enterprises (VE). In spite of their complexity and importance, less attention has been given to the integration of these platforms with the shop floor. Actually it is extremely important as
the effective operation of a VE can only be accomplished successfully with reliable data that comes from the heterogeneous enterprises’ shop floors. Therefore, open, interoperable and easy-to-use interfaces are mandatory for that integration.

The development of industrial flexible systems has arisen to support the enterprises necessities about quality, flexibility and efficiency. One of the key aspects for this development is the integration of industrial devices, usually heterogeneous (different functionalities and protocols). A flexible industrial system is composed of a group of numeric control machines, industrial robots, automatic assembly and inspection devices, interconnected by means of automatic material handlers and storage systems, controlled by integrated computer systems. Flexible systems have given a basis to improve the enterprises competitiveness on the market through costs reduction, increasing products quality, and providing products delivery in a diversity way with a shorter time to fit the market demand.

This kind of integration means to make all the required entities involved in the processes exchange information among them, like: manufacture devices with machines, the machines among machines and other external applications through the network. The traditional architectures proposed so far have worked with the idea of a central controller/supervisor to all the shop floor integration, usually a PC with a supervisory system. However, this approach increases the difficulty about systems scalability, system upgrading and an appropriate use of the industrial networks on the shop floor. As a way to solve these problems, the increasing computational power processing with lower prices, the architectures are evolving from turn-key, homogeneous, centralized and hierarchical, to open, heterogeneous, distributed and decentralized architectures [1], [2]. On the other hand, this has brought up some difficulties to manage distributed systems in all of its different levels. A large number of approaches are arising to support the system development with these requirements. The use of the Multi-agent Systems approach emerges as a prominent alternative and it is used in this work.

There is a large number of works applied on information integration in the several existing levels inside the enterprises, as presented in Rabelo [2], Usher [3], Barber [4], Guyonnet [5] and Silveira [6]. However, most of the implementations mentioned in these works have focused on the individual aspects of communication and integration, each one restricted to the borders of the application and/or technology. For example, the multi-agent community has put its efforts on the KQML / FIPA-ACL language and ontologies, without considering the robustness of the communication infrastructure and the integration with other open (industrial) systems. The computer networks and industrial networks communities have been
developing communication and interoperation protocols to support the integration between manufacturing equipments and instrumentation networks, but they seldom give attention to the application requirements and the high level protocols. Even the multi-agent systems described in the literature do not use to take into account the real industrial requirements.

This work proposes a global integration environment for industrial multi-agent systems. This environment is open, uses a number of standards, and provides a transparent way to integrate the standards CORBA, KQML, MMS and XML in all the required levels [7], extending the work done by Rabelo [2] and Ariza [8].

This proposed approach addresses the support of Virtual Enterprises (VE) where a VE is a temporary alliance of enterprises that come together to share skills or core competencies and resources in order to better respond to business opportunities and whose cooperation is supported by computer networks [9]. The VE can make use of the Industrial Agents to help in the supervision and execution of tasks getting/sending information from/to the shop floor. In this way, the VE members can work with the real state on shop floor bringing agility and flexibility related to the opportunities that are arising on the market place.

This paper is organized as follows. Section two points out the main problems on systems integration from the communication protocols point of view. Section three describes the proposed approach. Section four specifies the entities involved in the implemented scenario and shows the messages exchanged among the system’s agents in all required levels. Section five stresses some preliminary conclusions upon the proposed approach.

This work has been developed in the scope of IFM (Institute Factory of Millennium) Project [10].

2. THE PROBLEM

No consistent efforts have been observed towards to a global vision related to the information integration in multi-agent-based industrial systems. The current efforts are not focused proposals to join the several specific initiatives into a single platform. This global vision is one of the main requirements in industrial systems, especially in modern industrial multi-agent systems. This reality has imposed some difficulties to the information integration, which is an essential basis for (agile) decision processes that should be consistent and indeed coherent with the state of the shop floor. The problem becomes more serious when this communication is
desired to occur on open platforms, working with robust protocols, as most standardized as possible, and supporting an efficient communication among all involved actors – including people.

