Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

Controlled Simplification of Material Flow Simulation Models

Daniel Huber
Wilhelm Dangelmaier

Business Computing, esp. CIM
Heinz Nixdorf Institut
Fürstenallee 11
Paderborn, 33102, Germany

ABSTRACT

In this paper a method for controlled simplification is presented, which is able to create simplified models with specific properties concerning complexity and behavioral deviation automatically. The method requires a finite set of model component classes, of which instances a user-defined model can be created. Two techniques for simplification are used: aggregation, where a large set of components is substituted by a small set, and omission, where components are deleted without compensation. A set of simplification rules concerning the simplification techniques, the component classes and the model structures *line* and *parallel line* are defined. These rules are used by a simplification algorithm, which is embedded in a control loop of complexity measurement and behavior measurement.

1 INTRODUCTION

Models in material flow simulation, which is a application of discrete event simulation, are of growing size and level of detail (Chwif, Pereira Barretto, and Paul 2000). The computational resources necessary to simulate such models are high and even on modern computers, the runtime is long. Experimenting with models of long runtime is inefficient, especially if many different system configurations shall be analyzed. To reduce the runtime of simulations the amount of used computational resources can be increased, e.g. by parallelizing the simulation, or the complexity of the simulation model can be reduced. The complexity of a discrete event simulation model is defined as the measure, that reflects the requirements in runtime and computational resources to simulate it for a certain amount of simulation time (Schruben and Yücesan 1993). The process of reducing the complexity shall be called simplification and the result of it a simplified model. In the literature the term detail is often used synonymously for complexity and the terms reduction and abstraction for simplification. All these synonyms often imply different ”flavors” of the topic, mostly without the strong focus on simulation runtime.

One major character of model simplification is, that it always comes along with a change in model behavior. The deviation in model behavior between the simplified model and the original model is increasing with the extend of simplification (Hung and Leachman 1999, Johnson, Fowler, and Mackulak 2005). Thus it can be reasonable for certain applications not creating the simplest model possible, but to use a stepwise method and create a model with a desired combination of complexity and behavioral deviation. These are applications, where the deviation should not extend a certain level in order to create valid output data for a certain experimental frame. One of these applications is the dynamic, multiresolution modeling created by Dangelmaier and Mueck (2004).

In this paper a method to create simplified models with specific properties concerning complexity and behavioral deviation by controlled simplification is proposed. The simplification is embedded in a control loop, in which the complexity and the behavioral deviation is measured, compared with the desired values and the simplification is controlled to create a model with desired properties. To do so an algorithm for controlling the simplification was created, as well as a complexity metric. The simplification is accomplished by using a set of simplification rules, also created for this work, which is a part of a not yet published dissertation, hence some parts in this paper are shortened.

This paper is organized as follows: At first the related work in the fields of model simplification and complexity metrics will be presented. Followed by a general survey of the method for controlled simplification and a detailed presentation of all its components. The paper concludes with experimental results and a conclusion including further research planned.

2 RELATED WORK

2.1 Model Simplification

The related work in the field of model simplification can be separated in two parts: Definitions of general aims and methods, and the implementation of methods to achieve these aims. General aims and methods are for instance presented by Zeigler et al. (2000) and Frantz (1995). Zeigler suggests the methods aggregation, omission, linearization, deterministic / stochastic variable replacement and formalism transformation. Frantz created a comprehensive taxonomy of simplification techniques, which includes all of Zeiglers methods.

