# Rockwell Automation Agents for Manufacturing

Vladimír Mařík, Pavel Vrba
Rockwell Automation Research Center
Pekařská 695/10a
Prague, Czech Republic
Tel: +420-377422411
{vmarik, pvrba}@ra.rockwell.com

Ken H. Hall, Francisco P. Maturana
Rockwell Automation, Advanced Technology
1 Allen-Bradley Drive, Mayfield Heights
Cleveland, OH, USA
Tel: +1-4406463041
{khhall, fpmaturana}@ra.rockwell.com

## ABSTRACT

The paper provides an overview of the agent-based solutions developed by the Rockwell Automation company for the purposes of industrial control. Using agent-based manufacturing control, a higher degree of flexibility and reconfigurability of manufacturing solutions as well as higher robustness of the industrial systems can be achieved. Specific solutions connected with the proposed agent architecture, with implementation of the real-time control agents as well as with the information transfer among the SW agents and the real-time agents in a Programmable Logic Controller (PLC) are presented. Attention is also paid to the simulation of both the agent-based manufacturing facilities and their control systems. A simulation environment MAST for material handling systems has been implemented in JADE. The opportunity to re-use directly the simulation software on the agent control level is one of the most important features of MAST.

## Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Application and Expert Systems – Industrial Automation

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – Intelligent agents

I.6.0 [**Simulation and Modeling**]: General

## General Terms

Algorithms, Design, Experimentation

## Keywords

Multi-agent systems, manufacturing, PLC, real-time control, simulation, material-handling

## 1. INTRODUCTION

Rockwell Automation comp. (RA) is one of the world-leading manufacturers in the field of industrial automation, dominating the U.S. market in the area of discrete automation and control. The family of ControlLogix$^{TM}$ programmable logic controllers (PLCs) represents the flag-ship products. Programmable logic controllers are dedicated to real-time, strictly synchronous, and hierarchically organized control with reaction times in the order of tens of milliseconds.

The centralized and hierarchical approaches used traditionally in the production control, planning and scheduling or supply chain management appear to provide insufficient capabilities to handle the high degree of complexity and the increasing requirements for flexibility and robustness.

As the complexity of the manufacturing business environments is growing permanently, the technology of multi-agent systems (MAS) plays an increasingly important role in developing the concepts of highly distributed, robust and flexible manufacturing control. These issues naturally encouraged the development of new manufacturing architectures and solutions leveraging the MAS research results – the manufacturing control system is being considered as a community of highly distributed, autonomous and efficiently cooperating and asynchronously communicating units integrated by the plug-and-play approach.

The agent-based solutions seem to be the most attractive especially for manufacturing system reconfiguration purposes in the applications where the number of possible configurations of the equipment is impractically large. That's why the highest priority is given to the agent-based solutions for reconfiguration purposes (for example in the material handling, assembling and complex diagnostic tasks etc.).

The major difference between the "classical", strictly hierarchical control and the agent-based solutions is that the overall behavior of an agent system is more likely *emergent* than strictly deterministic – the behavior of the distributed system emerges from dynamically changing patterns of inter-agent interactions and asynchronously executed decision-making processes of particular agents that are not – in principle – influenced by any central control element. The emergent behavior, sometimes called the *aggregate behavior*, is expected to form one of the major benefits of this approach [5]. However it still constitutes a significant barrier against the wider deployment of the agent-oriented engineering today.

One of the possible solution to diminish the occurrence of emergent behavior would be to apply appropriate policies across the system. This includes for instance adopting rigid and preset organizational structures limiting the nature and the scope of the agent interactions or applying the interaction protocols whose properties can be formally analyzed. However, all these techniques lead either to a restriction of agents' autonomy or to the constraints on the flexibility and dynamic character of agent interactions. Hence, the major benefits of the agent-oriented engineering resulting from these characteristics are significantly suppressed. The other solution that copes not only with the issues of the emergent behavior, but also with the evaluation of the agent-based control systems in general, is the *agent-based simulation*. It is clear since the very beginning that such a simulation represents substantially more complex processes than in case of "classical" centralized control systems.

