Online ISSN 1848-3380, Print ISSN 0005-1144
ATKAFF 57(2), 452–465(2016)

Ismael F. Chaile-Alfaro, Lluís Ribas-Xirgo

Running Agent-based-models Simulations Synchronized with Reality to Control Transport Systems

DOI 10.7305/automatika.2016.10.1243
UDK 004.896.031.43-047.58:656.52.017.2-5

Original scientific paper

Adaptable, flexible and evolvable manufacturing systems and warehouses are so complex to manage that control systems have to be divided into several aspects. One of these is internal transportation, which has to do with all tasks involved in fulfilling a set of so-called transportation orders, i.e. commands to collect and deliver material from origin to destination spots. A common approach to design the controllers for these applications begins by modeling them as multi-agent systems and continues to final deployment through a cascade of transformations. To minimize development costs of internal transportation controllers, we have proposed a model of construction that includes components that synchronize the events from reality simulation and the ones from actual reality. By using these *synchronizers*, further development is required only for those parts of the initial multi-agent controller models with real counterparts. In this paper, we review the model and the architecture of the proposed internal transportation system controllers and we illustrate the whole design process through the development of a controller for an automated laboratory. Indirectly, we prove the validity of the architecture and of its key component, the synchronizers.

Key words: Agent-based modeling, Multi-agent systems, Agent-based control systems, Transportation control, Real-time systems, Multi-robot synchronization

Odvijanje simulacija agentskih modela sinkroniziranih sa stvarnošću za upravljanje transportnih sustava. Prilagodljivi, fleksibilni i razvijajući sustavi proizvodnje i skladištenja toliko su složeni za vođenje da je sustave upravljanja potrebno podijeliti u nekoliko aspekata. Jedan je od njih unutarnji transport koji je potrebno provoditi uz ispunjavanje svih zadataka iz skupa tzv. transportnih naredbi tj. naredbi za sakupljanje i isporuku materijala od polaznih do odredišnih točaka. Uobičajeni pristup u sintezi regulatora za ove svrhe započinje modeliranjem transporta kao višeagentnih sustava te se nastavlja do konačne implementacije kroz kaskadne transformacije. Za minimizaciju razvojnih troškova regulatora unutarnjeg transporta, u radu je predložen pristup sinteze uz uključenje komponenata koje sinkroniziraju simulacijsko i stvarno okruženje. Korištenjem ovih sinkronizatora, daljnji razvoj potreban je samo za one dijelove inicijalnog višeagentnog regulatora koji imaju svoje stvarne pandane. U radu je također dan pregled modela i arhitektura predloženih regulatora za transportni sustav te je prikazan cijeli proces sinteze kroz razvoj regulatora za automatizirani laboratorij. Posredno, također je validirana i ključna komponenta arhitekture odnosno sami sinkronizatori.

Ključne riječi: Modeliranje zasnovano na agentima, višeagentni sustavi, upravljanje zasnovano na agentima, upravljanje transportom, sustavi u stvarnom vremenu, višerobotska sinkronizacija

1 INTRODUCTION

Factories, warehouses, ports, and many other industrial facilities automate their control and management systems (e.g. production planning, logistics, etc.) to become more efficient [1]. Robots play a key role in these large systems automation. Particularly, they move material, objects, and people, too, within facilities. Nevertheless, this massive use of robots poses several challenges to management and control systems ([2]-[7]).

Internal transportation [8] is one of the critical aspects demanding specialized applications for control and supervision. Complexity of these tasks can be high, particularly when facilities are required to be adaptable and flexible (to produce or handle a range of similar products with different characteristics), evolvable (to update their machinery) and robust, understood as fault tolerance (to provide easy replacing/taking out of components from the plant when some of them fail). It is no surprise, thus, that those applications use agents and agent technology [9] to benefit from scalability, evolvability and robustness emerging

from them. For example, some proposals and actual systems use agents to manage traffic in urban areas ([9]-[13]), in warehouses ([1], [14], [15]), and to control automated-guided vehicle (AGV) systems ([16]-[20]).

The development processes of agent-based controllers for transportation systems begin by system specification, i.e. by creating multi-agent system (MAS) models. In this paper, we shall refer to these models as agent-based models (ABMs), not to be confused with the acronym for agent-based modeling, used to study the collective behavior of a set of agents [21]-[23].

Like in many design processes ([24]-[26]), initial ABM instances can be verified by simulation, among other alternatives. In fact, simulation software is used to model systems and validate test scenarios ([27]-[29]).

Early system specifications are gradually refined down to a degree of detail enough to be executed by the platforms that would run it (computers and robots).

There is a bunch of available multi-robot system (MRS) simulators [30-34] that can be used to both model a system and then, verify it. Also, all of them enable designers to seamlessly control real robots from the simulations. However, they are more focused on the individuals than on the system.

Using an ABM simulator makes it easy for designers to check system behavior but transition to actual robots is harder. In fact, ABM simulation is mainly used to support decision taking [35], [36] or help to manage elements represented by agents in supply chains [37], [38], in traffic [39] or in carpooling applications [40].

Our proposal is to keep as much of the ABM as possible in its original form, in order to lighten the whole development process [41] while enabling a full top-down approach from system level. This approach requires including synchronization between the model simulation and the reality, with some overhead in time and modeling constraints. However, the advantages may overcome these limitations: development time is shortened, as well as further system deployment and maintenance.

The model of computation and architecture must be defined before designing an internal transportation system controller. In our case, we propose an ABM organization (Section 2) that separates agents into two main classes, the one to control individual mobile robots and the one to account for other elements in the system, including a supplier of transportation orders.

The model of computation assumes that every automated-guided vehicle (AGV) in the system has a representative which is in charge of its commanding and of communicating with other agents.

The general idea is a traffic system that can be self-regulated from individual choices, requiring a minimum level of assistance from agents at a higher level of hierarchy. In other words, the transport orders from applications are handled by transportation agents in an autonomous manner, with minimal information from other agents, including those who may act as planners and routers. Partly, this can be achieved by enforcing transportation agents to obey some traffic rules (e.g. setting priorities at crossings) and by appropriately designing the layout of the traffic network (e.g. by minimizing conflicts using traffic circles).

Following the conventional design process, ABM instances are verified and, when functionally correct, built through progressive transformations of original specification into implementable descriptions. Typically, architectures do include as many resources as agents so to make the binding easy or, in other word as close as possible to one-to-one correspondence, and a set of communication resources. In our case, though, the architecture (Section 3) contains a computing resource to run the entire system simulation, including the ABM and resources to communicate it with the AGVs, which correspond to parts of the ABM [42]. This fact enables to reduce the size of the system specification that has to go through the implementation process but requires an extra type of resources: the synchronizers.

Therefore, the whole framework that we propose resembles those of [43] or [44] and [45] (simulation only) on development environments for agent-based systems and uses an ABM of the transport system that accepts inputs from the rest of the system and outputs control data for the physical transportation units as well as other data to the system. However, our framework significantly differs from them for the use of an ABM simulator with built-in synchronizers to simplify the development process (Section 4) and to minimize development costs.

The main contribution of this paper is to provide complete experimental evidence for the validity of the approach through complete deployment of a realistic prototype of an automated laboratory. The example has been chosen as we already had experience with modeling these cases [46], [47] and because this type of plants are relatively small, can be operated with AGVs that would be very similar to inexpensive, tiny hobby robots, with simple traffic networks, allowing to focus our efforts on synchronized simulation with reality.

