7 AGENT-BASED SIMULATION: MAST CASE STUDY

Vladimír Mařík
Department of Cybernetics, Czech Technical University in Prague, Czech Republic & Rockwell Automation Research Center, Pekařská 10a, 155 00 Prague, CZECH REPUBLIC, marik@labe.felk.cvut.cz

Pavel Vrba
Rockwell Automation Research Center, Pekařská 10a, 155 00 Prague, CZECH REPUBLIC pvrba@ra.rockwell.com

Martyn Fletcher
Agent Oriented Software Limited, PO Box 318, Cambridge CB4 1QJ, UK martyn.fletcher@agent-software.co.uk

Summarizing all the specific features of the simulation systems needed for agent-based systems, the paper documents that the simulation tools of this kind do represent rather complex development environment for agent-based systems than one-purpose simulation software obvious in the case of "classical" centralized systems. The agent-oriented simulation tools explore and combine methods of real-time emulation, qualitative simulation, testing and diagnostic algorithms with classical methods of both the discrete and continuous simulation approaches, techniques of advanced visualization and run-time interfacing. The MAST simulation tool and the detailed description of its extension for the Cambridge Packing Cell Testbed are used as a case study.

1. INTRODUCTION

In the case of multi-agent systems, under the term "simulation" we understand processes which are – in comparison to "classical" centralized systems – more complex and include more tasks than just single simulation in the classical meaning. There are many quite specific requirements and expectations put on simulation of the agent-based systems:

a) First of all, we expect that the qualitative evaluation of emergent behavior of an agent-based system will be provided.

b) Agent-based system can be simulated only by another agent-based system – the centralized approach is not adequate. It is necessary to stress, that the existing simulation tools like Matlab or Arena are not sufficient for this purpose.

c) The simulation of both the controlled process and the agent-control system has to be provided. Because of the direct reusability of the agent-control

algorithms, the agent-control part is, as a matter of fact, emulated as well. In such a way the agent-based simulation is usually organized as the interaction of two emulations.

d) In the case of the agent-based simulation, there are two kinds of interfaces expected, namely a nice and instructive human-machine visualization interface, and – as a rule – a machine-machine runtime interface between the agent-control system and the controlled process/manufacturing equipment (either emulated or physically connected – see below).

The process of developing and implementing an agent-based system relies on several phases and widely explores the simulation principles of diverse nature. This is especially true in the case of systems without any central element where unexpected emergent behavior can appear. The following stages of the design process based on simulation can be gathered like this:

(1) Identification of agents: The design of each agent-based system starts from a thorough analysis of (i) the system to be controlled or manufacturing facility to be deployed and (ii) the control/manufacturing requirements, constraints, and hardware/software available. The result of this analysis is the first specification of *agent classes* (types) to be introduced. This specification is based on the application and its ontology knowledge. The usual design principle – following the object-oriented methodology – is that each device, or each segment of the transportation path or each workcell is represented by an agent. In very up-to-date systems, also the product itself or semi-product can be considered as agent able to negotiate its own processing/assembling with the agents of the manufacturing environment.

(2) Implementation/Instantiation of agent classes from the agent type library. This library is either developed (step 1), or re-used (if already available). Particular agents are created as instances of the definitions in the agent type library. Furthermore, the implementation of communication links among these agent instances is established within the framework of initialization from these generic agent classes (for instance agents are given the names of their partners for cooperation). In such a way, the first prototype of an agent-based control/manufacturing system (or similarly, a supply-chain management system) is designed.

(3) Own Simulation: Behavior of a complex agent-based system is rather emergent than deterministic (Steels, 1994). The decision-making knowledge stored locally in the agents along with the patterns of inter-agent interactions result in an aggregate global behavior of the system, which cannot be precisely predicted in advance. Yet the direct experimental testing of the global behavior with the physical manufacturing/control environment being involved is not only extremely expensive, but non-realistic as well. Simulation is the only way out. For this purpose, it is necessary to have:

- A good model of the controlled process or the manufacturing facility or the virtual enterprise. This model must depict all the entities within the factory/enterprise and their interfaces to the external world.

