Simulation of Multi-agent Manufacturing Systems using Agent-based Modelling Platforms

José Barbosa[1,3,4], Paulo Leitão[1, 2]
[1] Polytechnic Institute of Bragança, Quinta Sta Apolónia, Apartado 1134, 5301-857 Bragança, Portugal
[2] Artificial Intelligence and Computer Science Laboratory, R. Campo Alegre 102, 4169-007 Porto, Portugal
[3] Univ. Lille Nord de France, F-59000 Lille, France
[4] UVHC, TEMPO research center, F-59313 Valenciennes, France
E-mails: {pleitao, jbarbosa}@ipb.pt

***Abstract*- Multi-agent systems (MAS) are driving the way to design and engineer control solutions that exhibit flexibility, adaptation and reconfigurability, which are important advantages over traditional centralized systems. The understanding, design and testing of such distributed agent-based approaches, and particularly those exhibiting self-* properties, are usually a hard task. Simulation assumes a crucial role to analyse the behaviour of MAS solutions during the design phase and before its deployment into the real operation. Particularly, Agent-Based Modelling (ABM) tools are well suited to simulate MAS systems that exhibit complex phenomena, like emergent behaviour and self-organization. This paper discusses the simulation of agent-based manufacturing systems and introduces the advantages of using ABM tools. The NetLogo platform is used to illustrate the benefits of such tools in the manufacturing world on the specification of a MAS system for a washing machine production line.**

I. INTRODUCTION

Lately, Multi-Agent Systems (MAS) are being used to solve the emergent challenges in manufacturing control systems demanding for flexibility, adaptation and reconfigurability. Multi-agent systems offer an alternative way to design and engineer control systems, differing from the conventional approaches due to their inherent capabilities to adapt to emergence without external intervention [1]. MAS solutions are based on the decentralization of control over distributed structures, providing modularity, robustness and autonomy, and solving at least 25% of the manufacturing problems [2].

The required software to develop agents is shorter and simpler than the software required by centralised approaches, leading to an easier development, debug and maintenance [3]. However, the analysis, test, debug and validation of the behaviour of the agent-based systems, which are distributed by nature, are usually difficult and time consuming, requiring the use of tools that support the correction of misunderstandings and errors during the design phase and before its deployment into the real operation. The use of simulation platforms that support a rapid prototyping and proof-of-concept is useful to overcome this question. In fact, simulation takes even more importance since is one of the easiest ways to represent, test and therefore understand the system behaviour. However, simulation platforms are usually developed case-by-case taking into consideration the particularities of the agent-based system and don't support efficiently the simulation of complex phenomena.

Agent-Based Modelling (ABM) platforms are tools that allow the modelling/simulation of complex adaptive systems by using agents, providing a way to output the simulation results in a graphical manner according to several designed scenarios. The simulation results can be used to extract conclusions about the system´s behaviour and consequently to refine the specification of the agent-based model. These tools provide an easy and powerful simulation capability which enables a fast testing and prototyping environment.

The objective of this paper is to discuss the use of modelling/simulation tools, and particularly the use of ABM tools, to support the development of agent-based manufacturing control systems. For this purpose, the existing ABM platforms will be briefly analysed and compared and then the ABM NetLogo environment will be used to model and simulate an agent-based control system for a washing machine production line case study.

The rest of the paper is organized as follows: Section 2 overviews the importance of simulation of agent-based systems and Section 3 discusses the simulation of agent-based systems by using ABM platforms and briefly introduces the NetLogo modelling and simulation environment. Section 4 uses a case study to illustrate the applicability of ABM tools to model and simulate agent-based solutions. At last, Section 5 rounds up the paper with the conclusions.

II. SIMULATION OF AGENT-BASED SYSTEMS

Simulation can be defined as the use of mathematical models to recreate a situation, often repeatedly, so that the likelihood of various outcomes can be accurately estimated. The model is a description of the system, with the detail of the model ranging from a simple representation to a complex behaviour of all intervenient involved in the system. The simulation extends the modelling process by adding time to the model and with that, the model behaviour can be observed for a better analysis.