## 2. THE FIRST APPLICATION

The very first Rockwell Automation (RA) industrial agent project was to increase the machine utilization of a steel rod bar mill in the mid nineteenth of the last century. The rod mill is shown in Figure 1. The mill makes steel rods by reheating steel and rolling the steel to size using multiple rolling stands and cooling the steel along a defined temperature profile using multiple cooling boxes. The production process recipes for most of the steel rods require the use of neither all of the rolling stands nor all of the cooling boxes. Hence the system had built-in redundancy and flexibility since it could use any combination of cooling boxes and rolling stands from the subset of working units to produce a given steel rod recipe, as long as the required temperature profile was followed. The recipes that the operators had been using, however, specified particular subsets of cooling boxes and rolling stands. If some piece of specified equipment was broken, the operators would not run the recipe, and the order could not be filled. The aggregate desired behavior of the MAS was to select and configure a subset of cooling boxes from the working units to satisfy the recipe requirements. This was implemented by enabling each cooling box or unit to assess its own health and bid on its part of the operation. The bids were used with a very accurate simulation of the steel cooling process to enable rebidding until a suitable subset of units and configurations was found [7].

The process used to develop the agents for each of the participating units was to first understand the requirements. The operators were interviewed to understand what aspects were important to control the quality of the steel. Once the requirements and the priority of the requirements were understood we began to investigate various proposed solutions. Using the agent concept we assigned functionality to the devices in the system. The cooling box agent implemented in custome "C" code running on a PC was interfaced with the existing control system and was given the task of determining its health and capacity to cool by maintaining a history of water flow. A central node, also implemented on a PC using a combination of custom "C" and Fortran, contained the steel cooling simulation that arbitrated the bids to obtain the correct cooling curve. A simulation of the entire process was used to evaluate the agent systems performance. Changes were made to the behaviors of each agent until the overall desired behavior was achieved. The agent-based control system did not directly control the bar mill but instead recommended a configuration to the operator. Because of safety concerns and possible damage to equipment the risk was too high to enable direct control by this new technology. Although the agent system preformed very well in all the tests, to release the system for production would require testing all steel recipes with all possible subsets of cooling boxes.

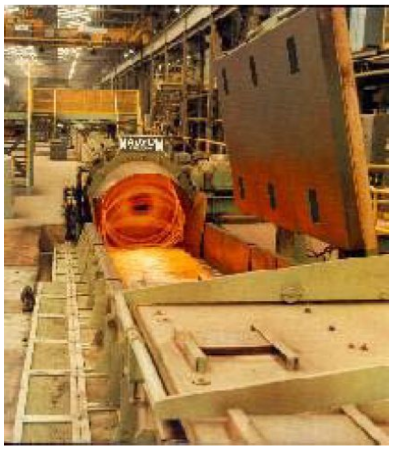


Figure 1. Bar steel production

The first experiments with the agent-oriented control philosophy, and especially the real-life industrial implementation developed together with BHP Billiton, Melbourne for a steel work in Australia led to a better understanding of the requirements and needs for this new kind of approach. Although the agent system preformed well in all the test cases, the plant was reluctant to use the "new technology" in production without the experts readily available. BHP sold the rod mill as part of its restructing shortly after the conclusion of the first experiments. Despite the fact that there was a central element in the agents' community, the company started to think how to change/modify its pilot products towards enabling agent-based control solutions. Our experience with the agent system pointed at two necessary improvements. The first was the ability to combine the agents with the legacy control system to eliminate the need for extra hardware and communications. The second was an introduction of an agent framework and development environment to enable creating and maintaining agent systems easier.

## 3. REAL-TIME CONTROL AGENTS

Regarding the lowest, real-time control level, the major characteristics of agents is that they are tightly linked with the physical manufacturing equipment – for instance, one agent opens and closes the valve in the piping system, another one controls the pump, next one controls the movement of the AGV (Automated Guided Vehicle) in the material handling system and the other one can be responsible for controlling the operations of a CNC machine.

These agents are referred to as *holons* or *holonic agents* [3] and are usually implemented as modules that encapsulate both the real-time (RT) control subsystem and the software agent (Figure 2). The RT-control subsystem is responsible for the direct handling of the information from physical sensors and actuators in real-time and is programmed in a low level language (usually in the ladder logic). However, the software agent part of the holonic agent needs to be implemented in some higher-level programming language due to the complex nature of the decision-making, communication and negotiation processes carried out by the agents. As the agent design usually follows the object-oriented principles – the agents are responsible for the local control of particular components of the manufacturing equipment – it is apparently reasonable to implement the agents in some object-oriented language like C++ or Java.

