Development of a Simulation Model to Analyze the Performance of Decentral Rescheduling Algorithms in Production Systems

Julian Sundermeier*, Felix Gehlhoff, Alexander Fay

Helmut-Schmidt-University / University of the Federal Armed Forces Hamburg, Institute of Automation Technology, Holstenhofweg 85, 22043 Hamburg, Germany; *Sundermeier.julian@hsu-hh.de

SNE 30(1), 2020, 15 - 22, DOI: 10.11128/sne.30.tn.10504
Received: June 10, 2019 (Selected ASIM SST Hamburg 2018 Postconf. Publ.), Accepted: October 10, 2019
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Real production systems that rely on manual productionand transportation processes are especially prone to disturbances. Thus, schedules often have to be revised on the spot by the operators using simple heuristics. These methods do not generate optimal solutions. Other approaches use agent-based control systems where the agents enable decentralized rescheduling based on local information. The following paper describes a simulation model of a production system that allows a quantitative comparison of the alternative control and planning logics by using the simulation software FlexSim®.

Introduction

In today's competitive environment, manufacturing companies are confronted with the need of increasing flexibility [1]. This can be attributed to e.g. shortened product life cycles, increasing variety, and competitive pressure, which requires short and reliable delivery times [2]. Therefore, the production system must not only be able to produce efficiently, but moreover it needs to be resilient to disturbances and adaptable to changing demands. In addition, a modularized system architecture is advisable to enable system reconfigurations during ongoing production without having to adjust the control code.

Classic optimization or scheduling methods often struggle with these requirements [3]. Although these techniques can calculate an optimal solution, they require a considerable amount of time for complex problems. Furthermore, when using e.g. linear programming, it is neglected that significant and frequent changes to the production schedule create considerable noise and thus problems within the factory. Therefore, in many real production systems, classic optimization processes are usually not suitable for dynamic rescheduling. Instead, there are mostly two approaches to deal with rescheduling. One is that the operators make the dynamic control decisions on their own. Often these decisions consider only a fraction of the available information, e.g., what the operator assumes to be the most important job to fulfil next. Some companies employ software with simple algorithms which mimic such heuristics, e.g. the right shift (i.e. the postponement of all follow-up orders corresponding to the delayed order). This type of control results in production schedules that lag (far) behind an optimal solution, which could have been attained by using a more sophisticated algorithm.

In order to meet the stated requirements for flexibility and quality of the solution, several researchers propose agent-based control approaches. The agents enable decentralized decision-making and rescheduling based on local information. Usually, this interaction is much faster than classic optimization methods and allows a focused adaptation of the plan. However, the results achieved with decentral decision-making are usually not globally optimal.

The distribution of control intelligence and the information linked to it makes production systems more robust and represents a major goal in Industry 4.0 [4].

Despite an abundant amount of available literature on the development of multi-agent systems [1], there is still a lack of comparison – qualitatively and quantitatively – of agent-based control in case of disturbances with 'traditional' solutions [5]. Benchmark systems are one way to cope with this problem [6], but they mostly do not consider manual and thus less predictable processes.

They also do not provide much help in terms of analysing the effects of rescheduling instead of just applying decentral control algorithms that take local decisions on the spot. Recent approaches, however, already give advice on how to measure qualitative factors such as robustness of a system that reacts to dynamic scenarios [7].

This paper describes a simulation model that enables the quantitative and qualitative comparison of different rescheduling algorithms – e.g. simulated operator decision-making, right-shift algorithms and agent-based dynamic rescheduling – to analyse the aforementioned effects. The context of the simulation model and the agentsystem is outlined in Section 1. Section 2 gives a brief introduction into the simulation software that was used to build the model. The communication scheme between the simulation and the decision maker (e.g. an MAS (multiagent system)) is explained in Section 3. This is followed by a description of the simulation model itself in Section 4. Section 5 briefly outlines the functionalities of the MAS and examines the interactions between simulation and the agents. The paper closes with a short validation of the developed simulation concepts in Section 6 and concludes in Section 7 with a brief summary and outlook.