The use of simulation environments can provide several advantages [4]:

- The system or part of the system can be tested, debugged and validated without the need to use the (real) physical equipments.
- The reproduction of different scenarios, abnormal conditions or dangerous tests can be done easily and safely in this virtual world.
- Data can be reused for operator training and maintenance, and the simulations can be repeated as many times as necessary to the correct understanding and tuning of the system control.
- The simulation process can be compressed (by accelerating the time span), obtaining results that in real environment take long time.

An example is pilots that are trained in conditions simulating high-altitude flights.

The visualization is an add-on into the simulation. Indeed, humans understand better a system or a concept when they see things happening in a graphical way and not only in a textual way. In fact, "*a picture is worth a thousand words*".

The simulation of control systems is well established as a practical tool of control engineers, with simulation techniques being used increasingly in last decades for the design and analysis of control systems. A good example is the use of Matlab and Simulink for modelling and simulating control systems. In manufacturing, the simulation allows the detection of errors, mistakes and misunderstandings during the design phase and before going to the implementation and commissioning. This allows reducing time and costs in the development of control systems. A good indicator of the importance of simulation in engineering can be stated with the fact that some universities have already dedicated curricula to address simulation topics [5].

Agent-based systems, due to its distributed nature, introduce new requirements for modelling and simulation, and the understanding of the system's behaviour can be increasingly difficult as the system grows in complexity.

Several environments for the simulation of multi-agent systems are reported in the literature, namely in [6]. A well-known example in the manufacturing domain is the MAST (Manufacturing Agent Simulation Tool) simulation environment [7], developed by the Rockwell Automation, focusing the dynamic product routing. MAST was used to simulate two real scenarios [8]: the holonic packing cell at the University of Cambridge, UK and the pallet transfer system at the Automation and Control Institute (ACIN) of the Technical University of Vienna. Another example is found on [9] where a Virtual Reality based approach is used to model and simulate a holonic application to die-casting industry.

Nevertheless, these platforms are developed case-by-case and according to the application particularities, requiring a significant effort to simulate the behaviour of agent-based manufacturing control systems. Additionally, the complexity associated to the simulation of distributed systems is increased in presence of complex phenomena, like adaptation, self-organization and chaos, which are common characteristics of complex adaptive systems. Normally, emergent phenomenon has behaviours that differ from classical sciences and the classical methods, like top-down techniques of non-linear systems, is not anymore sufficient. This suggests the use of computational platforms that simplifies these tasks and ensures a framework to simulate/validate strategies during the design phase.

Note that when talking in simulation and MAS, two different directions are possible, namely the simulation of MAS systems and using MAS systems for the simulation of control systems. In this work the focus is centred in studying the simulation of designed agent-based control systems and not the use of agent-based approaches as simulation environments to perform the simulation of control systems.

III. Agent-based Modelling and Simulating Environments

ABM is a class of computational models for simulating the simultaneous operations and interactions of multiple autonomous agents aiming to recreate and predict the occurrence of complex phenomena. ABM tools allow the modelling of a system or process by using a MAS system, and posterior simulation in presence of complex phenomena. However, in this work the intention is also to consider the use the ABM tools to simulate agent-based control systems.

These platforms are being used to simulate agent-based models for different application domains, such as economics, chemical, social behaviour and logistics. An interesting example in the manufacturing domain, described in reference [10], is the use of the NetLogo platform to simulate the dynamic determination of the best path to route the products in situations characterized by the occurrence of disturbances.

A special remark to the use of ticks (universal time) in simulation environments instead of the real time clock, otherwise it is impossible to compare different simulation results (which are dependent of some parameters such as the processing power of the PC processor).

A. Analysis of existing ABM platforms

Several ABM tools are currently available on the market presenting different functionalities, graphical interfaces and also programming languages. As examples, it is possible to refer Repast [11], Swarm [12], NetLogo [13] and Mason [14]. The scope of this work is not to survey in detail the available ABMs but instead to briefly analyse and compare the tools based on previous surveys that already provide detailed analysis of the most important available ABM tools.