Most implementations use the methods aggregation and formalism transformation. The latter is better known as simulation metamodelling and shall not be discussed further in this paper, since it is not possible to create a stepwise simplification using these techniques (Völker and Gmilkowsky 2003). Aggregation is used by Brooks and Tobias (2000), Johnson et al. (2005), Rose (2000), Rose (2007) and Hung and Leachman (1999). All these works use a quite similar approach to simplify models of production systems. In a nutshell: Identify the bottlenecks in the model, keep these unchanged and aggregate all other entities in the model into delays and buffers. Hung and Leachman and Johnson et al. identify the bottlenecks in the system by running a simulation of the original model and evaluating the utilizations of machines. Brooks and Tobias use the key figure effective capacity, presented later on in (3), to identify bottlenecks. With this equation bottlenecks can be identified without simulation, but by mere model analysis. When using machine utilization or effective capacity, machines in the model can be sorted by value and a set of strongest bottlenecks can be defined, which must not be aggregated. Johnson et al. point out, that the correct identification of bottlenecks is crucial for keeping behavioral deviation low. Additionally, the larger the set of preserved bottlenecks the lower the deviation. As Hung and Leachman present, the runtime of simulation is reduced by aggregation, because the number of events per simulation time is reduced. The number of events is reduced, because the model consists of less - and not more complex - components. An example for the impact of simplification: They created a two stepped simplification with 19.6% respectively 8.9% runtime in relation to the original model with 0.73% repectively 5.24% behavioral deviation (measured by mean errors in cycle times).

2.2 Complexity Metrics

Wallace (1987) and Schruben and Yücesan (1993) developed complexity metrics for simulation models represented by graphs. Wallace distinguishes between transformation complexity and control complexity. The first is measured by the weighted number of variables with read, write and read/write access for each node (weights: 1,1.5,2). The latter is a quantifier for each node, calculated as quotient of node degree and total number of nodes. Schruben and Yücesan developed several metrics, one is the sum of node complexity (measured by the number of state variables of the node) and out degree of the node. Another metric is based on the cyclomatic number of the graph, thus based on cycles in the graph created by adding a super source and -sink. In tests they showed that latter metric could not compete with the first metric, regarding correlation between metric and runtime.

3 GENERAL SURVEY

The developed method of controlled simplification works on a graph-like model, where nodes represent components like machines, buffers, conveyor belts etc. and edges represent the logical links between components. Token are representing jobs and are handed via these links from one component to the other. Each component is an instance of a specific component class. To allow automatic model analysis for simplification, a finite set of component classes must be defined.

Model simplification is done by using the methods aggregation and omission. To optimize the overall runtime of the bundle of simplification and the later simulation, no data obtained by simulation runs is used, but only information stored directly in the model. Thus changes in the configuration of the original model or the production program can be easily inserted into the simplification. To identify the systems bottlenecks, an enhanced version of the effective capacity of Brooks and Tobias is used.

The aggregation of model components is defined by a set of aggregation rules, each relevant for certain sets of components in certain model structures. The aggregation rules define how sets of components can be substituted by smaller sets of components with only a slight change in model behavior. Rules for omission are also defined by a set of rules, defining the significance of components for model behavior. If the significance is judged to be small, components can be excluded from the model.