1 Application Context

The simulation model resembles a factory where heavy single workpieces have to be moved between subsequent production steps. The factory comprises different production stages in two hall bays. In at least one process step, alternative workstations allow parallel processing of workpieces. Manually operated cranes in each hall bay connect the workstations. A limited number of operator teams operates the cranes. It is one goal of the simulation to replicate the estimated durations of these transport operations, as they have a significant impact on the overall productivity of the factory. These estimated durations stem from the ERP system of the considered factory and look similar to the following example in Table 1.

Start	Destination	Time needed [min]
Workstation 1	Workstation 2	30
Workstation 2	Workstation 3	45
	...	

Table 1: Excerpt of the ERP system transport data.

These times include the time that is necessary to load and unload the workpieces but do not include the time needed to reach the workpiece, i.e. the time to move the crane from its previous position to the start position of this transport.

Shuttle cars connect the two hall bays. Two product variants are produced that are made of one main body and one additional part. These additional parts require different work- and transportation processes (e.g. only one crane for transportation instead of two for the main body). One major problem is the coordination of the crane system. Due to various restrictions (e.g. the weight distribution on the ceiling construction), this task becomes particularly challenging.

If disturbances occur during production, today's human endeavours mainly constitute the right-shift after a delayed order and on the spot decisions by e.g. crane operators. This has to be replicated within the simulation model. Furthermore, it is necessary that the simulation is able to follow a specified production schedule, which is stored in a database.

In addition, there must be the possibility that an agent-based decentralized planning algorithm (or any other algorithm) can dynamically adjust or replace the production schedule when disturbances occur and thus control the simulation. If the simulation is controlled by an external planning algorithm it strictly follows the production schedule that is provided by the algorithm via the database. It does not execute any planning or rescheduling functions itself (e.g. right-shift).

2 FlexSim® Simulation Software

FlexSim® is an object oriented, discrete-event simulation software distributed by FlexSim Software Products Inc. (Orem, Utah, USA). Also declared as application-oriented simulation package [8], it is especially common for the simulation of production and intra-logistic processes. FlexSim® offers a GUI with a 2D or 3D view of the model and several standard library objects that can be included into the model via drag & drop.

Generally, FlexSim® can process numerical and string values which can be stored in global variables or tables or locally in labels on an object. For further information on the capabilities of FlexSim® see [9].

FlexSim® has been chosen due to two advantages compared to other simulation software [10]: on the one hand, all library objects can be edited and complex logic can be integrated into their event-triggers by using

FlexScript, an embedded programming language. Logic can also be added by use of C++, but this code needs to be compiled before the model can be run. On the other hand, FlexSim® provides a standard interface to a MySQL database, which can be used for the necessary access to the production schedule. The use of a standard interface facilitates the data exchange, as no new interface has to be configured.

3 Connecting to the Simulation via MySQL

To ensure the required data exchange, the communication scheme depicted in Figure 1 was used.

Both, the decentralized planning algorithm and the simulation model in FlexSim® act as MySQL-clients

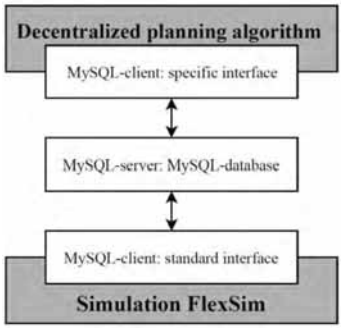


Figure 1. Communication scheme between simulation and agent-system.

via appropriate interfaces to a MySQL-server with the MySQL-database. The database itself contains the three tables 'productionplan', 'resources' and 'monitoring'.

In the table 'productionplan' (see Table 3), all orders are listed according to their unique value of the label 'ID', which is used inside the simulation model. Starting at column two, every seventh column contains the title of a production step. The value of the element that is at the intersection of a row and such a column indicates the assignment of a workstation for an explicit order. The columns in between are used for data collection of the target start/finish and actual start/finish date of a process and whether the processing of an order has already been started or completed.

In the table 'resources' (see Table 4), all workstations are listed according to their unique value of the label 'Object_ID' which is used in the simulation model. In addition, the table provides information on the respective operating status of a workstation and contains information on when and where a disturbance has occurred.

For the decentralized planning algorithm only those entries are relevant which concern the occurrence of a disturbance. For this reason, a third table called "monitoring" has been created, consisting of only one element. This element serves as a binary indicator of whether rescheduling is required or not. Thus, the decentralized planning algorithm only has to monitor this value and only become active when there is a change.