Reference [15] presents a survey on free java libraries to support agent simulation, being analysed four ABM tools, namely Repast, Swarm, Quicksilver and VSEdit (www.vseit.de). On a first approach, details such as type of license, quality of the provided documentation, type of existing support and viability of future product support and maintenance were analysed. Taking into consideration the previous aspects, the Repast and Swarm platforms were rated with higher classifications. On a more technical analysis, the authors classified aspects like support for modelling, simulation control and ease of use. Also in the analysed technical parameters, the Repast and Swarm platforms were rated with the higher classifications.

A second survey [16] has evaluated eight ABMs and separated them into three groups based on their license: open source, shareware/freeware and proprietary. On the first group, Mason, Repast and Swarm were included, being Java the programming language used by all tools. According to the survey the main difference is related to the integration of statistics by the Swarm and Repast tools; in opposite Mason doesn't provide this functionality. The second part of the survey evaluated the shareware/freeware tools, namely StarLogo (education.mit.edu/starlogo/), NetLogo and OBEUS tools. The StarLogo and NetLogo are very similar, being the last one easy to use and with good documentation support; on contrary, OBEUS misses a good technical documentation support. In the class of the proprietary tools, AgentSheets (www.agentsheets.com) and AnyLogic (www.xjtek.com) tools were evaluated, but this class is not considered in this analysis, since in this work, the focus is to use open source or shareware/freeware tools.

Another survey [17] compared some ABM tools to study their behaviour on complex adaptive systems and their capabilities against requirements for analysing self-organization, adaptation and networked causality. For this purpose a simple model was developed and the most rated ABMs until now received also the highest classifications. A special attention should also be given to the evaluation made by [18] where five ABM are evaluated: NetLogo, Repast, MASON and Swarm (Objective-C and Java versions). The authors developed some models to test some key points in the ABMs like execution speed, parameters display and behaviour as complexity grows.

A summary of some of the most important key points regarding the evaluation of the MASON, NetLogo, Swarm and Repast can be found at Table 1.

TABLE I – CHARACTERISTICS OF SOME ABM PLATFORMS

	MASON	NetLogo	Swarm	Repast
Availability (free)	✓	✓	✓	✓
Maturity	🙁	😐	🙂	😐
Programming effort	🙁	🙂	😐	🙁
Change of properties	🙁	😐	🙁	🙂
User interface	🙁	🙂	🙁	🙂
Simulation speed	🙂	😐	😐	🙂
Documentation	🙂	🙂	😐	😐

Legend: 🙂 *Good;* 😐 *Fair;* 🙁 *Poor*

As conclusion, there is no perfect platform to be used, being the choice of the correct ABM dependent of the task to be performed and the skills of the person who will make that task. In short, to starters and to a certain degree of complexity, the NetLogo platform is the right choice, due to the conjunction of ease of learning and power capabilities, combined with the good available documentation. When the complexity of the system grows up, requiring simulation speed, Repast is a good choice instead of NetLogo, losing however the user friendly aspect. Another conclusion from these surveys is that the users must also be aware of the constant change and evolution of these tools and that their exploration and comparison is a hard task.

B. *NetLogo Programming and Modelling Platform*

Since the NetLogo tool was chosen to be used in this work, due to its good relation between programming effort and simulation speed, this section briefly describes this tool.

The NetLogo application runs on a Java Virtual Machine, therefore it is able to run on major available platforms, like Windows, Linux, Mac and Solaris. However, its programming language is based on the Logo programming language [19] and not in Java, making it very easy to be used even by persons with low skills in programming.

NetLogo world is, basically, composed by two types of agents, the stationary agents (or patches) and the mobile agents (or turtles). The patches are arranged in a grid way, so they can form the world in over that the turtles move around. There is a third kind of agent that is the link agent, which connects turtles so they can form networks, graphs and aggregates. NetLogo is fully customizable, for example, the user can set the size of the patches and/or the world, and set the size, shape or colour of the turtles.

