61 THE GLOBAL AUTOMATION PLATFORM: AN AGENT-BASED FRAMEWORK FOR VIRTUAL ORGANIZATIONS

Franco Guidi-Polanco[1]
Claudio Cubillos[2]
Giuseppe Menga[2]
[1]*Pontificia Universidad Católica de Valparaíso, Chile*
franco.guidi@ucv.cl
[2]*Politecnico di Torino, Italy*
{claudio.cubillos, menga}@polito.it

This work presents our agent-based architecture for the development of Global Automation Systems. These systems consist of software applications that manage all the processes in a network of enterprises, in distributed, decentralized and autonomous way.

1. INTRODUCTION

Global automation is a new concept that transfers and extends classical process control and factory automation ideas to a large scale distributed environment. It exploits the current trends of information and telecommunication technology, especially embedding on the Internet data acquisition and control devices.

Global automation systems consist of software tools and applications that process, handle, and control the business, logistic, production (and today the surrounding environment) activities of a network of enterprises. They can be considered as the nerve system of a world-wide organization. They facilitate the exchange of information among customer offices, business units, departments, production cells, and distribution centers. They allow a variety of functionalities to be automated, that otherwise would require the direct intervention of human operators. Global automation systems are built on complex concurrent, distributed, and autonomous modules that integrate functionalities drawn from a variety of application areas (e.g., telecommunication, databases, virtual reality, artificial intelligence).

The need for an architectural reference model to implement global automation systems has been envisioned. Such model has been identified as an agent-based framework that offers a communication infrastructure and a uniform system architecture. We call this infrastructure the "Global Automation Platform" (GAP). The GAP is a framework sufficiently flexible to be used at multiple levels in a

global automation system, ranging from the monitoring and control system for a single production cell (the virtual SCADA), to the integration of the supply chain of a multi-national corporation (the virtual factory), up to the construction of a global organization.

Concretely, this work proposes an agent-based framework for the development of global automation systems. Such a framework is responsible to support three aspects of these systems: (1) the definition of base elements from which global automation components will be extended; (2) the support for the interactions among participants; and (3) the basic services required in global automation systems.

2. BACKGROUND: THE GLOBAL AUTOMATION

In the era of Computer Integrated Manufacturing (CIM), an automation system was conceived as a strong and rigid hierarchy of control layers: Facility, Shop, Cell, Workstation and Equipment. According to the USA-NBS CIM reference model (Maclean et al., 1983), each layer is populated with a set of control modules (the device controller, the workcell controller, the cell controller, etc.) with precise responsibilities. In particular, a higher layer control module coordinates the control modules below it. The flexibility of such systems is limited to the possibility of reconfiguring the production process off-line by re-programming each control module. A local area network or a hard-wired field-bus represents the communication medium between the different factories' sub-systems. The inter-factory communication is usually handled via telephone, fax, or e-mail. An example of this old style architecture is described in our past research (Brugali et al., 1999).

In the era of Internet, global automation systems are conceived as flat interconnections of autonomous and decentralized decision making/control modules dominated by the two concepts of "heterarchy" and "proactivity": the former means that no hierarchy in decision making is enforced, the latter that each partner takes the initiative to reach a decision (e.g. planning production) and the global behavior of the system is an "emerging behavior" (Bemelman et al., 1999). Control modules have decision-making capabilities and coordinate their activities by exchanging data and events according to a peer architectural model and common protocols.

Thus, creating global automation systems requires the adoption of an organizational model that recognizes autonomous and cooperating entities as system's elementary building blocks. These entities have the ability to determine their actions and to develop cooperating societies.

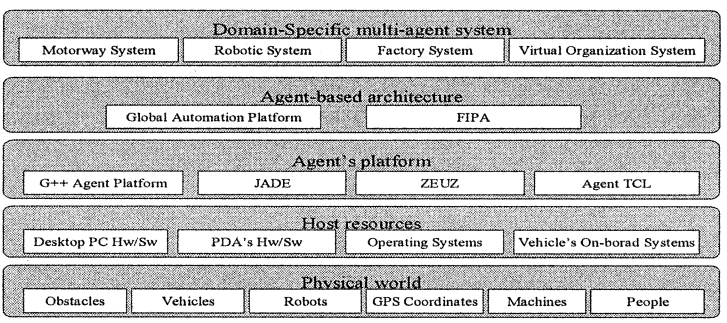

Fig. 1. The Abstract Architecture for MAS.