2 AGENT-BASED MODEL OF TRANSPORTATION SYSTEM CONTROLLER

Transportation systems are composed of carriers or AGVs and their application part, which tell the former ones what to do but not how it has be done, just as in agent systems. Therefore, agent-based controllers for transportation systems like the ones used in manufacturing plants and

warehouses take orders from application-specific components (e.g. production planners and management systems) and transform them into requests for vehicles in accordance system states, including individual vehicle information. (Note that transport order tasks are allocated in a distributed fashion [9].)

The proposed ABM follows the previous organization (Fig. 1) by classifying agents into application-specific and transportation. Agents use a common agent communication language (ACL) so that they can interoperate and that addition and removal of components does not affect the system functionality.

Application-specific agents ($\{A_i\}$) link the transportation agents ($\{B_i\}$) with the rest of the elements in the application. From the transportation viewpoint, the role of $\{A_i\}$ is to facilitate the operation of $\{B_i\}$ by providing them with information other than the one they can gather individually and to communicate their state to other system components. To name a few, $\{A_i\}$ include the human-machine interface (HMI) and interfaces with remote terminal units (RTUs) in manufacturing plants and with the enterprise resource planner (ERP).

Transportation agents $\{B_i\}$ correspond to AGVs and, in fact, are their controllers. They communicate to each other to solve traffic conflicts (i.e. avoiding collisions in e.g. junctions) up to a certain degree. For instance, only the highest priority B_i (priority is defined when B_i receives the transportation order) can enter a junction spot while the rest must wait. (In case of a draw, the B_i with the highest ID gets the clearance.)

When those conflicts cannot be locally solved, they raise the issue to a traffic coordination agent.

Thus, designers should be aware of which traffic situations can occur and whether they can be solved locally or with the participation of other agents.

Provided that ABMs can be simulated and that these simulations can be run concurrently with physical agents, the difference between expected and sensed behaviors can be minimized via additional controlling levels [48] or locally within each agent.

However, the main problem of using ABM simulators as controllers is that ABMs must run in real time with the physical requirements of systems and their applications.

2.1 ABM application to automated laboratories

Laboratories of clinical analyses have progressively been transformed into a kind of complex manufacturing facilities, able to produce thousands of analyses per hour from blood and other biological samples. In these facilities, samples are dropped into tubes that are placed in racks which are delivered to different analyzing machines by using a conveyor system [49].

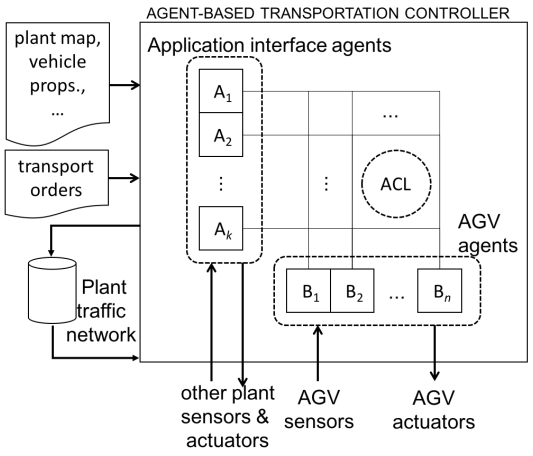

Fig. 1. Multi-agent model of a controller for transportation systems

Unfortunately, further variability is added because some tests must be repeated, not all racks stop at the same analyzers and several analyzers can provide the same information, though with different workload capacities.

As a result, the complexity of managing this kind of laboratories is quite high, even though they use relatively simple transport infrastructures. In these systems, small AGVs can successfully replace conveyors [46], [50] adding more degrees of freedom to the system and relieving plant managers from operating with lots of data to gain flexibility and robustness [51].

To include most of the characteristics of actual plants of automated laboratories, our study case includes four different analyzers: one ion-counting unit, one for coagulometry and two biochemical units, as most of the samples require evaluation of biochemical parameters.

The plant layout is based on the one of the conveyor-belt system in [52] (Fig. 2) with conveyors replaced by autonomous AGVs (Fig. 8), thus not requiring much infrastructure. In this case, to simplify vehicle operations, robots move around following a line with marks, which are used by AGVs to self-locate within map. In details, marks are used to indicate specific decisions points like ports or stop points before accessing to junctions (cross points like 3 in Fig. 8). Mark types are determined by AGVs in accordance with their location in the plant.

2.1.1 Transportation agents (B_i)

Each AGV is aware of its own position and able to recognize the environment and to communicate with other agents to coordinate movements. AGVs use information about the plant to determine to which analyzer they should

Fig. 2. Automated laboratory [52] with internal transportation system based on conveyor belts

go, to satisfy their loads requirements as fast as possible. Currently, in our model, AGVs randomly choose among compatible goals (i.e. they can go to one or the other of the biochemical analyzers on a random basis), as the focus of this work is to validate the proposed ABM-based controller.

When an AGV arrives at its destination, it docks at the port of the corresponding analyzer so to begin with its work. In case it is busy, AGV puts itself on hold in a parking area (short wait) or goes on to a compatible destination or to the re-circulation lane (long wait).

2.1.2 Application-specific agents (A_k)

$\{A_k\}$ are used like interface agents between B_i and whatever element in the real plant or its model. Their main task is to help with specific issues of transportation.

As for example, A_1 could be the agent representation of the Laboratory Information Management System (LIMS) in charge of the overall planning of transportation orders, as well agents A_2 to A_5 represent the interface with real analyzers in the laboratory.

As A_k are application-specific, they change according to the application and what it is represented.

For the case study experiments, the LIMS agent randomly generates work orders and representatives of the analyzers randomly decide whether sample tests are successful or not, i.e. must be repeated.

3 TRANSPORTATION SYSTEM CONTROLLER'S ARCHITECTURE

Components to build such type of controllers are mobile robots and computing resources, either embedded or not, and a communications network. Mobile robots are AGVs with some on-board computer that runs an embedded controller, and the other computing resources include, at least, a computer running the multi-agent based model simulation of the system.

3.1 Transport agent architecture

Transportation system controller is organized as a set of B_i agent controllers, which control vehicle sensors and actuators. Each B_i controller is divided into several layers: the lowest one is in charge of controlling the vehicle in accordance with the requests from the topmost one. The top layer is the one capable of communicating with other agents and, therefore, of considering system state when taking decisions on accepting and completing transport orders.

Resulting architecture for a B_i (Fig. 3) includes an intermediate, *interface* layer, which offers all services to communicate the top layer with the bottom one in a safe and secure way. Basically, services enable the high level layer (T_i) sending requests to the low level layer and to get answers from it.

With this organization, $\{T_i\}$ are detached from the corresponding low level layers (V_i and R_i). Consequently, implementation of both levels are independent if they share the communication language. Note that, even though both levels can use the same language and services as if they were different agents, they are not.

In short, any transport agent B_i is divided into two levels, the highest one T_i, and the lowest one, which is either virtual (V_i) and/or real (R_i). In our approach, though, both V_i and R_i can co-exist because of the interface layer synchronizing T_i with the lower level (V_i and R_i).

As mentioned previously, the main advantage of this approach is that simulation and control can run concurrently, with simulation helping to maintain a symbolic view of the system for all agents and to foresee results of individual choices.

The virtual system's representation includes the state of the plant as well as the state of the lower level of B_i. In fact, all $\{V_i\}$ interact with a plant environment simulator E and, as a result, all $\{T_i\}$ have access to system-wide information without communicating with other agents. For instance, they know the symbolic position of any other AGV to determine collision-free routes or to solve conflicts at crossings. The *symbolic position* is the position in the traffic network as represented in the simulated plant environment, which has to be accurate enough for the application, even though it does not correspond to a realistic representation on a screen.

