A Discrete-event Modeling of ARGESIM Benchmark C20 'Complex Production System' using Flexsim Simulation Software

Eike Schulz*, Josef Paul, Sebastian Schreiber, Alexander Fay

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

Abstract. ARGESIM proposes a new benchmark description ARGESIM C20 'Complex Production System'. This article describes a first implementation of this benchmark by use of the simulation software *Flexsim*. The simulation model is controlled by a heterarchical control algorithm implemented in *Flexsim*, which covers routing and scheduling and is based on reservations. The results are presented according to the requirements of the benchmark description.

Introduction

In [1], Schreiber and Fay propose an advanced reference system for the benchmarking of manufacturing control systems. As a first implementation a simulation model of the presented manufacturing system has been modeled by use of the simulation software *Flexsim* [2]. In addition, a heterarchical control algorithm has been developed that can cope with the challenges of the benchmark description.

This algorithm has also been implemented in the *Flexsim* model. Furthermore, interfaces have been implemented which allow the subsequent integration of other internal and external control algorithms into the model. Another implemented part of the simulation model are routines for the generation of output data for external analysis of each simulation run.

In this article, a brief introduction into the functionality and capabilities of *Flexsim* is given in Section 1. The designed simulation model is described in Section 2 the developed control algorithm in Section 3. Results of the validation of model and algorithm are presented in Section 4.

1 *Flexsim* Simulation Software

Flexsim is an object-oriented, discrete-event simulation software distributed by Flexsim Software Products Inc. (Orem, Utah, USA). Although declared as general purpose software, it is especially suitable for production and intra-logistic tasks. It offers a graphical user interface with a 2D or 3D view of the model and several standard library objects that can be included into the model via drag & drop. In addition to the 2D and 3D view, *Flexsim* offers a tree-view which shows all objects, tools and variables which the model consists of as a hierarchically structured tree. By creating a separate tree node for each component of the model, *Flexsim* assures its object-orientation.

All library objects can be edited by the user and complex logic can be placed in their event-triggers by using *Flexscript*, an integrated programming language. Logic can also be added by use of C++, but this code needs to be compiled before the model can be run. In addition, it is possible to bind DLLs to add user-specific functionality.

Generally *Flexsim* can process numerical and string values. These can be stored in global variables or tables or locally as subnodes of individual objects (so called 'labels'). Communication inside the model is possible by sending messages (which can contain up to three numeric values) from one object to another or by directly referencing another node. An interface to Excel is used for import and export of Excel/CSV files. For further information on the capabilities of *Flexsim* see [3].

SNE Simulation Notes Europe – Print ISSN 2305-9974 | Online ISSN 2306-0271
SNE 22(1), 2012, 55-60 | doi: 10.11128/sne.22.bn20.10119

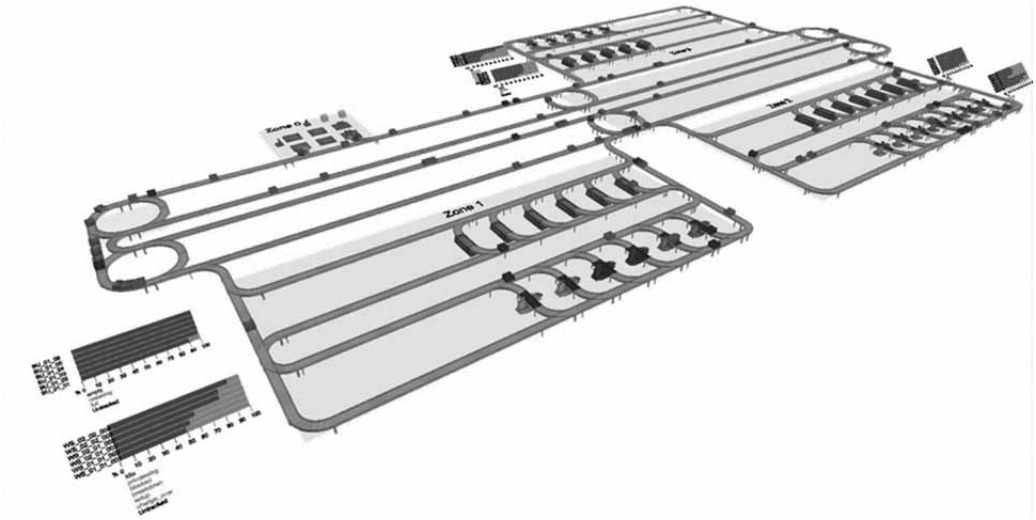

Figure 1. 3D view of the simulation model.

