Agent-based Modeling and Simulation of a Small Scale Cyber-Physical System using NetLogo

José Barbosa*‡, Paulo Leitão*†
* Polytechnic Institute of Bragança, Campus Sta Apolónia, 5300-253 Bragança, Portugal, Email: {jbarbosa, pleitao}@ipb.pt
† LIACC – Artificial Intelligence and Computer Science Laboratory, Rua Campo Alegre 1021, 4169-007 Porto, Portugal
‡ INESC-TEC, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

***Abstract*—The Cyber-Physical System (CPS) paradigm promotes the decentralization and distribution of the logic control as well as the integration of cyber and physical counterparts. In parallel, self-organization allows the dynamic and automatic system re-configuration responding to condition and environment changes. Modeling and simulation assume a crucial importance in the design of such complex, distributed, and self-organized systems, in the way that the detected and debugged errors may be corrected before the deployment into the real system, as well different strategies can be tested and evaluated. Agent-based modeling tools are computational frameworks able to analyze, experiment and compare systems populated by cooperative agents, supporting the fast prototyping of agent-based solutions exhibiting self-* properties. In this paper, the NetLogo tool was used to model and simulate the agent-based control layer of a small scale CPS, which control uses self-organization principles.**

I. Introduction

In the advent of the generalization of communication infrastructures and inexpensive processing power, many opportunities are appearing for the deployment of intelligent, adaptive and flexible (complex) systems. Cyber-Physical Systems (CPS) are among these new systems that are benefiting from the technology generalization and increase of the system intelligence, promoting the achievement of better system efficiency and agility. CPS advocates the integration of computational applications with physical devices, being designed as a network of interacting cyber and physical elements [1]–[3], which capture, monitor, analyze and adapt the process by continuously and dynamically self-determining the optimized operating parameters.

This implies the growing of the system complexity, where more machines are connected (using IoT technologies), and the generated data is collected and processed (using Big data and advanced data analytics) to support production improvements, e.g. tracing better the production and early detect the occurrence of errors, deviations and patterns. In order to keep up with this increase of the system complexity, the control system layer is becoming distributed, bringing the decisional processing capability into the individual entities that compose the system. This distribution of the control system brings contradictory features: at the local side, simplifying the development of individual control entities, but at the global side increasing its complexity. This system complexity, which is reaching unprecedented levels, requires the use of new methodologies and approaches for a proper design and simulation of these systems before the commission phase [1].

Multi-Agent Systems (MAS) derive from the distributed artificial intelligence field and are amongst the most promising technologies to promote distribution, decentralization, intelligence, autonomy and adaptation, contributing to achieve flexibility, robustness, responsiveness and reconfigurability [4]. In fact, MAS are based on a network of distributed, modular, intelligent and autonomous computational entities, known as agents, which represent physical or logical objects in the system. The overall system behavior is achieved by the interaction among distributed agents (note that each agent is autonomous and is able to interact with other agents when it doesn't possess knowledge and skills to reach alone its objectives). Thus, MAS offers a suitable approach to design CPS systems by decentralizing the control system by distributed, autonomous and cooperative entities, differing from the conventional approaches due to its inherent capabilities to adapt to emergence without external intervention.

Despite this, agent-based systems can be also used to support the modeling and simulation of CPS, enabling the detection and correction of errors and misunderstandings during the design phase and before its deployment into the real operation, as well as the evaluation of the use of different control strategies. In fact, the system designer can use the agent-based simulation as a rapid prototyping and proof-of-concept, where errors are identified and corrected, and the system configuration and/or control logic are modified, at an early stage, to achieve the desired system performance. Agent Based Modeling (ABM) tools use agent-oriented programming environments as the way to model and simulate systems displaying complex and emergent behaviors [5]. Several frameworks are currently available, e.g., NetLogo, Repast and AnyLogic, each one presenting different features [6], [7].

In this paper, the NetLogo ABM tool [8] is used to design, model and simulate a small scale CPS located at Polytechnic Institute of Bragança. The developed model is used to assess the system performance and detect its operation boundaries, particularly focusing on the range for the frequency of the different products arrival rate into the production system and the capacity of the intermediate buffers.

