Introducing agent-based simulation of manufacturing systems to industrial discrete-event simulation tools

Lennart Büth, Nik Broderius, Christoph Herrmann, Sebastian Thiede
Institute of Machine Tools and Production Technology, Chair of Sustainable Manufacturing and Life Cycle Engineering,
TU Braunschweig
38106 Braunschweig, Germany
L.bueth@tu-braunschweig.de

***Abstract*—Growing competition and increasing market dynamics force manufacturing companies to increase flexibility in their manufacturing systems. Thus, new approaches, which exceed the limitations of traditional manufacturing systems, are needed. This also requires rethinking the simulation modeling behind those systems. However, simulation software tools used in industry, referred to as industrial grade software tools, and actually used simulation models are not sufficient for this purpose. Therefore, this paper aims at building up an agent-based simulation model of a flexible manufacturing system in an industrial grade software tool. For that aim, a general three-step approach for implementing an agent-based logic into an industrial grade discrete-event simulation tool is presented. An exemplary application is based on a matrix-structured manufacturing system that provides by its structure and by the individual acting entities a flexible and scalable approach without a fixed cycle time. The example is modeled within the industrial grade discrete-event simulation software Tecnomatix Plant Simulation and compared to a reference model, recreating the same results with small deviations.**

Keywords—multi-agent simulation; discrete-event simulation; simulation software; manufacturing system

I. INTRODUCTION

The manufacturing industry is subject to a growing competition and increasing market dynamics in a global environment [1], [2]. Furthermore, customers demand individual products and short delivery times [3]. Manufacturing companies try to counter those challenges by increasing the flexibility of their production [4], [5]. However, with regard to the trade-off between efficiency and flexibility, traditional manufacturing systems (MS) rapidly reach their limits [6]. Due to high fixed costs for equipment and machines as well as for human resources, a high utilization is required. Following, the design and control of manufacturing systems has to be rethought [5]. Moreover, since flexibility leads to more complexity, also the simulation approaches for production planning and the respective simulation software are committed to new requirements.

Although flexibility in manufacturing systems has been addressed in many ways in research and practice, the respective simulation of these systems was generally not in focus [3], [7]. As a rather new approach the simulation method agent-based simulation, which is build up on individually acting entities, was successfully deployed for flexible and complex systems [8], [9]. Also for the modeling and simulation of flexible manufacturing systems, first studies have shown that agent-based simulation (ABS) is a promising substitute for classic simulation methods like discrete-event simulation (DES) [6], [10]. However, the respective simulation models are generally built in simulation environments and as simulation software for specialists as well as without a specific manufacturing context. An implementation in an industrial grade software tool, referring to a simulation software tool that is mainly designed for manufacturing purposes and commonly used in industry, has not been introduced yet.

Against this background, this publication aims at developing a systematic and generalized approach (section III) to implement the agent-based simulation method in an industrial grade discrete-event simulation tool for manufacturing systems. Further, the approach is applied in a case study and verified (section IV).

II. AGENT-BASED SIMULATION IN MANUFACTURING

In a first step an overview about simulation of MS and the paradigm of multi-agent systems (MAS) within them is given, setting the basis for the development.

A. Simulation tools

There are four main methods in simulation modeling: Discrete event (DE), Agent-based (AB), System dynamics (SD) and Dynamic systems (DS) [11]. They can be distinguished by considering if the system states change mainly continuously or mainly in discrete time steps as well as by the level of abstraction a method can represent [11]. On a system level, this publication is concerned with the DE and AB methods. The market includes many alternatives for simulation tools, [12] analyzes 19 different tools and stating their popularity.

There are two common ways to classify simulation software [13]. The first possibility is to classify it by the simulation modeling methods it is capable of. The second way is referring to the range of application the software provides. In this paper, both ways are combined with focus on software which is capable of modeling DES and/or ABS, resulting in the classification of five categories, differentiating in versatility and the specification of their application (see Fig. 1). The most general category is “General programming languages”, offering the most versatility but being also the most time

978-1-5386-0837-1/17/$31.00 ©2017 IEEE

consuming to use, followed by "simulation programming languages". "General purpose simulation tools" offer a broad range of possible applications and different simulation paradigms. Within the manufacturing oriented simulation tools, it can be differentiated between generalized ones and "one application manufacturing tools". These last two categories are regarded as industrial grade tools due to their popularity and dissemination in the manufacturing industry. As shown in Fig. 1 these tools are mostly concerned with the DE simulation paradigm, resulting in the research question "How can DES tools be expanded to ABS tools?".

