Benchmarking Simulation Software Capabilities Against Distributed Control Requirements: FlexSim Vs AnyLogic

Ali Attajer, Saber Darmoul, Sondès Chaabane, Fouad Riane, Yves Sallez

Benchmarking Simulation Software Capabilities Against Distributed Control Requirements: FlexSim Vs AnyLogic

Ali Attajer, Saber Darmoul, Sondès Chaabane, Yves Sallez, Fouad Riane

Benchmarking Simulation Software Capabilities Against Distributed Control Requirements: FlexSim Vs AnyLogic

Ali Attajer[1,2,3], Saber Darmoul[1], Sondes Chaabane[2], Fouad Riane[1,3], and Yves Sallez[2]

[1] Ecole Centrale Casablanca, Bouskoura Ville Verte, 27182, Casablanca, Morocco
{ali.attajer, saber.darmoul, fouad.riane}@centrale-casablanca.ma
[2] LAMIH, UMR CNRS 8201, Université Polytechnique des Hauts-de-France, UPHF, Le Mont Houy, 59313 Valenciennes, France
{sondes.chaabane, yves.sallez}@uphf.fr
[3] LIMII, Hassan First University, Settat, Morocco

Abstract. Industry 4.0 communication and data management technologies enable the development of distributed, product driven control architectures, where intelligent products can play active roles in manufacturing control processes. Although simulation is a widespread practice to test, evaluate, compare and validate different design alternatives, there is still a lack of papers that assess and discuss the capabilities of available simulation software to meet and implement the requirements of such distribution as a design alternative. This paper provides an analysis of distributed, product driven control requirements and benchmarks them against the capabilities of two commercially available simulation software, namely FlexSim and AnyLogic. A comparison of the strengths and weaknesses of each software is provided through a case study.

Keywords: simulation software, distributed control, intelligent product, simulation benchmark.

1 Introduction

The advent of the industry 4.0 paradigm introduces a set of information and communication technologies that allow both information processing to be distributed, and decision-making to be decentralized over several autonomous and intelligent production entities, including smart manufacturing assets (machines, robots, material handling devices, etc.), augmented operators and intelligent products [1]. This distribution/decentralization particularly encourages the design and development of distributed, product driven control architectures, where intelligent products can play more active roles in operational and decision-making processes [1].

As manufacturers are often reluctant to experiment with new control architectures directly on their production systems [2], mainly due to risk aversion considerations (loss of production capacity, functionality, quality, performance, etc.), they prefer first assessing the control architecture using simulation before implementing it on real scale. Indeed, simulation is a widespread practice that offers a methodology and a set of tools

[3] to test, evaluate, compare and validate different design alternatives at lower costs and almost without risk. However, as it will be discussed in section 2, there is still a lack of papers that provide guidelines to benchmark the capabilities of available simulation software against the requirements of distributed product driven control in order to select the simulation software that offers the set of capabilities that better meet those requirements. The aim of this paper is then to provide such guidelines based on the benchmarking of two representatives of available simulation software: FlexSim and AnyLogic. FlexSim (https://www.flexsim.com/) is considered due to its high ranking [4], and as a representative of discrete event simulation software. AnyLogic (https://www.anylogic.com/) is considered as a representative of multi-agent based modeling and simulation software [5].

The remaining of the paper is organized as follows: section 2 reviews the related works with respect to the use of simulation in distributed manufacturing control. Then, section 3 provides an analysis of distributed control requirements, which are illustrated on a case study (cf. section 4) and further matched against the capabilities of FlexSim (cf. section 5.1) and AnyLogic (cf. section 5.2). Finally, a conclusion provides a comparison of the strengths and weaknesses of each software, and some future works are presented.