The rest of the paper is organized as follows: Section II briefly depicts the ABM concept and describes some relevant features provided by existing tools. Section III introduces the

978-1-5090-6161-7/17/$31.00

small scale production system used as experimental testing scenario and Section IV describes how the agent-based system was modeled using NetLogo. Section V discuses the results from the simulation of the agent-based model, and finally, Section VI rounds-up the paper with the conclusions.

II. Agent Based Modeling and Simulation

System simulation can be defined as the imitation of the operation of a real-world process or system over time [9], using a model that represents the characteristics and functionalities of such process or system. Basically, during the simulation process, the model is exercised by manipulating the input parameters, and is analyzed how they affect the output performance indicators. Simulation is being used to study, evaluate, test and optimize complex systems from many domains, such as economical and financial systems, production systems, public institutions, video games, layouts and stocks systems, and communication protocols.

The use of simulation brings several benefits for the development of complex systems, namely [5]:

- The system can be analyzed, tested and validated without the use the real equipment.
- The reproduction of abnormal or dangerous scenarios can be done easily and safely.
- The simulation can be repeated as many times as necessary to achieve the correct understanding of the system.
- The simulation process can be compressed to obtain faster the results.

In manufacturing domain, the simulation can be used to evaluate the effect of capital investment in equipment and physical facilities, to quantify the system performance, to predict the performance of an existing or planned system, and to compare alternative solutions for a particular design problem [10]. For this purpose, traditionally, simulation uses computational software tools, such as Delmia from Dassault Systems and Arena from Rockwell Automation, which considers discrete event modeling to depict the behavior of a complex system as a series of well-defined and ordered events.

ABM constitutes an alternative way to design, model and simulate complex and emergent systems, such as those following the CPS paradigm, in the sense that they use an agent-oriented programming approach to implement non-linear relations among individual and distributed cooperative nodes. ABM tools are computational frameworks to analyze, experiment and compare systems populated by cooperative agents, reproducing a variety of patterns observed in the real system, and particularly supporting complex phenomena, such as evolution. Note that the agent-based simulation is different from MAS frameworks, since [11]:

- MAS frameworks allow developing agent-based systems, but they don't provide a simulation infrastructure since, e.g., they miss a scheduler and the notion of a "clock".
- ABM tools provide an agent-based infra-structure to support the system's simulation but they aren't adequate for developing agent-based systems since, e.g., they are not compliant with FIPA (Foundation for Intelligent Physical Agents) specifications.

Typically, an ABM is composed by 3 types of components: i) the environment representing the world, ii) the mobile agents that represent the entities that can move and take decisions on the world, and iii) the relation among agents, representing the existing connections among the world entities.

A set of ABM tools is currently available, namely Repast [12], Swarm [13], NetLogo [8] and Mason [14], varying in the offered functionalities, graphical capabilities and programming languages. Several surveys, e.g. [6], [15], [16], compare them according to the provided features. The main conclusion extracted from these surveys is that there is no perfect platform, being the choice dependent of the task to be performed and the skills of the user [5]. In short, for beginners and to a certain degree of complexity, the NetLogo tool is the right choice, due to the ease of the learning process, combined with the good available documentation. When the complexity of the system grows up, requiring higher simulation speed, other tools are a better choice, as example Repast. Additionally, the constant change and evolution of these tools must be taken into consideration, e.g., the RepastHPC tool that use the available resources in High Performance Computing (HPC) platforms.

ABM tools can be applied to different domains, ranging from economics to production systems passing by electric power systems. In particular, several examples can be shown of using ABM tools to model and simulate systems related to manufacturing processes. A flexible manufacturing system, composed by independent conveyors, disposed in two levels and connected by two lifters, was modeled and simulated by using NetLogo [7]. In this work, a particular attention was devoted to the study of the maximum pallet arrival rate to the system, without overflowing it, determining in this way the maximum achievable operational throughput. A special attention was also devoted to the study of routing techniques to be deployed, namely the feasibility to deploy a stigmergic technique, based on the ant's food foraging. A part of a washing machine production line was modeled and simulated using also NetLogo [5], focusing on the particular step of the production process where two independent parts get assembled together (operation named as "marriage") and on the analysis of the necessity of having a redundant tub welding duplication. In another application, Repast was used to simulate the system dynamics in a supply chain management in manufacturing, using agents to model the players along the supply chain, namely suppliers, manufacturers [17].