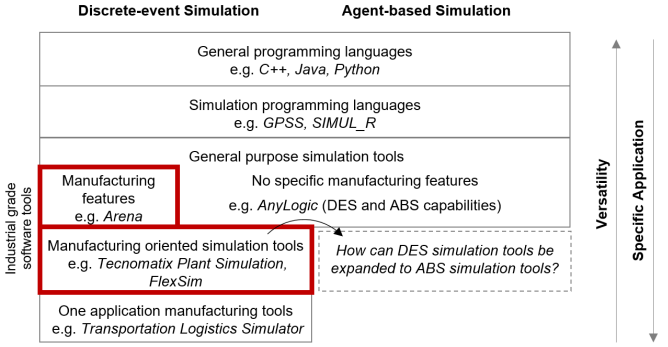

Fig. 1. Classification of simulation software and research question

B. *Properties of Agents and Multi-agent systems*

Russel and Norvig generally define an agent as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" [14], see Fig. 2. An agent is connected to its environment by some kind of sensor and the perception of the environment by its sensor triggers an action [15]. The first key concept of the definition is thus the situatedness of the agent in its environment [16]. Wooldridge and Jennings distinguish in [17] two levels (or degrees) of agents. The first and unconscious level, called Weak-Notion-of-Agency, includes agents that have, in addition to the situatedness, these four basic properties [17]:

- **Autonomy:** Agents act without any direct interventions of a user and have at least some control over their internal state and actions.
- **Pro-activeness:** Agents do not only react, but they take the initiative to change the environment in a goal-directed manner.
- **Reactivity:** An agent perceives the dynamic of its environment and responds to changes in it.
- **Social ability:** Agents interact with other systems or communicate by some kind of agent-communication language.

Fig. 2. Simple model of an agent and its interaction with the environment [9]

The topic of MAS emerged from the fields distributed artificial intelligence and distributed problem solving. Thus, it can be defined as a "loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver" [16]. A MAS has these four main characteristics: Each agent has a limited viewpoint as it has incomplete information or capabilities for solving the problem, asynchronous computation, the data is decentralized and there is no global system control [16].

Multi-agent systems can accommodate different types of agents, each specialized in its function. Agents communicate (at least partly) with the basic components of their environment and optionally some agents communicate with each other to share information or provide some service. Agents in an agent-based system might seem less intelligent than individual agents. However, due to their ability of cooperation they are capable of solving problems that are more complex. The proper model of a MAS builds an important building block for the ABS. It consists of two parts [18]:

- **Model of agents:** In this part the active entities are characterized as agents by their own knowledge and behavior; moreover, the agents' architecture and their way of communication is defined.
- **Model of environment:** In this part the system's (in most cases) passive entities and their state variable are defined and generally represented by objects; further, all elements of the physical world (e.g. suppliers) and of the information world (e.g. databases) that influence, but are (in most cases) not part of the controlled system are defined.

III. General Implementation Approach

As Borschev and Filippov have illustrated, DES and ABS are no contradictory methods but can easily correspond [11]. This fact can be used to combine the elements of DES with the functions of a DES tool to implement an ABS in an industrial grade software tool designed for DES. As an ABS is generally run in discrete time steps the timing tools of a DES like system clock and event list can also be applied for it. Further, both methods are built up on entities which are passive in case of DES and active as well as individual in the case of ABS. Most industrial grade simulation software tools provide all similar functionality like intuitive handling, object-oriented structure and custom programming. These functions offer several starting points to transform the elements of a DES into an ABS logic. Fig. 3 illustrates the developed procedure to transform the elements of a DES into an ABS logic, described and derived in the following. On the left hand side of the figure it is exemplified which general elements of a DES are used and the right hand side displays the functions of the simulation software that are used. The breakdown into these three steps results from the simple model of an agent and its interaction, as illustrated in Fig. 2. In the first step the environment is built, in the second step the agent is implemented and in the third step the interaction between the agent and environment are introduced.