As previously mentioned, with the evolution of the supervision model to distributed systems, all the centralized services that were located in a single supervisor were “split” into several distributed supervisors to improve the systems performance and reliability. These distributed supervisors can have their intelligence and autonomy enhanced through an “agentification” [11], where each industrial device is represented by an industrial agent. Thus, they can work cooperatively to solve the supervision problems.

This work aims to harmonize and to encapsulate the industrial multi-agent systems communication with all the required actors, maintaining the essential properties of each protocol within the borders it was designed. It means to allow an external application to access shop floor data with a leaner interface, without knowing the syntax details, particularities and incompatibilities of each machine’s controller that is desired to communicate with. This is extremely important as the lack of “transparency” is a traditional problem for a suitable integration between high-level and low-level information inside the enterprises.

3. PROPOSED ENVIRONMENT

It is proposed an environment where several technologies and communication standards are integrated: multi-agent systems, the KQML, CORBA and MMS protocols, and the XML language. More specifically, a communication layer that allows one agent to communicate with other agents (or other systems) through CORBA, using KQML messages modeled in XML, and the messages content expressed in MMS [7].

CORBA (Common Object Request Broker Architecture) [12] is used as a supporting infrastructure protocol to the communication between agents and shop floor, among the agents themselves, and between heterogeneous/distributed agents and other heterogeneous systems. These heterogeneous systems can be agent-based or not. CORBA is a standard communication architecture for distributed systems. It allows a heterogeneous and distributed collection of objects to interoperate through a common interface, where the “visible” services of each agent/system can be invoked.

MMS (Manufacturing Message Specification) [5] [13] [14] works as an application protocol. It allows the data exchange among devices on shop floor, agents and human operators, through remote access. MMS is an ISO standard which defines a set of messages to monitor and to control industrial
devices from an external application. Although MMS is not an object oriented language, it models the system in objects and provides an interface for each of them. The most important object is the Virtual Manufacturing Device (VMD). A VMD virtualizes the services of an industrial device, making them available for the external applications which need to have access to. The MMS makes transparent the details of the industrial devices implementation.

KQML (Knowledge Query Manipulation Language) [15] was chosen as a high level application protocol. It is message-oriented and allows the communication among the agents involved in the communication with industrial devices. KQML can be used as a language allowing an application program to interact with an agent system, i.e. the knowledge sharing, trying to solve problems in a cooperative way. KQML includes some primitives ("performatives") that are used to "communicate" facts, to make questions, and to find another agent groups [16].

XML (Extensible Markup Language) [17] works like a common/neutral language to support the interoperability among the several protocols and languages used. XML is a language that uses tags, like HTML. It is becoming the standard for data exchange among distributed and heterogeneous applications. The XML messages structure is previously defined in a DTD (document type definition) to formally describe its structure. A DTD provides a grammar that indicates which structures can happen, and in which sequence they should occur.

Agents can be defined as autonomous and distributed software modules that exchange information when they need through the network to solve problems that are beyond their resolution capacities [2]. A multi-agent system - therefore a collection of agents - can have several control architectures (hierarchical, federated, completely decentralized, etc.) and a consequent number of agent “classes”. Every class is created according to the characteristics of each system. In the proposed model there are three agent classes, working with shop floor devices communication: CommonAgent, AgentManager and AgentMachine (Figure 1).

The agents belonging to the CommonAgent class are the ones that do not know how the shop floor functionalities work in fact, even though, they know how to communicate with the agents inside the AgentManager class. The CommonAgents represent all the “high level” actors, like users, agents inside a multi-agent system or other systems, which want to communicate with the industrial devices. Whenever the CommonAgents want to get information from industrial devices or to send information for these devices, they should make their interactions with the industrial devices via an
AgentManager. The CommonAgents use the KQML protocol to exchange high-level messages with the AgentManagers.