3. THE ABSTRACT MODEL FOR AGENT-BASED ORGANIZATIONS

In general terms we envision the use of agents as a way of representing software architectures. This vision is constructed as the abstract model depicted in Figure 1. The abstract model is composed by the following layers:

- *Physical world*: it is composed by things pertaining or observed in the real world (e.g. objects in mobile robot's environment, wired or wireless communication networks, computational systems, humans, traffic lights, etc.), and concepts conventionally adopted for its characterization (e.g. geographical coordinates obtained from a GPS service, temperatures, data transmission latency, water flows measurements, etc.). The physical world is conceptualized as a multidimensional space surrounding agents accomplishing physical-related tasks.
- *Host resources*: represent a computing system in terms of CPU, runtime memory, data storage, data communication, operating system, and peripheral devices (e.g. thermometers, valves, engines, etc.) where the agent software is executed. The hosts can be seen from two perspectives: (1) internally, they offer the runtime environment for the agent platform, so hosts must satisfy a set of minimum hardware/software requirements imposed by the platform's software (or in an opposite point of view, the agent platforms must be designed to be executed in specific categories of devices); and (2) externally, the host resources offer the "physical body" for agents, that is, the means for agents to perceive the physical world, and/or to influence it with their actions. Examples of common environments are desktop computers, vehicles' on-board units, PDAs, and mobile robots.
- *Agent platform*: corresponds to the software that offers the base classes to build agents, and to virtualize environment-dependent services (e.g. interfaces to peripheral devices, motors, databases, network communication, etc.). It also offers the execution environment that controls the entire agent's life-cycle, and regulates its interactions with other agents, and other resources. As it was stated above, agent platforms must be designed for their execution in devices with different hardware (e.g. PDAs, mobile phones, desktop computers) and software capabilities (in terms of operating systems and programming languages). Examples of agent platforms are JADE (Bellifemine et al., 1999), and AgentSheets (AgentSheets, 2004) (a comprehensive list of agent platforms can be found in (AgentLink, 2004)).
- *Agent-based architecture*: represent a reusable architecture to support the development of different kind of agent-based systems. The architecture specifies a set of common services (e.g. directory facilitator, yellow pages, etc.), and a framework of communication/content languages (e.g. ACL (Genesereth and Ketchpel, 1994), KQML (Finin et al., 1993), etc.) and interaction protocols (e.g. FIPA Request Interaction Protocol (FIPA, 2002), the Contract Net protocol (FIPA, 2002b), etc.), necessary to achieve interoperability among agents. The services offered by the architecture can be implemented by service agents (such as a yellow-pages agent), or as environment-dependent service (e.g access to

some kind of physical device). An example of a particular agent-based architecture is specified by FIPA standards (FIPA, 2002c), which was conceived to obtain interoperability between different and generic agent systems.

- *Domain-specific multi agent system (DSMAS)*: corresponds to a concrete instance of a multi-agent system, where domain-dependent agents are designed to represent real-world services and systems, and interactions among them are well defined. In most cases agents at this level are abstractions of real entities pertaining to the application domain. DSMAS architectures can be reused within the scope of the context they were created for. A reusable DSMAS architecture constitutes an agent-based framework for the development of systems within its domain. DSMAS architectures are supported by the services offered by the agent-based architecture.

The model proposed above has three main characteristics:

- *Decouples design responsibilities*: the model presents the different aspects related to a multi-agent architecture in a separated way. Therefore, the design responsibilities can be clearly identified and assigned to different development projects or teams.
- *Promotes high cohesion within each layer*: components within each layer are closely related from the functional and communicational point of view, in such a way that their interactions are optimized.
- *Clearly emphasizes the environment*: traditional agent architectures consider the environment implicitly, in most cases just as a mere communication supplier. In our model, the environment is distinguished as a physical and a virtual one.