Each R_i interact with elements at the plant (P) and is "controlled" by commands from T_i, which stands for the topmost controller level of B_i. Commands depend on replies from R_i, but also on differences between V_i and R_i replies, which are monitored and controlled by the synchronizing interface, on messages from other $T_{j\neq i}$ and A_k, and on global information stored in E.

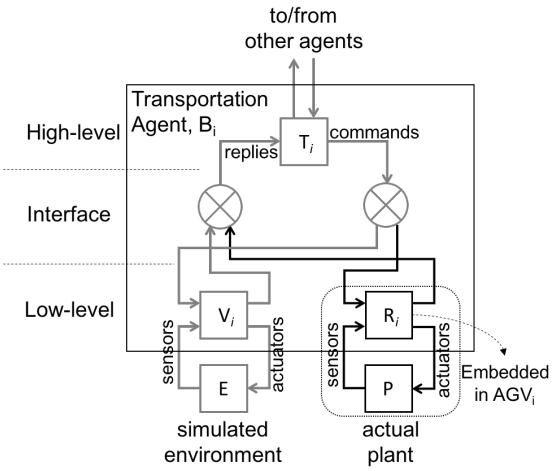

Fig. 3. Architecture of transportation agents

3.2 System architecture of transportation controller

Our approach [53] controls transport systems by model simulation. To that end, the proposed ABM's architecture organizes agents into two classes. One for the external elements ($\{A_k\}$) and another for the transportation agents $\{B_i\}$. In Fig. 4, besides $\{A_k\}$ and $\{B_i\}$ components $\{(T, V, R)_i\}$, ABM relies on other architectural platform resources to be run, namely agent communication language (ACL) services, intra-agent communication services, and symbolic environment simulator (E).

In cases where decisions taken individually, in a distributed fashion, might be inefficient, there can be an agent helping to control the traffic. Note that, most of the time, $\{B_i\}$ can move around with only local information and, eventually, they have to solve conflicts with others, thus creating temporary hierarchies among them. In some scenarios, particularly in high-density traffic networks, traffic coordinator agents would minimize inter-agent communication to solve conflicts and the number of conflicts (see [54] and [55] for recent examples on algorithms these agents should include).

Figure 4 illustrates resulting control loop with this ABM. Topology of the plant and number of AGVs are among the variables that configure the model that is used for controlling the real plant.

The model is run under inputs that come from external agents and physical elements and generates outputs for the latter ones. This control loop might be too slow for many applications unless physical elements have embedded some controllers and relation with the ABM is done at a higher level of abstraction. However, even with this solution, ABM simulation has to execute fast enough to interact at real time with physical elements. This requires agents to have simple communication protocols that enable negotiations to occur within a few messages and to be efficient in taking decisions, which usually goes against reflexive, elaborated behaviors.

The higher level modules of transportation agents, $\{T_i\}$, get orders from agents representing other modules of the application ($\{A_k\}$) and try to fulfill them. To do so, T_i of each B_i must negotiate with application agents $\{A_k\}$ and other workmates which jobs they take and, when in transit, how they can be completed efficiently.

In taking decisions, $\{B_i\}$ have knowledge of their own state and the state of their lower-level counterparts ($\{V_i\}$).

Results of deliberations are transformed into requests to the $\{V_i\}$ and also to the real robots $\{R_i\}$. The last set of requests is, in fact, the output of the ABM controller. The inputs include the replies to the before mentioned requests from robots, hence closing the loop between controller and controlled system.

Apart from controlling AGV operations, ABMs can be used for functional validation and for plant characterization, which includes AGV characterization. Functional validation refers to use the ABM without real counterparts of agents and, particularly, without $\{R_i\}$. On the other hand, characterization refers to the process of measuring actual parameters from the reality, including travelling times and energy consumption at each segment and node of the traffic network.

The framework also includes mechanisms to measure worst-case execution times (WCETs) of models, and to monitor whether the control loop is closed fast enough when compared with events coming from reality.

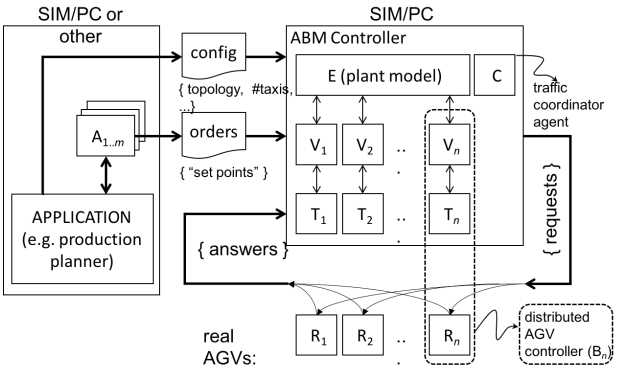

Fig. 4. ABM control loop for multi-AGV transport systems

A typical partition of the system would bind all agents into a central computer (e.g. SIM/PC on Fig. 4) and $\{R_i\}$ into real AGVs. Complementary resources to implement the system are the ones to support $\{T_i\}$ and $\{R_i\}$ communication, as well as the ABM simulation with the rest

of the application, i.e. with the part extraneous to transportation. Finally, there are the synchronizers, which will be described next.

3.3 Synchronizing simulation and reality

The synchronizer is the key resource of our architecture because it is in charge of providing the basic means to have a concurrent and online simulation and representation of reality.

The synchronizer is an interface module within each agent. It works like a middleware between the high-level T_i and low-level $(V, R)_i$of each agent to synchronize incoming messages from lower levels to outgoing channel to the higher level. As synchronization deals with time-tagged messages, we shall refer to them as events.

There are two classes of events: the ones caused by requests from T_i and the ones that are not. The former has to be synchronized as hard-real time events while the latter allow some mismatch between reality and simulation thus requiring a soft real-time synchronization of its events.

Hard real-time synchronization events (HSEs) include requests emitted by $\{T_i\}$ and the corresponding replies from $\{(V, R)_i\}$, that are expected to occur simultaneously.

Soft real-time synchronization events (SSEs) include informative messages from $\{(V, R)_i\}$, which refer to the occurrence of conditions that are autonomously managed by the low-level. For instance, detecting an obstacle or running low in battery are situations that each component of low-level ($(V, R)_i$) handle locally without requiring immediate attention by corresponding high-level. However, it is important that $V_i$and R_i run synchronized to keep virtual representation more accurate to reality and, in case of mixed-reality operation, to have reality (R_i) working together with simulated-only (V_i), which means that they have not a real counterpart.

3.3.1 Synchronization mechanism example

Synchronizing simulation with reality for the *i*-th transportation agent (B_i) means, on one side, keeping simulation messages waiting for their message counterparts from reality and, on the other, advancing simulation to trigger events which have occurred in reality but not yet in simulation.

Synchronizer modules deal with the former cases as well as with event mismatching errors, i.e. with cases where events cannot find their counterpart in pre-defined time gaps.

Events are denoted *h* or *s*, depending on them being HSE or SSE. We shall use subscripts to indicate the source and a star in superscript when they correspond to synchronizing error events. Subscripts can be *T*, *S*, *R* or *V* depending on events coming from high-level, *synchronizer*, real AGV or simulated (virtual) AGVs, respectively. For example, h_T corresponds to a HSE from T_i, $h_S{}^*$, to a HSE from the synchronizer reporting some error, and s_R and s_V, to SSE from real and simulated AGVs, respectively. (Fig. 5 shows all possible event types in a block diagram of the synchronizer.)