An important aspect of this solution is the existence of a *run-time communication interface* allowing to transfer the information from the RT-control subsystem (i.e. data from sensors, diagnostic subsystems, etc.) to the software agents and, vice versa, to propagate the control actions decided by the agents to the RT-control subsystem and thus to the physical actuators. Moreover, the agent control subsystem should be designed in such

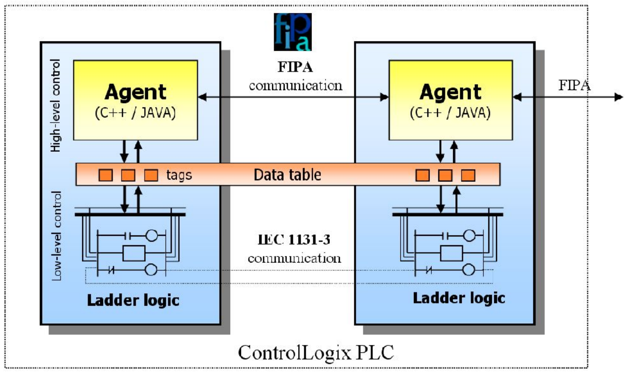


Figure 2. Holonic agent architecture

a way that it can be easily integrated with the existing industrial automation control architectures including the PLCs (Programmable Logic Controllers), industrial hardware or software visualization solutions (for example operator panels), etc. In such an architecture, everything is naturally concentrated around the so-called data-table of the PLC – the common data memory of the PLC, that is (i) used to hold the sensor and actuator information related to the manufacturing equipment connected through the PLC's I/O modules, (ii) used by the low-level RT control programs that process the data-table values (inputs and internal states) and compute the output values propagated to actuators and (iii) directly accessed by the visualization tool to display the state of the controlled process.

This led to the idea to use the PLC data-table as the mentioned run-time interface that provides the agents with a direct access to the PLC's data table in order to act upon the real data from the physical manufacturing process. As will be shown in Sect 4., such an interface, referred to as *universal run-time interface*, is considered to be used also for the simulation purposes. Within the simulation, the agent-control algorithms are validated before they are applied to the real facility. Apparently, the testing in the real manufacturing environment is simply unconceivable or would be at least extremely expensive and, maybe dangerous. The most straightforward solution is to simulate the manufacturing facility/process as well. In such a case, the universal run-time interface provides a connection between the agents and the simulated equipment.

In the following two Sections, the experience of Rockwell Automation in developing the agent-based manufacturing control systems is presented.

## 3.1 ACS Platform

The ACS – Autonomous Cooperative System was developed as the C++ based agent platform dedicated to the Logix family of PLC controllers. The agent platform enables to run the agents directly inside the PLCs (ControlLogix, FlexLogix, etc.), supports the agent management (registration, deregistration, services look up, etc.) and ensures the transport of messages among agents. The ACS platform is designed with respect to minimizing the use of memory and CPU resources of the PLC so as to not impact the performance of real-time control programs that run in parallel with the agents. The agents use a specially designed communication language – JDL (Job Description Language) for the message exchange in the RT-tasks as well as for the planning purposes. The JDL messages can be converted into FIPA-compliant messages by adding an appropriate FIPA wrapper to allow the ACS agents to communicate with other FIPA-compliant agent platforms (for example JADE or FIPA-OS).

The first application of ACS was aimed at the development of a reconfigurable and diagnostics driven control system for HVAC (Heating, Ventilation and Air Cooling) and CWS (Chilled Water System) of a US Navy ship [4] with the goal to increase its survivability as a part of the Shipboard Automation project funded by ONR (Office for Naval Research). Here, the C++ control agents access the sensor and actuator values in the data table of the ControlLogix (or its software emulator SoftLogix) controller using the C++ implementation of the universal runtime interface. The firmware of the ControlLogix is extended to allow running the C++ agents directly inside the controller in parallel with the ladder logic programs. Each element of the physical HVAC/CWS equipment (valve, cooling unit, piping section, etc.) or group of them is controlled by an individual agent. A reconfiguration process is triggered upon failure detection by a built-in diagnostic module. After detecting a failure an alternative solution – for example finding alternative path for water in the piping system to avoid the broken part – is found via negotiation among the agents.

The solution that has been successfully tested in the RA Advanced Technology Labs in Cleveland as well as in the U.S.NAVY facility in Philadelphia consists of 116 agents which are run on 6 ControlLogix controllers. Both the appropriate tool for development of agents and control systems and the visualization HMI (in the Rockwell Automation RSView32 tool) were developed as well.