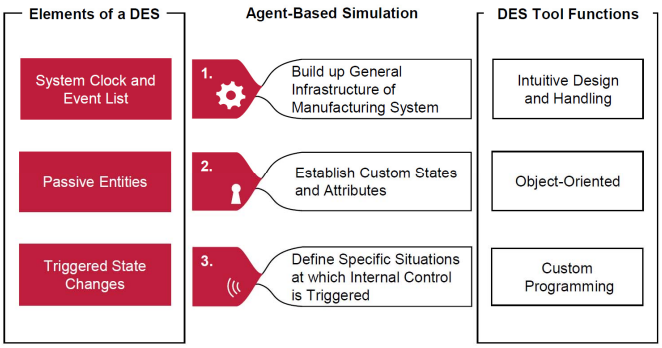

Fig. 3. Three general steps for implementing an agent-based logic into a discrete-event simulation tool

A. *1st Step - Build Up General Infrastructure of Manufacturing System*

In the first step the general infrastructure of the agent-based MS for the agent is built up, representing the environment of the agent. This step is generally equal to building up a traditional simulation of a MS as described in the VDI guideline 3633 [19]. The only difference is that the infrastructure might be more complex to offer the needed flexibility for the MS and its elements. Hence, elements like workstations, buffers, sources and drains are introduced to ensure a steady material flow. A workstation (WS) and a buffer together can be regarded as a working unit. The elements can be either simply connected without any further logistical consideration or a logistic concept can be introduced. This could for example be implemented by means of conveyor lines or tracks with a transporting unit that connect the working units. As an agent-based MS is in general built up to propose a certain flexibility, all working units should be connected to all other working units so a product can freely move between them.

B. *2nd Step - Establish Custom Status and Attributes*

In this step it is determined which elements will be regarded as agents and what will be their individual and unique knowledge. This step is important to provide the agents with some kind of autonomy. As exemplified before, this is one main property of agents. To achieve autonomy, passive entities of a DES are equipped with individual attributes and a unique state by using the object-oriented structure of the DES tools. Thus, the entities gain a certain autonomy which is a first important condition to regard them as agents. The agents to be introduced could include, but are not limited to the elements: product, buffer, workstation, line and drain. All of these elements could possibly take decisions or communicate with the other elements. Therefore, they are regarded as potential agents. Which elements exactly will be taken as agents and how large the level of autonomy is, is dependent from the purpose of the simulation model. A possible attribute for the product agent is the process time for each work package (WP). This attribute is inherent to every product variant and includes the time each WP needs for being processed. Consequently, each product "knows" how long it will need for processing and can hand this information to other agents. Another obligatory attribute is the actual status of a product regarding its actual WP. Thus, the product is aware of which process step is to be machined next. The WS should be equipped with the boolean attributes giving information which WP of which product variant can be processed. Attributes can be implemented in tables as provided in most DES tools. By extending the agents with more attributes different functions can be implemented and the autonomy of the agents is improved.

C. *3rd Step - Implement Control Logic*

The third step can be regarded as the main step to provide the model with an agent-based logic. Here, the interaction of the agents with the environment is defined by implementing pro-activity, reactivity and the social ability. To do this, the DES capability that the entities trigger state changes when an event happens is used to implement an internal control of the agents. Events like a product entering a WS (entrance control) can be easily combined with the call of a custom programmed method. In this method the rules for communication between the agents and the goals by which the agents act can be set. Thus, the agents are represented perceiving the environment by sensors, reacting and making decisions.

To conclude, Fig. 4 shows which property of agents are achieved in which step by which means. By building up the infrastructure of the MS in step 1 it is ensured that agents later can be regarded as situated and connected to some kind of environment. In the 2nd step the agents are determined and provided with an internal state to gain autonomy. In step 3 the agents are provided with pro-activeness, reactivity and social ability. Pro-activeness is reached through the goal directed manner in which the agents act through control strategies which can be implemented in the custom-programmed lines of code. Reactivity is reached through forcing an action of the agent when a specific event like entering a WS happens. Additionally, in the custom programmed methods the way of communication and thus the social ability of the agents is realized. After the 3rd step all capabilities of an agent which were mentioned in *II.B* are assigned in the simulation model and its elements. Thus, the model can now be regarded as an agent-based MS.