The AgentManagers are representatives of the AgentMachines inside the general system community. They make possible the communication of industrial devices with the agents, the other agentified systems or not, and the users (CommonAgents). The AgentManagers are also responsible for the “intelligent processes”, like as the authorization of who can see something or who can act in a certain way with the industrial device (for example, to turn on/off the machine, to stop the process, etc.). When an AgentManager receives a request to work with a device or to give some information, it makes the communication transparent and makes a “bridge” with the AgentMachine. This bridge is built up through a KQML/XML message, and the real command is specified in MMS.

![Diagram](image)

*Figure 1*

The AgentMachines represent the industrial devices themselves. In fact, they act with the legacy systems usually built when initial implementations of local supervision systems are carried out. To this end, it is necessary to wrap the functionalities of those systems by constructing the respective VMD. In this way they can be activated both locally and remotely by other applications. Since this layer is represented by an agent, it is called “agentification” of industrial devices controllers [11]. Therefore, the AgentMachine will communicate in MMS with the VMD inside each legacy system (*Figure 1*). Rabelo [2], Ariza [8], SISCO [14] and Shang [18] have
highlighted the advantages to use MMS through VMD, regarding the interoperability and independence provided. It aims to support the information access to the shop floor through a (multi-agent) application without knowing the details of how the functions are implemented in the real devices. The AgentMachine is also responsible to report to the AgentManager any problem detected during the process execution. It provides conditions to the AgentManager actuates according to the real state of the shop floor.

4. IMPLEMENTED SCENARIO

4.1 Scenario

A prototype was developed considering a Flexible Manufacturing Cell (FMC) composed of the following equipments and respective PLCs (Programmable Logical Control):
- 1 position table: to feed the milling machines;
- 2 milling machines: for milling, one vertical milling machine and one horizontal milling machine;
- 1 robot: to pick and place the parts inside the manufacturing cell.

In fact, these equipments were simulated, considering that the current FMC installation does not support some requirements of the involved protocols. Nevertheless, all processes are distributed and hence each machine’s controller and the machine itself is a distributed process launched in the network / different PC.

As these industrial devices are service “providers”, their respective VMDs are necessary to exist in order to allow other applications to access them through MMS. The communication between the AgentMachine and the VMD is made using CORBA IDLs (Interface Definition Language). An IDL file describes the data type, the operations and the objects that a client can request and the server can put available. For example, an IDL file could describe a VMD interface defining MMS messages like CreateProgramInvocation, DeleteProgramInvocation, Start,DataExchange and Status. A client application using this IDL would be able to request the services execution to the “CORBAlized” VMD, and the server would execute the real function.
4.2 Communication Messages

4.2.1 Implemented MMS Messages

The basic actions made by the AgentMachine to the initialization and operation are described below, taking the position table as an example.

4.2.1.1 Position Table’s PLC

It was necessary to create a program (called CP_MESA) invocation responsible for the initialization of the position table. In this case, the service Create_Program_Invocation is used to execute the procedure CP_MESA, responsible to start the operation with the position table. It is done through a service called Start in the CP_MESA program invocation object. The position table operations are invoked via remote procedure calls (class Interlocked_Control, defined in a Companion Standard for PLCs [6]). The MMS messages implemented in the position table’s VMD are related to the following procedures:

- mm_Create_Program_Invocation (CP_MESA) – it creates an invocation to the CP_MESA program, responsible for the position table starting;
- mm_Delete_Program_Invocation (CP_MESA) – it finishes CP_MESA program invocation;
- mm_Start(CP_MESA) – it executes the procedure CP_MESA to start the position table operation;
- mm_Data_Exchange (int pos) – it rotates the pallet to the position that it will be handled by the robot or by the vertical milling machine;
- mm_Status () – it returns the physical status of the industrial device (position table).

The MMS messages “translate” the attributes and methods described in industrial device’s VMD to the client applications. The client applications are responsible to handle them properly.