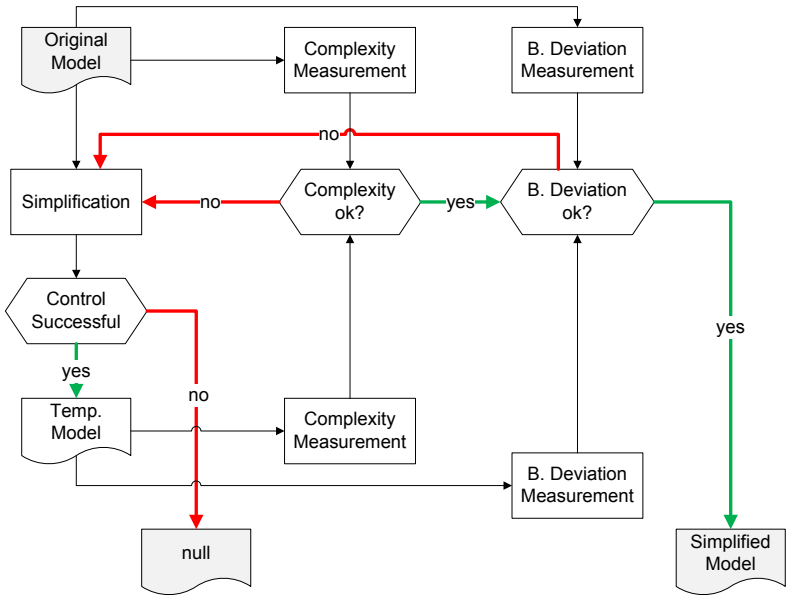

Figure 1: Simplification Control Loop

The simplification is embedded in a control loop shown in Figure 1. The complexity of the temporary model is calculated after each simplification iteration and compared to the complexity of the original model. If the desired complexity is reached ($\mathscr{C} \leq \mathscr{C}_{Target}$) the behavioral deviation is measured by executing a set of simulation runs and comparing certain production key figures (Work in Process, Cycle Time and Throughput) over time. This is done by calculating the area between the trajectories of the original model and the simplification for each key figure. If the deviation is also sufficiently small the simplified model is created. If the complexity or deviation are not sufficient, a new iteration of the simplification is started, controlled to reduce complexity or deviation. If control is not successful, i.e. the properties of the temporary model could not be changed, no simplified model is created.

The complexity is calculated with a combined approach of Wallace and Schruben and Yücesan, which has only a very short runtime. The measurement of the behavioral deviation has a very long runtime, because simulation runs are necessary, at least one set (consisting of three or more simulations) for the original model and one set for the simplification. By using a two level control loop, the number of simulations necessary can be kept as small as possible. If the target complexity cannot be achieved, no simulation is necessary. If the simplified model shows a sufficiently small behavioral deviation after simulating, two sets of simulation runs had to be executed therefor, one for the original and one for the simplification. If the deviation is too high, one additional set of simulations of the simplification has to be executed for each attempt to reduce the deviation, because the simplified model is changed in every attempt.

4 MODEL COMPONENTS

Now the finite set of components shall be defined. This set was used to create and evaluate the methods, but the set can be extended, if the rules of simplification are as well.

- Source
  Sources create tokens and send them to linked successor separated by a time interval. The tokens can be of various types tt with a defined ratio per type. The time interval can be deterministic or a random distribution.
- Sink
  Sinks terminate token.
- Machine (M)
  Machines delay token on their path through the model and can change their type. The delay time is defined by the service time t_c and the setup time t_s, both deterministic or a random distributions. Machines can operate in batch

mode, where processing begins after a number of ls token have arrived in the machine. Machines can also break down, defined by $mtbf$ and $mttr$, both deterministic or a random distributions.

- Assembly Machine
  In difference to normal machines, which only have one input and one output channel, assembly machines have two or more input channels. Processing begins after a number of $ls(Input)$ token arrived in each input channel. There is one main input channel and token arriving in this channel leave the assembly machine after processing, all other token are terminated.
- Buffer FiFo (BF)
  Buffer with size bs, working with the first in first out method.
- Buffer LiFo (BL)
  Buffer with size bs, working with the last in first out method.
- Sequencer (SQ)
  Buffer with size bs, which sorts token by type. If there is a sequence of one type of desired length, all token of this sequence are send to the successor.
- Conveyor Belt (CB)
  Buffer with size bs, working with the first in first out method. Additionally to the Buffer FiFo, token can only leave after they waited for the delay t_c.
- Distributer (D)
  Distributers have one input channel and several output channels. Each output channel has a probability with which it is chosen, to define the successor a specific token is send to.
- Joint (J)
  Joints have several input channel. All token leave this entity over one output channel.

5 COMPLEXITY METRIC

The complexity of a model is defined by its components and the links in the model. Because all components are instances of the component classes, the complexity of the components of a model can be calculated using the complexity of the component classes $\mathscr{C}_v$. $\mathscr{C}_v$ is the adaption of Wallace's metric and calculated with (1),

$$\mathscr{C}_v = \mathrm{EppT} \cdot \frac{\sum \mathrm{CoV}}{10} \tag{1}$$

where $EppT$ is the number of events defined in a component class to get a token from the input channel to the output channel. $EppT$ is equivalent to Wallace's both complexities – transformation and control – and is used because Wallace is working on event-like objects rather than components. CoV is the complexity of the variables used in a component class, similar to the transformation complexity of Wallace. Strings and random variables are weighted with a factor of 1.5 and arrays with a factor of 2 in relation to basic deterministic variables. The data type of variables, in contrast to the access type, was chosen, because the type a variable has a much higher impact of the access complexity than the access type. Because the number of events, and thus the event routines in which the variables are used, has a more important part in component class complexity (cf. section 2.1) than the sum of weighted number of variables, $\sum \mathrm{CoV}$ is divided by 10 (which was chosen arbitrarily) to reduce the impact of the term. The $\mathscr{C}_v$ of the used component classes are presented in Table 1.