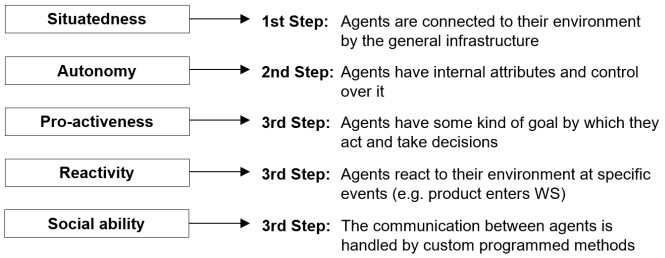

Fig. 4. Properties of agents: Steps of implementation

IV. Case Study and Verification

In the following, the developed general approach is applied in a case study. As example, a Matrix Structured Manufacturing system (MMS) is used, originally introduced, modelled and assessed by [6] using the general-purpose simulation tool *Anylogic*. The agent-based simulation is transferred to an

industrial grade discrete-event simulation tool, *Tecnomatix Plant Simulation*. Fig. 5 highlights the consecutive procedure for the case study, which is explained in the following.

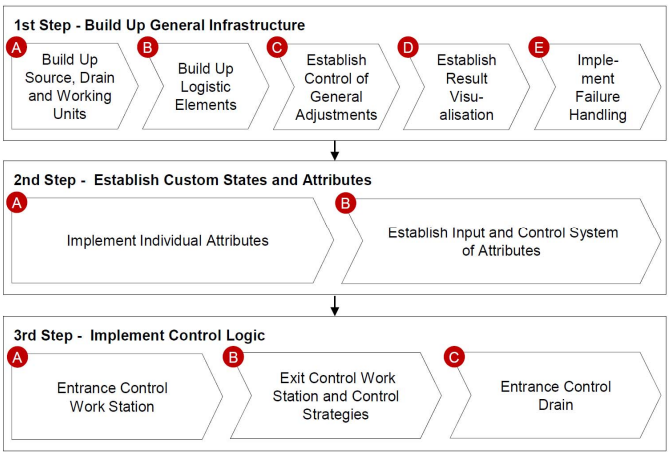

Fig. 5. Case study: Conceptual roadmap for the implementation of an agent based matrix manufacturing system into a discrete-event simulation tool (Tecnomatix Plant Simulation)

A. Build Up General Infrastructure of Manufacturing System

Step 1.A: First, the general elements source, drain, buffer and workstation (WS) have to be built up in the simulation frame. It is important to use the inheritance functions of *Plant Simulation* in such a manner that all elements of one class (e.g. all workstations) are children of a parent element. This allows switching adjustments just for the parent objects and all children inherit that adjustment as well. The source produces movable objects like products. How many variants and in which distribution are to be produced can be adjusted in the tab "Attribute". Further, the frequency and duration of product production can be adjusted. For a better handling of products that have already been produced by the source but cannot leave it, a starting buffer should be implemented. This element has infinite capacity (Capacity = −1) and should be positioned immediately behind the source.

Step 1.B: In the next sub step, all elements are connected by logistic elements like tracks (combined with a transporting unit) or conveyor lines. The decision which elements should be connected to each other should be taken in regard to the modeled MS. Generally, it is appropriate to connect every work unit with every other work unit to achieve maximum flexibility. Therefore, from every WS (and from the starting buffer) a logistic element is put to every buffer. Naming and inheritance are to be assessed according to Step 1.A. The same is true for the positioning of the logistic elements. In a method the position and angle of every line is calculated in two nested loops according to the position of each WS.

It is important to check that the WS and a logistic element as well as a logistic element and a buffer are connected by the element "connector", to ensure that a WS has information about all its succeeding elements. The connectors can be faded out in the *Plant Simulation* menu. Without the connectors, the products will not flow from the WS on to the conveyor line. Furthermore, all WS have to be attached to the drain by a connector. Thus, the product is directly transferred to the drain after its last WP was processed. A connection of a WS and the drain by logistic elements would also be reasonable. Schönemann et al. did not consider logistical elements and the products could freely flow along the whole simulation frame. Consequently, the implementation in step 1.B is a slight amendment to the original MMS to get one step closer towards a realistic logistical system. Fig. 6 shows a screenshot of the modeled infrastructure with main elements.