AnyLogic is also taking significant attention, providing good programming functionalities, documentation and support for high quality graphical libraries, which enables the creation of visually rich environments. Different strategies to handle the ramp-up of small-lot and complex products, e.g., airplanes, were simulated by using AnyLogic [18]. The developed simulation model includes a part of the Hamburg Airbus A350 assembly line, where two major components are assembled. The aim of this work was to study the management

and mitigation strategies to properly respond to disturbance situations, such as missing material or non-conformities.

III. Small-size Flexible Production System

The experimental case study considered in this work is a real small-scale production system composed by one IRB 1400 ABB robot, two punching machines and two indexed lines, as illustrated in Figure 1, the last ones supplied by Fischertechnik™.

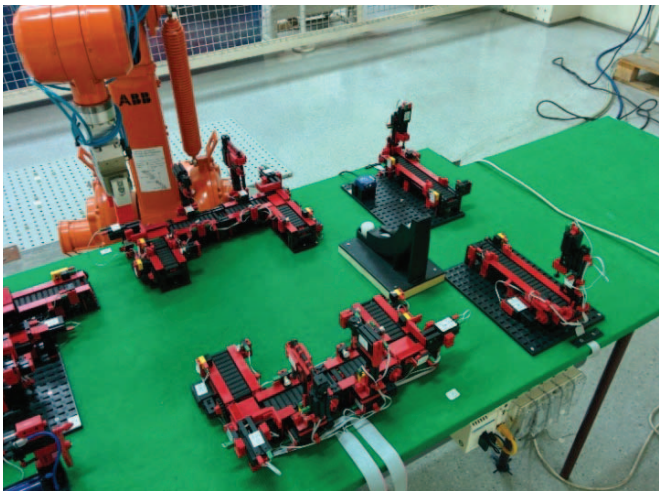

Figure 1. Layout of the small-scale flexible production system.

Technologically, the punching machines are composed by one motor, two light sensors to detect the parts in the beginning of the conveyor and in the punching position, and two switch sensors to detect the end of the movement of the punching device. The indexed lines are composed of two workstations interconnected by several conveyors, allowing to process four parts simultaneously. Four switch sensors are used to determine the range of movement of the embolus and five light sensors are used to detect the presence of the parts in the conveyors and in the processing positions. The low-level logic control of the Fischertechnik™ devices is running in a Modicon M340 PLC through proper IEC 61131-3 programs.

The manipulator robot executes the transfer of parts between the machines using proper RAPID programs. Finally, a human operator performs visual inspection operations to verify the compliance of the parts according to the specifications. The human operator interacts the system through the Omron NS8-TB10-V1 Human-Machine Interface (HMI) device, connected to the system using the Omron C200HG PLC. Table I summarizes the available skills in each resource, depicting alongside the respective processing times.

Two different types of parts can circulate in the system, each one having a particular process plan, as illustrated in Table II, which represents the sequence of operations to be performed. The circulation of parts within the production system is tracked by a radio-frequency identification (RFiD) reader, allowing to uniquely identify each part.

Table I
Representation of the resources' skills.

Resource	**{Skill, time}**
Manipulator robot	{transfer, 3}
Punching machine A	{punch_1, 5}
Punching machine B	{punch_1, 7}
Indexed line A	{drill_1, 7}, {drill_2, 6}
Indexed line B	{drill_1, 5}, {drill_2, 9}
RFiD reader	{read, 2}
Inspector	{inspection, 3}

Table II
Representation of the process plans for the products catalog.

Sequence	**Part "A"**	**Part "B"**
#1	read	read
#2	punch_1	drill_1
#3	drill_1	drill_2
#4	drill_2	punch_1
#5	inspection	inspection

The objective of this work is to develop an agent-based model to control the production system, which will be simulated under different scenarios to study the system behavior. Furthermore, the model can be used as a testbed for testing and validating different control architectures, operating under disruptive scenarios, such as the introduction of new products and the occurrence of machine failures. For this purpose, the NetLogo tool is selected to be used as modeling and simulation environment due to its good balance between programming effort and simulation speed.