The NetLogo GUI is structured in a tab way and is composed by 3 tabs: Interface, Information and Procedures. The *Interface tab* is the graphical part of NetLogo, i.e. allow the user to insert buttons, create graphics and see the world behaviour. The *Information tab* can be used to retrieve and/or change some information about the objective, functioning or bugs that the model may have. This is useful for the users (that are not the designers/developers) as a starting point to know the expected behaviour of the model. The *Procedures tab* is the place where the code is built, i.e. creating the model with the desired characteristics and expected behaviour. Table II illustrates some simple commands used in NetLogo to perform actions over agents.

TABLE II - SIMPLE COMMANDS IN NETLOGO

Desired action	**Encoding**	**Comments**
Create an agent	crt 1	Creates 1 turtle
Move agent one patch upper way	set heading 0 fd 1	Faces agent in upper way Move agent 1 patch
Check if patch ahead is empty	if not any? product-on patch-ahead 1 []	Checks if on the next patch is any agent called product
Remove first item from an array (e.g. service-list)	set service-list remove-item 0 service-list	Removes the first (0) item from the array named "service-list"
Count the total number of pallet on the system	count product-on patches	Counts the products (i.e. pallets) that are in the system (i.e. patches)

The examples given in the previous table shows that the programming language of the NetLogo tool is very intuitive and therefore easy to understand.

IV. Experimental Implementation

Aiming to illustrate the applicability of ABM platforms, a washing machine production line will be used as case study to accommodate an agent-based control system that will be modelled and simulated in the NetLogo environment. The use of simulation in this work has supported the task of specification of an agent-based control system for the process control, adjusting the definition of the autonomous agents' behaviour and the interaction among them.

A. *Description of the Case Study*

The case study used in this work is a part of a washing machine production line, following a product-driven control approach. This simplified production line is composed by 11 machines that are linked together by conveyors, as illustrated in Fig. 1, including two particularities:

- The first one is centred on a workstation (WS) where a marriage operation occurs, consisting in joining two different components (i.e. Rear Tub and Drum) that arrive from two independent conveyors.
- The other is the existence of an operation that can be performed in one of two available and similar machines (i.e. tub welding machines).

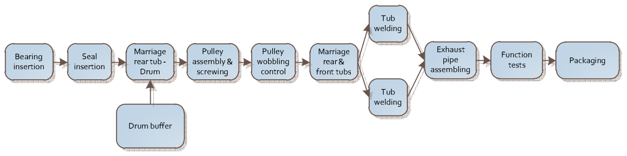

Fig. 1. Layout of the production line case study.

All other operations are single machine operations that are placed on a sequential order, each one having a processing time, according to the type of product to be processed. The production line also comprises a station (WS9, functional tests), where a quality control check is made to all produced products. This station is in charge to run a proper quality check program and the product is labelled with the inspection results for posterior analysis.

The products enter the line with a process plan that must be fulfilled. The process plan is set to the product taking into consideration the variables (e.g. type of the rear tub) and operation parameters (e.g. thickness of welding process) according to the type of washing machine to be manufactured.

B. *Implementation Details*

The agent-based model to control this production line was developed in NetLogo. The agent-based system is composed by 3 types of agents: Product Agents (PA), Quality Control Agents (QCA) and Resource Agents (RA). The Rear Tub and Drum are examples of PA agents, the machines and conveyors are examples of RA agents and WS9 is a QCA.

The behaviour of the PA agent is very simple. Basically the PA is created with a process plan containing the details and sequence of operations that must be fulfilled. During its life-cycle the PA agent will interact with the RA agents in order to guarantee the execution of the product according to the process plan. The results of the operations' execution are stored for posterior analysis and to support traceability. The behaviour of the PA agent can be summarized in the following pseudo-code:

```
Procedure PA
    while process plan is not completed do
        selects next operation
        asks conveyor to move pallet to next WS
        waits the pallet arrival to the WS
        notifies RA about the parameters to execute the operation
        waits the end of the operation execution
    end while
end
```

The RA agent presents a different behaviour from the PA agent. The RA waits to be requested to perform an operation, indicated with the arrival of a pallet to the workstation. When the processing starts, the RA agent changes its state to not available and executes the proper operation. When the processing is finished, the RA agent notifies the PA agent and is again available to execute a new operation (after the removal of the processed pallet). The pseudo-code describing this behaviour is the following:

```
Procedure RA
    while true do
        waits for PA requests
        executes operation
        notifies PA about results of the operation execution
    end while
end
```

The QCA agent represents the quality control station that executes a quality check. In this case, it simulates a quality control check in a random manner, notifying the results to the PA agent: OK if the quality check is according to the standards and KO if not. The behaviour of the QCA agent is very similar to the RA behaviour, being the difference related to the type of operations that they execute.