The links in the model are taken into account by using the cycles in the model graph created when adding and connecting a super source and super sink, as Schruben and Yücesan do. The basic idea for using cycles is: Components being in more cycles than other are more likely higher frequented by token, thus they schedule and execute more events. Example: A *line* of components is connected with a distributer to which three *parallel lines* and a closing joint are connected (Figure 2 top). There are three cycles in the model ($Z_{blc} = 3$). Every component in the single line is on three cycles $Z_v = 3$, every component in the parallel lines is on one cycle $Z_v = 1$. If the distributer distributes equally, the number of token and thus the number of events executed by components in the single line is three times higher than of components in the parallel lines. The components in the single line have a higher influence on complexity than the components in the parallel lines (cf. Section 2.1). Hence the complexity of a model is defined by (2).

$$\mathscr{C} = \sum_{v \in V} \mathscr{C}_v(v) \cdot \frac{Z_v(v)}{Z_{blc}(v)} \tag{2}$$

Table 1: Used $\mathscr{C}_v$

Component class	$\mathscr{C}_v$
Source	3.0
Sink	1.4
Machine	3.4
Assembly Machine	7.5
Buffer FiFo	2.7
Buffer LiFo	2.7
Sequencer	5.1
Conveyor Belt	4.0
Distributer	3.8
Joint	2.2

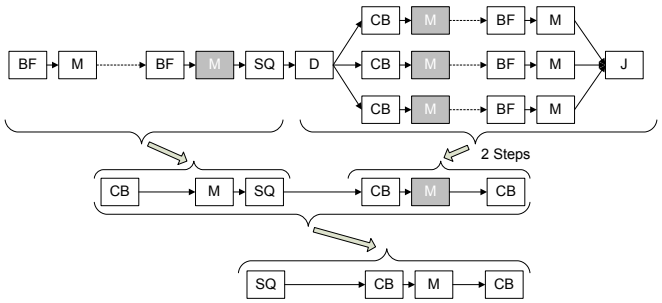

Figure 2: Simplification of a Line and a Parallel Line

An additional problem arises in more complex models, if assembly machines are used. If the line connected to the main input channel is on significantly more cycles than those lines for add-on token (which is common for realistic models), the importance of add-on lines is underrated. This becomes particularly important for the bottleneck calculation presented in the next section. Since add-on tokens are terminated in assembly machines, their material flow ends there and thus their flow can be considered independently from the main flow, concerning cycle calculation. For cycle calculation the assembly-machine-nodes are split to separate the main flow from the add-on flow. The part of the assembly machine still connected to the add-on flow is now considered a sink. By doing so, the model graph is not completely connected without the super source and super sink and blocks are created. Z_{blc} is the number of cycles in one block.

6 SIMPLIFICATION RULES

The effective capacity of Brooks and Tobias to identify the bottlenecks in the system, presented in (3), shall be adapted to this work and improved by using cycles in the model. Where ls is the batch size, t_c is the cycle time, *sui* is the set-up interval, t_s is the set-up time, $mtbf$ is the mean time between failure, $mttr$ is the mean time to repair and gp is the quotient of good parts.

$$C_{effective} = \frac{ls}{t_c} \cdot \frac{sui}{t_s + sui} \cdot \frac{mtbf}{mttr + mtbf} \cdot gp \tag{3}$$

Because the production program is unknown in this work, the set-up interval is unknown and additionally gp is not defined in the set of component classes. Based on the defined component classes, machines (V_M) and conveyor belts(V_{CB}) can be bottlenecks. The effective capacity for these components is calculated with (7) and (8). $C_{eff,cyc}$ (6) is the effective capacity weighted with the cycle quotient $\frac{Z_v(v)}{Z_{blc}(v)}$ defined above. For better comparability $C_{eff,rel}$ is introduced in (5) by normalizing the $C_{eff,cyc}$ in the interval $[0,1]$. Concluding the set of bottlenecks is defined as the set of components with the highest