Furthermore, for every pair of events from low-level synchronizer must produce an equivalent, outgoing event, i.e. for any pair of h_V and h_R in response to an h_S caused by some h_T, there should be an h_S to high-level. If something fails, then appropriate sync. error events, $h_S{}^*$, are sent to high-level and, eventually, to low-level.

A typical, error-free communication protocol (Fig. 5, top left) starts by an $h_T$request, which is sent to synchronizer, which, at its turn, sends the request (h_S) to low-level. After each h_T, synchronizer waits for corresponding h_V and h_R. If both occur at the same control cycle and corresponding messages are equal, sync sends the acknowledgement (h_S) to high-level, with message contents from low-level. If messages are not the same, synchronizer emits an h_S* error event to both levels. If events from simulation (V_i) and reality (R_i) do not occur simultaneously (i.e. at the same *instant*), synchronizer either waits for h_R or causes the simulation to catch up to reality, i.e. causes to h_V happen.

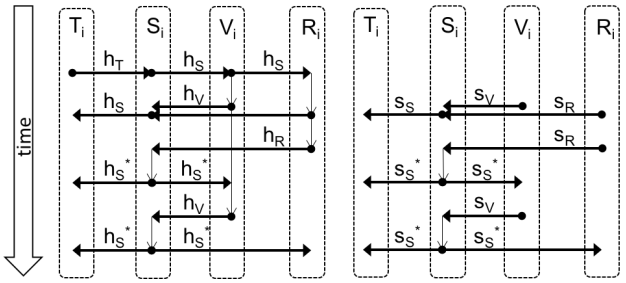

Fig. 5. Synchronizer's (S_i) event interface protocol for hard (left) and soft (right) real time events

Obstacle detections and other SSEs happen at low-level and help synchronizing virtual representation and reality, though they admit some mismatch. In this case, synchronizer tries to pair each $s_R$with corresponding s_V using a similar strategy than with $h_V$and h_R, however, it allows a tolerance in the time s_R and $s_V$occur. Again, when messages are equal s_S is sent to *high-level* and in error case, s_S* is sent to both levels. The sending of s_S* to V_i if no s_V has occurred allows $V_i$to perceive unexpected reality events and the other way round, i.e. sending s_S* to R_i if no s_R has occurred enables R_i perceiving simulation-only stimuli.

Each synchronizer uses three different states to classify incoming events, whether they are HSEs or SSEs:

1. *In-time* when events from reality and simulation arrive at the same time or within a tolerance margin.
2. *Ahead* when events from reality arrive first than from simulation. This is the worst case in terms of synchronization and should be avoided as much as possible, because reality cannot wait simulation.
3. *Behind* when events from simulation arrive first than from reality and simulation has to wait reality and update its parameters to the new behavior of reality.

Synchronizers are built upon two parallel finite state machines (FSM), namely one to deal with HSEs and another for SSEs. There is an extra state, so called *wait*, in the HSE FSM to act as the entry point for h_T to start hard real-time synchronization. To start a SSE, there is no need of any h_T, it just starts when receives a $s_R$and ors_V.

As synchronizer is independent from the application, we need to include in the events' data frames (Fig. 6) some parameters to help solving problems caused by mismatch of events in terms of time or message contents.

T_{out} is a timeout for h_R with respect to an h_S caused by h_T. If $h_R$does not occur within this period of time, an error event h_S* is sent to both levels. In this case, h_S* has a message that contains the *timeout message* from the initial h_T. Previous mechanism outlines the behavior of the *event discovery method* (EDM) of the synchronizer.

I_{max} is the maximum number of allowed ABM runs to cause the simulation to fire a h_V corresponding to a previous h_R. In case this *immediate synchronization method* (ISM) fails (i.e. the synchronizer does not receive the h_V before I_{max}), a h_S* with a predefined message is sent to both levels.

SSEs are allowed to occur at different time points within a period shorter than T_{tol}, thus $T_{tol}$is a tolerance time threshold before starting an EDM for s_R.

T_{outs} is a timeout for SSEs from reality, just as T_{out} for h_R. In a similar way than with HSEs, synchronizer starts an ISM to generate a s_V for any unmatched s_R, and that these mismatches cause s_S* that can be used to appropriately update the virtual representation of the system.

TIME	DATA FRAME					
			HSE		SSE	
Count	Message	I_{max}	Timeout message	T_{out}	T_{tol}	T_{outs}

Fig. 6. High-level events' frame format

Note that, while inter-agent communications happen inside simulation by using a FIPA compliant ACL, intra-agent communications between layers are managed by synchronizers, which include functions to send and receive messages within the simulation and to/from outside. In our study case, intra-agent communications between simulated and real parts are, obviously, wireless, by Bluetooth technology with 8N1 format at 38400 bps.

4 DEVELOPMENT METHODOLOGY AND PLATFORM

The proposed development methodology assumes that the system's specification follows the given computational model and that its implementation is done following the proposed architecture. As illustrated on Fig. 7, process begins with specifying the agents of the system and, once validated (possibly with estimated data about the application), proceeds with the refinement and synthesis stages. Depending on whether the synchronizers are used or not, these processes should be applied to the whole controller description or only to the low-level parts. Next subsections will detail each stage.

4.1 ABM description

Instances of systems' ABMs are built by describing the functionality of each agent. In order to simulate resulting ABM-based controllers, designers have to:

1. Develop an environment module *E* which will be in charge of updating the symbolic representation of the system in accordance to agent actuators and provide information of the system state through agent sensors. Note that data from *E* can be used by an agent to create a HMI of the system and seen by other agents as a kind of blackboard.
2. Create the application-specific agents. Initially they can be stubs to provide/collect data to/from the rest of the system, but they will have to interface will true application components at the end.
3. Configure and adapt AGVs, particularly, their low-level. For that, they can use previous examples as a reference and perform a customization of the code. These customizations imply a rough estimation of AGVs' parameters with respect to the plant (delays, energy consumption, battery charging/discharging profiles, *et cetera*), at least from nominal data. Note that each B_i is composed of T_i and V_i at simulation level and that V_i interacts both with T_i and E while T_i interacts with other agents ($B_{j\neq i}$ and A_k).
4. Deploy final system.

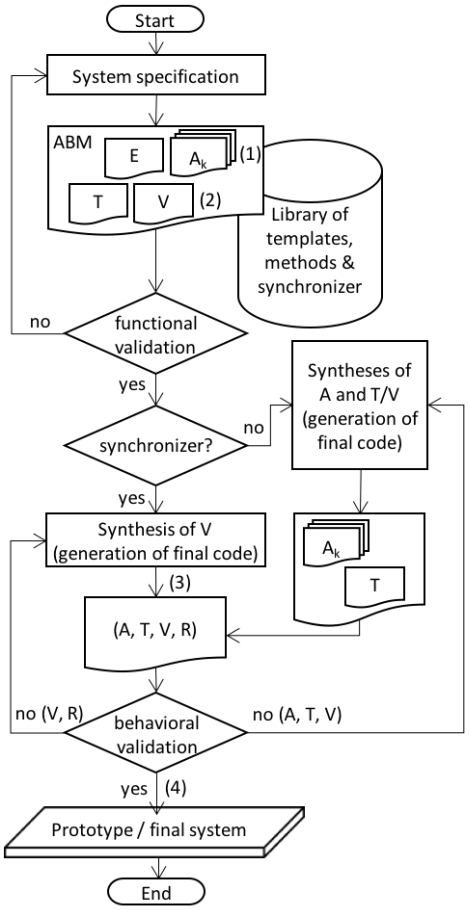

Fig. 7. Transportation system controllers' development methodology with and without synchronizers

Take into account that, even though agents communicate within a single model, they should use an ACL compliant with the FIPA-ACL specification [56] to make it possible to distribute the ABM model execution.