ID	Step1	Target start time 1	Target finish time 1	Processing started 1	Actual finish time 1	Processing finished 1	Step 2	...
1	Value of the label "Object_ID" indicates on which workstation from the table "resources" the order should be processed	Elapsed simulation time at which process step 1 should start	Elapsed simulation time at which the processing of process step 1 has actually started	=1, if processing of process step 1 has started; =0, otherwise	Elapsed simulation time at which process step 1 should end	=1, if processing of process step 1 has ended; =0, otherwise	Value of the label "Object_ID" indicates on which workstation from the table "resources" the order should be processed	...
...	...	...	...	...	...	...	...	...

Table 3: Schematic structure of the table 'productionplan'.

Object_ID	designation	WIP/BU	On/Off	Error_Type	Error_Occure_Time
1	Designation of the workstation where the label 'Object_ID' has the value 1 in the simulation	=1 if an order is being processed in the workstation / buffered in a buffer; =0, otherwise	=1 if the workstation is functional; =0, if there is a disturbance and the workstation is not available	=1 if a defect in the wokpiece itself has been detected; =..., is not considered in this paper	Past simulation time at which the disturbance occurred
...	...	...	...	...	...

Table4: Schematic structure of the table 'resources'.

This value also serves as an auxiliary value for the algorithm to detect whether it has already responded to a disturbance or not. The database that serves as an interface between the simulation and a planning algorithm is integrated within the process depicted in Figure 2.

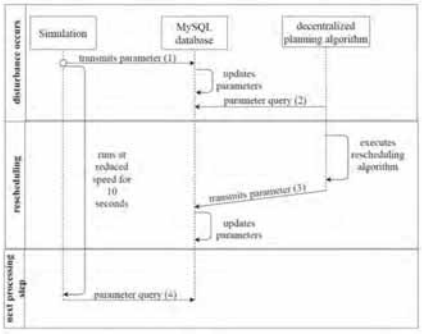


Figure 2: Sequence diagram of disturbance handling between simulation and MAS.

This process shows how the simulation and the decentralized planning algorithm interact in case of disturbances. Section 5.2 explains the details of this interaction. Due to the intended application of the decentralized planning algorithm within a real factory (see section 5.1) there should be no more than ten seconds between the occurrence of a disturbance and further processing after the rescheduling algorithm has been executed.

For the execution of the rescheduling algorithm, the decentralized planning algorithm might need up to nine of the ten seconds available. In the remaining second, there are four accesses to the database (see numbers one to four in Figure 2). First, a disturbance occurs in the simulation and the simulation sets corresponding parameters in the database. Second, the decentralized planning algorithm reads these parameters and executes the rescheduling algorithm. Third, after the execution is completed, the production schedule is updated in the database. Finally, the simulation queries the database to receive the updated production schedule and runs the model accordingly.

To validate if this concept is feasible it should be determined whether the access time to the database can fulfil a real-time criterion of a maximum of 250 ms (four accesses in one second). Within that access time, the connection to the database must be established and all query parameters exchanged. To analyse the access times a proof of concept model has been created (see Figure 3).

A source object creates ‘flowitems’ that are buffered in a queue object. The transport from the queue object to the processor object can be done either by an operator or by a transporter. The respective assignment is stored in a MySQL database. After processing on the processor object, the ‘flowitems’ leave the system via a sink object. If either the operator or the transporter fails during a simulation run, an agent system is activated which adjusts the assignments in the database to transport all ‘flowitems’ from the remaining resource.

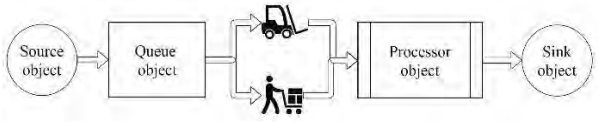
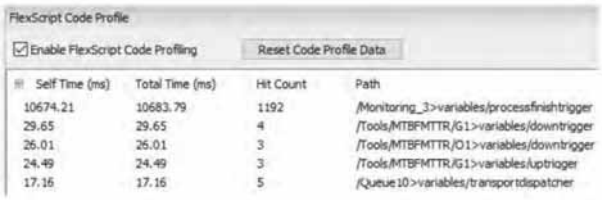


Figure 3: Structure of the proof of concept model.