Aiming to analyse the behaviour of the system in different scenarios (i.e. changing the condition parameters), an interaction area is configured with some buttons and sliders, allowing the user to change the processing time of each machine, and to simulate a malfunction on the WS7 and WS11 (i.e. the tub welding machines). An area to visualize the results was also included, considering the representation of some performance parameters, namely the evolution of the Work In Process (WIP), the number of finished washing units (WU), the number of the finished WUs with defect and the average Manufacturing Lead Time (MLT).

Fig. 2 represents a general view of the implemented agent-based control system (left area reserved for the interaction with the user and results and the right side for the layout of the production line).

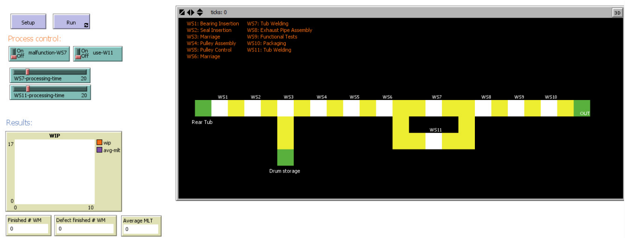

Fig. 2. Layout of the production line case study.

C. *Discussion of the Experimental Results*

The tests performed on the designed agent-based control model allowed to conclude about the system's behaviour in different scenarios.

In a first scenario, all the machines of the production line possess the same processing time, in this case 20 ticks, and no delays occur. This scenario is illustrated in Fig. 3.

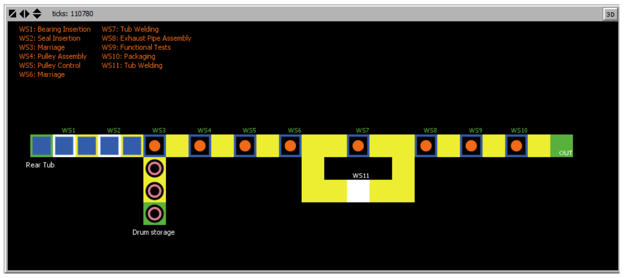

Fig. 3. Test scenario with all machines having the same processing time

As expected, in this test scenario the line is well balanced and there is no congestion in the system in terms of products to be processed. This scenario was useful to verify the correctness of the agent-based control system specification and also to detect some small misunderstanding and consequently make some adjustments in the control system.

A second scenario considers that the machine WS7 (one of the tub welding machines) gets slow in the operation execution due to some kind of problem, being the processing time associated to WS7 changed to 25 ticks. As expected the system behaviour will change. In fact, the resulted behaviour, illustrated in Fig. 4, shows that the increased processing time provokes an initially degradation of the balance of the line, being the pallets automatically re-routed to the alternative machine (WS11). It is also possible to verify that the control system adapts dynamically to the changing environment conditions.

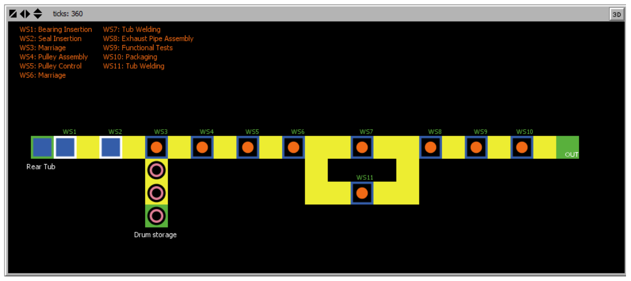

Fig. 4. Test scenario with the WS7 having an increased processing time

The simulation of the two previous scenarios allows the observation that the machine WS11 is not used when the line is balanced (in this case when all the processing times are tuned to 20 ticks), and in the second test scenario it is used to compensate the increased processing time of the WS7. Additionally, the MLT is increased since the processing time of one machine is now bigger than in the previous scenario.