2 Simulation in distributed control of manufacturing systems

In product driven control architectures, intelligent products [6] can play active roles in manufacturing control [1], [7], [8]. Although simulation is widely used to assess the behavior and performance of such architectures [2], [3], [9], most often, authors do not argument the selection criteria of the simulation software they use. Usually, authors do discuss the strengths and weaknesses of their control architectures. However, they do not discuss the capabilities, ease of use, strengths and weaknesses of the simulation software they used. They do not report on their user experience with the simulation software to implement those architectures. These are some of the key observations that motivate this paper. As a matter of fact, if no guidelines are available to help select a simulation software before implementing a product driven control architecture, then misleading choices could be made, or additional efforts (e.g. programming) could be spent using a software that does not provide the necessary or satisfactory capabilities that implement the sought after requirements.

Indeed, the Simulation Software Survey [10] is a useful source of information to summarize the main characteristics of a variety of simulation software packages available in the market. Some references focused on ranking simulation software [4], comparing their capabilities [11] and providing frameworks for simulation software selection [12]. However, all of this is made independently of the distribution/decentralization decisions and independently of any targeted control architecture. Several studies described requirements to develop benchmarking testbeds [13], particularly using simulation [14], to assess the behavior and performance of distributed control architectures. In [15], a recent review of the benchmarking initiatives aimed at Holonic Manufacturing Systems performance assessment shows that very few of them exist. None of the

above-mentioned references provides an analysis of the capabilities of available simulation software to meet requirements of distributed product driven control. Therefore, this paper is an effort to fill in this gap. The paper compares the strengths and weaknesses of two available simulation software with respect to the implementation of distributed, product driven control. FlexSim is considered due to its high ranking, and as a representative of discrete event simulation software [4]. AnyLogic is considered as a representative of multi-agent based modeling and simulation software [5]. Both software provide trial and evaluation versions available on-line that could be used to implement the case study considered in section 4.

3 Requirements for simulation in distributed control

Distribution decisions are design decisions, which behavior and performance can be evaluated using simulation. However, for a successful simulation and evaluation of product driven control architectures, the simulation software (i.e. simulators) have to satisfy several types of requirements, as shown in **Fig. 1**.

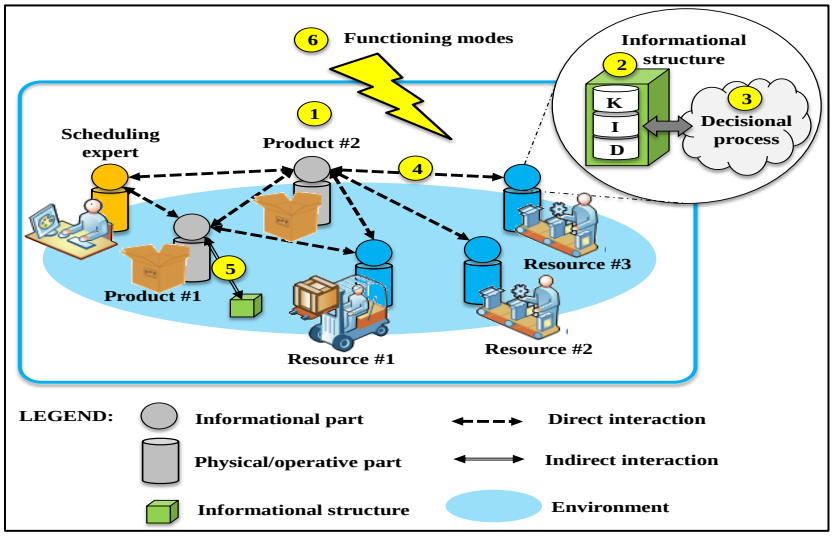

Fig. 1. Illustration of the requirements for simulation.

1. **Production entities** (item numbered (1) in **Fig. 1):** simulators shall be able to consider several features of several types of production entities:
   (a) To achieve product intelligence, simulators should enable products to be aware of their (design, operational and customer order) specifications, context and set of required services to be manufactured.
   (b) The resources that offer different services to the products (e.g. transformation services for machines, maintenance and quality control as support services, transport and storage services for material handling systems);

(c) Decision entities, human and/or artificial, to synchronize, coordinate and perform analysis and decision-making processes.