4.2 Functional validation

It consists of checking whether the system works properly and of foreseeing the characteristics of the transportation system (e.g. number of AGVs and average speed) according to the application requirements.

4.2.1 Requirements analysis

Application requirements' analysis must be performed to determine the characteristics of the ABM. For clinical analysis laboratories, the workload is in the range of 4,000 to 12,000 samples/day. The lowest value is drawn from the fact that a typical workload for an automated laboratory at a hospital is 1 million samples/10 millions of tests/determinations per year [49], which gives a throughput of 4167 samples/day if we consider working days only. Maximum value is taken from one of the leading-edge automated laboratories, the Cobas 8000 [57], which can reach an order processing time of as little as 36 s (by using 5-sample racks on conveyors).

The time to fulfill an order depends on the vehicle characteristics (velocity, number of tubes it carries, *et cetera*), the analyzers' throughputs, the plant layout and the route they follow (e.g. analyzers they go to, or need to use the re-circulation lane).

Realistic computation of the values for the characteristics parameters must take into account differences between prototype and final plants.

Our experimentation plant (Fig. 8) is twice smaller than the real one, and average response time of an order is 236.3 s, approximately.

Worst cases require each AGV travelling a distance of 22 (11x2) m (i.e. visiting each different analyzer and the re-circulation lane to repeat some test), consuming an average time of 240.8 (120.4x2) s, plus 36 s taken by the analyzers. Therefore, each AGV takes 276.8 s to complete an order.

The best case occurs when each AGV travels a distance of 14.6 (7.3x2) m long (it goes just to one analyzer and does not use the re-circulation lane), so the total time (AGV travel + analyzer time) is around 195.8 s.

System throughput not only depends on the time to complete an order but also on the number of analyzers, on the analyzers' characteristics (particularly, their throughput and quality of tests), on the number of transportation agents, on their coordination efficiency, and, last but not least, on the LIMS's ability to pass work orders to AGVs in an effective sequence.

Estimation of throughput uses simple scenarios because of the high complexity of its computation for real cases. For instance, we have used two scenarios to determine whether the proposed plant can manage typical automated laboratory workloads and how has to be configured, i.e. its ABM characteristics.

The first scenario is made of an AGV carrying orders constantly during 24 hours/day. It is assumed that AGVs have battery autonomies of 8 hours and the same re-charging time, thus, at least two AGVs, one moving through the plant and the other re-charging are required to cover a full working day. Under this scenario, the lowest system throughput is about 1560 samples/day (worst case response time of an order), which covers around 37.4% of the minimum daily workload (4167 samples/day).

The second scenario is an extension of the first one to three AGVs that transport orders 24 hours/day at the same time, where separation between each other is at least of 4

meters (to avoid possible delays related to coordination between transportations agents and analyzers). At any time, there are other three robots re-charging batteries. The system throughput for this case is around 4680 samples/day, which exceeds the minimum workload in 12.3%.

We shall see later, in Section 4.2.3, that the proposed controller is able to control up to 20 AGVs in real time, so it can also handle this scenario, with only 6 robots. Consequently, the proposed system can manage the minimum workload for an automated laboratory of a hospital.

The fact that AGVs move by following lines can create traffic bottlenecks when the number of AGVs grows.

To study if some bottlenecks emerge, some simulations have been done in two different scenarios with 20 AGVs each one, as this is the limit of the experimental plant. Initial positions in the plant have been on the topmost lane (15 idle AGVs) and on the leftmost lane (5 AGVs ready and waiting for transportation orders) of the layout.

In the first scenario, every AGV received random orders to fulfill and, in the second one, each AGV accepted orders that forces to go to each analyzer. From the simulation of these scenarios, a main bottleneck emerges at junction 3 (bottom left part of Fig. 8) and sometimes there were minor bottlenecks in crossings to/from analyzers. Therefore, after three non real-time simulations for each scenario during 9050 ticks each one, results show that concentration of AGVs on the spot around junction 3 is 15% higher for the scenario with random orders than scenario with constant ones.

Although other results can be found for different simulation times, initial position of AGVs, *et cetera*, those concentration points have to be treated to avoid degrading overall throughput. Possible solutions include modifying the plant layout by creating e.g. roundabouts or by extending the one-dimensional layout to 2D in those areas, and improving work order assignment and sequencing.

4.2.2 Characterization of transportation agents B_i

Model accuracy depends on how good is the characterization of the actual plant. Fortunately, synchronizers allow model simulations to run concurrently with real plants even if inaccurately characterized. However, accurate characterization of models improve synchronization quality, which can be defined in terms of the quantity of unnecessary delays in the simulation execution and the number of extra iterations in the simulation loop to keep up with events from reality.

Static data such as traffic network and nominal characteristics of vehicles such as average speed and energy consumption can be used for system validation and as a set of initial values for the model. However, in order to control a real plant, parameters should be as accurate as possible and, for this reason they have to be estimated from a series of test runs [58].

Plant characteristics are of two types: one that define traffic network and the other for the functional and non-functional behavior of the B_i. We assume the traffic network be constant and defined by a topological graph known to all B_i of the system (plant layout on Fig. 8).

In a simple version, cost data consist of the time to travel from a node to another and the time devoted at each node to decide which outgoing arc to take.

For every order request from a T_i to a R_i, the delay time that takes to T_i to get a reply from R_i is recorded. This delay is compared to the previous one in the same node or arc of the map graph and updated accordingly so that further decisions of T_i and the reactive behavior of V_i are more accurate to the reality.

4.2.3 Estimation of controller characteristics

To test the maximum load of the system, a series of simulations with different quantities of robots performing random transport orders were done to estimate the worst-case execution time (WCET).

The average WCET was 16ms, and the maximum communication time was 20 ms, although this value corresponds to the case for 20 AGVs, it was extrapolated from real data obtained from cases with up to 4 robots. Consequently, the control loop takes 36 ms at worst and the maximum cycle frequency for the ABM controller is about 28 Hz. With this controller period, real time frequencies of events must be 14 Hz or lower.

Taking into account the geometry of the traffic network of the experimental plant and the average speed of robots, that frequency allows the simulation controlling 20 real robots $\{R_i\}$ in real time with a spatial resolution under the cm, which is acceptable for the previously-presented laboratory.

After system functionality and estimates were validated, a prototype with three AGVs was used to verify the characterization, WCET control, and deployment stages.

4.3 Synthesis of low-level

In the proposed development process (Fig. 7) the only part of the system to be implemented is the low-level one, as the rest is kept as a simulation model. In the study case, the prototype of the automated laboratory used Parallax Boebots and only the Netlogo code for low-level (V_i) had to be ported to PBASIC for the Basic Stamp board that controls the robots.

4.4 System prototyping

Within the proposed methodology, system prototypes are ready as soon as $\{B_i\}$ have their $\{V_i\}$ embedded into physical counterparts (AGVs), and some environment has been created for them.