IV. Development of the Agent-based Model

This section describes the agent-based model developed in the NetLogo framework for the simulation and assessment of the aforementioned small-size flexible production system.

The agent-based system is composed by 2 types of agents (that are implemented by using turtles, similar to the mobile agents previously described):

- Product Agents (PA), which are responsible to manage the production process of the catalog of products in the assembly system by using the available resources (following a specific process plan that contains the details and sequence of operations that must be executed).
- Resource Agents (RA), which are responsible to manage the execution of processing tasks by the resources disposed in the assembly system, e.g., punching machines, indexed lines and robot.

The PA's behavior, summarized in the Algorithm 1, is simple and comprises the necessary actions to guarantee the proper execution of each operation according to the process plan. For this purpose, after determining the next operation to be executed, extracted from the process plan, the PA interacts with RAs to negotiate its assignment taking into consideration the processing time, processing cost and availability of resources.

The RA presents a behavior, represented in Algorithm 2, which is cyclically waiting for requests to perform an operation, that is assigned by the PA after a negotiation

Algorithm 1 PA Behavior

```
initialization
i ← 0
j ← size of processPlan
while i < j do
    assign operation(i) to an available resource agent
    ask robot agent to transfer part
    request resource agent to start execution
    while in execution do
    store achieved results in csv file
    i++
```

procedure. When the part arrives, the RA changes its state to not available and the associated resource executes the proper operation. When the processing is finished, the RA notifies the PA and after the removal of the processed part, it becomes again available to execute a new operation.

Algorithm 2 RA Behaviour

```
initialization
while true do
    if PA processing request received then
        execute operation
        while in execution do
        notify PA about operation results
        store achieved results in csv file
```

One of the key features of using an ABM tool to model a manufacturing process is the capability to release the designer from hard or difficult programming tasks. As an example, the process plan of part A (see table II) is simply described by assigning a variable with the following array:

```
set processing-sequence [4 1 2 3 5]
```

where each number in the sequence represents the required operation to be performed that matches with the resources' skills (e.g. *read* is denoted as operation number 4 and *punch_1* is the operation number 1). The re-configuration of the process plan can be easily done by re-arranging this variable. In the same way, the machines' skills and processing times are fixed by setting the *skill* and *processing-time* variables. As example, the skills and processing times for the indexed line B machine are defined as follows (see table I):

```
set skill [2 3]
set processing-time [5 9]
```

The screenshot of the agent-based model developed in NetLogo is depicted in Figure 2. At the center, a representation of the real system is depicted, mapping the available machines and their relative localization. The current state of each machine is provided, according to the list {free, waiting, processing, broken, buffer}, and from the product perspective, the ID of each product in execution is also displayed. In this way, the user has, in real time, the possibility to see the current status of the system.

The analysis of the system behavior is possible by running the system according to different configurations. For this purpose, an interaction area (displayed on the left side) is configured with some buttons and sliders, allowing the user to change some system parameters, namely the processing times and failure rates of the different resources, the part arrival interval rate and the number of parts to be produced. Furthermore, the usage of a buffer and its capacity can also be defined by using an input text box.

The output results are displayed by using text boxes (on the right side) and graphics (on the bottom side), allowing the visualization of the system operative key performance indicators (KPI). These results are also stored in a *cvs* file for posterior analysis, and also to support traceability.

V. Simulation of the Agent-based Model

At this stage, the agent-based model is ready to be simulated. In this work, the analysis of the system behavior considers three major KPIs, namely the makespan, the system operability (i.e. the product arrival rate that do not create a system deadlock situation), and the resource occupancy.

Some testing scenarios were designed to assess the described small scale system, exploiting the impact of changing the product arrival rate, the use of buffer (and its capacity) and the introduction of disturbances in the resources. In the experiments, the catalog of orders included the production of 500 products of type A and 500 of type B.