4. AGENT TECHNOLOGIES IN GLOBAL AUTOMATION

The adoption of agent systems as enabling technologies for the development of distributed organizations' infrastructures is currently matter of research. In particular, the agent technology seems not only to satisfy the demand for high flexibility requested by enterprise-wide integration (Rimassa, 2004), but also to provide approaches to support autonomous self-configuration and self-adaptability of their activities in their operational environment (Gou et al., 2003), (Niessen, 2001).

Historically, the research behind agent systems addressed the agent microarchitecture, that is, the internal structure that describes the underlying intelligence model. This was especially motivated by trends coming from distributed artificial intelligence (DAI). More recent platform implementations are aware to accomplish the FIPA standard, which is focused in achieving interoperability among heterogeneous agent systems. Now that agent internals and interoperability challenges have been achieved, the next step is to provide a more enhanced environment, able to empower the relationships not only among agents, but also between agents and their physical environment.

In this context, the GAP corresponds to a high-level reusable software infrastructure designed to implement Global Automation Systems. It is an agent-based framework, which is made up of a communication infrastructure, logically structured as an information bus, which is embedded in a uniform system

architecture. Agent properties that we have specially considered while developing the GAP were autonomy, social ability, reactivity, pro-activity and mobility

4.1 The physical environment

The GAP provides the agent-based architecture to interface the physical environment (*host resources* and *physical world* in the model of Section 3) with the domain-specific agents in a distributed system. The components of the physical environment depend directly on the domain that the system pertains to.

4.2 The G++ Agent Platform

Several agent platforms are currently offered for the development of multiagent systems. However, to build the GAP we have developed our own agent platform, that we call the G++ Agent Platform (Guidi-Polanco et al., 2004). The G++ Agent platform is a Java framework for the development of agent-based systems, focusing in architectural aspects related to components distribution and communication. Our work was motivated by the need for creating and integrating autonomous systems through geographical scale cooperation networks. Under such a scenario, the possibility of leaving the agent with all the responsibility for its integration with the environment (and consequently other peers) implied the agent overload and the replication of a series of complex functionalities. Due to this, was noticed that existing platforms were not suitable to accomplish these requirements.

Inspired in communication models from telecommunication systems, the platform provides a strong and reliable communication infrastructure that supports wireless communication among agents hosted in physically mobile devices.

Another distinguishable feature of the G++ Agent Platform is the possibility to integrate within the system other non-agent systems (e.g. legacy applications) by using the provided API.

4.3 The Agent-Based Architecture

The Global Automation Platform (GAP) is the agent-based architecture which offers a collection of components and services specially intended for the development of global automation systems. Specifically it offers *communication standards* and a set of supporting *services*. The former defines the languages that will be used for exchange of information between entities participating in global automation systems. The latter, the set of services made available to support distributed integration and collaboration. Three services are offered at this level:

- *Messaging*: it provides persistence and reliability in direct messaging between senders and well-defined receivers.
- *Event distribution*: it implements the asynchronous publish/subscribe communication model.
- *Service brokering*: it supports dynamic reconfiguration of the relationships between service providers and consumers.
- *Gateways*: allows interactions through systems pertaining to different domains

5. CASE OF STUDY: A MOTORWAY CONTROL SYSTEM

Generally the motorway network of a country is split into a number of sub-networks under the responsibility of several motorway administrations (see Figure 2). Each administration is in charge for the management (technical and financial) of its domain, and it maintains a relationship with the other administrations and with the national bodies (Ministry of Transport, Firemen, Civil Protection, Police). The functions of each administration are at two levels: (1) *Low level*: toll collection, on-road data acquisition, and traffic control; and (2) *High level*: maintaining the network database, monitoring of the whole network information flow, planning the network load (high level traffic planning), supporting centralized network facilities (data base of traffic events related to network sections within their own responsibility, call center for traffic information), managing emergencies, maintaining relationships with the other administrations.