- A good simulation tool for running the model of controlled process/manufacturing facility/virtual enterprise to provide the emulation of the physical manufacturing/control environment. Standard simulation tools like e.g. Matlab, Arena, Grasp, Silk, AnyLogic etc. can be used for these purposes.

- A suitable agent runtime environment for running the agents – reused from phases 1 and 2 – and for modeling their interactions. On the basis of the results of agent platforms comparison (Vrba, 2003a), the JADE platform as open-source or JACK (Howden, *et al.,* 2001) as a commercial tool can be recommended.

- System integration strategies developed and implemented in the form of sub-system interfaces. It is necessary to have the following two *run-time interfaces*: (i) an interface between the agent-control and the process emulation and (ii) an interface to link the agent-control with the physical manufacturing equipment. In the ideal case these two interfaces should be compatible (or identical at best) to enable the designer to switch from simulation/emulation system to the physical manufacturing/control system or virtual enterprise as appropriate.

- HMI (human-machine interfaces) for all the phases of the system design and simulation.

4. **Implementation of the target control /manufacturing system**: In this stage, the target control, manufacturing, production management or supply-chain system is re-implemented into the (real-time) running code. This implementation usually relies on ladder logic, structured text or function blocks at the lowest level of control. However, the higher-level control – carried out by the agents – is almost entirely reused, i.e. the same agents used in the simulation phase (3) are used also for the physical control. For instance in the ExPlanTech production planning MAS system (Říha, *et al.,* 2001), there was 70% of the agent code reused from the simulation prototype. Therefore, the choice of the multi-agent platform in the phases (3) and (4) is critical – it is advised to operate with the same agent platform.

The simulation phase is much more crucial for the development process of agent-based systems than it has been for the development of "classical" centralized systems. It enables besides others:

- to **predict the behavior of the system as a whole**. The fact there is no central unit in the agent-based system represents a critical barrier in a wider applicability of the agent-oriented ideas. The simulation runs help to understand the system behavior and to detect the patterns of emergent behavior. Considering that the behavior of a MAS is emergent, to ensure that all types of possible behavior were explored/covered by simulation still remains a painful problem (the situation is similar to that of system testing).

- to **predict and test the optimal scenario for the agent-based system** development

- to select **the most optimal negotiation framework and strategy** for individual units in the system

- to **directly link the simulation with real-life manufacturing/control processes**. That means that whereas a part of the agent-based system is engaged fully in the

real-life activities, the remaining part can be just simulated. The shift of the borderline between the simulated (in more precise terms "emulated") and the real part of the system can be carried out in a quite smooth way. This would help to speed-up the initial "commissioning" process significantly.

The requirements on simulation tools or platforms for MAS-oriented solutions call for new types of simulation systems (simulation platforms) with embedded MAS principles. One of pioneering systems of this kind, presented in this paper, is the MAST simulation tool being developed by Rockwell Automation (Vrba, 2003b). Another example of such a system is the agent-based control and simulation of the scaled-down form of chilled water system for the US-Navy ships (Maturana, *at al.*, 2004).

2. MAST – Manufacturing Agent Simulation Tool

The development of the MAST modeling and simulation tool started about three years ago as one of the pilot projects of Rockwell Automation Research Center in Prague aimed at the investigation of holonic systems, i.e. the exploitation of multi-agent technologies in manufacturing control. The original idea was to implement the agent-based solution for material handling domain, particularly the transportation of materials/products between various manufacturing cells on the factory shop-floor using conveyors and AGVs. The attention has been paid mainly to the identification of agents (see Section 1, phase 1), i.e. the definition of agent types for basic components, like manufacturing cell, conveyor, diverter, AGV, etc., and the specification of communication protocols and scenarios used for the inter-agent cooperation (Vrba, 2003b).