2. **Informational structures** (item numbered (2) in **Fig. 1):** simulators shall enable considering entity attributes and properties, related to product and process specifications (e.g. bills of materials, routings, machining and process parameter settings, tolerances, services required to obtain a given product, etc.), as well as indicators and descriptors of the evolution of manufacturing processes (e.g. key performance indicators, statuses and reports describing normal, tolerable, satisfactory and/or abnormal operating conditions). To achieve product driven control, simulators shall enable intelligent products to handle informational structures that are compliant with a DIK model [16]:
   (a) ***D (for Data)*** represents the properties and statues of production entities and processes, as well as the events generated by, or occurring to, production entities in interaction with each other and within their environment. Data can be considered as raw facts, without meaning, issued from measurements (e.g. data acquisition from sensors, such as velocity, temperature, pressure, etc.);
   (b) ***I (for Information)*** obtained by some data processing to add meaning to raw data, for example to have answers to questions, such as "what" event happened, "when" and "where" did it happen, "who" or "what" generated it, "how" is it described and eventually "who" is in charge of dealing with it;
   (c) ***K (for Knowledge)*** represents expertise and can be seen as groups of information that are linked by semantic relations.
3. **Interactions:** To achieve product driven control, simulators shall enable intelligent products to support different types of interactions with production entities:
   (a) Direct interaction (item numbered (4) in **Fig. 1**), for example using direct communication channels and exchanges of messages.
   (b) Indirect interaction (item numbered (5) in **Fig. 1**), using the environment and communication channels, such as blackboards or stigmergy [17].
   (c) More complex interactions, such as negotiation protocols, should be enabled.
4. **Decision-making** (item numbered (3) in **Fig. 1):** To achieve product driven control, simulators shall enable intelligent products to integrate different decision-making processes, which are the set of activities, coordinated and synchronized within business processes, that lead to the satisfaction of production objectives as well as performance, behavior and quality of service constraints and requirements.
   (a) Depending on distribution design choices, the decision-making can be supported either by products, or resources, or decision-makers, or else by any combination of these entities.
   (b) Decision-making processes use informational inputs to generate decisions and informational outputs that will be stored in informational structures.
5. **Functioning modes** (item numbered (6) in **Fig. 1):** To achieve product driven control, simulators shall enable intelligent products to represent and be aware of all operational settings of the manufacturing system, in terms of both normal, degraded and disturbed operational conditions.

It is worth noting that production entities and informational structures are common to many production systems and thus easily handled by simulators. However, decision-making processes, interactions and functioning modes are rather business dependent, specific to each production system, and particularly related to the distribution design choices and mechanisms of the suggested control architectures. The implementation of these requirements will challenge the capabilities of simulation software, in terms of ease of use, ease of configuration, ease of custom code programming and existence of pre-built libraries. The case study of section 4 is built to evaluate the above requirements through different interactions between entities.

4 Case study

Let us consider an automated manufacturing system, as the example one shown in **Fig. 2**. The system is composed of a main unidirectional conveyor loop (shown in blue color in **Fig. 2**) servicing production resources located aside secondary loops (shown in black color in **Fig. 2**), and two scrapping areas (shown in orange color in **Fig. 2**). The production resources include a raw material (RM) automated storage and retrieval system (AS/RS), two equivalent machines M1 and M2 and an AS/RS to store work in progress (WIP) and finished products. As the main purpose of the paper is not focused on complex product design and manufacturing, product routings with only one operation (to be performed interchangeability either on machine M1 or machine M2) are considered. Machines are subject to failures, and products are subject to quality defects. The scrap areas receive WIP products if they do not meet quality requirements. Decision and quality control points are located on the main conveyor loop as milestones so that intelligent products check updates about indicators and make decisions.