The resulting prototype for the study case ([58]-[60]) is shown in Fig. 8, with AGV emulators in the foreground and the projection of the user interface screen in the background.

Robots determine its position in the plant by detecting marks (short crossing lines over layout guide lines). Marks correspond to nodes of a directed graph that tell robots how to manage next step according to their current state, including node location, and to the order currently in course. This mechanism lets robots know their position between marks but not its exact location between them. Fortunately, exact positioning is only required for local decisions. When robots move, they use sensors to follow the line and detect marks and other sensors for detecting obstacles.

Robots wait for LIMS instructions at the end of the return lane. After receiving a transport order, they proceed to the loading dock (bottom left corner) and begin their journey. If nothing abnormal happens, they contact the LIMS just at the bifurcation between the return lane and the recirculation lane (top right bifurcation) to decide which line to take in terms of the successfully done tests. The recirculating lane (line at the middle) can also be used by AGVs carrying samples that still wait for acknowledgement of their tests or for their repetition, in case of test failures.

At the beginning of the returning lane, AGVs have their tube racks unloaded, and, at the waiting queue, they have their batteries re-charged (if needed), and follow their pace to the programming spot.

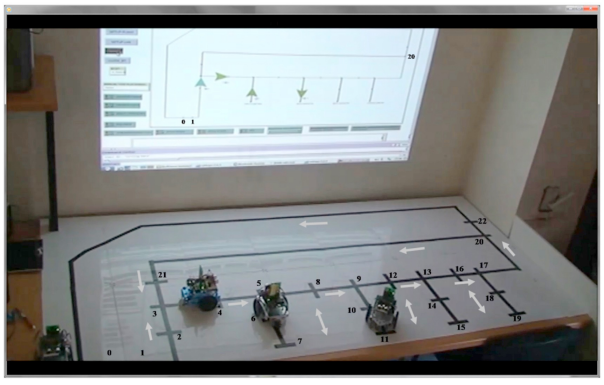

Fig. 8. Plant prototype (bottom) and HMI (top)

4.5 Behavioral validation

It requires real time monitoring and on-line characterization of transportation agents.

The rest of the section is devoted to explain how are AGVs and controller characterized with respect to a given plant, the mechanism included in the platform to ensure that real-time control constraints are met and the deployment of the study case.

4.5.1 Pre-runtime characterization of AGVs

Off-line robot characterization is very important to have a good initial matching between simulation and reality, and to improve decisions and robot actions made by $\{T_i\}$. After that, continuous on-line updates of these data might be necessary to adapt controller to dynamically changing system characteristics.

The characteristics of each robot with respect to the plant traffic network have been obtained by averaging the travel time at each segment for 30 runs. The full circuit is 11 m long and took an average time of 112.654, 125.456 and 123.169 s per robot to be completed.

Besides, during pre-runtime characterization, it has been the first opportunity to detect noise influence from different sources such as wireless communications' delay variability and erroneous data from sensors. In the first case, influence of noise that cause some failure has not been reported for this testbed, possibly because Bluetooth transceivers (at AGVs and at the computer) are closer enough to keep signal to noise ratio inside operational values that avoid failures. Also, there have not been detected interferences between AGVs' transceivers and other electromagnetic sources. Unfortunately, line detection sensors were too sensitive to sun and fluorescent light, which caused detection errors that were solved by protecting them from that influence.

4.5.2 Computation of controllers' WCET

With the ABM simulation running on an inexpensive laptop (64-bit MS Windows 7 OS on a 1.65GHz AMD E-450 CPU, 4 GB RAM machine) that communicated with three robots via serial protocol over Bluetooth, the WCET for the controller of a 3-robot system was 431 ms

Taking into account that robots move at speeds up to 10 cm/s and that the closest nodes in the automated laboratory prototype are a bit larger than that, each T_i needs to handle, at least, one HSE per second. Therefore, the controller should be able to reply to 3 events per second, which is not possible with that WCET and real robots would had to wait when execution times are close to it.

Fortunately, the average control cycle period duration was roughly 2 ms, and execution times over 333 ms account only for 0.0023% of the total, and those over 36 ms (20 taxis) are roughly the 1.17%, as illustrated on Fig. 9.

Note that, even in these cases, the system can work properly because of synchronizers, though it might run not as efficiently as it could.

However, to get rid of this problem it is possible to use better machines with tailored OSs to improve execution times and avoid interferences with controller processes.

4.5.3 Synchronization quality

The more accurate is the characterization of the physical elements in a plant the better will be the control and, subsequently, the efficiency of the system.

Synchronization quality is given in terms of percentages of synchronizers "in-time" states with respect to the total number of HSE states they go through. For instance, in pre-runtime characterization, simulation tends to be optimistic and goes ahead reality about 33% of the time, as time delays at segments are initially set to zero. However, most of the time (66%), simulation and reality go together and a mere 1% of time is used to run ISM to make simulation keep up with reality.

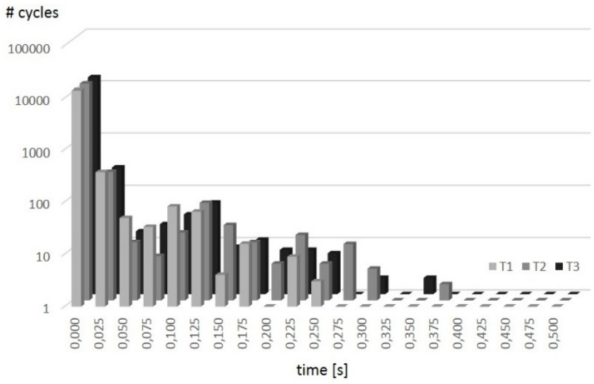

Fig. 9. Control cycle execution time logarithmic histogram of three robots, T_1, T_2 and T_3

4.5.4 Real-time monitoring

To guarantee real-time monitoring and control, all delays are compared to the WCET of the main control loop body to be sure that no inputs from the plant will be lost or taken into account out of time. Therefore, the control loop has a cycle period compatible only with robots whose embedded controllers are able to understand quite complex instructions, with execution times larger than the WCET of the model.

In case delays are closer to WCET, there are alternatives to preserve coherence between simulation and reality such as including time-stamps into the messages or minimizing the WCET by appropriately modifying the scheduling of agent execution [61].

5 CONCLUSIONS

Systems that run applications on the industrial domain must solve the internal transportation aspect. In this paper, we have proposed a framework to rapidly design and deploy the corresponding subsystems directly from agent models.

The proposed MAS architecture organizes agents into two classes, the application-specific ones and the transportation ones or taxis. The latter follow a three-tier architecture, that includes an intermediate layer to synchronize the lower level parts, which can also be run on the actual robots.

Simulators of ABMs with such an architecture can be used: 1) for functional validation; 2) for plant characterization, which includes testing whether real time requirements are met, parameter identification, and controller setup, and 3) as a model for the controller of the transportation system, including a mixed-reality environment for monitoring and supervising in human-assisted operation.

We have shown how synchronizer maintains coherence between symbolic system representation and reality so that transportations agents can take timely decisions. Furthermore, it replaces traditional direct monitoring so that system representation is the outcome of a system simulation that runs synchronized with reality. As a consequence, it is possible to make simulation-only elements interact with real ones.

Experimental results show that the proposed strategy minimizes the time-to-prototype as well as the time-to-market, provided that the development platform is the same that the deployment one.

REFERENCES

[1] R.-S. Chen, K.-Y. Lu, and C.C. Chang, "Intelligent warehousing management systems using multi-agent", in: Int. J. Comput. Appl. Technol. 4 (2003), pp. 194–201.