To be able to test and validate the agent functionality without being connected to the physical manufacturing equipment, the simulation, or more precisely the emulation of the manufacturing environment had to be implemented as well (see Section 1, phase 3). Basically, simulated movement of virtual products triggers virtual sensors that send signals to appropriate agents while the control actions taken by agents are propagated back to the simulation through virtual actuators. The simulation part of the MAST tool is tightly linked with the GUI (HMI-interface) that provides the visualization of the simulation and allows the user to interact with it.

The agent behavior is aimed at the transportation of products among user-requested manufacturing cells. The agents cooperate with each other via message sending in order to find the optimal routes through the system – a cost based model is applied where each conveyor provides a transportation at predefined cost. The work cells are interconnected via a network of conveyors and diverters (switching components) that route transported products through the conveyor network following the optimal, i.e. least-cost routes. Main stress is put on the failure detection and recovery – the user can simulate a failure of any component and trace the reaction of agents looking for another delivery routes while avoiding the broken component. It is important to say that there is no central control element – the decision making and control processes are distributed over the agents that work autonomously and use message sending and on-line service discovery for mutual collaboration. The *plug-and-operate* approach is thoroughly applied for system

integration allowing to add/delete/change any component/agent through the GUI on the fly.

For the implementation we decided to use the JAVA language because of the variety of JAVA-based FIPA-compliant agent development tools being available today, most of them as open sources. Originally, the development started with the FIPA-OS agent platform but due to the performance and memory consumption issues we selected the JADE platform instead. The main advantage of using the object oriented language – JAVA in this case – is that the description of a particular agent type is represented by a single JAVA class containing the general specification of agent's attributes and the set of rules according to which the agent behaves. Such a class is then used to create as many instances as required (see Section 1, phase 2) without the need to program the behavior of each agent instance individually.

3. A PPLICATION OF MAST TO CAMBRIDGE PACKING CELL

3.1 The Cambridge Packing Cell Overview

Although there is a number of academic and industrial research organizations active in the holonic/agent-based control field, there are very few holonic systems deployed in real factories making real products today. Main reasons for this situation are the higher investments needed to implement the agent-based manufacturing system and also a number of research issues that remain to be resolved like the appearance of unpredictable, emergent behavior of a community of agents (see Section 1) or missing framework for evaluation of the holonic system's performance and applicability (Fletcher, *at al.*, 2003a).

To give the opportunity to evaluate different holonic design and development strategies, the holonic packing cell has been constructed in the Center for Distributed Automation and Control (CDAC) at the Cambridge University's Institute for Manufacturing. This lab provides a physical testbed for experiments with agile and intelligent manufacturing with focus on two particular areas: (i) Automatic Identification (Auto-ID) systems and (ii) agent-based control systems (Fletcher, *at al.*, 2003b).

The automatic identification is an emerging technology designed to uniquely identify a specific product in a supply chain. It replaces the bar code with an Electronic Product Code (EPC) embedded in a radio frequency identification (RFID) tag comprised of a silicon chip and antennae. The EPC numbers, usually in form of 96 bit code, are read wirelessly via high frequency radio waves – the RFID readers then pass the information to a computer or an application system (MES/ERP).

In the Cambridge packing cell, the RFID tags are attached to Gillette personal grooming items (razors, shaving gel, deodorant and shaving foam) and also to boxes to which these items are packed. The orders are placed by a user that can select any three out of the four Gillette product types to be packed into two types of gift boxes. The layout of the packing cell is given in Figure 1. It consists of three conveyor loops (Montech track) to transport the shuttles with boxes – the navigation of shuttles into and out of the loops is controlled by two, independently operating gates that are provided with EPC codes of passing boxes from the RFID readers. Shuttles

are held at two docking stations so that the robot (Fanuc M6i) can pick and place items into boxes. The items are held in a storage unit in four vertical slots (each for a particular type of the Gillette item); the items are picked up by the robot from the bottom of the slot and, in the case of unpacking operation, picked from the box and placed to the top of the slot.