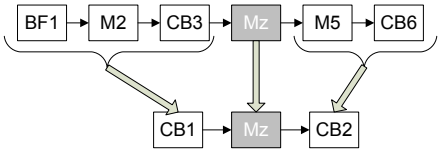

Figure 3: Simplification of a Line

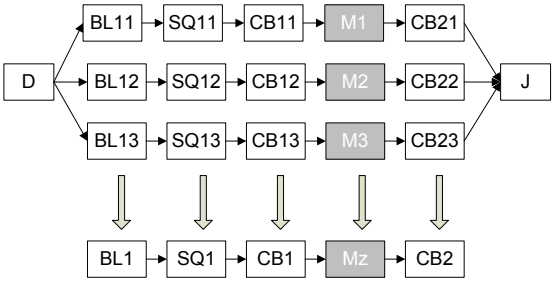

Figure 4: Simplification of a Parallel Line

$C_{eff,rel}$ (4).

$$V_{bn} = \left\{ v \in (V) \mid C_{eff,rel}(v) = max_{\forall w \in V}(C_{eff,rel}(w)) \right\} \tag{4}$$

$$0 \leq C_{eff,rel}(v) = \frac{min_{\forall w \in V}(C_{eff,cyc}(w))}{C_{eff,cyc}(v)} \leq 1 \tag{5}$$

$$C_{eff,cyc}(v) = \frac{C_{eff}(v)}{\frac{Z_v(v)}{Z_{blc}(v)}} \tag{6}$$

$$C_{eff}(v) = \frac{ls(v_M)}{t_c(v_M) + t_s(v_M)} \cdot \frac{mtbf(v_M)}{mttr(v_M) + mtbf(v_M)} \tag{7}$$

$$C_{eff}(v) = \frac{bs(v_{CB})}{t_c(v_{CB})} \tag{8}$$

By normalizing the global bottlenecks in the model have $C_{eff,rel}(v) = 1$, and local bottlenecks can be defined as components with the highest $C_{eff,rel}(v)$ in a given set of components.

The simplification rules are based on structures, i.e. model parts with a specific link structure, and component classes. The structures used in our work are lines and parallel lines (Figure 2). Lines are built of components with one input channel and one output channel linked sequentially. Parallel lines are similar lines between one distributer and one joint. A model does not need to consist only of these two structures, but only components in structures can be simplified by aggregation.

Following, two aggregation rules shall be presented with which most structures can be simplified (The complete work contains several more, but are only used in special cases or can be derived from the following.). All simplification rules in aggregation are based on the work of Brooks and Tobias, as well as on the general idea of Hung and Leachman and Rose, etc.

Let there be a line with serially numbered components 1 to n of the classes Machine, Buffer FiFo and Conveyor Belt (see Figure 3). The bottleneck machine Mz in the line is identified with (4) and $1 \leq z \leq n$. Then an aggregation is possible by using the rule in Table 2. The parameters mtbf and mttr are changed for the bottleneck machine, as is the description for exiting token *tt*, such that token leaving the simplified line have the same description as the tokens in the original line. The breakdown parameters are changed to assure, that the simplified structure has the same availability as the original structure. This is done by applying equations of reliability analysis (Birolini 1991), which shall not be discussed in this paper. The cycle times and delays of all machines and conveyor belts in front of and behind the bottleneck are compensated by the delays of the conveyor belts CB1 and CB2. The buffer sizes *bs* of buffers and conveyor belts, as well as batch size buffers of substituted machines *ls* are also compensated by CB1 and CB2. In summary: CB1 substitutes all components in front of the bottleneck, CB2 all behind the bottleneck. The compensation of the set-up time is an optional term, which can be omitted by the control algorithm, to influence behavioral deviation if the production program is quite homogeneous and set-ups are rare. By preserving the bottleneck machine, the stochastic character is preserved, because never all components with random variables are removed from the model. Even if a conveyor belt is the bottleneck in a line, an additional machine with the next bigger $C_{eff,rel}$ is preserved to keep the stochastic character of the model.