Another interesting scenario is to consider a malfunction in the WS11 (one of the alternative tub welding machines), maintaining the same processing time of WS7. The resulted behaviour, illustrated in Fig. 5, shows that the pallets are now moving (re-routed) only to the machine WS7 to be processed.

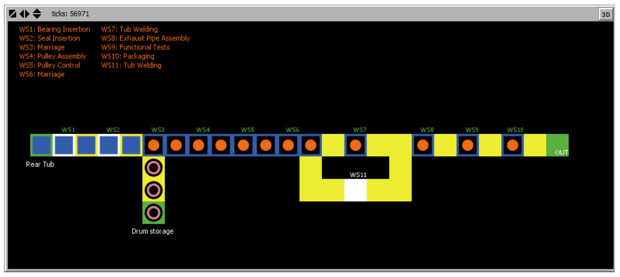

Fig. 5. Test scenario where WS11 is unavailable

In this case, and since the line is not well balanced and only one tub welding machine is available, a congestion in the upstream sequence of the production line appears, and consequently the MLT is significantly increased due to the time spent by the pallets stocked in the line. Also the WIP parameter is increased.

Fig. 6 summarizes the WIP (maximum value) and MLT parameters for the three scenarios simulated.

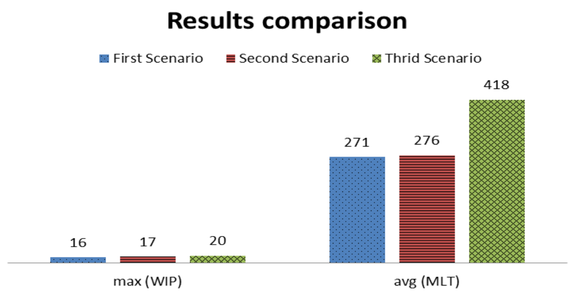

Fig. 6. Summary of the performance parameters for the experimental tests.

After the observation of the graphical results it can be stated that the number of products on the systems directly depends of the congestion of the line, i.e., if the line is not balanced, e.g. due to a slow performance of a WS, the WIP rises, creating a congestion on the process line. Other conclusion drawn from the graphical results is the time necessary to manufacture, in average, a product. These results show that if congestion occurs, the average MLT rises abruptly due to the fact that products must wait long time to be processed. Also regarding to the average MLT, it can be observed an increase of 5 ticks from the first to the second scenario. This is a consequence of the increase of the processing time of the WS7 for the same amount of ticks.

All these conclusions are easily extracted by the simulation provided by the use of an ABM tool, which can easily support the design and test of different if-then scenarios. The analysis of the behaviour of the agent-based control solution under different scenarios allowed to support the design of the agent-based solution, tuning and refining its specification.

V. CONCLUSIONS

The design, debug and testing of multi-agent systems, due to its distributed nature, assumes a crucial role, being its system behaviour not easily understood and usually time consuming. For this purpose the use of simulation environments are crucial for the development of agent-based solutions, and particularly the use of ABM tools that can be successfully used as an intermediary tool to help on the specification and debugging of multi-agent systems. These tools can help to reduce the time and effort due to their potential to help, in an easy way, on the specification of the agent-based control system.

This paper discusses the use of ABM tools in simulation of agent-based manufacturing systems and compares some of the most relevant available ABMs. Aiming to illustrate the applicability and benefits of using these tools in the simulation of agent-based systems, a production line case study for washing machines was considered. For this purpose, the NetLogo tool was used as a modelling and simulation platform to design the agent-based control system. This environment allowed the rapid solution prototyping that can be used as an auxiliary tool for the specification of a multi-agent system to be implemented on the washing machine production. Through the simulation of different scenarios, some conclusions about the system's behaviour can be extracted and used later to refine and tune some conceptual parameters in the specification of the agent-based manufacturing control system.

As a future work, the developed model will continue to be used to finalize the specification of the multi-agent control system for the production line. Also, and due to some NetLogo limitations and aiming to consider more complex problems and functionalities, the logical step is to move into a more powerful ABM, for example the RepastS tool.