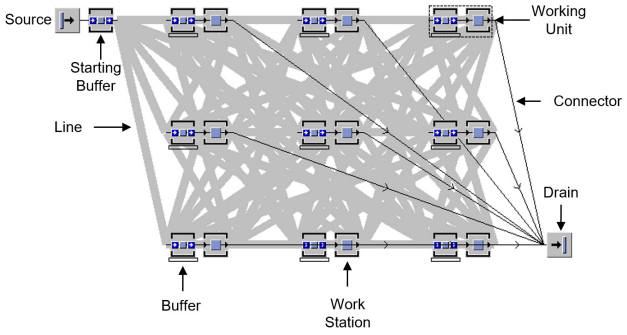

Fig. 6. General infrastructure of case study in Plant Simulation with main elements

Step 1.C: Next, the user interface for the control of general adjustments is established. This ensures that also users that have no knowledge about *Plant Simulation* can run the simulation and evaluate different layouts. The adjustments should be either copied to the elements when the simulation is started in the initialization method (e.g. speed of line) or be directly changed in the method which controls button inputs. The latter is also true for the start, stop, reset button and controls options, e.g. if several WP of the same product can be processed consecutively on the same WS.

Step 1.D: The visualization of the results allows a direct insight into the capabilities of the system and its adjustments. A very important information is the ratio of time that each WS and the system in total is in a specific state. Possible states are working, waiting, blocked or failed. For tracking the states, a table (Utilization) can be positioned in the result center. The ratios of the states from the single workstations can directly be read from the attributes of a workstation. The total ratio is then derived by calculating the average. With two global variables, both the actual total utilization and the cumulative total utilization can be tracked over the whole simulation run. For that, both variables are always recalculated when a product enters a WS and when it leaves a WS, both using programmed methods. *Plant Simulation* offers tools like diagrams that can be used to visualize the change of variables like the utilization over time.

Step 1.E: Failures can be either manually implemented to the WS or at initialization according to inputs of the control center. This can be done analogous to Step 1.A or 1.B. As there are problems occurring when a machine fails directly when the product is about to leave the WS, a method handling these cases has to be introduced. This method is called when a WS failed. Another problem is that once failures have been activated, the simulation will run forever, as there is always a

new failure in the event list. Therefore, failures have to be deactivated when the last variant has left the system. This has to be implemented in the entrance control of the drain.

B. Establish Custom Status and Attributes

In this step, the passive entities are provided with individual attributes. This can be regarded as a first step towards an agent-like behavior. The conveyor line, buffer, drain and source cannot be regarded as full agents. Although they communicate with the product and WS agent for example reporting how many and which products they store, they rather have a passive role.

Step 2.A: Individual attributes can be implemented by opening the properties of the respective parent elements in the class library. For the different product variants, the entities in the folder "MUs" should be copied (Variant1 and Variant2). All attributes are individual to every variant. In the properties, the tab "User-defined" has to be chosen. The Attribute "TimeForWPs" is a table consisting of the process time that the respective product needs for every WP. The attribute "Status" indicates which WP is to be processed next. Both attributes in combination represent the knowledge of the product agent about its manufacturing process. The WS agent has one attribute that provides him with the knowledge about which work packages are possible to process for each variant.

Step 2.B: For simplifying the input of attributes and for allowing flexible changes of the attributes an input and control system has to be established. The information about which WP is process able for all WS can be stored in a table. The information from this table is transferred at every initialization to the individual attribute of each WS by two nested loops in the initialization method.

C. Implement Control Logic

The third step is the key step in transforming the DES to an ABS. Now the situations when an agent takes action should be defined and the agents should be prepared with the capabilities to decide in a goal directed manner. To that end, we use the function of a DES software tool to allocate custom-programmed methods to events occurring to the elements of the simulation. The three events important for implementing the MMS are when a product enters the WS, when it leaves the WS and when it enters the drain. For each of these events a method can be built that handles the decisions and communications of the agents. In the simplified illustration in Fig. 7 the three events and the four allocated actions of agents (I. to IV.) are displayed. In the following, the three events are further outlined with a focus on the control strategies in Step 3.B.