2 Reference System Model

A special demand imposed on this implementation of the 'Complex Production System' benchmark had been to provide the possibility to control it by different control algorithms, both control algorithms implemented in *Flexsim* (by means of the above-mentioned programming languages) and external ones, which communicate with the simulation model via OPC interfaces.

To fulfill the requirements regarding flexibility and adaptability, all parts of the model have been based on standard library objects: work stations (WS) on the standard *processor* object, the conveyors on the standard *conveyor* object, and the FIFO-buffers (BU) on the standard *queue* object. Furthermore, in each instance of an object class (e.g. WS or conveyor) the corresponding triggers have the identical code. The only individual difference is the instance's name. This name follows a defined nomenclature which for example in case of the WS looks like '*WS_xx_yy_zzz*', where the abbreviation 'WS' characterizes the object class; the first two numbers 'xx' show the type of WS (i.e. I-V); the two numbers 'yy' show the generation of the WS; the last three numbers 'zzz' are an instance specific number for this specific type of WS.

Based on this nomenclature, all other individual parameters, e.g. tool capacity and operation capabilities, are stored in global tables and variables and are retrieved on reset of the model. During runtime they are stored locally in labels on each object. By this means, the user can on the one hand easily change parameters for several objects by just editing one global table, and on the other hand quickly create new objects by just cloning an existing one, editing its name and adding a line for the new parameters in the corresponding global table. This enhances flexibility and adaptability of the simulation model. The WS setup and the layout of the conveyors follow the description in [1]. A picture of the production system is shown in Figure 1.

Although *Flexsim* is not designed to make a separation between system's behaviour and control system itself, such a separation is advisable to run different control algorithms in combination with the simulation model of the production system. Therefore, basic functions (i.e. import of order books and retrieval of object capabilities) have been separated from functions that are used specifically by some control algorithms (e.g. machine and buffer scheduling and routing decisions).

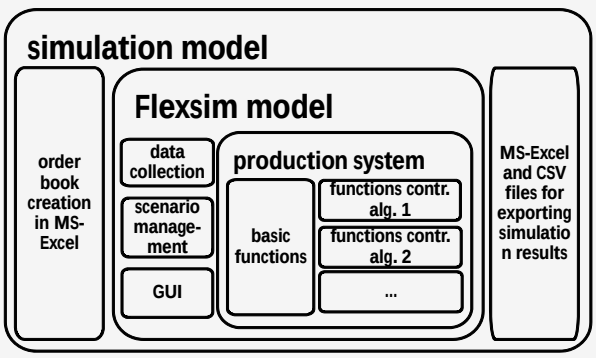

Figure 2: Extent of the simulation model.

The code of the latter kind of functions is capsulated within "case select" blocks, where a global variable indicates the currently active control algorithm. During runtime, only the code of the corresponding "case" is executed. To enable the simulation model to be used with other control algorithms, only a new "case" needs to be defined and implemented.

Another part of the model are the routines for importing order books and exporting validation data. To make later comparison to other implementations of the ARGESIM C20 Benchmark possible, the order books as well as the validation data are stored as Excel/CSV files. Before the start of a simulation, the selected order book is imported into a global table. During the simulation run a variety of data is collected (e.g. state changes of all WS and buffers, entry and exit times of all products into WS and buffers). At the end of each simulation run, this data is aggregated and exported into Excel files which already include formulas to calculate the criteria proposed in [1].

There are two main reasons for not using the *Flexsim* internal analysis tools. The first is that Excel/CSV files can easier be accessed and retraced by others, the second is the possibility to use the add-in RExcel for further data analysis. The general structure of the simulation model as described in this paragraph is shown in Figure 2.