Fig. 3 illustrates the occupancy of the several resources disposed along the production system considering an arrival time of 20 ticks (simulation time unit) without the use of the intermediary buffer and in a non-disturbance scenario.

A quick analysis in the graphic concludes that the critical resource is the robotic manipulator, which presents the highest occupancy rate. In fact, the manipulator has a final occupancy rate of 74.03%. The indexed machines exhibits lower but quite similar occupancy levels (32.76% for the indexed A and 33.85% for the indexed B). The punching machines differ in the final occupancy rate due to the different processing times, namely the punching A machine, with a processing time of 5 ticks, has a resource occupancy of 12.08%, while the punching B machine, with a processing time of 7 ticks, has an overall occupancy of 17.62%.

Analyzing these occupancy results, it can be concluded that the system still has some operational slack due to the relatively low occupancy levels, particularly by the robotic manipulator. This means that a lower arrival rate of the products can still be managed, ensuring in this way a higher throughput. Aiming to determine the upper and lower bounds of the system operation, a set of experiments was performed by varying the arrival time of the products and by changing the buffer capacity, whose results are summarized in Fig. 4.

This set of experiments allowed to verify that decreasing the product arrival rate for 18 ticks, naturally increases the resources overall occupancy, e.g., the punching B has 18.62%, the indexed A has 36.75% and the robotic manipulator occupancy time is now of 81.93%, while the makespan decreases

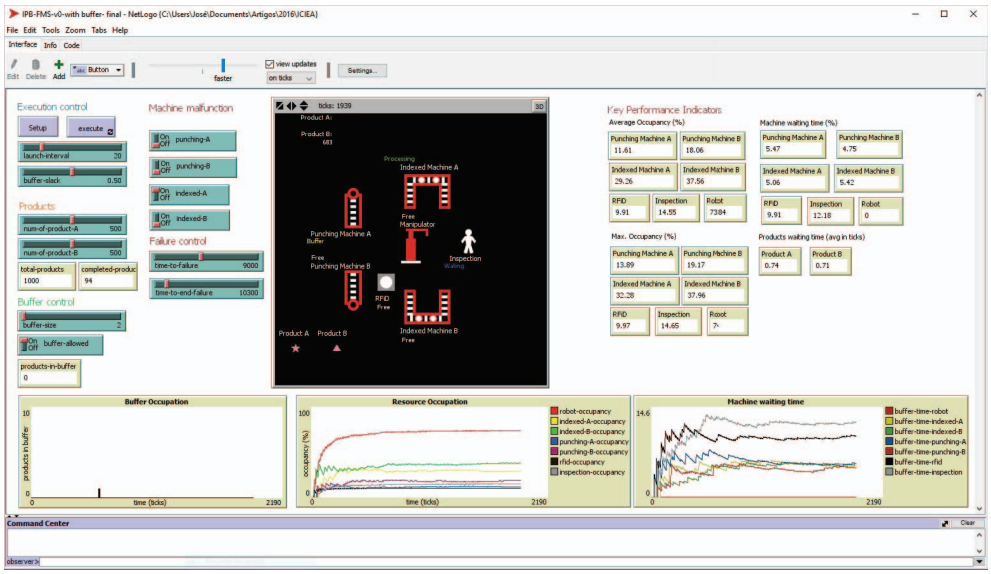

Figure 2. Screenshot of the agent-based model in the NetLogo framework.

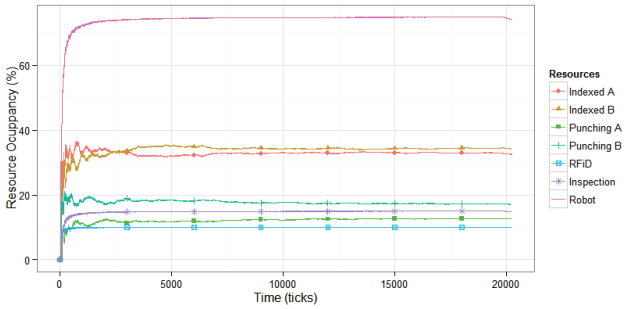

Figure 3. Evolution of the resource occupancy according to the time arrival.