Step 3.A: Both actions that are triggered when the product enters the WS are managed in a method "Entrance-ControlWS". The first action in this step is that the product agent and the WS agent exchange information about the process time the product needs for the next step and the state is incremented by one. Further, the WS agent reports to all preceding WS (and the starting buffer) the information that a new actually blocked product can start to this working unit. All products on preceding WS (and the starting buffer) are checked if they fulfill five conditions to be moved. They still need to have one more WP left, they must be waiting and the actual WP needs to be process able at the WS. Further, the maximum amount of products should not be exceeded (only important for a product from the starting buffer) and the WS should not be already reserved by another product. The first product which fulfills all five relevant conditions is moved. In addition, other strategies would be possible here. For example the product with the lowest or highest state (e.g. according to a prioritization by customer due date) or from a bottleneck WS could be moved.

Step 3.B: In this sub step the control strategy of where the product will go after its actual WS (or starting buffer) should be implemented in a method "ExitControlWSAndStart". This method calls a method "OutControl" where the actual functionality is managed. First, an array of all lines that succeed the actual WS and lead to another WS that is capable of processing the product is returned by a method "FreeSuccMU". Then a loop checks all successors in the array referred to the time the product would need until it is processed. This calculation is realized in a method "ProcTimeSucc" and depends on the control strategy which has been chosen. In this case study three possible control strategies have been implemented in the simulation model: **Random:** A random time between 0 and 1 seconds is returned leading to a random choice of the product agent. **Shortest distance:** Only the transportation time on the line is considered. **Shortest time until processing**: The remaining transportation time and the remaining process time of products on the following WS, buffer and line are considered. If all succeeding WS are blocked, the product waits at the WS until it is "pulled" by the control mechanism introduced in Step 3.A. However, if the earlier exemplified control option to process consecutive WPs in one WS is set to true, the product will check if the next WP can be processed on the same WS as well. In that case, the product is moved to the entrance of the respective WS, initializing the entrance control again.

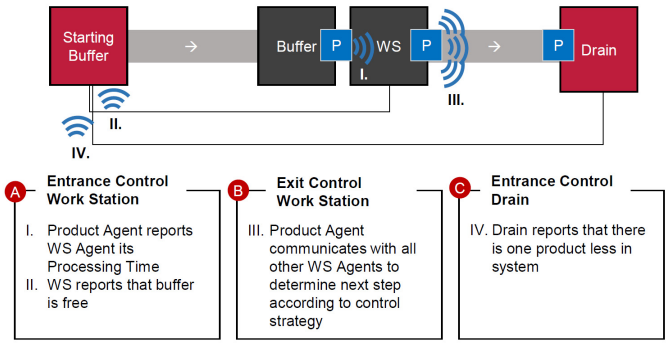

Fig. 7. Overview of triggered situations

Step 3.C: This is the last step of the case study implementation. It is executed when the maximally allowed amount of products was reached before the product has entered the drain. In this case, the drain reports to the starting buffer that the system is now capable of receiving one more product. This information initiates the product agent first in line on the starting buffer to start its decision routine and control strategy. This is managed by the method "OutControl" analogous to Step 3.B.

D. Verification of the Case Study

To verify the implementation, the results regarding the performance indicators duration and utilization of the system are compared with the initial *Anylogic* based simulation model of [6]. To compare both models the initially used parameters (speed of line, buffer size, amount of products, time between products, max. products in system and consecutive WP on one WS) are carried over to the ABS implementation in *Plant Simulation,* using the "shortest time until processing" strategy. To highlight possible deviations of used random function in the sources of the different models, a simple production line model also used by [6] was remodeled in *Plant Simulation* and compared. Table I shows the average utilization and duration of 10 simulation runs and the deviation between both simulation results. Each run included 1000 products.

TABLE I. COMPARISON OF THE INITIAL MODEL BY SCHÖNEMANN ET AL. AND THE MODEL IN PLANT SIMULATION

Configuration		***Initial Model***	***Plant Simulation***	***Deviation***
Line	Utilization	69.08 %	69.03 %	-0.07 %
	Duration	5.933 days	5.945 days	+0.19 %
Matrix	Utilization	86.45 %	88.03 %	+1.79 %
	Duration	4.175 days	4.130 days	-1.07 %

The results reveal that the difference for the line configuration between the initial model and the model in *Plant Simulation* is very small. For the matrix configuration, the deviation is +1.79 % regarding the utilization and −1.07 % regarding the duration. This shows that the effect of the differences in logistic handling (see VI.A) have a greater effect on the matrix configuration. However, as the deviation can still be assumed as rather small, it is to state that the *Plant Simulation* model and the initial *Anylogic* model refer to the same operation of a MS. Thus, the verification can be assumed successful.