3 Heterarchical Control Algorithm

The first control algorithm that has been implemented for the ARGESIM C2E benchmark is of heterarchical nature and has been implemented directly in *Flexsim*. The control algorithm has to ensure that all items of the order book are produced according to their operation sequences and that all production constraints are met. It also has to manage tool replacements of WS type II and the operation changes of WS type V. Therefore the key elements of the control algorithm are its abilities regarding the scheduling of WS and buffers and the routing of the items between them. The scheduling is done by means of reservation routines for WS and BU slots. These routines are the most complex ones in the model and are the core of the algorithm.

As a pre-condition for the routing, all objects that can be selected as a target (i.e. WS, buffers, sink) and all split junctions have a unique ID that is assigned on reset of the model. The split junctions, which are called "decision points" (DPs) in the model, are the points where the routing decisions take place. When an item with a target ID enters a DP, it is checked which of the exit ports leads to the shortest path to the target. Because of the fixed layout of the transport system, the shortest paths can be calculated already at reset of the model. This is done by use of the 'Floyd-Wharshall all-pairs shortest-path' algorithm.

The results are stored in global tables and can be accessed by each DP. For Scenarios 10 to 12, shortest-path calculation can be repeated after the breakdown or repair of a conveyor to update the distance-table and to allow rerouting of elements. In addition to the search for the shortest path, there are some fail-safe procedures implemented in the DPs. These prevent system to get into a block or deadlock state.

Every time an item enters a DP, a procedure called *Check_Item* is executed before the routing decision starts (see Figure 3). In this procedure it is first checked whether the item has a reservation or not. If not, the reservation sequence is initialized, and the check is completed. If the item already has a reservation, it is checked if there are any production constraints connected to this reservation and if they are still met. It is also checked if the item can still arrive on schedule at the designated WS. If there are any problems recognized with the existing reservation, the necessary steps are taken to cancel it and initialize the reservation sequence again. The *Check_Item* procedure is designed as a user defined function (called 'user command' in *Flexsim*) that can be automatically executed from every DP.

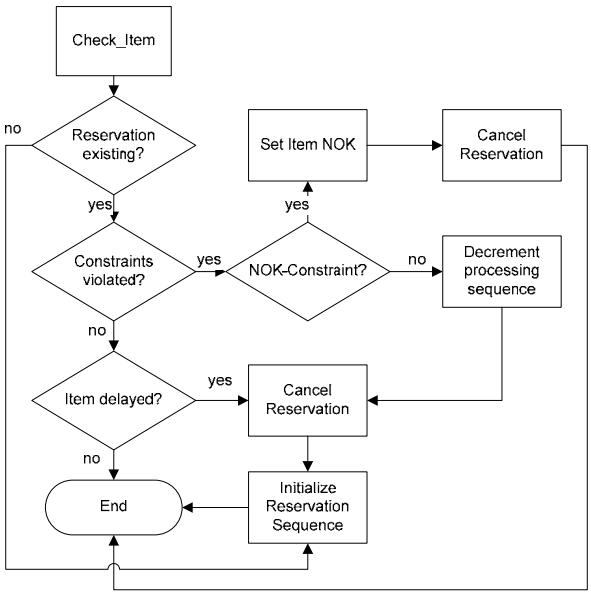

Figure 3. Check_item procedure at each decision point.

Sc.	Task A makespan[h] Median	IQR	Task B throughput time [s] Median	IQR	Task C No of tardy jobs Median	IQR	Task C total weighted tard. [h] Median	IQR	Task D No of NOK-1 Median	IQR	Task D No of NOK-2 Median	IQR
1	21.38	0.00	914.4	445.8	626.5	0.0	25.03	0.00	13.0	00.00	18.0	0.00
2					669.5	0.0	27.99	0.00				
3					745.5	0.0	33.69	0.00				
4	21.47	0.16	982.1	511.0	953.5	28.3	76.09	5.47	25.5	13.00	30.0	8.50
5					1019.5	46.5	80.05	5.73				
6					1086.5	33.3	92.82	10.42				
7	21.90	0.25	1214.0	919.4	2425.0	98.8	478.44	41.60	151.5	23.75	174.5	40.25
8					2516.0	66.0	508.58	58.86				
9					2584.5	81.8	536.99	60.21				
10	21.88	0.13	1260.0	1005.3	2744.5	135.8	647.85	125.91	178.5	38.25	221.0	46.25
11					2780.0	130	637.26	141.89				
12					2801.5	134.8	669.69	100.43				

Table 1: Results of tasks A-D for 30 (90) independent simulation runs of one order book of 5,000 items (OB_1000).