ACKNOWLEDGMENT

This work has been financed (or partly financed) by the EU Commission, within the research contract GRACE coordinated by Univ. Politecnica delle Marche and having partners SINTEF, AEA srl, Instituto Politécnico de Bragança, Whirlpool Europe srl, Siemens AG.

REFERENCES

[1] M. Wooldridge, *"An Introduction to Multi-Agent Systems"*, John Wiley & Sons, 2002.

[2] V. Marik and D. McFarlane, "Industrial Adoption of Agent-Based Technologies", *IEEE Intelligent Systems*, 20 (1), 2005, pp. 27-35.

[3] H. V. D. Parunak, "Foundations of Distributed Artificial Intelligence", G. O'Hare and N. Jennings (eds), Applications of Distributed Artificial Intelligence in Industry, John Wiley & Sons, pp. 139-164, 1996.

[4] P. Leitão, J.M. Mendes, A. Bepperling, D. Cachapa and A.W. Colombo, "Engineering Tools for the Integration of Service-oriented Production Systems" Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'09), Moscow, Russia, 3-5 June, pp. 1763-1768, 2009.

[5] L.J. De Vin and M. Jagstam, "Why we Need to Offer a Modeling and Simulation Engineering Curriculum," Proceedings of the Winter Simulation Conference (WSC'01), 2001, vol. 1, pp.1599-1604.

[6] A. Uhrmacher and D. Weyns, *"Multi-Agent Systems: Simulation and Applications"*, CRC Press, Inc., Boca Raton, FL, USA. 2009

[7] P. Vrba and V. Marik, "Simulation in Agent-based Control Systems: MAST Case Study", Proceedings of the 16th IFAC World Congress, Prague, 2005.

[8] P. Vrba, V. Mařík, "Capabilities of Dynamic Reconfiguration of Multiagent-Based Industrial Control Systems", *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 40, n. 1, 2010, pp. 1-11.

[9] M. Bal, M. Hashemipour, "Applications of Virtual Reality in Design and Simulation of Holonic Manufacturing Systems: A Demonstration in Die-Casting Industry", Lecture Notes in Computer Science, vol. 4659, 2007, pp. 421- 432.

[10] Y. Sallez, T. Berger, D. Trentesaux, "A Stigmergic Approach for Dynamic Routing of Active Products in FMS", *Computers in Industry*, vol. 60, pp. 204–216, 2009.

[11] M.J. North, T.R. Howe, N.T. Collier, and J.R. Vos, "A Declarative Model Assembly Infrastructure for Verification and Validation", S. Takahashi, D.L. Sallach and J. Rouchier (eds.), Advancing Social Simulation: The First World Congress, Springer, Heidelberg, 2007.

[12] N. Minar, R. Burkhart, C. Langton, M. Askenazi, "The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations", Santa Fe Institute, Report No.: 96-06-042, June, 1996.

[13] U. Wilensky, "NetLogo", http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, 1999.

[14] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan, "MASON: a Multiagent Simulation Environment", *Simulation*, 81 2005, pp. 517-527.

[15] R. Tobias and C. Hofmann, "Evaluation of Free Java-libraries for Social-scientific Agent based Simulation", *Journal of Artificial Societies and Social Simulation*, 7, 2004.

[16] C.J.E. Castle and A. T. Crooks, "Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations", Technical Report 110, Centre for Advanced Spatial Analysis, University College London, UK, September 2006.

[17] M.J. Berryman, "Review of Software Platforms for Agent based Models", Technical report, Defence Science and Technology Organisation, Edinburgh, Australia, April 2008.

[18] S.F. Railsback, S.L. Lytinen, and S.K. Jackson, "Agent-based Simulation Platforms: Review and Development Recommendations", Simulation, vol. 82, n. 9, pp. 609-623, 2006.

[19] W. Feurzeig, S. Papert, M. Bloom, R. Grant and C. Solomon, "Programming-languages as a Conceptual Framework for Teaching Mathematics", SIGCUE Outlook, vol. 4, n. 2, pp. 13-17, 1970.