V. CONCLUSION

A general procedure was developed to transfer ABS of manufacturing systems into industrial grade DES tools. The initial research question was thus successfully addressed. With the procedure, the agents in the MS are equipped with all capabilities to model agent-like behavior. Nonetheless, all control is handled by different custom-programmed methods that are not inherent to the agent. State charts inherent to the agents that individually handle the behavior of an agent are a functionality that industrial grade software tools do not offer in comparison to tools like *AnyLogic*. Therefore, it could be argued that the system only acts like an agent-based system, as the respective logic's of the different agents have to be provided centrally.

The exemplary application showed that the implementation process is not simple due to the need of many custom-programmed methods. Comparing the realization process of an ABS in an industrial grade DES tools with a general purpose simulation tool, the needed effort seems similar. With existing knowledge regarding the use of DES tools companies can benefit using the developed implementation approach, initializing the transition towards more flexible manufacturing systems.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 680435.

REFERENCES

[1] Y. Koren and M. Shpitalni, "Design of reconfigurable manufacturing systems", Jour- nal of Manufacturing Systems, vol. 29, no. 4, pp. 130–141, 2010.

[2] B. J. Pine, Mass customization: The new frontier in business competition. Boston, Mass.: Harvard Business School Press, 1998.

[3] W. Terkaj et al., "A review on manufacturing flexibility", in Design of Flexible Production Systems, T. Tolio, Ed., Springer Berlin Heidelberg, 2009, pp. 41–61.

[4] A. Sethi and S. Sethi, "Flexibility in manufacturing: A survey", International Jour- nal of Flexible Manufacturing Systems, vol. 2, no. 4, 1990.

[5] G. Askar et al., "Flexibility planning in automotive plants", in Management logistis- cher Netzwerke : Entscheidungsunterstützung, Informationssysteme und OR-Tools, Physica-Verl., 2007, pp. 235–255.

[6] M. Schönemann et al., "Simulation of matrix-structured manufacturing systems", Journal of Manufacturing Systems, no. 37, pp. 104–112, 2015.

[7] Y. Koren, The global manufacturing revolution: Product-process-business integration and reconfigurable systems. Hoboken, N.J: Wiley, 2010.

[8] W. Shen and D. H. Norrie, "Agent-based systems for intelligent manufacturing: A state-of-the-art survey", Knowledge and Information Systems, vol. 1, no. 2, pp. 129– 156, 1999.

[9] M. J. Wooldridge, An introduction to multiagent systems, Reprint. Chichester: Wiley, 2008.

[10] P.Vrba, "Simulation in agent-based control systems: Mast case study", International Journal of Manufacturing Technology and Management, vol. 8, no. 1/2/3, p. 175, 2006.

[11] A. Borshchev and A. Filippov, Eds., From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools, 2004.

[12] L. M. S. Dias et al., "Discrete simulation tools ranking : A commercial software packages comparison based on popularity", 2011.

[13] A. M. Law and W. D. Kelton, Simulation modeling and analysis, 2. ed., internat. ed. New York: McGraw-Hill, 1991

[14] S. J. Russell and P. Norvig, Artificial intelligence: A modern approach. Upper Saddle River: Prentice Hall, 1995.

[15] D. Pawlaszczyk, Skalierbare agentenbasierte Simulation: Werkzeuge und Techniken zur verteilten Ausführung agentenbasierter Modelle: Techn. Univ., Diss.–Ilmenau, 2009. Ilmenau: Univ.-Verl., 2009.

[16] N. R. Jennings et al., Autonomous Agents and Multi-Agent Systems, vol. 1, no. 1, pp. 7–38, 1998.

[17] M. J. Wooldridge, Ed., Intelligent agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages, Amsterdam, The Netherlands, August 8 - 9, 1994 ; proceedings, 2. printing. Berlin: Springer, 1995, vol. 890.

[18] M. Paolucci and R. Sacile, Agent-based manufacturing and control systems: New agile manufacturing solutions for achieving peak performance. Boca Raton, Fla: CRC Press, 2005.

[19] Simulation of systems in materials handling. VDI Guideline 3633, 20.