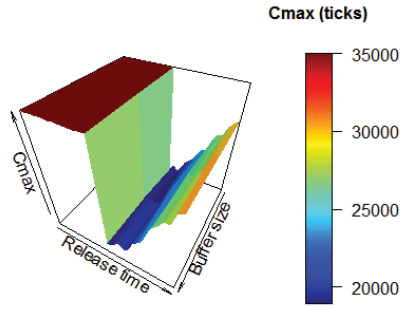

Figure 4. Evolution of makespan according to time arrival and buffer capacity.

by 4.92% to 19.250 ticks, in contrast to the 20.247 ticks achieved in the initial 20 ticks arrival scenario.

The system operational lower bound, i.e. the minimum product arrival rate for a non-disturbance scenario that ensures that the system doesn't fall into a non-operational status is 14 ticks. The non-operational situation occurs when the robotic manipulator has an occupancy higher than around 96% (empiric value by conducting several simulations).

It is also possible to observe that the combination that better exploits the system's operation is the one without intermediate buffer and with a product release time of 15 ticks. In fact, the use of a buffer, with small capacity, has minor effect on this arrival interval rate, being only considerable for high capacity. Nevertheless, the use of a buffer can have advantages in situations of variable production rates and where resources may have processing delays.

Some simulation tests considering a scenario with resources' malfunction were also conducted to determine how the system would behave under such conditions. Namely, a malfunction is introduced, for every scenario, in the indexed A machine between the ticks 9000 and 10300. Fig. 5 depicts the resource occupancy for such scenario for a launching interval of products of 20 ticks. As it can be observed, when the indexed A machine becomes unavailable, its occupancy decreases while the occupancy of indexed B machine increases to compensate this unavailability. It can also be observed that at the same time, the robotic manipulator also suffers a temporary occupancy decrease.

Naturally, after the recovery of the indexed A machine, all the resources tend to follow its nominal occupancy (under the tested conditions).

Fig. 6 summarizes the achieved results for the upper and lower bounds of the system operation considering the machine breakdown scenario. An interesting conclusion is the achievement of the same values for the minimum vale of the product arrival rate (14 ticks), and the optimal value that maximizes the makespan (15 ticks), which maybe justified by the small downtime of the indexed A machine.

Summarizing, from the simulation of the agent-based model

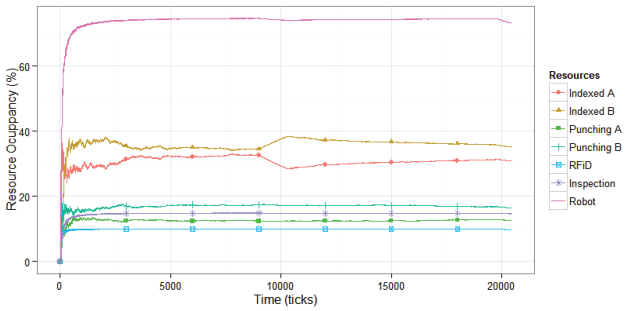

Figure 5. Evolution of the resource occupancy according to the time arrival for a machine breakdown scenario.

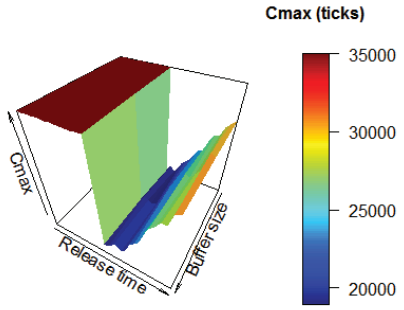

Figure 6. Evolution of the makespan according to the time arrival and buffer capacity for a machine breakdown scenario.

it was possible to observe that, as expected, since the system is robotic centric, the robotic manipulator has the highest occupancy ratio, raising to approximately 94% under controlled situations. The indexed lines occupancy are evenly distributed while one punching machine is, tendentiously, more occupied that the other. Additionally, when a buffer is introduced, the robot occupancy level increases to nearly 99% without introducing a major performance benefit.