Traffic control is the activity of regulating access to the highway, and settings on-road signals in order to avoid traffic saturation and optimize transit times. Traffic control is achieved -at cell level- by regulating incoming traffic at the toll plazas. Inputs for the traffic control process are forecasts of this incoming flow and forecasts of the traffic conditions of neighbor cells. These forecasts make it possible to determine the optimal traffic flow (the one that actually should enter the motorway) that minimizes the global transit times. If the current incoming flow exceeds the computed prevision, traffic control will introduce a delay at toll plazas. Priority Schedulers of each cell can negotiate among these outcomes in order to distribute the access priorities given to different cell traffic controllers during the day, with the objective of achieving a globally optimum compromise.

Our model for traffic monitoring and control subdivides administrations into disjoint *motorway cells*. A motorway cell is an autonomous control module that performs low-level monitoring and control of local traffic and exchanges traffic status data and events with neighbor cells. The graph of motorway cells conforms to a cellular interconnection architecture that supports decentralized control. The short-term traffic control of the whole motorway network emerges as an intrinsic property

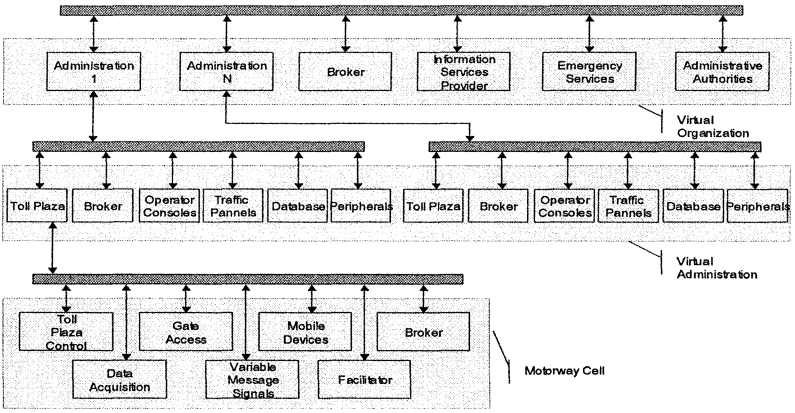

Fig. 2. The motorway system

of the cellular cooperative structure. In the following subsections, the traffic and control motorway system is described using the abstract model of Section 3.

5.1 *Physical environment* (level 1) and *host resources* (level 2)

A cell's *physical world* is mainly composed by:

- *Operative Toll-Plazas*: perform classical toll-plaza activities (e.g. collecting tolls, collection of data to transfer to the higher decisions level) including traffic access regulation at gates under the control of a Major Toll-Plaza.
- *Major Toll-Plaza*: extends the standard function of a toll-plaza with the responsibility of cell's traffic monitoring and control. It also performs traffic access regulation at their own access gates following its own traffic decisions.
- *Fixed Devices*: are the set of sensors, traffic signals and other traffic control facilities assigned to fixed places along the motorway.
- *Wireless Mobile Devices*: are sensors, signals, and other intelligent units mounted on users' and maintenance vehicles moving along the motorway.

In the motorway system, the *host resources* correspond to every kind of computing device available for the execution of agents. These devices can be fixed computing systems such as toll-plaza's desktop computers, or wireless mobile computers mounted on signals and vehicles.

5.2 The *agent platform* (level 3)

The base environment for the traffic monitoring system is made-up by the *G++ Agent platform* and the set of services defined for global automation systems (message and event distribution, and gateway agents).

5.3 The *GAP agents and services* (level 4)

The GAP services are used to achieve interoperability among different participants (internal and external to an organization). For example the *Brokering service* is responsible for maintaining the address of agents pertaining to each administration domain, in such a way that they could be automatically identifiable by other agents interested in interacting with them. The *Gateway service* enables agents to be reachable by other agents pertaining to different organizations. The *Messaging service* is used to support reliable communication mainly with wireless devices. The *Event service* is used by monitoring agents of toll-plazas to subscribe to relevant events occurred within their domain, in such a way to opportunely react in case of accident or traffic jams.