For controlling all the operations of the packing cell, the agent-based control system has been implemented (Fletcher, *at al.*, 2003a). It comprises of the following *resource agent* classes with number of instances shown in brackets: Robot (1), Storage (1), Docking Station (2), Gate (2), Track (1), Box Manager (1) and Production Manager (1). Additionally, there are the *order agents* associated with particular gift box orders (one agent per one order) and *product agents* representing all available boxes in the system. The processing of particular order starts with the negotiation between the order agent and product agents to select the appropriate box in which the items will be packed. The product agent representing selected box then uses its own intelligence and cooperation with resource agents to determine how best to be packed: (a) the order agent queries the storage unit if it is able to provide the requested items, (b) the order agent negotiates with docking stations to reserve a processing slot, (c) there is a negotiation with the gates to ensure a proper routing to docking stations using the information from RFID readers, (d) once at the docking station, the product agent requests the robot to pack the items into the box (this includes a negotiation between the robot and storage unit).

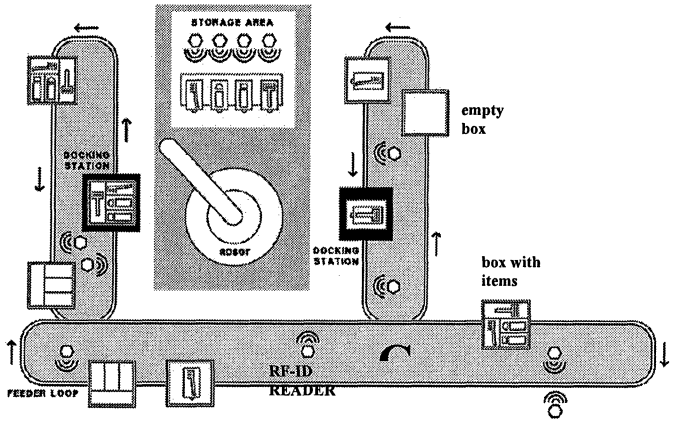

Figure 1 – Layout of the Cambridge packing cell

The agent control system – implemented in JAVA and using the JACK Intelligent Agents platform – is running on a Personal Computer. To provide the agents with an access to sensor and actuator parameters of the cell, the *blackboard* mechanism is introduced. The blackboard, running also on a PC, holds the synchronized copy of the data registers of an Omron PLC (connected via Ethernet) related to the sensors and actuators data. The agents can read from and write to the

blackboard and thus observe the current status of the cell and perform desired control actions.

Recently, the CDAC testbed has been considerably extended with new manufacturing equipment. It includes mainly a second robot and storage area to increase the flexibility of the system allowing to (i) process more boxes concurrently, (ii) choose the appropriate place where the box will be packed based on availability of requested items in storages and (iii) simulate the failure conditions. New shelving storage system is used to hold the shuttle trays with both the empty and packed boxes as well as with the raw items that can be used to feed the storage areas. The shelving storage system is operated by a gantry robot that transports the trays with boxes or raw items between the particular shelves and the new docking station in the feeder loop where the tray is placed onto the waiting shuttle.

It is obvious, that such a substantial hardware extension of the lab along with the new manufacturing scenarios being introduced requires a new agent-based control system to be deployed. It has been recognized, that the agent-based solution for the material handling tasks used in the MAST tool can be easily extended to provide the graphical simulation of the CDAC lab (see Figure 2 for a screenshot) and, eventually, to be directly used for the physical control of the packing cell. This paper reports on the primary results achieved in realizing the CDAC-related extensions of the MAST tool. Particularly, newly implemented agents, like the RFID reader, Docking Station, Robot, etc. are described and the issues regarding the use of MAST for the physical control of the cell are discussed.