In the case where more general, system-wide decision making, not directly linked with the physical devices is needed or required, it is necessary to introduce – besides the control and diagnostic agents – the higher level agents. In principle, it is possible to separate the responsibilities of the global agent system in such a way that:

- System-wide observations and optimizations are concentrated in high level agents which are based on high-level abstractions and which are decoupled from the physical devices.
- Control behavior is concentrated in the lower level agents which are constrained in a time critical realm and which operate in too limited time intervals to converge to high-value solutions.

As the high-level agents we use the agents developed by John Hopkins University (JHUAPL agents) which are able to consider and re-consider the activities of the control agents from the system-wide (ship-wide) point of view. They understand the ship’s operational modes, various interactions of ship systems, change the global priorities and perform the model-based diagnostics. The control agents developed by Rockwell Automation (RA agents) are responsible for the real-time control of the HVAC/CWS equipment, for efficient chilled water resource utilization, for re-configuring water paths, performing basic equipment diagnostics as well as for localization of leaks. The JHUAPL and RA agents communicate by exchanging FIPA/JDL messages.

## 3.2 Java-based agents

Recently, Java is being widely considered as an alternative to C++, mainly due to the portability of Java programs between different hardware platforms and operating systems, networking features, web-browser integration, etc. To enable the Java agents to access the ControlLogix PLC data table, the Java implementation (in form of API) of the universal runtime interface has recently been developed. To select an appropriate Java-based agent platform as an agent run-time environment for the ControlLogix PLC, we have conducted an evaluation of the majority of currently available Java-based agent platforms [8]. For the benchmarking, only the following subset of them has been considered: JADE, FIPA-OS, ZEUS, JACK and newly AGLOBE [6] (not included in the original study in [8]). As the agent platform is going to be used as a runtime environment for the real-time control agents, there were specific requirements on its properties taken into account:

- *Speed of the message passing* among agents – we measured the *average roundtrip time* (avgRTT) as an average time needed for a pair of agents to exchange one message (i.e. to send a message and get a reply). We tested different number of agent pairs (1, 10 and 100) communicating concurrently under different scenarios: (i) agents running on the same computer and (ii) agents distributed on two different computers (see Figure 3 for results with 10 agent pairs).
- *Memory constraints* – there is a limited amount of memory available for user applications on the controller. Within the RAM memory of the controller, which can for example be about 8 to 16 MB, the agent platform runtime environment, the agents themselves and also the low-level control code (ladder logic) have to fit inside. There are also smaller PLC-like devices that can have only 256KB of memory available, what would be a strong limiting factor for integrating the runtime part of the agent platform.
- *Costs and maintainability of the source code* – there are freely available agent platforms (usually distributed under a kind of an open source license) as well as the commercial ones. Both these groups have their pros and cons. In case of open source platforms they are for free and you are provided with the source codes and allowed to modify them. This should be necessary in order to port the platform to PLC-based controller. On the other hand, in case of commercial agent platforms you have to pay possibly thousands of $US per each installation and the source codes are not available for you to modify them. However, you will probably get a better support.
- *FIPA compliancy* – we envision the compliancy with the FIPA standards as a crucial property of our agent-based solutions. The main reason is to ensure the interoperability, not only among the holonic agents at the lowest real-time control level but also between the holonic agents and other agents at higher levels of information processing within the company, for example data-mining agents, ERP agents, supply chain management agents etc.

The results of this study show that among the FIPA-compliant open source agent platforms, JADE seems to be the most suitable agent runtime environment for agent-based manufacturing solutions. Its major competitor, FIPA-OS is on average twice to three times slower than JADE and particularly, it has serious problems with the scalability – in the case of 100 agent pairs the tests in FIPA-OS were not successfully finished at all (platform failed to deliver all the messages). If the FIPA compliancy does not play an important role, a good alternative to JADE is the A-GLOBE platform that is being developed at the Gerstner laboratory of the Czech Technical University in Prague (http://agents.felk.cvut.cz/aglobe/). Among the commercial agent platforms, we can highlight JACK that keeps pace with both the A-GLOBE and JADE and moreover offers the implementation of the BDI (Belief-Desire-Intention) model.

In our effort to move from C++ to Java we have finally decided not to use any of the existing platforms as the agent runtime environment for the ControlLogix controllers. Instead, we have ported the ACS platform mentioned above to the Java language. However, we are aware of the great popularity of the JADE platform. That is why we have implemented the first prototype of the CIP-based MTP (Message Transport Protocol) as a plug-in into JADE recently. CIP stands for Common Industrial Protocol and is used as an internal communication protocol for the ControlLogix controllers as well as for the Java version of the ACS platform. The CIP MTP can be used by JADE as a communication layer for bidirectional sending of messages between the JADE and the ACS agents.