In the proof of concept model, parameters from the database were queried and / or manipulated by firing individual event triggers. To determine the required access time for such an operation, the FlexSim® internal FlexScript Code Profiler tool was used (see Figure 4).


FlexScript Code Profile

Enable FlexScript Code Profiling | Reset Code Profile Data

Self Time (ms)	Total Time (ms)	Hit Count	Path
10674.21	10683.79	1192	/Monitoring_3>variables/processfinishtrigger
29.65	29.65	4	/Tools/MTBFMTTR/G1>variables/downtrigger
26.01	26.01	3	/Tools/MTBFMTTR/O1>variables/downtrigger
24.49	24.49	3	/Tools/MTBFMTTR/G1>variables/uptrigger
17.16	17.16	5	/Queue10>variables/transportdispatcher

Figure 4: Excerpt from the FlexScript Code Profiler tool.

Lines two through four include function calls that both query and manipulate parameters from the database. In the simulated time, the functions were called four times (see second row, third column) or three times (see third and fourth row, third column). Overall, the function calls in total according to the second column have required between 29.65 ms and 24.49 ms. This results in an average access time of 5.666 ms per function call for reading and writing a parameter into the database. The fifth row contains a trigger in which only one value is entered (not read) into the database. This results in an access time of 3.432 ms per function call. In summary, access times are short enough to guarantee the real-time capability of 250 ms.

4 Simulation Model

This section first discusses the conception of the simulation model. For this purpose, the basic requirements and the particular restrictions are emphasized. Subsequently, the respective implementation will be briefly discussed.

4.1 Conception

A basic requirement for the control of the simulation model is that the orders run through the system according to the given production schedule. It must be ensured that the sequence of the production processes is adhered to and the actual start and finish dates are entered into the database.

Beside the basic material flow, particular restrictions must be respected during implementation. This includes e.g. compliance with follow-up constraints arising from production requirements.

The crane system is a key component of the production system, because it is indispensable for the material flow. In this context, a transport can be understood as a process characterized by a movement to the pick-up location and a transport time. These times correspond to the set-up and processing time in a production process. For this reason, a crane in the simulation model can also be represented by the standard object 'processor'. Although this abstraction restricts the visualization of the material flow in the model, it also offers an advantage. Namely, the routing of the 'flowitems' can be more easily controlled because the behavior of the input and output ports of a processor object can be specified via appropriate event triggers. Both, the time for the movement to the pick-up location as well as the transport time are stored in the database. If the production schedule was created by a decentralized planning algorithm, the time required for the movement to the pick-up location is eliminated since this movement is anticipated in advance.

In order to avoid coordination problems due to overlapping work areas of the cranes in a hall bay, capability profiles are created which define the executable transport processes of a crane. As these profiles are mutually exclusive, there are only very few possible collision points. These can be disregarded due to the very low probability and the capability of human operators in the real application to avoid the actual collision (e.g. by waiting until the other crane has been removed).

A last challenging problem occurs from the restriction that FlexSim® does not provide an option within its standard commands to cancel the processing of a 'flowitem' on a processor object. However, this is exactly what should happen when a material defect is detected so that the 'flowitem' can be transported to another workstation for a post-processing.

4.2 Implementation

To facilitate later adaptation or reconfiguration of the model, only standard library objects have been used for the implementation.

The routing of the 'flowitems' and the associated adherence to the process sequence is ensured by the two labels 'ID' and 'NextStep', which are stored on each 'flowitem'. While the value of the label 'ID' is constant, the value of the label 'NextStep' is adapted dynamically after the successful completion of a production step.

This is done via the 'OnProcessFinish' trigger of each processor object that represents a workstation in the simulation model. The value of the label 'NextStep' starts at 2, so it points to the second column of the table 'production-plan', where the assignment of the workstation for production step 1 takes place (compare Table 3). After the production step finished, the value is increased by the value 7 so it points to the ninth column etc.

For the modeling of the cranes, a pull system was implemented, which ensures that a 'flowitem' is not transported until its processing is completed and the following workstation is idle. If both conditions are fulfilled, the 'flowitem' gets pulled and the value of the label 'NextStep' is read out. Thereafter, the 'flowitem' is processed (this represents the ongoing transport) and finally send to the target workstation. The processing times correspond to the estimated times from the ERP and are chosen according to the destination. Each crane is connected to other workstations through its output ports, which reflects the implementation of its capability profile.