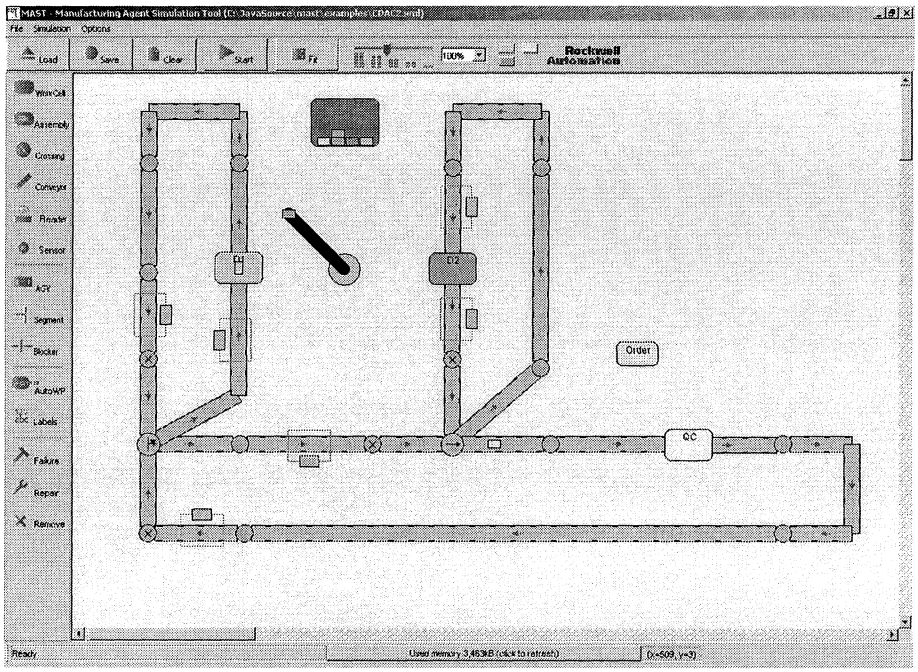

Figure 2 – Screenshot of MAST simulating Cambridge packing cell

3.2 The RFID reader agent

One of the major aim of the CDAC testbed is to demonstrate that the agent control can be integrated with the Auto-ID infrastructure. Physically, the lab is equipped with the RFID readers that send information about read EPC codes via Ethernet to a central server called Savant. The scanning period of the reader is quite high (e.g. 100 Hz) and within each scan all RFID tags that are in range of the reader are read at once. This results in large amount of redundant information generated by each reader that is sent to Savant. So the main purpose of Savant is to provide the EPC data filtering and storage in form of records containing the EPC number, name of the reader where it was read, timestamp and the flag if the RFID tag entered or left the reader. Such an information can be obtained from Savant by any client application using the SOAP protocol.

In an agent-based solution that integrates the Auto-ID technology it is reasonable to have a mediator agent that provides the EPC data to the other agents via standard agent communication protocols. To be able to easily add the Auto-ID support in the MAST tool, we decided not to use the Savant server – signals from emulated RFID readers are sent directly to the associated RFID reader agents (using a standard JAVA method call) that are doing the filtration locally. We have rather focused on implementing the mechanism of providing the EPC data to other agents. It is based on the FIPA **subscribe-inform** protocol which support has been embedded into all agent classes in MAST (by inheriting from general `MASTagent` class). Generally, this mechanism allows any agent (let say A) to subscribe for being informed by other agent (B) each time a particular event happens at the B-agent. In the case of the RFID reader agent, there are two events about which the reader informs the subscribed agents: (i) the product with an RFID tag entered the range of the reader and (ii) the product leaved the reader's range. In both these cases the reader agent sends the *fipa-inform* message including the EPC code(s) of the product to all agents that previously subscribed for one of these events. For subscription the *fipa-subscribe* message is sent to reader agent containing the `workPieceIN` string for the former or the `workPieceOUT` string for the letter type of the event (`workPiece` keyword is used in MAST to represent a product).