This rule is designed to minimize the deviation concerning the production key figures work in process, cycle time and throughput. All delaying times are preserved in the simplification as are the buffer capacities. Waiting times, as a part of

Table 2: Aggregation of mixed lines

Org. Comp.	Component Parameter	Simpl. Comp.	Aggregated Component Parameter
Mi	$t_c(\cdot)$	Mz	$t_c = t_c(M_{bn})$
Mi	$ls(\cdot)$	Mz	$ls = ls(M_{bn})$
Mi	$t_s(\cdot)$	Mz	$t_s = t_s(M_{bn})$
Mi	$mtbf(\cdot)$	Mz	$mtbf = mtbf_{eff}$
Mi	$mttr(\cdot)$	Mz	$mttr = mttr_{eff}$
Mi	$tt(\cdot)$	Mz	$tt = tt(Mi_{max})$
BFi	$bs(\cdot)$	CB1	$t_c = \sum_{i=1..z-1} t_c(Mi) + \sum_{i=1..z-1} t_c(CBi) \,[+\sum_{i=1..z-1} t_s(Mi)]$
CBi	$t_c(\cdot)$	CB1	$bs = \sum_{i=1..z-1} ls(Mi) + \sum_{i=1..z-1} bs(CBi) + \sum_{i=1..z-1} bs(BFi)$
CBi	$bs(\cdot)$	CB2	$t_c = \sum_{i=z+1..n} t_c(Mi) + \sum_{i=z+1..n} t_c(CBi) \,[+\sum_{i=z+1..n} t_s(Mi)]$
		CB2	$bs = \sum_{i=z+1..n} ls(Mi) + \sum_{i=z+1..n} bs(CBi) + \sum_{i=z+1..n} bs(BFi)$

Table 3: Aggregation of homogeneous Parallel Lines

Org. Comp.	Component Parameter	Simpl. Comp.	Aggregated Component Parameter
D1	$bs = 1$	BL1	$bs = \sum_{\forall BL1r} bs(\cdot)$
Jn	$bs = 1$	BL2	$bs = \sum_{\forall BL2r} bs(\cdot)$
BL1r	$bs(\cdot)$	SQ1	$bs = \sum_{\forall SQ1r} bs(\cdot)$
BL2r	$bs(\cdot)$	SQ2	$bs = \sum_{\forall SQ2r} bs(\cdot)$
SQ1r, SQ2r	$bs(\cdot)$	CB1	$bs = \sum_{\forall CB1r} bs(\cdot)$
CB1r, CB2r	$bs(\cdot)$	CB1	$t_c = \overline{t_c}(CB1r)$
CB1r, CB2r	$t_c(\cdot)$	CB2	$bs = \sum_{\forall CB2r} bs(\cdot)$
Mr	$t_c(\cdot)$	CB2	$t_c = \overline{t_c}(CB2r)$
Mr	$ls(\cdot)$	M	$t_c = \frac{1}{m}\overline{t_c}(Mr)$
Mr	$t_s(\cdot)$	M	$ls = \overline{ls(Mr)}$
Mr	$mtbf(\cdot)$	M	$t_s = \frac{1}{m}\overline{t_s}(Mr)$
Mr	$mttr(\cdot)$	M	$mtbf = mtbf_{eff}$
Mr	$tt(\cdot)$	M	$mttr = mttr_{eff}$
		M	$tt = tt(Mi_{max})$

the cycle time, which are not explicitly contained in the model, are preserved by preserving the bottleneck and the buffer capacities. But waiting times are expected to be a major factor in behavioral deviation.

With the full set of aggregation rules, all lines can be simplified to the following minimal line:

BL1 - SQ1 - CB1 - CBy - Mz - CB2 - BL2 - SQ2

Buffer LiFo and Sequencer cannot be substituted, because they re-sort the material flow. Because a line is part of a larger model with preceding and successive lines or parallel lines, Buffer LiFo and Sequencers positioned in front of or behind the bottleneck must be preserved separately. If the bottleneck in a line is a conveyor belt, it (CBy) must also be preserved.