[2] M. Al khawaldah, A. Nuechter, "Multi-Robot Cooperation for Efficient Exploration". Automatika, 2014, 55(3), 276–286.

[3] S. Behnke, "Dynamaid, an Anthropomorphic Robot for Research on Domestic Service Applications". Automatika – Journal for Control, Measurement, Electronics, Computing and Communications, 2011, 52(3), 233–243.

[4] E. Ivanjko, E. K. Filter, "Sonar-based Pose Tracking of Indoor Mobile Robots". Automatika, 2004, 45(3-4), 145–154.

[5] E. Di Lello, A. Saffiotti, R. Ecology, "The PEIS Table: An Autonomous Robotic Table for Domestic Environments". Automatika – Journal for Control, Measurement, Electronics, Computing and Communications, 2011, 52(3), 244–255.

[6] F. Mastrogiovanni, A. Scalmato, A. Sgorbissa, R. Zaccaria. "Robots and Intelligent Environments: Knowledge Representation and Distributed Context Assessment". Automatika – Journal for Control, Measurement, Electronics, Computing and Communications, 2011, 52(3), 256–268.

[7] D. Matko, "Mobile Robots Tracking Using Computer Vision". Automatika – Journal for Control, Measurement, Electronics, Computing and Communications, 2005, 46(3-4), 155–163.

[8] S. Schreiber, A. Fay, S. Member, "Requirements for the benchmarking of decentralized manufacturing control systems", in: Emerging Technologies and Factory Automation, 2011.

[9] J.Z. Hernández, S. Ossowski, R. Juan, A. García-serrano, "On Multiagent Co-ordination Architectures: A Traffic Management Case Study", in: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001: pp. 1–9.

[10] O. Banias, R. Precup, D. Curiac, "Multiagent architecture applied in decentralized real-time urban road traffic control", in: Applied Computational Intelligence and Informatics, 2009: pp. 271–276.

[11] A. Guerrero-Ibáñez, J. Contreras-Castillo, R. Buenrostro, A. Martí, and A. Muñoz, "A Policy-Based Multi-Agent management approach for Intelligent Traffic-Light Control", in: Intelligent Vehicles Symposium, 2010: pp. 694–699.

[12] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, "SUMO – Simulation of Urban Mobility", in: Third Int'l. Conf. on Advances in System Simulation (SIMUL 2011), 2011: pp. 63–68.

[13] F. Mastrogiovanni, A. Scalmato, A. Sgorbissa, R. Zaccaria. "Robots and Intelligent Environments: Knowledge Representation and Distributed Context Assessment". Automatika – Journal for Control, Measurement, Electronics, Computing and Communications, 2011, 52(3), 256–268.

[14] M. Cossentino, C. Lodato, S. Lopes, P. Ribino, "Multi Agent Simulation for Decision Making in Warehouse Management", In: Computer Science and Information Systems, 2011: pp. 611–618.

[15] H. L. Liang, J. Verriet, R. Hamberg, and B. van Wijngaarden, "Graphical Configuration of Agent-Based Warehouse Management and Control Systems". In Advances on Practical Applications of Agents and Multi-Agent Systems, Advances in Intelligent and Soft Computing, (2012): pp. 265–268.

[16] P. Farahvash, T.O. Boucher, "A multi-agent architecture for control of AGV systems", Robotics and Computer-Integrated Manufacturing. 20 (2004) 473–483.

[17] A. Wallace, "Multi-Agent Negotiation Strategies and the Flow of AGVs", International Journal of Production Research. 45 (2007).

[18] S.C. Srivastava, A.K. Choudhary, S. Kumar, M.K. Tiwari, "Development of an intelligent agent-based AGV controller for a flexible manufacturing system", The International Journal of Advanced Manufacturing Technology. 36 (2007) 780–797.

[19] M. Hafidz, H. Lin, T. Murata, "Dynamic Task Assignment of Autonomous AGV System Based on Multi Agent Architecture", in: International conference on Progress in Informatics and Computing (PIC), 2010: pp. 1151–1156.

[20] R. Erol, C. Sahin, A. Baykasoglu, V. Kaplanoglu, "A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems", Applied Soft Computing. 12 (2012) 1720–1732.

[21] Open Agent Based Modeling Consortium, https://www.openabm.org/page/modeling-platforms, last access 22 Dec 2014.

[22] A. Kashif, X. Hoa, B. Le, J. Dugdale, "Agent-based framework to simulate inhabitants' behaviour in domestic settings for energy management", in: International Conference on Agents and Artificial Intelligence (ICAART), 2009.

[23] P. Fonseca i Casas, J. Casanovas, X. Ferran, "Passenger flow simulation in a hub airport: An application to the Barcelona International Airport", Simulation Modelling Practice and Theory. 44 (2014) 78–94.

[24] J. Babić, S. Marijan, I. Petrović, "Introducing Model-Based Techniques into Development of Real-Time Embedded Applications". Automatika, 2011, 52(4), 329–338.

[25] C. Mitsantisuk, M. Nandayapa, K. Ohishi, S. Katsura, "Design for Sensorless Force Control of Flexible Robot by Using Resonance Ratio Control Based on Coefficient Diagram Method". Automatika – Journal for Control, Measurement, Electronics, Computing and Communications, 2013, 54(1), 62–73.

[26] D. Vasquez, T. Fraichard, R. Siegwart, "Towards Safe Vehicle Navigation in Dynamic Urban Scenarios". Automatika – Journal for Control, Measurement, Electronics, Computing and Communications, 2009, 50(3-4), 184–194.

[27] S. Avenue, "Modelling and Simulation of Transportation Systems: a Scenario Planning Approach". Automatika, 2009 50, 39–50.

[28] I. Bačić, K. Malarić, M. Pejanović-Djurišić, "Simulation Model for Evaluation of the DVB-SH-A Performance". Automatika – Journal for Control, Measurement, Electronics, Computing and Communications, 2014, 55(2), 170–181.

[29] T. Sandu, N. Denz, B. Page, "Model-Driven Software Development and Descrete Event Simulation — Concepts and Example". Automatika, 2009, 50(1–2), 17–27.

[30] I.Y.H. Chen, B. MacDonald, B. Wunsche, "Mixed reality simulation for mobile robots", in: IEEE International Conference on Robotics and Automation, 2009: pp. 232–237.

[31] Player, Cross-platform robot device interface & server, http://playerstage.sourceforge.net/index.php?src=player, last access 21 Nov. 2015.

[32] Stage, 2D multiple-robot simulator, http://playerstage.sf.net /index.php?src=stage, last access 21 Nov. 2015.

[33] Willow Garage, "ROS: Robot Operating System", https://www.willowgarage.com/pages/software/ros-platform, last access 21 Nov. 2015.

[34] Coppelia Robotics, "V-rep: Virtual Robot Experimentation Platform", http://www.coppeliarobotics.com/, last access 21 Nov. 2015.

[35] P. Davidsson, L. Henesey, L. Ramstedt, J. Törnquist, F. Wernstedt, “An analysis of agent-based approaches to transport logistics”, Transportation Research Part C: Emerging Technologies, 13. 2005.

[36] A. Jaoua, D. Riopel, M. Gamache, “A simulation framework for real-time fleet management in internal transport systems”, Simulation Modelling Practice and Theory. 21 (2012) 78–90.