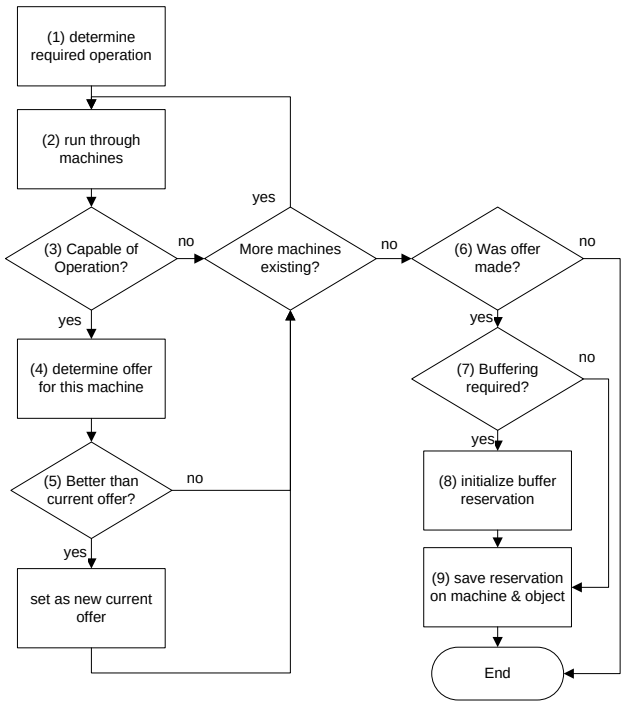

Figure 4. Reservation process.

The reservation process itself works as shown in Figure 4. It is distributed among the three user defined functions *Reserve_WS*, *Reserve_BU* (for buffer reservations) and *Cancel_Res* which are called directly from the *Check_Item* procedure. At the beginning of the process, the required process step is determined and all machines are considered to find the best possible offer for this item and operation. In this algorithm, the offer with the earliest possible start is selected. No optimization rules like “earliest due date” or “total weighted tardiness” are used.

Each item tries individually to find the best offer for itself. If an offer can be made it is checked if the item can be sent directly to the WS or if buffering is required. In the latter case, the function *Reserve_BU* is started, and in case a buffer place was found, the reservations are stored on the item and on the corresponding buffer or WS. The ID of the buffer or WS that has been reserved is then used as target ID for this item.

Although an even more heterarchical way of decision making would have been preferable, the workaround with the DPs had to be made because flowitems cannot start logic on their own in *Flexsim* but only activate triggers in other fixed resources.

Other important parts of the control algorithm are the user defined functions which are responsible for the reservation of time slots for tool replacement and operation changes. These functions are called directly from the workstations when the tool abrasion limit is undercut or a certain number of operation change requests is reached.

By making reservations for these changeover times, it is possible to already plan normal item reservations after a tool or operation change. With this no reservation has to be cancelled because of tool abrasion or a required operation change.

4 Results

According to [1] results are presented for the four criteria makespan (Task A), throughput time (Task B), tardiness (Task C) and number of violated constraints (Task D) by use of the statistics median and interquartile range (IQR). After validation of model and algorithm and finalizing all parameters, simulation runs with three order books were made.

The order books OB1000, OB1001 and OB1002, each consisting of a set of 5,000 orders, were tested witch 30 independent simulation runs per order book. Results for OB1000 are presented in Table 1. Results for the other two order books can be found in Appendix A.

Because the control algorithm does not account any due dates and does not make any reservations before an order's release date, it works exactly the same in all three operational scenarios. Due to this the results (except for tardiness) could be aggregated for each of the four manufacturing scenarios. The median and IQR for tasks A, B and D, are hence based on 90 independent values.