The actual start and finish date of a production step is set by the 'OnEntry' and 'OnExit' trigger of a processor object which represents a workstation.

Compliance with the follow-up constraints can easily be achieved due to the routing concept used.

In order to simulate the detection of a material defect, including all subsequent repair processes, a slightly more cumbersome solution was implemented. For this, a queue object was created for each workstation and then connected via the center port. After that, a separate mean time between failure / mean time to repair (MTBF / MTTR) object was created for each workstation, which is not intended to control the failure behavior of a work-station, but instead simulates the detection of a material defect. For this reason, the MTTR has been set to a very small value.

The 'OnBreakdown' trigger contains the functions that cause the canceling of the current production step.

The 'flowitem', which is located on the workstation whose 'OnBreakdown' trigger was fired, is duplicated and the duplication is send to the connected queue object. Then the original 'flowitem' is destroyed, which represents the cancellation of the processing. When executing this command while processing a 'flowitem' on a processor object, the FlexSim® engine can no longer execute the other scheduled events of the processor object (e.g. completion of processing).

For this reason, another command must destroy all scheduled events. At the same time, this command causes the affected processor object to never receive 'flowitems' again, even if it is intended in the production plan. However, this can be removed by executing a command that resets the properties of the processor object.

After all commands of the 'OnBreakdown' trigger have been executed, post-processing of the duplicated 'flowitem' must be scheduled by the decentralized planning algorithm. This process – among others – is explained in the next section.

5 Interaction Between Simulation and Decentralized Planning Algorithm

This section examines the interactions that take place between the decentralized planning algorithm (henceforth, an MAS is assumed to fulfil this role) and the simulation in case of disturbances as well as the functionalities of the MAS itself. The detailed design of the MAS is not a part of this paper, however, it is briefly outlined which functionalities are provided by the MAS.

5.1 Functionalities of the MAS

Starting point for the interaction between the MAS and the simulation is the creation of an initial schedule. The MAS in place is designed similar to approaches like [11] or [12]. Workpiece agents are responsible for the fulfilment of all necessary production steps for the workpiece which they represent. Requirements regarding the number and type of workpieces to be produced as well as the production plan for each type of workpiece are stored in the database. A production plan looks similar to the following example.

ID	Product Name	Step	Operation (Op)	First Operation	Last Operation
1	A	1	Op_1.1	1	0
1	A	2	Op_1.2	0	0

Table 2: Exemplary excerpt of a production plan.

All information regarding the resources is also stored in the database. The agent system uses the same table for this information as the simulation. Each resource has a certain capability. These capabilities are mapped to operations that can be accomplished by the resource.

This design, which uses a database to store all necessary information for products and resources, makes the MAS easily adjustable to changing needs and production technologies as there are no changes necessary within the code of the agents itself. When all agents are created, which happens automatically based on the information in the database, the workpieces use an auction-based process based on the Contract Net [13], which is the most common coordination mechanism for MAS [14]. As the MAS is implemented in JADE [15], the workpiece agents can search for resources that provide certain operations by using the so-called Directory Facilitator of the JADE platform.

The main goal of the agent system is to reschedule in case of disturbances. Thus, the scheduling process is not focussed on creating the optimal solution for the factory for all orders at the same time but instead schedules one job after another and tries to keep as close to already scheduled operations as possible. This results in a completely decentralized approach in which no supervisory component or agent knows all information and each agent manages its own schedule.

Each workpiece schedule is created including the transport processes, as these might be a bottleneck within the production system. To achieve this, the operations at workstations (WS) are scheduled as depicted in Figure 5, similar to [12].

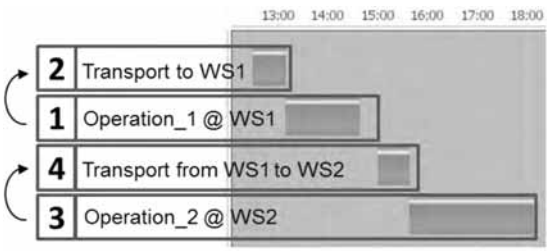


Figure 5: Exemplary scheduling in a regular case.