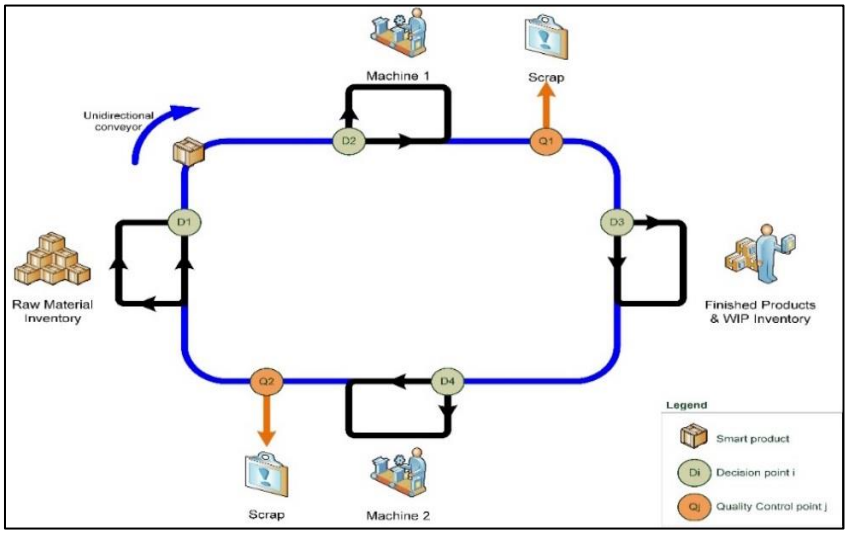

Fig. 2. Illustration of the case study

4.1 Product decisions

Initially, a product leaves the raw material AS/RS without having a schedule. The product moves on the conveyor and crosses decision point D_1, where it acquires its decision indicators (cf. section 4.2). According to this data, and using a decision mechanism, such as the one described in section 4.2, the product selects the machine that will perform the next operation in its routing. Then, the product moves on the conveyor. When it crosses decision point D_2, it acquires the decision indicators and updates the decision it made earlier accordingly. As a generalization, the product can update its decisions based on indicators each time it crosses a decision or a quality control point. At decision points, a product can make one among four possible product decisions (PD):

- **PD1.** Enter the resource loop;
- **PD2.** Stay on the main conveyor to wait until the resource is available;
- **PD3.** Go to the alternative resource loop;
- **PD4.** Return to stock and wait for the next production horizon.

When a product leaves a production resource (M1 or M2), it crosses a quality control point, where it acquires indicators about its quality. According to this data, and using a decision mechanism (cf. section 4.2), the product can make one among four possible product decisions (PD) at quality control points:

- **PD5.** go to finished products inventory if quality indicators are acceptable;
- **PD6.** rework on either machine M1 or M2 if quality indicators are tolerable and machines are available and reliable (rework machine has to be selected);
- **PD7.** go to WIP inventory and wait for rework on either machine M1 or M2 if quality indicators are tolerable and machines are either unavailable or unreliable (wait for a pre-specified period of time before updating the decision);
- **PD8.** go to scrap otherwise.

These decisions are taken using a decision mechanism such as the one detailed in the next paragraph.

4.2 AHP based decision mechanism

The Analytic Hierarchy Process (AHP) described in [18] was adapted to the purpose of this paper to enable products to make decisions and consequently update their next step at each decision and quality control point in reaction to availability and reliability disturbances. Starting from the raw material AS/RS, the global objective for each product is to reach the finished products AS/RS. As in [18], three types of criteria are considered, related to production costs, processing and transportation times and machine reliability. Each criterion type is associated with a set of indicators. A product applies AHP at decision points to select a decision among PD1 to PD4, and at quality control points to select a decision among PD5 to PD8. First the product acquires the indicators associated with the type of decision point. Then the product performs pairwise comparisons between decisions according to each indicator. Then,

comparisons of decisions according to indicators are aggregated to comparisons of decisions according to criteria. Finally, the decision that best suits the global objective is selected. We refer to reference [18] for more details with respect to the different steps of implementation.

5 Capabilities benchmarked against requirements

The case study is implemented in FlexSim and AnyLogic to evaluate their capabilities to implement a product driven control model.