## 4. SIMULATION IN AGENT-BASED SYSTEMS

It is obvious that the experimental testing of the agent-control system with the target physical manufacturing/control environment being directly involved is not only extremely expensive, but for certain applications even non-realistic. *Simulation* is the most convenient solution. There are quite specific requirements and expectations put on simulation of agent-based systems:

- Simulation of both the controlled process/manufacturing facility and the agent-control system has to be provided. For this purpose it is necessary to have:
  a) A good model of the controlled process/manufacturing

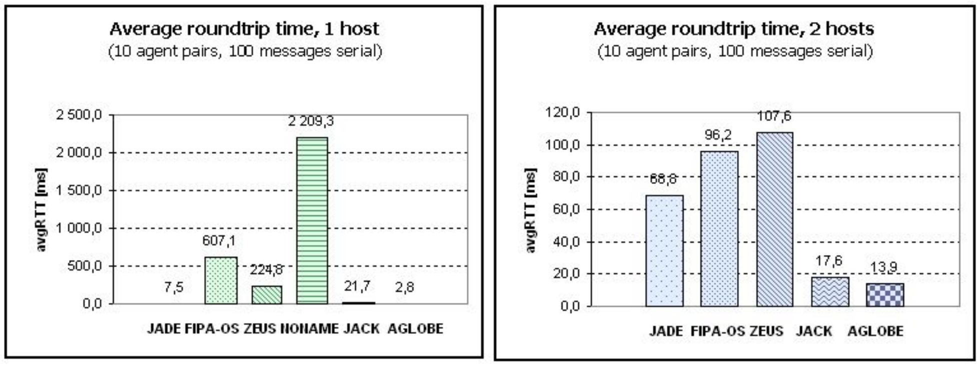


Figure 3. Agent platforms benchmarking results

facility and a suitable simulation tool for running this model – this will de facto provide the *emulation* of the physical manufacturing environment. It is encouraged to use one of the commercially available simulators like Matlab, Arena, LabView, Silk or AnyLogic for these purposes instead of developing an own ad-hoc simulator.

b) A suitable agent runtime environment for running the agents and modeling their interactions (agent platforms like JADE, JACK, A-GLOBE, etc.).

- The agent-control algorithms used in simulation are usually reused (90 – 100%) for the real-life control of the manufacturing equipment. Thus the agent-control part is – in the simulation phase – emulated as well. That is why the simulation can be carried out only by another agent-based system, usually organized as an appropriate interaction of two emulations.
- It is necessary to have the following two r*untime interfaces*:

a) an interface between the agent-control and the emulation of the controlled process/ manufacturing facility (for the simulation phase) and

b) an interface to link the agent-control with the physical controlled process/manufacturing facility (for the real-life control)

These two interfaces should be compatible and – in an ideal case – identical to enable the designer to switch from simulation/emulation to the physical manufacturing/control system in a step-wise manner. As mentioned in previous, Rockwell Automation designed so called *universal run-time interface* (and implemented appropriate C++ and Java API) that enables efficient interactions among different components by sharing the control data – mainly sensors and actuator values – in the data-table of the industrial PLC controller (the values in the data table are usually referred to as *tags*). These components include:

- *Physical manufacturing environment* that is linked to the PLC through its I/O modules. The PLC automatically supports the transfer of the input data from sensors and their storage into pre-prepared tags as well as the transfer of particular tag values as the output data to actuators.
- *Subsystem that emulates the manufacturing equipment* (for instance Matlab or Arena simulation). Data from emulated sensors and actuators have to be also automatically transferred by the interface between the emulation subsystem and the tags in the data-table.
- *Agent control system* which is responsible for carrying out the agent-based control algorithms. The decision-making of agents is based on the cooperation with the other agent as well as directly on the tag values in the data-table referring to sensors/actuators of the particular part of the manufacturing equipment the agent is in charge of.
- *Visualization module* providing a graphical insight into the manufacturing process state. The graphical tools of the commercial simulators used for the emulation purposes (for example Simulink in Matlab) can be utilized, however all the displayed data must be based only on the information from the data-table. Thus the interface must also ensure the transfer of tag values between the data-table and the visualization subsystem. Indeed, the industrial SCADA (Supervisory Control And Data Acquisition) systems like iFix (General Electric), WinCC (Siemens) or RSView (Rockwell Automation) can be used for process visualization as well because they naturally directly operate with the PLC data.
- *Low-level real-time control programs* usually programmed in ladder logic.