VI. Conclusions

Generally, today's systems are growing in complexity. Systems in the manufacturing domain are no exception, where machines, products, processes, clients or, basically everything, is being connected and have influence in each other. This complexity growth imposes the use of new design, modeling and simulation tools that capture this phenomena, allowing to foresee beforehand possible bottlenecks.

This paper uses an agent-based modeling tool as the way to model and simulate a small-scale production system. The developed model allowed to make a system initial analysis, assessing the product arrival rate and the resources workload.

After an intensive simulation, the user was able to proceed with the necessary system adjustments. Since the robot is the major system operation bottleneck, some system variations could be tested, e.g., introducing another robot, increasing the buffer capacity or changing the system control. The operation boundaries that ensure that the system is working under control and without deadlocks, can also be easily detected.

In the future, this model will be used to perform feasibility tests in self-organized manufacturing control architectures. This will enable a first tune of the control architectures, refining and assessing the entities interactions and self-organizing mechanisms before developing and deploying them.

References

[1] E. A. Lee, "Cyber Physical Systems: Design Challenges," in *Proceedings of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'08)*, 2008, pp. 363–369.

[2] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial Automation based on Cyber-Physical Systems Technologies: Prototype Implementations and Challenges," *Computers in Industry*, vol. 81, pp. 11–25, Sep. 2016.

[3] ACATECH, "Cyber-Physical Systems: Driving force for Innovation in Mobility, Health, Energy and Production," ACATECH – German National Academy of Science and Engineering, Tech. Rep., Dec. 2011.

[4] P. Leitão, "Agent-based Distributed Manufacturing Control: A State-of-the-art Survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, 2009.

[5] J. Barbosa and P. Leitão, "Simulation of Multi-agent Manufacturing Systems using Agent-Based Modelling Platforms," in *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN'11)*, jul 2011, pp. 477–482.

[6] M. J. Berryman, "Review of Software Platforms for Agent Based Models," Defence Science and Technology Organisation, Edinburgh, Australia, Tech. Rep., 2008.

[7] J. Barbosa and P. Leitão, "Modelling and Simulating Self-Organizing Agent-based Manufacturing Systems," in *Proceedings of the 36th Annual Conference of the IEEE Industrial Electronics Society (IECON'10)*, 2010, pp. 2696–2701.

[8] U. Wilensky and B. Rand, *Introduction to Agent-Based Modeling: Modeling Natural, Social and Engineered Complex Systems with NetLogo*. The MIT Press, 2015.

[9] J. Banks, "Introduction to Simulation," in *Proceedings of the 1999 Winter Simulation Conference*, 1999, pp. 7–13.

[10] O. Benedettini and B. Tjahjono, "Towards an Improved Tool to Facilitate Simulation Modeling of Complex Manufacturing Systems," *International Journal of Advanced Manufacturing Technology*, vol. 43, no. 1/2, pp. 191–199, 2008.

[11] P. Leitão, U. Inden, and C.-P. Rückemann, "Parallelising Multi-agent Systems for High Performance Computing," in *Proceedings of the Third International Conference on Advanced Communications and Computation (INFOCOMP'13)*, 2013, pp. 1–6.

[12] M. J. North, T. R. Howe, N. T. Collier, and J. R. Vos, "A Declarative Model Assembly Infrastructure for Verification and Validation," *Advancing Social Simulation The First World Congress*, no. 1, pp. 129–140, 2007.

[13] N. Minar, R. Burkhart, C. Langton, and M. Askenazi, "The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations," 1996.

[14] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "MASON: A Multiagent Simulation Environment," *Simulation*, vol. 81, no. 7, pp. 517–527, 2005.

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

[16] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, "Agent-based Simulation Platforms: Review and Development Recommendations," *Simulation*, vol. 82, no. 9, pp. 609–623, 2006.

[17] A. Größler, M. Stotz, and N. Schieritz, "A Software Interface Between System Dynamics and Agent-Based Simulations – Linking Vensim® and RePast®," in *Proceedings of the 21st International Conference on System Dynamics Soc.*, 2003.

[18] A. Schirrmann, "Analysis of Control Strategies for A/C Production Ramp-Up," in *AnyLogic Conference 2014, San Francisco, CA, November 12 – 13*, 2014.