5.1 FlexSim capabilities

FlexSim offers a user-friendly interface and a wide library of standard objects that enable drawing a simulation model quickly. Different library objects are used to build a simulation model (see **Fig. 3**) that corresponds to the case study illustrated in **Fig. 2.**

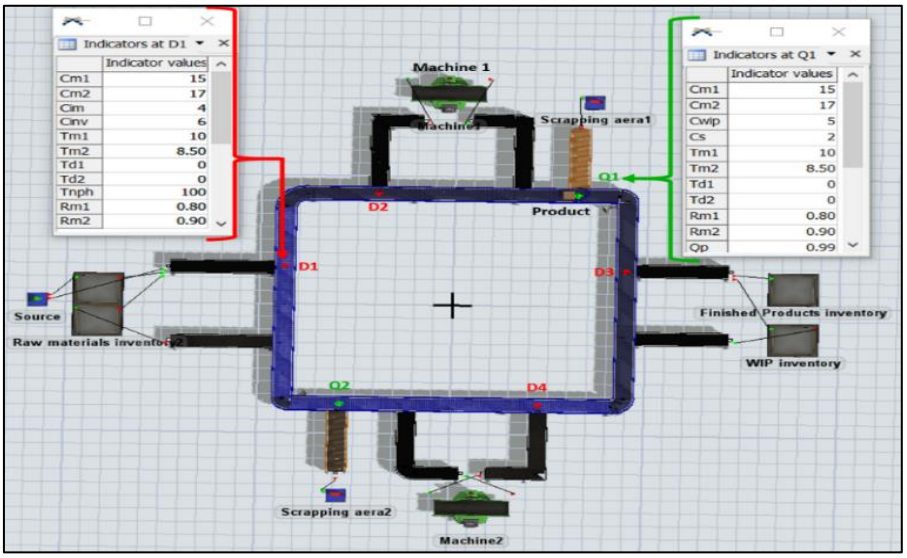

Fig. 3. Simulation of the case study using FlexSim

Production entities: FlexSim offers a rich and user-friendly library containing simulation model objects that can be used to design simulation models. A source node is used to simulate and customize the product arrivals, and to assign the processing cost and time indicators to products as flow items. Three queuing objects for raw material, WIP and finished product inventories are constructed for the waiting areas. Conveyors are constructed to move the flow item. Machines are simulated using two production servers. Two sink nodes are used to simulate the scrapping areas 1 and 2.

Informational structures: FlexSim allows different ways to store and process data and information. It can route items through different resources based on data embedded

in a flow item since its creation at source nodes. It can connect with external data sources (e.g. MS Excel spreadsheets) and databases, such as ERP, MES, HMI, PLC and OPC servers and exchange data using Open Database Connectivity1(ODBC) connection. FlexSim defines specific indicators for each model object. For example, the holding costs associated with inventory are defined in the queuing objects. All indicators are communicated by objects and stored in two global tables named "Indicators at Di" and "Indicators at Qj". The role of the tables is to provide data to decision and quality control points in order to perform the AHP calculations. Global tables enable indirect communication between products and production resources. When a decision is assigned to a product, FlexSim stores this decision in a global list that can be exported to Excel.

Interactions: In FlexSim, library objects are connected to define different process flows and to allow the exchange of physical flows between model objects. The flow of information between objects can be implemented by sending direct messages on state conditions. Custom communication protocols between objects can be programmed on the FlexSim snippet using either FlexScript (FlexSim's proprietary language) or C++. Products are considered as inert flow items that move through the model objects according to their predefined routings, without having active roles or interactions with other entities. At decision and quality control points, the supervisor can change the criteria and weightings of the AHP mechanism using FlexSim interfaces.

Decision-making: as products are inert flow items, they cannot directly process information or do calculations, and consequently cannot be directly endowed with decision mechanisms. To solve this problem, the AHP mechanism is implemented on decision and quality control points (i.e. outside the product). When a product reaches a decision or quality control point, a customized logic encoded on each point allows updating the product routing. Such logic can be programmed using FlexScript or C++ programming languages. FlexSim can directly compile custom code written in C++ via its snippet. It can create .dll[2] files in C++ and link them to FlexSim. It can connect with other programming environments and languages, such as Python and R.