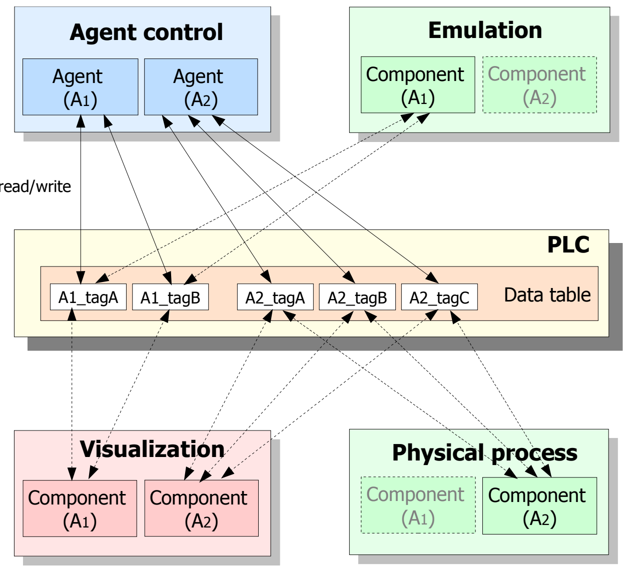


Figure 4. Universal run-time interface using PLC data-table

The important feature of the proposed interface is smooth shift of the control functionalities from the agent-based simulation towards the real-life control. It allows replacing of the emulation subsystem with the real physical manufacturing equipment by preserving the same tag names referring to the sensor and actuator values. Thus it is not necessary to do any modifications in the agents or in the visualization subsystem. Moreover, as shown in Fig. 4, the shift can be sequential, step-by-step. While some parts of the manufacturing equipment can already be physically connected and controlled by the agent(s) (for instance the component $A_2$ in Fig. 4), the other part of the system can still be emulated (component $A_1$ in Fig. 4). Thus, there is actually no strict borderline between the agent-based simulation and the real-life agent-based control.

Currently, the agent-based simulation of the agent-based solutions is being intensively used in two of the Rockwell Automation projects:

- The already mentioned agent-based control of the HVAC/CWS system of the U.S. Navy ships. In this case, the physical shipboard automation system is simulated in MATLAB/Simulink and linked via the OPC (OLE for Process Control) to the ControlLogix to interact with the agents [4].
- The second project that explores the universal-runtime interface is aimed at the agent-based control and simulation tool for the material handling applications. As a result, so-called MAST – Manufacturing Agent Simulation Tool has been developed.

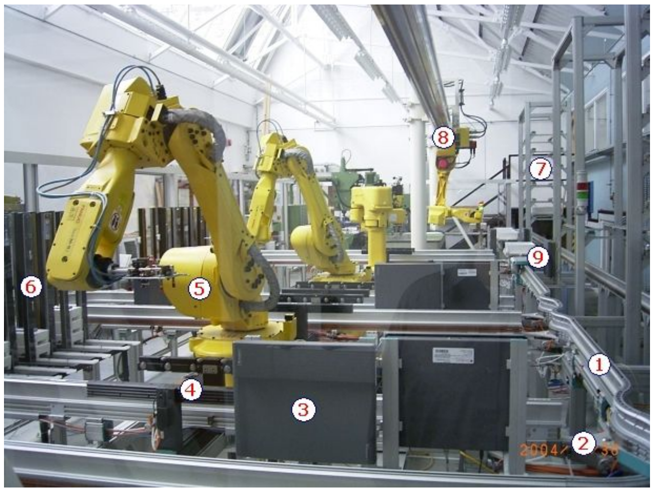
## 4.1 MAST Simulation Environment

The simulation environment MAST has been primarily intended as the agent-based demo application depicting the major benefits of the agent technology applied to some exemplary manufacturing task − the transportation of workpieces (products) among the manufacturing work cells (machines) on the factory shop floor using the conveyor-based transportation system.