3.3 The Gate agent

The gate agent is responsible for routing the shuttles around the track. The gate is located at the crossing point of two conveyor loops (see Figure 1) and switches the shuttle coming from one of the two input tracks to one of the two output tracks in order to let the shuttle to remain in the current loop or to be rerouted to the other loop. How the particular shuttle will be switched obviously relates to the target docking station to which the box carried by the shuttle should be transported.

For the implementation of the gate agent in MAST we took the advantage of the existing diverter agent and the concept of least-cost product routing. Basically, the diverter holds an up-to-date routing table that contains the names of the work cells (docking stations) that are reachable using the diverter's output conveyors (tracks). These routing tables are determined by mutual cooperation of diverters, conveyors and workcells that exchange knowledge about reachable destinations along with

costs of the delivery in a back-propagation manner (for further details see (Vrba, 2003b)). Once the product enters the diverter, the diverter agent searches the routing table for the product's destination workcell name in order to find out which of the output conveyors will direct the product to its destination following the least-cost route.

The modification for the Cambridge packing cell lies in the integration of the Auto-ID technology. The gate agent cooperates with the RFID readers using previously described subscription mechanism to get the EPC data of products incoming from input tracks to the gate. The issue is, that the EPC code, i.e. the ID of the product (represented as 24 characters string) does not directly contain the name of the destination docking station. To resolve this, the gate agent queries the product agent, that is supposed to be registered in the agent platform under a name that equals to the ID of the product (box). In the case that the EPC data sent by the reader contains more IDs (both box and items in it are marked with the RFID tags), the gate agent have to contact the yellow pages services in the agent platform to determine, which of the IDs relates to the existing product agent. The product agent then informs the gate agent about its destination so that the gate can properly navigate it.

3.4 The sensor agent

The sensors are used to detect the presence of shuttles at particular places on the track. The associated sensor agents support the same subscription mechanism as RFID readers to inform the other agents. Sensors are used particularly in combination with gates to allow only one shuttle moving through the gate at the same time. For this purpose, there are usually two sensors in front of the gate (at input tracks) and two sensors at the exit from the gate (at output tracks). The input sensors are closed by default, which means that the shuttle is stopped by the sensor in front of the gate. Once the gate performs a switch-operation for selected waiting shuttle, the sensor agent that blocks this shuttle is informed (the subscribe-inform mechanism is used again) – the sensor opens and the shuttle enters the gate. The exit of the shuttle is detected by one of the output sensors that inform the gate so that another waiting shuttle (if there is some) can be processed.

3.5 The robot and storage area agents

The robot agent has been implemented to control the packing/unpacking operations performed by the Fanuc M6i robot. The packing operation starts by receiving a request from the product agent when the shuttle carrying the box reaches the docking station. The message sent to robot includes the specification of required items (e.g. two gels and one razor). The robot agent starts the negotiation with the storage area agent to get the index of the slot from which the item of given type can be picked up (there are four vertical slots each for one type of the Gillette item). If the item is present at the bottom of the slot, the robot picks it and places it into the box; if not present, the robot continues with the other required item. When all items are processed (either packed to box or missing in the storage), the robot agent informs the product agent that the operation has finished.

3.6 The order and product agents

The order agents are responsible for processing the user orders for packing the customized gift boxes. The box can contain any combination of three Gillette grooming products choosing from four different product types (razor, shaving gel, foam, deodorant). Automatically generated order agent (one per order) starts the negotiation with product agents that represent available boxes in the system (either empty or already packed). If there is an empty box of requested type available and the associated product agent is willing to cooperate (i.e. is not currently "working for" another order agent), it is committed the processing of the order.