Functioning modes: FlexSim enables defining customized indicators to represent product quality, and customized routines to change the values of these indicators. For example, the value q_p is created to represent a measurement of a product dimension. This quality parameter is susceptible to random events that can change its value (to model product defects). The quality indicator can be directly consulted by the various FlexSim objects. Machine failures can be generated by the MTBF/MTTR fault profile in the FlexSim "toolbox". Using probability distributions, FlexSim can model the first failure time, the down time, and the up time of the machine. Based on mean of machine's up time and the machine operating time, the machine reliability is calculated using the exponential distribution.

---

[1] ODBC is a standard application programming interface (API) for accessing database management systems (DBMS).

[2] Dynamic-link library (DLL) is Microsoft's implementation of the shared library concept in the Microsoft Windows and OS/2 operating systems.

5.2 AnyLogic capabilities

AnyLogic enables modeling all entities (products, resources, storage, and scrapping areas) as agents using agent-based modeling.

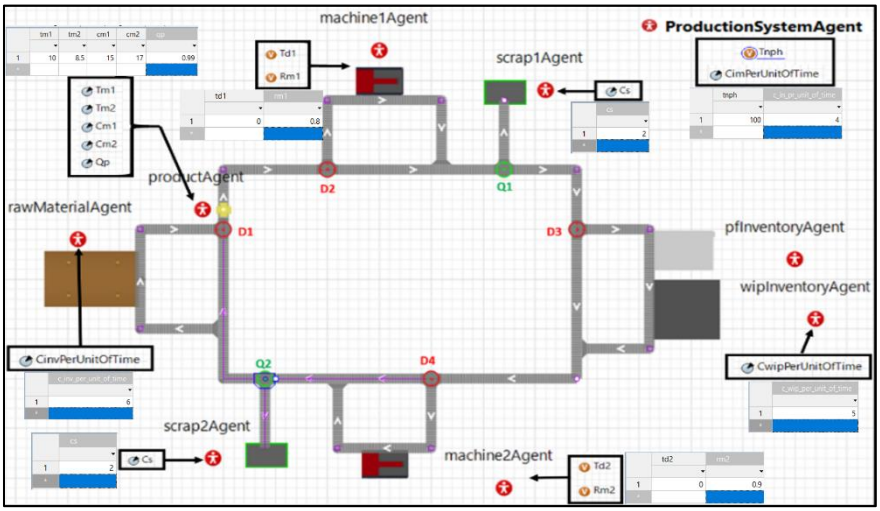

Fig. 4. Simulation of the case study using AnyLogic.

Production entities: The Process Modeling Library of AnyLogic is used to build a simulation model that corresponds to **Fig. 2**: A source that generates *productAgent*(s); Three queuing agents for raw material, WIP and finished product inventories; a conveyor agent that moves *productAgent*(s) at a certain speed, preserving order and space between them; *delay agents* are associated with machines M1 and M2; Two agents are associated with the sink nodes that represent the scrapping areas 1 and 2. Their role is to destroy incoming *productAgent*(s). A *systemProductionAgent* embeds all other agents. AnyLogic enables defining positions on the conveyor where product agents can make calculations. Four decision points D_i (red points in **Fig. 4**), and two quality control points Q_j (green points in **Fig. 4**) are created to enable such calculations.

Informational structures: Product cost and time indicators are defined as *productAgent*(s)' related parameters since their creation. Each product knows its production cost and processing time on M1 and M2. Before a product is processed on a machine, the quality indicator q_p does not contain any value. AnoLogic assigns the value of the indicator q_p embedded in *productAgents* using a set of statistical distribution functions to simulate product defects. *ProductAgents* can consult each time the other indicators from all agents. The native Java environment supports custom Java code, external libraries, and external data sources.