The agent library that was developed represents basic components of material handling systems such as work cell, conveyor belt, switch (diverter), etc. The cooperation of agents is focused on finding the optimal/shortest transportation routes between the work cells interconnected via a network of the conveyor belts, diverters and intersections. It is important to stress that there is no central control element – decision making processes are distributed over the agents that work autonomously without being affected by any central, higher-level controller. The important feature of the proposed solution is the fault tolerance and structure flexibility. A failure of any component can be emulated (for example a failure of the conveyor belt) what causes the agents to start the negotiations concerning the alternative transportation paths while avoiding the broken component. New components can be added to the system or the existing ones can be removed while the rest of the system still continues in its operation – the newly added agents representing, for instance, new transportation capabilities are integrated on-line and used to transport workpieces while the removed agents just inform the community that they no longer exist and thus cannot be used. A more detailed description of the decision-making making processes and the negotiation scenarios can be found for example in [9].

MAST is entirely implemented in Java language and uses the JADE agent platform [1]. From its first prototype developed more than three years ago, the MAST tool grew up into a comprehensive agent-based control and simulation environment consisting of the following parts:

- *The agent control part* that contains a library of Java/JADE classes representing the material-handling components.
- *The emulation part* that is used to simulate the behavior of the physical manufacturing environment. Since the Java interface to share the sensor/actuator values in the data-table was not available at the time the MAST development started, it was decided not to use some commercial simulators like Matlab or Arena. Instead, an own ad-hoc simulator was developed in Java. The emulation model of the material handling system consists of the Java objects representing particular components with their virtual sensors and actuators. These emulation objects are appropriately linked with each other according to the structure of the system. The emulation engine moves the virtual workpieces over these components and activates the virtual sensors. This causes to inform the appropriate agents through the runtime interface.
- *The runtime interface* that was originally implemented as a direct link (Java method calls) from each emulation object to the appropriate agent (sensor signals) and vice versa from the agent to the emulation object (actuator signals). Currently, this tight link has been split and the universal run-time interface presented in this paper has been used instead. Thus, the sensor/actuator values are shared in the data-table of the ControlLogix/SoftLogix PLC and read/written-down by the emulation part, the agent control part, and the visualization subsystem via the Java API.



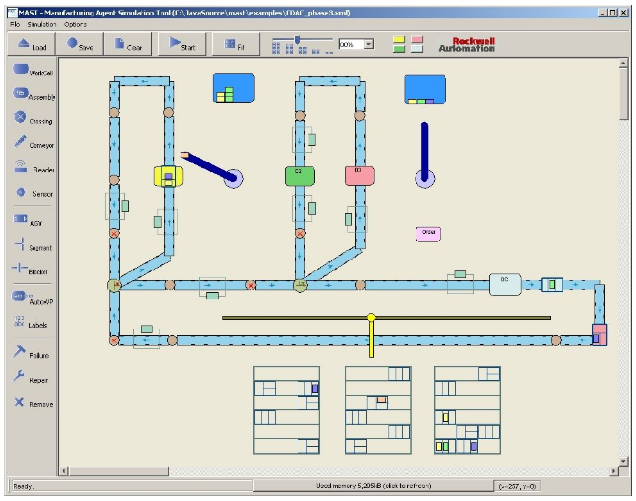


Figure 5.Cambridge packing cell and its simulation in MAST environment

- *The GUI* for the graphical drag-and-drop design of the material handling system as well as for the visualization of the simulation. Through the GUI, the user can send workpieces between the work cells, introduce failures of different components and even change the structure of the system at runtime.

Recently, the MAST environment has been extended to simulate all the components of the holonic packing cell of the Center for Distributed Automation and Control (CDAC) at the University of Cambridge, U.K. This lab provides a physical testbed for experiments with the Radio Frequency Identification (RFID) technology and the agile and intelligent agent-based manufacturing control [2]. These technologies are demonstrated on packing of the individually tailored Gillette gift boxes that can be filled by any combination of three out of four Gillette grooming items (gel, razor, deodorant and shaving foam).
The MAST's agent library has been extended with a set of new agents to represent and control particular components of the lab's equipment (numbering corresponds with the labels in Fig. 5): (1) *conveyor loops* (Montech track) to transport the shuttles with boxes (there is one main feeding loop and two subsidiary loops leading to robots); (2) *gates* that navigate the shuttles out of the

main loop to the subsidiary loops and vice versa; (3) *RFID readers* that read the IDs (EPC codes) of passing boxes – the data are provided by the readers to the gates to be able to properly navigate the shuttle; (4) *docking stations* at which the shuttles are held while a box is being packed; (5) *Fanuc M6i robots* that pack the boxes by the items picked up from the storage units; (6) s*torage units* for temporary holding of the items in four vertical slots (each for a particular type of the Gillette item); (7) *rack storages* that hold shuttle trays with both the empty and packed boxes as well as with the raw items that can be used to feed the temporary storage areas and (8) *gantry robot* that picks the box out of the rack storage and drops it to the shuttle and vice versa.