The workpiece agent uses an estimation of the transport time as the earliest possible start date for the initial production operation request or CFP (call for proposal).

After the transport operations have been scheduled, the agent sends messages to the production resource agents that inform them about the actual arrival and departure times. In case the transport process is the bottleneck, the production step is cancelled and re-booked at the earliest possible arrival time. This time is derived from the best proposal of all transport resources. If a disturbance has happened that demands a deviation from the initial production plan – here a repair process at another workstation – the scheduling process follows a similar approach. The difference is that not only the production operation has to be scheduled but also the buffering and repair process. The production operation has to be scheduled first because the necessary duration, i.e. the end of the buffering process, depends on the earliest start of the production process.

5.2 Interaction in case of disturbances

Different situations can trigger a rescheduling process by the MAS. These vary from equipment breakdowns to process delays. The focus of this paper is on defects that are detected in the workpiece itself. When combining the MAS with the simulation, the simulation has to set the triggers for the MAS in the database that will start a rescheduling process (as shown in Figure 2).

The information provided include the error type and occurrence time in the resource table at the applicable resource. In addition, the parameter in the table 'monitoring' is set to true. The simulation also goes into a mode of slower simulation speed. This is necessary because the FlexSim® engine does not offer an opportunity to completely stop a simulation run for a certain period of time. The dynamic adaptation of the simulation speed uses the object orientation of FlexSim®. Thus, the 'OnBreakdown' trigger, which is fired anyway in the detection of a material defect, spawns a new 'flowitem' in a specially created Queue object. This 'flowitem' fires the 'OnEntry' trigger of the Queue object, which reduces the simulation speed to one hundredth of the original simulation speed. In addition, the 'OnEntry' trigger sends a message to a second Queue object with a delay of 10 real-time seconds. Upon receiving the message, the second Queue object increases the simulation speed to its original value.

This procedure does not prevent the case that, during the time when the simulation is running at a reduced speed, another event occurs which relies on information from the database. However, this case is considered very unlikely and thus acceptable.

The MAS monitors a parameter that triggers its activities. If this parameter is set, the agent detects this change and reads all necessary disturbance information from the database. Afterwards the parameter in table 'monitoring' is set back to false. The information at which resource and what time the disturbance occurred enables the agent to conclude which workpiece agent is involved and needs to be contacted. The disturbance information is sent to the corresponding agent. The workpiece agent in turn up-dates its own schedule with the process finished values from the database that the simulation has set for each operation. The workpiece agent then determines which actions are needed for the disturbance that occurred. In this case, all operations that are not finished yet have to be cancelled and the already explained error handling with booking of a buffer place is started.

After the new schedule is complete, the agent sends this schedule to an agent that has the capability to insert this schedule correctly into the database (DB Connector). A socalled Sniffer Agent that comes with the JADE platform can automatically monitor and visualize the exchange of messages. An abbreviated (as can be seen from the number of messages on the left) and commented example is shown in Figure 6.

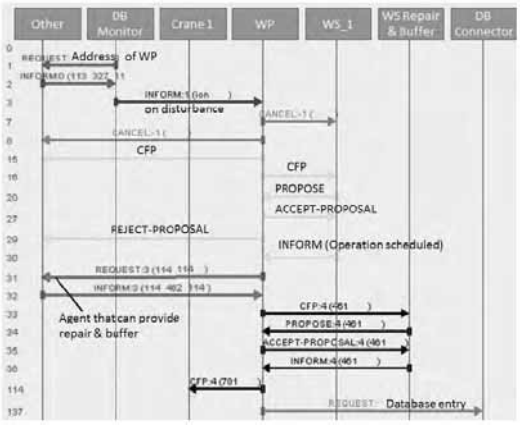


Figure 6: Sequence diagram of disturbance handling within the MAS.

The simulation now uses these actualized values to determine the correct destinations of the objects within the simulation.

6 Validation and Analysis

It is necessary to validate whether the simulation model does correctly implement the production schedule that the agent system has provided and show the effects of default solutions in case of incorrect schedule data. In addition, it must be analyzed whether the production schedule is correctly implemented after a rescheduling by the agent system.