Let there be a parallel line consisting of several similar minimal lines, a distributer and a joint with (see Figure 4). Such a structure with $r = 1..m$ parallel lines can be aggregated by using the rule in Table 3.

By using this rule the parallel lines become a single line with similar behavior, thus the distributer and the joint can be deleted. All buffer sizes of buffers, sequencers and conveyor belts are cumulated over the lines.

There are two possibilities to handle machines: Batch sizes can be cumulated and for the cycle and set-up time the mean is used; or the resulting batch size is the mean and resulting cycle and set-up times are the means divided by m. The drawback of the second possibility is the loss of buffer size, because the batch buffers of $m-1$ machines are lost. The second possibility was chosen despite the drawback, because the risk of creating instabilities in the simplification is lower. The buffers succeeding the parallel line in the original model are sized for batch sizes of $\overline{ls(Mr)}$, but in possibility one the

batch size in the simplified model is $\sum_{r=1..m} ls(Mr)$. This can result in blocking and starvation of machines in the simplified model, where there is no such behavior in the original model.

Additionally to aggregation, omission is used for simplification. Zeigler and Frantz mention omission as a method to simplify models, but recent implementations could not be found. Components can be deleted from the model, if their influence in model behavior can be regarded as small. Components, which re-sort material flow or junctions in the material flow like distributers, joints and assembly machines cannot be deleted. The change in model behavior would be significant or worse the model structure could be corrupt. An indicator $0 \leq q(v) \leq 1$ is used to evaluate the influence in model behavior. If $q(v) \leq q_{Thres}$, the influence is estimated to be small and the component v can be deleted.

One major use of buffers in manufacturing systems is to decouple machines. When buffers of adequate size are put between machines, blocking and starvation of machines can be reduced. Thus a buffer can be deleted, if blocking and starvation in the simplified model will not occur in an other amount than in the original model. If we assume, with no loss in generality, that every buffer is placed exactly between two machines, which have randomly distributed cycle and set-up times, no buffer could be deleted. The only possibility is, that there are other components with buffer-character, like conveyor belts or sequencer, between the same two machines the buffer is. To decide if the buffer can be deleted, the buffer size necessary to prevent blocking and starvation of the two relevant machines have to calculated. This is a quite difficult problem, when using the defined component classes, thus the simple approximation in (9) shall be used. It states that a buffer can be deleted, if the cumulated buffer size of all components with buffer character V_{BM} minus the buffer size of the buffer in question is greater than the maximum of the batch sizes of the relevant two machines ls_{max} (10).

$$q_{BF}(v) = \begin{cases} 0, & \text{if } \left(\sum_{V_{BM}} bs(\cdot)\right) - bs(v) \geq ls_{max} \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

$$ls_{max} = max\left(ls(M1), ls(M2)\right) \tag{10}$$

$$q_M(v) = max\left(C_{eff,rel}(v), \frac{t_c(v) + t_s(v)}{t_{CT}(v)}\right) \tag{11}$$

$$q_{CB}(v) = max\left(q_{BF}(v), q_M(v)\right) \tag{12}$$

$$q_D(ch) = \frac{\text{Probability for ch}}{\sum_{Ch(v)} \text{Probability for ch}} \tag{13}$$

Machines can be deleted, if they are not bottlenecks nor close to it, $C_{eff,rel}(v) \leq q_{Thres}$, and if their influence on cycle times is small (11). To calculate the influence on cycle times, $t_{CT}(v)$ is used, as the cycle with the minimum cumulated mean delay times the component is on. Concerning omission, conveyor belts combine the properties of buffers and machines, thus all three defined criteria must be considered (12). One line in a parallel line structure can be deleted, if the probability that this line, i.e. channel, is chosen is very low (13).