[37] L.A. Santa-Eulalia, G. Halladjian, S.D. Amours, J. Frayret, “Integrated methodological frameworks for modeling agent-based advanced supply chain planning systems: A systematic literature review”, In J. Ind. Eng. &Management (JIEM). 4 (2011) 624–668.

[38] J. Holmgren, P. Davidsson, J. a. Persson, L. Ramstedt, “TAPAS: A multi-agent-based model for simulation of transport chains”, Simulation Modelling Practice and Theory. 23 (2012) 1–18.

[39] Y. Yu, A. El Kamel, G. Gong, F. Li, “Multi-agent based modeling and simulation of microscopic traffic in virtual reality system”, Simulation Modelling Practice and Theory. 45 (2014) 62–79.

[40] M. Armendáriz, C. Juan, A. Peleteiro, C.R.P.H. Tudor, A.J.F. Kennedy, “Carpooling: A multiagent simulation in Netlogo”, in: 25th European Conf. on Modelling and Simulation (ECMS), 2011.

[41] I.F. Chaile and Ll. Ribas-Xirgo, "Agent simulator-based control architecture for rapid development of multi-robot systems," in International Conference on Systems, Control, Signal Processing and Informatics (SCSI 2015), INASE Joint ConferencesBarcelona, Spain, April 7-9, 2015.

[42] Ll. Ribas-Xirgo, Ismael F. Chaile, “Multi-Agent-based Controller Architecture for AGV Systems”, IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), 2013.

[43] A. Fernández-Caballero and J.M. Gascueña, “Developing Multi-Agent Systems through Integrating Prometheus, INGENIAS and ICARO-T”. In Proceedings of International Conference on Agents and Artificial Intelligence (ICAART), 2009.

[44] B. Shirazi, I. Mahdavi, N. Mahdavi-amiri, “Simulation Modelling Practice and Theory iCoSim-FMS: An intelligent co-simulator for the adaptive control of complex flexible manufacturing systems”, Simulation Modelling Practice and Theory. 19 (2011) 1668–1688.

[45] J.J. Gómez-Sanz, C.R. Fernández, J. Arroyo, “Model driven development and simulations with the INGENIAS agent framework”, Simulation Modelling Practice and Theory. 18 (2010) 1468–1482.

[46] Ll. Ribas-Xirgo, I. F. Chaile, “An Agent-based Model of Autonomous Automated-Guided Vehicles for Internal Transportation in Automated Laboratories”. In International Conference on Agents and Artificial Intelligence (ICAART), 2013.

[47] Ll. Ribas-Xirgo, A. Miró-Vicente, I. F. Chaile, and A. Josep Velasco-González, “Multi-Agent Model of a Sample Transport System for Modular In-Vitro Diagnostics Laboratories”, IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA), 2012.

[48] T. De Wolf, T. Holvoet, “Towards Autonomic Computing: Agent-Based Modelling, Dynamical Systems Analysis, and Decentralized Control”, in: First Int’l. Workshop on Autonomic Computing Principles and Architectures, 2003: pp. 470–479.

[49] Hospital Reina Sofía de Córdoba, “Reina Sofía Análisis Clínicos,” https://youtu.be/_u64EUrFF4I, 2009, last access 12 Dec 2014.

[50] Ll. Ribas-Xirgo, J. M. Moreno-Villafranca, I. F. Chaile. “On using automated guided vehicles instead of conveyors”. IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), 2013 (pp. 1–4).

[51] J. Himoff, G. Rzevski, M. Hinton, P. Skobelev, “MAGENTA Technology: Multi-Agent Logistics i-Scheduler for Road Transportation”, in: Proc. of AAMAS, 2006.

[52] Roche Cobas 8000 analyzer, https://youtu.be/lJaTt_S7zW8, last access 22 Dec 2014.

[53] I. F. Chaile-Alfaro and Ll. Ribas-Xirgo, “MASYM, a Framework to Deploy Synchronized Industrial Systems Based on Any ABM Simulator,” Lat. Am. Trans. IEEE, v. 13, n. 10, 3244–3252, 2015.

[54] T. Standley, R. Korf, “Complete Algorithms for Cooperative Pathfinding Problems”, in: 22nd International Joint Conference on Artificial Intelligence, 2011: pp. 668–673.

[55] J. Yu, S.M. Lavalle, “Planning Optimal Paths for Multiple Robots on Graphs”, in: International Conference on Robotics and Automation (ICRA), 2013: pp. 3612–3617.

[56] Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org/repository/aclspecs.html, last access 21/10/2014.

[57] Roche Cobas 8000 modular analyzer series brochure, http://www.cobas.com/content/dam/cobas_com/pdf/product/cobas-8000/cobas 8000 brochure.pdf, last access 22 Dec 2014.

[58] Ll. Ribas-Xirgo, I.F. Chaile, “Agent-based Automatic Guided Vehicle (AGV) System Development Framework”, https://youtu.be/pJmNw24aBdQ, last access 12 Dec 2014.

[59] I.F. Chaile, L. Ribas-Xirgo, “Controlling multiple AGVs with agent-based-modelling”, https://www.youtube.com/watch?v=2LdK5AhJhuQ, last access 12 Dec 2014.

[60] I.F. Chaile, L. Ribas-Xirgo, “Synchronizing automated guided vehicles (AGV) with agent-based model (ABM) for control”, https://youtu.be/A4tk7kRtjiA, last access 12 Dec 2014.

[61] P. Mathieu, Y. Secq, “Environment updating and agent scheduling policies in agent-based simulators”, in: Int’l. Conf. on Agents and Artificial Intelligence (ICAART), 2012: pp. 170–175.

Ismael Fabricio Chaile-Alfaro received the Ph.D. in Microelectronics and Electronic Systems, the M. Sc. degree in Micro and Nanoelectronics both from Universitat Autònoma de Barcelona (UAB), the master's degree in General Direction and Strategic Planning from ESERP Business School and the Electronic Engineering degree from Universidad Nacional de Tucumán, in 2016, 2010, 2010 and 2005, respectively. He worked in several companies in Argentina until 2008. In 2009 he moved to Spain and started to work as teacher assistant in C programming, Embedded systems and Physical Multi-agent systems at UAB; after that he worked as Linux Programmer and Embedded Software Engineer. He has authored or co-authored 9 papers published in journals and conference proceedings. His research interests include multi-robot and multi-agent systems, emergent distributed technologies, autonomous robots, real-time embedded systems, mixed reality, simulations and industrial SCADA.

Lluís Ribas-Xirgo received the M.Sc. and Ph.D. degrees in Computer Science from the Universitat Autònoma de Barcelona (UAB), Bellaterra, Spain, in 1992 and 1996, respectively. He is currently an Associate Professor with the Department of Microelectronics and Electronic Systems, UAB, where he is head of the research group on Software/Hardware Agent-based Distributed Electronic Systems. He has authored or co-authored more than 75 papers published in journals, books, and conference proceedings. His research areas include physical agent systems, embedded systems and robotics, with emphasis on design and verification methodologies and tools. He is also very active in learning technologies and applications.

AUTHORS' ADDRESSES

Ismael F. Chaile-Alfaro, Ph.D.
Assoc. Prof. Lluís Ribas-Xirgo, Ph.D.
Research Group on Distributed Electronic Systems,
Microelectronics and Electronic Systems Department,
School of Engineering,
Carrer de les Sitges, Campus UAB, ES-08193 Bellaterra,
Barcelona, Spain
email: chaileif@gmail.com, lluis.ribas@uab.cat

Received: 2015-03-14
Accepted: 2016-05-24