The product agent first contacts the yellow-pages services to obtain the list of storage areas. In the following contract-net protocol the storages are giving their bids in terms of how many of requested items they can provide; they also include name of the robot that operates the storage and the names of the docking stations. The product agent selects the storage offering the best bid and starts another negotiation with the docking station agents to reserve a processing slot (slots previously reserved for other product agents along with the priority of the order are considered). The product agent then changes its destination to selected docking station – any time the shuttle carrying a box associated with the product agent enters the input RFID reader of the gate, the gate contacts the product agent to get the name of its destination for proper navigation (see Section 3.3).

Once the processing of the box is finished by the robot (see Section 3.5), the product agent informs the order agent about the state of the order while releasing the shuttle from the docking station to return to the main feeder loop (see Figure 1). If the order is not fully completed, e.g. because there were not enough items in the storage, the product agent restarts processing of the order, i.e. contacts the storages again to obtain remaining items. Such a flexible behavior allows (i) to distribute the packing operation over several robots according to the current availability of items in storages, (ii) the product agent to put off the completion of the order until the missing items are inserted into a storage (e.g. by unpacking some boxes or by transporting raw items from shelving storage) and (iii) to integrate new storages and robots (possibly also new track loops) at runtime without the need to make any changes to program code of existing agent classes.

3.7 Using MAST agents for the physical control of the CDAC lab

As mentioned in Section 3.1, the idea is to use to MAST tool for the physical control of the Cambridge packing cell. However, there are still some issues that need to be resolved in order to reuse the MAST agents in the implementation of the target control system (see Section 1, phase 4). Particularly, the mechanism of accessing the sensor and actuator parameters have to be modified such that the MAST-agents will use the same interface for accessing either the simulation-part of the MAST tool (i.e. emulated hardware) or the physical manufacturing equipment of the lab.

In the current implementation, the virtual sensors and actuators of the simulation engine of MAST are directly linked with appropriate agents via standard JAVA method calls. Similar approach as the blackboard mechanism described in Section 3.1 will be used instead. The sensor and actuator data will be shared through the tags

(data table) of the ControlLogix PLC using the prototype JAVA API. This API allows to directly access the data table, i.e. to read and write the tags remotely via Ethernet link from any JAVA program running on a PC. For example, in the case of the sensor component (Sect. 3.4), there is a sensor detecting presence of a shuttle and an actuator used to stop/release the shuttle. For each sensor there will be two tags in the PLC's data table distinguished by the name of the sensor (e.g. for sensor `s1` there will be tags `s1_sensor` and `s1_actuator`). As the simulation moves virtual shuttles around the track and the shuttle arrives at the sensor `s1`, simulation sets the `s1_sensor` tag in the data table to `true`. It is scanned by the appropriate sensor agent for changes to which the agent reacts, in this case by informing the subscribed agents. The shuttle is stopped if the `s1_actuator` tag is set to `stop`. When the sensor agent decides to release the shuttle, it changes `s1_actuator` to `go` value to which the simulation reacts by releasing the shuttle.

It is obvious that this mechanism allows to simply connect the agents with the physical hardware of the lab instead of the emulation-part of the MAST tool. The only thing needed is to link the physical sensors and actuators with the PLC (through its I/O interface) and store their values under the appropriate names.

Another issue is how to get the EPC data from the readers to the RFID reader agents (Section 3.2). The most convenient solution is to implement a JAVA driver for the physical RFID reader to receive unfiltered EPC data via the Ethernet directly from the reader and do the filtration locally in the RFID reader agent. The other way is to use the existing Savant server solution (see Section 3.1), i.e. to equip the RFID reader agent with the ability to receive already filtered EPC data from the Savant.