4.2.2 Mapping MMS Messages

The mapping of VMD objects as a CORBA IDL can be done by inheritance of a generic VMD in a dynamic way. Starting from this generic VMD, it is possible to construct a specific VMD to the industrial device as explored in Ariza [19]. However, in the presented implementation, each VMD was mapped manually due to a reduced number of necessary MMS messages.

Each service described in a CORBA IDL corresponds directly to a MMS message that will access the VMD. For example, whenever an AgentMachine (as a client) invokes the program CP_MESA (responsible for
implementing the remote procedures call) of the position table, the service `boolean CreateProgramInvocation(CP_MESA)` in the IDL will call the message `mm_Create_Program_Invocation()` in the VMD object (server).

### 4.2.3 KQML Messages AgentManager : AgentMachine

During the implementation only four performatives were necessary: Ask-if, Tell, Reply and Sorry. The other KQML performatives were not used considering that the messages exchanged between AgentManager and AgentMachine are only used for notifications or information gathering. The Tell performative is used to communicate an action, for example, to change the pallet position in the position table, through the command `DataExchange(Pos)`. The Ask-if performative is used to request the physical state of the position table, through the Status() command. The Reply performative is used to send an answer for a requisition. The Sorry performative indicates that an AgentMachine cannot execute the service requested. The implemented performatives are shown in Table 1.

**Table 1 – Performatives**

<table>
<thead>
<tr>
<th>Performative/Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ASK-IF</td>
</tr>
<tr>
<td>:sender AgentManager</td>
</tr>
<tr>
<td>:receiver AgentMachine</td>
</tr>
<tr>
<td>:reply-with AgentManager</td>
</tr>
<tr>
<td>:in-reply-to NONE</td>
</tr>
<tr>
<td>:language MMS</td>
</tr>
<tr>
<td>:content Status()</td>
</tr>
<tr>
<td>:ontology NONE)</td>
</tr>
<tr>
<td>(TELL</td>
</tr>
<tr>
<td>:sender AgentManager</td>
</tr>
<tr>
<td>:receiver AgentMachine</td>
</tr>
<tr>
<td>:reply-with AgentManager</td>
</tr>
<tr>
<td>:in-reply-to NONE</td>
</tr>
<tr>
<td>:language MMS</td>
</tr>
<tr>
<td>:content Start()</td>
</tr>
<tr>
<td>:ontology NONE)</td>
</tr>
<tr>
<td>(REPLY</td>
</tr>
<tr>
<td>:sender AgentMachine</td>
</tr>
<tr>
<td>:receiver AgentManager</td>
</tr>
<tr>
<td>:reply-with NONE</td>
</tr>
</tbody>
</table>
4.2.4 KQML Messages in XML

The KQML messages exchanged among the proposed agents are modeled in XML, previously defined in a specific DTD (Figure 2).

```
<!DOCTYPE kqml SYSTEM "standard.dtd">

<kqml msg_id="1">
  <perf name="Start()"/>
</kqml>
```