5.4 *Domain agents* (level 5)

The traffic monitoring system is made up of two main components that are implemented by specializing the GAP framework agents: the General Administration Agent (GA-Agent) and the Motorway Cell Agent (MC-Agent).

- *GA-Agent*: it performs high-level monitoring and control of motorway traffic, and is also responsible for dealing with special events (e.g. emergencies). It accesses a database that stores information about high-level daily traffic measurements, reported events and toll data, which is the result of interactions either with MC-Agents within the administration, or external organizations.
- *MC-Agent*: is an element of a cellular decentralized distributed control system. The MC-Agent resides in the major toll plaza of the respective cell. It

encapsulates its own forecasting function and is able to control the traffic in its section. The cell has a state represented by *density*, *queue length* and *speed (flow)* as a time function of its internal traffic, which are the results of real-time processing of data collected by on-road sensors (e.g. vehicle detectors, video-cameras) and the forecasted incoming traffic. Each MC-Agent communicates directly with local devices, and through *Gateways* agents with its peers.

- *Device Wrappers:* they are agents that interface road traffic devices, such sensors or gates. In the case of gate access systems, they perform their operations under priority access cycles determined by the scheduling process.

6. CONCLUSION

A layered model for the implementation of global automation systems has been introduced. This model allows the development of agent-based software infrastructure in distributed domains, ranging from intra-factory applications to wide-range virtual enterprise organizations.

4. REFERENCES

1. AgentLink "Software Products for Multi-Agent Systems". Technical report. Europe's Network of Exccellence for Agent-Based Computing , 2004.
2. AgentSheets "Getting Started with AgentSheets". Available at http://agentsheets.com, last visited: January 20, 2004.
3. Bellifemine F., Poggi A., and Rimassa G. "JADE - A FIPA-Compleant Agent Framework". CSELT Internal Technical Report, 1999 .
4. Bemelman R., Tharumarajah A., Welgama P., and Wells A.J. "Application of a Behaviour-Based Scheduling Approach for Distributed Scheduling of an Assembly Shop". Production Planning and Control, vol. 10, n.3, 1999; 266-275.
5. Brugali D., Menga G., and Aarsten A. "A Case Study for Flexible Manufacturing System". Domain-Specific Application Frameworks (Fayad, Johnson, Schmidt eds.). John Wiley & Co. 1999 .
6. Edwards W.K. "Core Jini". Prentice-Hall, Inc. Upper Saddle River, New Jersey, 2001.
7. Finin T., McKay D., Fritzson R., and McEntire R. "KQML: An Information and Knowledge Exchange Protocol". In Proc. of the International Conference on Building and Sharing of Very Large-Scale Knowledge Bases (December 1993) .
8. FIPA. "FIPA Request Interaction Protocol Specification". Standard N. SC00026H .December 3, 2002.
9. FIPA. "FIPA Iterated Contract Net Interaction Protocol Specification". Standard N. SC00030H December 3, 2002.
10. Genesereth M.R., and Ketchpel S.P. "Software Agents". Communications of the ACM, vol. 37, n. 7 1994; 48-53.
11. Gou H., Biqing H., Liu W., Li X. "A Framework for Virtual Enterprise Operation Management". Computers in Industry 2003; 50: 333-52.
12. Guidi-Polanco F, Cubillos C, Menga G. The Agent-Based GAP: A Framework for Global Automation Systems, Proceedings of the Thirteenth IEEE Internatonal Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), Modena (Italy), 2004; 3.
13. Maclean C et al. A Computer Architecture for Small-Batch Manufacturing. IEEE Spectrum, vol. 20, 1983;5 .
14. Niessen, M.E. Agent-based Supply Chain Integration. Information Technology and Management 2001; 2:289-312.
15. Rimassa G. Applying Agent Technology for Enterprise-Wide Solutions. Proceedings of the Thirteenth IEEE Internatonal Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), Modena (Italy), 2004; 3.