4. CONCLUSION

The main idea presented in this paper is that the simulation of agent-based system requires substantially different class of simulation systems and tools in comparison to "classical" centralized systems. The simulation systems applicable in the field of multi-agent system

a) have to be designed as agent-based systems as they have to – as a substantial part of their activities – **emulate** the behavior of the real MAS system. The off-line simulation mode is expected to be complemented or replaced by a real-time control of the physical real-life agent-based system or its part.

b) are expected to carry out – in the off-line mode – the emulation with the goal to achieve the **qualitative simulation** (in the sense of AI terminology) of the behavior of the MAS system as a whole with the stress to capabilities to detect the types/classes (in the optimum case all the types/classes) of potential emergent behavior. That's why, the models of agents should be mainly strongly knowledge-intensive ones suitable for qualitative simulation purposes. The knowledge-oriented analysis of behavior of each type of agents is a very important part of the simulation system design.

c) are explored like **testing, evaluation or diagnostic tools** of the agent-based system, especially during the period of the system design. The testing should start from different initial conditions, under different failures of various components – but nobody can confirm that all the potential states of patterns of

behavior will be covered by the series of experimental simulation runs. Simulation – similarly to the case of software testing – cannot be considered as a complete evaluation of the system.

d) can use – for the emulation of the controlled process/physical manufacturing environment – existing standard discrete or continuous simulation tools (or their combination if both discrete and continuous processes have to be simulated concurrently)

e) should be equipped by an efficient **visualization module** as the main "output" of the simulation processes is the "movie" showing the behavior of the system

f) should **run in real time** and be equipped by a **run-time interface** to the real-life control hardware to enable the shift of the borderline between the simulation and real-time real-life control.

Describing the MAST tool and its extension for the Cambridge testbed, we have documented that all the features mentioned above are really needed. Especially the analysis, development and implementation of the RFID agents as well as the ideas behind the implementation of the real-time interface do represent the main technical contribution of this paper.

Summarizing all the specific features of the simulation systems needed for agent-based systems, we can conclude: **The simulation tools of this kind do represent rather complex development environments for agent-based systems than one-purpose simulation vehicles.**

5. REFERENCES

1. Fletcher M, McFarlane D, Thorne A, Jarvis D, Lucas A. Evaluating a Holonic Packing Cell. In Holonic And Multi-Agent Systems for Manufacturing, LNAI 2744, Springer Verlag, Heidelberg, 2003, pp. 246-257.
2. Fletcher M, McFarlane D, Lucas A, Brusey J, Jarvis D. The Cambridge Packing Cell – A Holonic Enterprise Demonstrator. In. Multi-Agent Systems and Applications III, LNAI 2691, Springer Verlag, Heidelberg, 2003, pp. 533-543.
3. Howden N. et al. JACK Intelligent Agents – Summary of an Agent Infrastructure. In Proceedings of IEEE International Conference on Autonomous Agents, Montreal, 2001.
4. Maturana F, Staron R, Hall K, Tichý P, Šlechta P, Mařík V. An Intelligent Agent Validation Architecture for Distributed Manufacturing Organizations. In Proceedings of the 6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services, Vienna, Austria, 2004.
5. Říha A, Pěchouček M, Vokřínek J, Mařík V. ExPlanTech: Exploitation of Agent-based Technology in Production Planning. In: Multi-Agent Systems and Applications II, LNAI No. 2322, Springer Verlag, Heidelberg, 2002, pp. 308-322.
6. Steels L. A Case Study in the Behavior-oriented Design of Autonomous Agents. In proceedings of the Third International Conference on Simulation of Adaptive Behavior, August 1994, Brighton, UK, pp. 445-452
7. Vrba P. JAVA-Based Agent Platforms Evaluation. In: Holonic and Multi-Agent Systems for Manufacturing, LNAI 2744, Springer Verlag, Berlin Heidelberg, 2003, pp. 47-58.
8. Vrba P. MAST: Manufacturing Agent Simulation Tool. In proceedings of IEEE Conference on Emerging Technologies and Factory Automation, September 2003, Lisbon, Portugal, Volume 1, pp. 282-287.