Figure 2

Whenever a KQML message (packed in XML) arrives, the agent that receives the message verifies if the message matches its specified DTD previously agreed. If a message does not match, the receiving agent replies to the sender reporting the inconsistency. If it does, the receiving agent unwraps the KQML-XML message and puts the KQML data inside an object called `kqmlMessage` so that the message can be treated properly afterwards. When an agent needs to send a KQML message, it wraps the
KQML information from the *kqmlMessage* object to a XML message (Figure 3) and then the message is sent out.

```xml
<Performative name="ASK-IF">
  <sender>AgentManagerMesa</sender>
  <receiver>AgentMachineMesa</receiver>
  <reply-with>AgentManagerMesa</reply-with>
  <in-reply-to>NONE</in-reply-to>
  <language>MMS</language>
  <content>Start()</content>
  <ontology>NONE</ontology>
</Performative>
</KQML>

*Figure 3*

The adoption of XML to represent KQML messages was made because XML acts as a “neutral language”, which eliminates some restrictions among legacy systems. Thus, it is enough for the agents to know which DTD should be used to format or to interpret the message hence the message content becomes “transparent” to be treated.

### 4.2.5 Other KQML Messages CommonAgents : AgentManager

The KQML messages exchanged between the *CommonAgent* and the *AgentManager* are different from the messages exchanged between the *AgentManager* and the *AgentMachine*. While the *content* attribute of the messages exchanged between the first ones contains MMS message, the second ones contains “abstract commands” that will be translated by the *AgentManager* into MMS messages and vice-versa, using a given *ontology*. An ontology defines a common vocabulary for those who needs to share information about a certain domain (like industrial devices). The ontology includes basic concepts and definitions from a domain and the possible relationships among existent domains [20].

The *Figure 4* shows how the messages exchange are among the agents.
The prototype developed has used the multi-agent platform Massyve Kit 3.0 [21] with C++ Builder 5.0 development tool [22]. The ORB CORBA used to provide the communication between the agents and the modules (that simulate the industrial devices) was ACE-TAO [23], ACE version 5.1 and SO version 1.1, in the Windows XP operating system.

5. CONCLUSIONS

This paper has proposed a platform to join several standard protocols to integrate the information from the enterprise lowest level (the shop floor – MMS), to the highest level (knowledge sharing – KQML), allowing the message exchange in a robust way in an open environment (with CORBA) and syntactically interoperable (with XML).

The proposal brings up a harmonized vision illustrating that it is possible to encapsulate the communication at the different enterprise levels. The several actors involved in the process does not have to know and do not have to understand all the protocols and the details of industrial devices involved in the communication.

During the tests a set of messages were exchanged between the agents and the (simulated) shop floor devices. The CommonAgents executed requisitions and waited for the answers about operations success, failure or mistakes. Errors were also simulated in the shop floor devices; the AgentMachines, responsible for the devices supervision, sent
“warnings/alerts” messages to AgentManagers noticing the problems correctly.

The messages transmission between the agents and the industrial devices were “instantaneous”, considering that all the software processes involved (agents and the simulated equipments) were launched in a LAN. Thus a more real evaluation of the time spent with communication could not be carried out at all. Besides that, even considering that the communications were carried out properly in all phases, it is possible to preliminary affirm that some communication aspects are extremely sensible to real time requirements of shop floor applications. The authors point out that it is a price to pay when is desired to bring higher levels of integration, flexibility, intelligence and autonomy related to the distributed industrial systems, accepting the “weight” of the technologies used in the supporting platforms.

Next steps of this work are: 1) to evaluate the advantages of using CORBA as the infrastructure to the whole enterprise, and to study deeper the use of a real time ORB to satisfy the usual temporal requirements of shop floor applications; 2) use of hybrid architectures, based on distributed/federated databases for agents, where the data is not sent by an agent to another by means of a high level protocol (as in the usual way), but rather accessing it via federated queries; 3) the conception of an ontology to describe the shop floor devices and their attributes.

6. ACKNOWLEDGEMENTS

This work has been supported by CNPq – The Brazilian Council for Research and Scientific Development.

7. REFERENCES

1 – Conference Proceedings

2 – Conference Proceedings

3 – Conference Proceedings
Leandro Loss, Ricardo J. Rabelo


4 – Conference Proceedings

5 – Technical Report

6 – Master Degree Dissertation

7 – Master Degree Dissertation

8 – Conference Proceedings

9 – Conference Proceedings

10 – Project
IFM [www.ifm.org.br](http://www.ifm.org.br)

11 – Journal Article

12 – Technical Report

13 – Technical Report
SISCO 1996 – System Integration Specialists Company, Inc. – Device GateWay; 6605 19 Mile Road, Sterling Heights, MI 48314-1408, USA. [http://www.sisconet.com](http://www.sisconet.com)

14 – Technical Report

15 – Technical Report

16 – Technical Report

17 – Book

18 – Conference Proceedings

19 – Conference Proceedings

20 – Technical Report

21 – Technical Report

22 – Technical Report
Borland; 2000 - http://www.borland.com

23 – Technical Report