Additionally, each manufactured workpiece (box in this case) is represented by the agent. The *workpiece agent* plays an active role in coordinating the packing operations by negotiation with the other agents. It includes negotiation with shuttle agents that provide transportation capabilities, interactions with the gantry robot agent to pick up the box from the rack storage and drop it onto a shuttle, negotiation with storage units to select the one that holds the requested items, interaction with a robot agent to pack the items into box etc.

The application of the MAST environment to the real-life control of the Cambridge packing cell is currently under progress.

## 5. CONCLUSIONS

According to our estimates, at least 25% of industrial automation problems can be efficiently solved by using the agent-based approach. The industrial case-studies document the robustness and flexibility/reconfigurability of the manufacturing systems based on the agent-oriented philosophy. The plug-and-operate approach is highly appreciated by the customers. On the other hand, many constraints, namely rather limited time for the decision making, constraints given by the properties of the physical equipment as well as limited number of acceptable manufacturing structures bring new requirements on agents‘ behavior. Usually just acceptable, not fully optimal solutions can be expected under these conditions.

Simulation plays even more important role than in the case of centralized systems. It is very often aimed at detecting of the emergent/aggregate behavior. The direct re-usability of the simulation code for control/diagnostic purposes belongs to the key factors reducing the commissioning time and expenses.

There are many aspects to be considered when agent-based solutions are being introduced to manufacturing. For instance, the migration processes from the classical control device to the autonomous agents are expensive, standards for autonomous agents aimed at the manufacturing domain are not developed adequately, etc. Other issue is the selection of suitable agent platform used as the agent runtime environment in PLCs. The use of Java-based, FIPA compliant agent platforms like JADE is encouraging, however none of them is directly dedicated to the manufacturing control domain.

## REFERENCES

[1] F. Bellifemine, A. Poggi, and G. Rimassa, “Developing Multi-agent Systems with JADE,” Proc. 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages, pp. 89-103, 2000.

[2] M. Fletcher, D. McFarlane, A. Lucas, J. Brusey and J. Jarvis, “The Cambridge Packing Cell - A Holonic Enterprise Demonstrator,” Proc. 3rd International / Central and Eastern European conference on Multi-Agent Systems, Prague, Czech Republic, 2003.

[3] V. Marik, M. Pechoucek, P. Vrba, and V. Hrdonka, “FIPA Standards and Holonic Manufacturing,” *Agent Based Manufacturing: Advances in the Holonic Approach*, ed. Deen, S. M., Springer-Verlag Berlin Heidelberg, pp. 89-121, 2003.

[4] F. Maturana, R. Staron, K. Hall, P. Tichý, P. Šlechta and V. Mařík, “An Intelligent Agent Validation Architecture for Distributed Manufacturing Organizations,” *Emerging Solutions for Future Manufacturing Systems*, Ed. Luis M. Camarinha-Matos, Springer Science+Business Media, New York, pp. 81-90, 2004.

[5] H.V.D. Parunak and R. VanderBok, “Managing Emergent Behavior in Distributed Control Systems”, Proc. IAS-TECH/97 conference, Anaheim, CA, 1997.

[6] D. Sislak, M. Rollo and M. Pechoucek, “A-Globe: Agent platform with inaccessibility and mobility suport,” *Cooperative Information Agents VIII*. Number 3191 in LNAI, Eds. Klusch, M., Ossowski, S., Kashyap, V., Unland, R., Springer-Verlag, Heidelberg, 2004.

[7] D. Vasko, F. Maturana, A. Bowles and A. Vandenberg, “Autonomous Cooperative Systems Factory Control,” Proc. PRIMA 2000, Australia, 2000.

[8] P. Vrba, “JAVA-Based Agent Platform Evaluation,” *Holonic and Multi-Agent Systems for Manufacturing*, LNAI 2744, Springer Verlag, Berlin Heidelberg, pp. 47-58, 2003.

[9] P. Vrba, V. Marik and M. Fletcher, “Agent Based Simulation: MAST Case Study,” *Emerging Solutions for Future Manufacturing Systems*, Ed. Luis M. Camarinha-Matos, Springer Science+Business Media, New York, pp. 61-72, 2004.