For PSc#1 (deterministic, scenarios 1-3) the control algorithm provides deterministic results, as expected. Only the throughput time has a positive IQR because all product types are mapped in one value. The results for all Tasks show the biggest leap from PSc#2 (some disruptions, scenarios 4-6) to PSc#3 (many disruptions, scenarios 7-9). The additional disruptions of the transport system in PSc#4 (scenarios 10-12) have in contrast rather moderate impact on the values.

References

[1] S. Schreiber, A. Fay. *ARGESIM Benchmark C20 'Complex Production System' – Definition and Call.* SNE Simulation Notes Europe, vol. 21 (3-4), 2011. doi: 10.11128/sne.21.bn20.10095

[2] *Flexsim*. http://www.flexsim.com/products/flexsim.

[3] M. Beaverstock, A. Greenwood, E. Lavery, W. Nordgren. *Applying Simulation with Flexsim*. Orem, Utah, USA: Flexsim Simulation Software, 2010.

Remarks. This work is based on the first author's master thesis *"Implementation and validation of a benchmark for a production system by use of a heterarchical control architecture"* (in German), written at and accepted by the Helmut-Schmidt-University, Institute of Automation Techn. 2011.

Submitted: February 10, 2012
Accepted: March 15, 2012

Appendix A - Further Results

	Task A		Task B		Task C				Task D		Task D	
	makespan[h]		throughput time [s]		No of tardy jobs		total weighted tard. [h]		No of NOK-1		No of NOK-2	
Sc.	Median	IQR	Median	IQR	Median	IQR	Median	IQR	Median	IQR	Median	IQR
1	21.20	0.00	942.4	468.9	725	0.0	31.18	0.00	35.0	0.00	18.0	0.00
2					785	0.0	35.72	0.00				
3					863	0.0	44.20	0.00				
4	21.23	0.14	993.5	524.0	1069.5	45.3	88.41	11.28	39.0	13.25	28.0	9.25
5					1115.5	43.3	96.83	8.39				
6					1210.5	34.3	104.19	15.75				
7	22.19	0.64	1204.0	925.3	2532.5	52.0	715.56	115.38	163.5	17.25	180.5	20.75
8					2570.5	66.0	703.60	83.58				
9					2624.5	44.0	731.19	135.36				
10	22.49	0.99	1247.0	993.9	2721.0	122.3	876.86	203.88	178.0	18.50	205.0	34.5
11					2779.5	88.5	865.58	162.33				
12					2859.5	121.3	983.91	199.74				

Table 2: Results of tasks A-D for 30 (90) independent simulation runs of one order book of 5,000 items (OB_1001).

	Task A		Task B		Task C				Task D		Task D	
	makespan[h]		throughput time [s]		No of tardy jobs		total weighted tard. [h]		No of NOK-1		No of NOK-2	
Sc.	Median	IQR	Median	IQR	Median	IQR	Median	IQR	Median	IQR	Median	IQR
1	20.98	0.00	908.6	429.4	676.0	0.00	30.04	0.00	21.0	0.0	14.0	0.00
2					721.0	0.00	32.55	0.00				
3					801.0	0.00	38.61	0.00				
4	20.91	0.12	990.2	510.0	1054.5	44.0	80.65	6.94	30.0	15.5	32.5	7.50
5					1106.5	28.8	87.25	5.99				
6					1184.0	43.5	96.53	10.78				
7	21.24	0.25	1255.0	986.6	2768.5	115.3	631.56	106.99	184.5	25.0	248.5	49.25
8					2839.0	91.8	648.38	86.52				
9					2883.0	109.3	669.96	141.72				
10	21.23	0.32	1306.0	1085.2	3064.5	198.3	829.57	222.02	221.0	50.0	295.0	68.25
11					3115.	277.3	849.98	437.49				
12					3197.0	176.8	1065.43	421.21				

Table 3: Results of tasks A-D for 30 (90) independent simulation runs of one order book of 5,000 items (OB_1002).