Interactions: AnyLogic is a multi-paradigm simulation software, which features Agent-Based Modeling (ABM). In ABM, the primary consideration is individual agents, their rules, behaviors and interactions with each other and with the environment. Agents living in one environment can directly communicate via sending messages to each other. The simulation runs on one computer system which means all agents share

the same ontology and the use of an Agent Communication Language is not necessary. For example, when the product reaches the decision point D2, if machine 1 is in the product routing, the *machine1Agent* sends a message to the *productAgent* to inform of its availability.

Decision-making: the AHP mechanism is implemented using the Java programming language. It is embedded directly on *productAgents*, and is only triggered when product agents reach decision or quality control points. AnyLogic can work with R[3] programming language through the use of the Java library "Rcaller", which increases the data analytics capability. Also, it can integrate Artificial Intelligence in simulation models using a link to Python language or Skymind's library to enable reinforcement learning. The artificial intelligence models can be constructed externally (e.g. in Python) and may be trained before their integration into AnyLogic.

Functioning modes: Machine failures can be generated by the "*ResourcePool*" block. Recurrent downtime and maintenance activities are scheduled by the different types of triggers or using the AnyLogic Schedule element. In our case, the downtime and maintenance tasks are defined using the properties of the "*ResourcePool"* block.

6 Conclusion and perspectives

This paper discussed the capabilities of two available simulation software to implement the requirements of distributed product driven control. A case study provided the product with some intelligence enabling it to play an active role in the decision-making processes using AHP mechanism. The implementation of the case study on AnyLogic and FlexSim allows to examine the capabilities of these simulation software according to the requirements of distributed product driven control. To do this, a matrix is established **(Table 1)** to map the simulation software capabilities to the case study requirements.

From **Table 1**, it can be noticed that AnyLogic is a consistent simulation software to implement and model a distributed product driven control in industrial context, due to the conjunction of multi-paradigm simulation and high capabilities in data analytics. AnyLogic offers good interoperability with standard programming languages, such as Java, Python and R, which extends its core capabilities to those of the rich libraries available in these languages. Agent based modeling can be combined with discrete event simulation, which further extends the capabilities of simulation using AnyLogic. Agent based modeling enables achieving product intelligence in terms of data processing, communication, interactions and decision-making.

Table 1. Summary of simulation software capabilities for product driven distributed control requirements. (Legend: 😊 Good, 😐 Fair, ☹ Poor)

	AnyLogic	**FlexSim**
Production entities		

[3] R is a programming language and free software environment for statistical computing and graphics.

	AnyLogic	FlexSim
Products	😊	☹
Resources	😊	😊
Decision-making entities (human and/or artificial)	😊	😊
Interactions		
Product-Product	😊	☹
Product-Production resource	😊	☹
Product-Human	😊	☹
Production resource- Production resource	😊	😐
Functioning modes and disturbances		
Machine disturbances	😊	😊
Product disturbances	😊	😐
Intelligence level of the entities (except product for FlexSim)		
Associate informational structures with production entities (except product for FlexSim)	😊	😊
Decision-making (except product for FlexSim)	😊	😊

On the other hand, FlexSim is very strong in 3D animation and is characterized by its user-friendly interface and ease of use. Compared with AnyLogic, which offers multi-paradigm simulation, FlexSim offers only discrete event simulation. In this type of simulation, products are represented and handled as flow items. This introduces limitations with respect to implementing product-based decision-making and interactions. Custom made developments and extra programming is needed to overcome these limitations and achieve product intelligence. As FlexSim offers less interoperability with other programming languages compared to AnyLogic, this is an extra limitation.

Because of the differences between packages, none of them is suitable for use with every type of manufacturing problem [11]. The most appropriate simulation software should be selected for the specific application being studied.

We are considering an extension of the work to take into account several types of disruptions in the production environment (e.g., late delivery of raw materials, conveyor breakdown, etc.) in order to progress more on this work and further develop our model. Even several products can be interconnected (a network of products able to communicate with each other), in this case the products can share their experiences when they make a decision, and they can update the set of actions.