In order to validate that a provided production schedule is implemented correctly, two labels were stored on each flowitem. The production schedule as provided in advance is stored on one label. On the other label, the actual process sequence that a 'flowitem' passes through is plotted during a simulation run. When a 'flowitem' reaches the sink and both labels match, the production schedule was implemented correctly. This result was achieved in the evaluation of several simulation runs. If the rescheduling algorithm was executed during a simulation run, probably the labels no longer match. In this case, it is currently necessary to manually check whether the updated production schedule has been adhered to.

On the other hand, it could be shown that the simulation model cannot implement incorrect production schedules. If the simulation implements such a production schedule, the material flow stagnates and the simulation run is aborted.

The provided target times could be adhered to with minor deviations. This results from the comparison of the corresponding values in the database with the actual times entered by the simulation.

7 Conclusion and Outlook

This paper presents an approach how control algorithms, such as a decentralized planning algorithm, can be tested in combination with a simulation model in FlexSim®, linked via a production schedule in a database. Since it is possible to link arbitrary algorithms to the database, the simulation model can be used to compare algorithms which implement different production schedules. Possible metrics that can be examined are e.g. the throughput time of the orders, the makespan or the impact of the occurrence of a disturbance.

References

[1] Leitão P. Agent-based distributed manufacturing control. A state-of-the-art survey. In: Engineering Applications of Artificial Intelligence 22 (7), pp. 979–991.

[2] Wiendahl HP, ElMaraghy HA, Nyhuis P, Zäh MF, Wiendahl HH, Duffie N, Brieke M. (2007) Changeable Manufacturing - Classification, Design and Operation. In: CIRP Annals 56 (2), pp. 783–809.

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

[4] Spath D, (Hrsg.) Ganschar O, Gerlach S, Hämmerle M, Krause T, Schlund S. Studie des Fraunhofer-Insti-tut für Arbeitswirtschaft und Organisation IAO: Produktionsarbeit der Zukunft – Industrie 4.0, Stuttgart 2013.

[5] Wior I, Jerenz S, Fay A. Automated transportation systems subject to interruptions in production and intralogistics - a survey and evaluation. In: International Journal of Logistics Systems and Management (IJLSM), Vol. 30, No. 4, 2018. DOI: http://dx.doi.org/10.1504/IJLSM.2018.10011675

[6] Schreiber S, Fay A. ARGESIM Benchmark C20 'Complex Production System' – Definition and Call. In: SNE Simulation Notes Europe, vol. 21 (3-4), 2011.

[7] Trentesaux D, Pach C, Bekrar A, Sallez Y, Berger T, Bonte T, Leitão P, Barbosa J. (2013) Benchmark-ing flexible job-shop scheduling and control systems. In: Control Engineering Practice 21 (9), pp. 1204–1225.

[8] Law AM. Simulation Modelling & Analysis. New York: McGraw-Hill, 2007, pp. 182-185

[9] Beaverstock M, Greenwood A, Lavery E, Nordgren W. Applied Simulation: Modelling and Analysis Using FlexSim. Orem, Utah, USA: Flexsim Simulation Software, 2011.

[10] Swain JJ. Simulation software survey: Simulated worlds. In: OR/MS Today, Vol. 42, No. 5, October 2015, pp. 36-49.

[11] Heinze M, Lüder A, Gantner W, Kühnle H, Peschke J. (2008) Structure and Functionality of a PABADIS'PROMISE Agent System. In: Klaus-Dieter Thoben (Hg.): ICE2008. The 14th International Conference on Concurrent Enterprising, a new wave of innovation in collaborative networks, Lisbon, Portugal, 23 - 25 June 2008.

[12] Badr I, Schmitt F, Göhner P. (2010) Integrating Transportation Scheduling with Production Scheduling for FMS: An Agent-Based Approach. In: IEEE International Symposium on Industrial Electronics (ISIE), 2010.

[13] FIPA Contract Net Interaction Protocol Specification, Geneva 2002. URL: www.fipa.org/specs/fipa00029/SC00029H.pdf (Stand 26.07.2018).

[14] Caridi M, Cavalieri S. (2007) Multi-agent systems in production planning and control. An overview. In: Production Planning & Control 15 (2), S. 106–118.

[15] Bellifemine F, Caire G, Greenwood D. Developing Multi-Agent Systems with JADE, Chichester 2007.