Acknowledgement

This research is financed by the project PHC-TOUBKAL/20/98.

References

[1] W. Derigent, O. Cardin, and D. Trentesaux, “Industry 4.0: contributions of holonic manufacturing control architectures and future challenges,” *J. Intell. Manuf.*, 2020.

[2] P. Leitão, V. Mařík, and P. Vrba, “Past, present, and future of industrial agent applications,” *IEEE Trans. Ind. Informatics*, vol. 9, no. 4, pp. 2360–2372, 2013.

[3] D. Mourtzis, “Simulation in the design and operation of manufacturing systems: state of the art and new trends,” *Int. J. Prod. Res.*, pp. 1–23, 2019.

[4] L. M. S. Dias, A. A. C. Vieira, G. A. B. Pereira, and J. A. Oliveira, “Discrete simulation software ranking — A top list of the worldwide most popular and used tools,” in *Proceedings of the 2016 Winter Simulation Conference*, 2016, pp. 1060–1071.

[5] S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M. P. O’Hare, “Agent Based Modelling and Simulation tools: A review of the state-of-art software,” *Comput. Sci. Rev.*, vol. 24, pp. 13–33, 2017.

[6] G. G. Meyer, K. Framling, and J. Holmstrom, “Intelligent Products : A survey,” *Comput. Ind.*, vol. 60, pp. 137–148, 2009.

[7] I. Kovalenko, D. Tilbury, and K. Barton, “The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems,” *Control Eng. Pract.*, vol. 86, no. March, pp. 105–117, 2019.

[8] J. Dias-Ferreira, L. Ribeiro, H. Akillioglu, P. Neves, and M. Onori, “BIOSOARM: a bio-inspired self-organising architecture for manufacturing cyber-physical shopfloors,” *J. Intell. Manuf.*, vol. 29, no. 7, pp. 1659–1682, 2018.

[9] L. Zhang, L. Zhou, L. Ren, and Y. Laili, “Modeling and simulation in intelligent manufacturing,” *Comput. Ind.*, vol. 112, p. 103123, 2019.

[10] J. J. Swain, “2019 Simulation Software Survey,” *Software Survey*, 2019. [Online]. Available: https://pubsonline.informs.org/do/10.1287/orms.2019.05.10/full/. [Accessed: 18-Apr-2020].

[11] A. M. C. Guimarães, J. E. Leal, and P. Mendes, “Discrete-event simulation software selection for manufacturing based on the maturity model,” *Comput. Ind.*, vol. 103, pp. 14–27, 2018.

[12] L. Fumagalli, A. Polenghi, E. Negri, and I. Roda, “Framework for simulation software selection,” *J. Simul.*, vol. 13, no. 4, pp. 286–303, 2019.

[13] S. Schreiber and A. Fay, “Requirements for the benchmarking of decentralized manufacturing control systems,” in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2011.

[14] L. Mönch, “Simulation-based benchmarking of production control schemes for complex manufacturing systems,” *Control Eng. Pract.*, vol. 15, no. 11, pp. 1381–1393, 2007.

[15] O. Cardin and A. L’Anton, “Proposition of an Implementation Framework Enabling Benchmarking of Holonic Manufacturing Systems,” in *Studies in Computational Intelligence*, 2018, vol. 762, pp. 267–280.

[16] R. Ackoff, “From Data to Wisdom,” *J. Appl. Syst. Anal.*, vol. 16 (1), pp. 3–9, 1989.

[17] P. Valckenaers, M. Kollingbaum, and H. Van Brussel, “Multi-agent coordination and control using stigmergy,” *Comput. Ind.*, vol. 53, no. 1, pp. 75–96, Jan. 2004.

[18] F. Ounnar and P. Ladet, “Consideration of machine breakdown in the control of flexible production systems,” *Int. J. Comput. Integr. Manuf.*, vol. 17, no. 1, pp. 69–82, 2004.