7 CONTROLLED SIMPLIFICATION

The simplification rules and the complexity and behavioral deviation metrics are integrated into the simplification control loop shown in Figure 1. The intention for using the control loop is to achieve a complexity-behavior-trajectory similar to the ideal shown in Figure 5. In the first iterations of simplification the complexity can be reduced without much behavioral deviation, but the number of these simplifications is limited. To achieve further complexity reduction an increasing amount of deviation must be allowed. The simplification is continued until the desired complexity is reached or no further simplification rules can be applied. In the first case the behavioral deviation is measured and if it is lower than the threshold the simplification is created.

Algorithm 1 shows the control loop in detail. The functions ANALYZEAGGREGATION (Lines 8 and 12) and ANALYZEOMISSION (Lines 15 and 19) as well as the procedures SYNTHESISAGGREGATION (Line 10) and SYNTHESISOMISSION (Line 17) are straight forward methods to find structures or components to simplify (Analysis) and apply rules and modify the model (Synthesis).

Algorithm 1 (Simplification Control Loop)**.**

Require: $\mathscr{C}_{Target}, \mathscr{V}_{Target}, \vartheta_{BN}, \vartheta_q$

1: *Calculate* $\mathscr{C}$
2: $q_{Thres} = \vartheta_q$
3: *List* $L_C = \{C_{eff,rel}(v)\}$

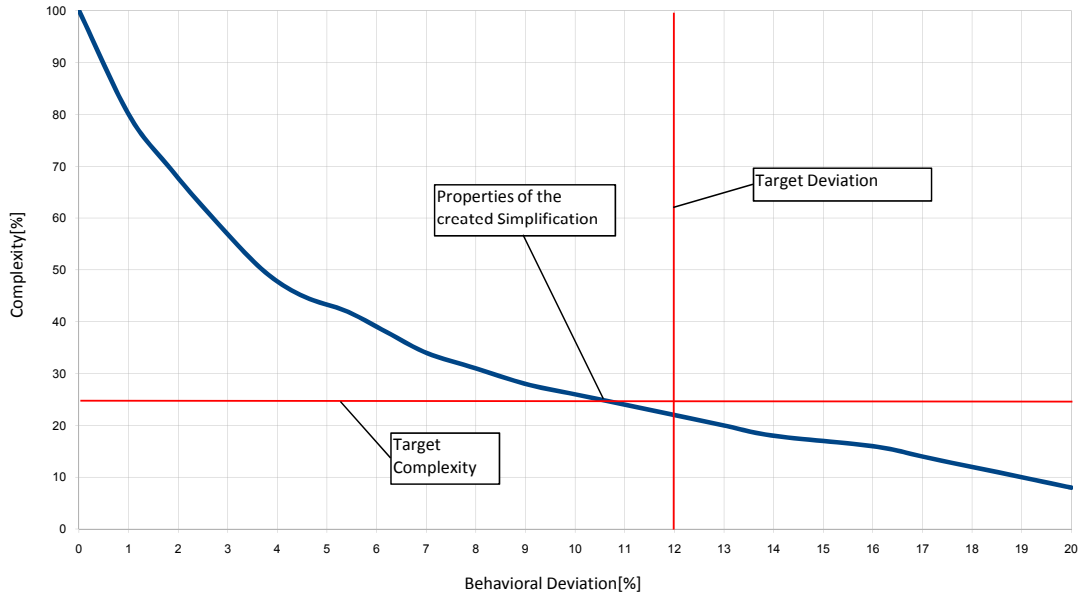

Figure 5: Complexity and Behavioral Deviation Trajectory of Simplification (Ideal)

4: L_C.sort(downward)
5: List $L_{BN} = \{v | L_C.indexOf(C_{eff,rel}(v)) \leq L_C.length \cdot \vartheta_{BN}\}$
6: L_{BN}.sort(downward)
7: **while** $L_{BN}.length > 1$ **do**
8: Structure s = ANALYZEAGGREGATION(L_{BN})
9: **while** $\mathscr{C} > \mathscr{C}_{Target}$ and $s \neq null$ **do**
10: SYNTHESISAGGREGATION(s, L_C)
11: Calculate $\mathscr{C}$
12: s = ANALYZEAGGREGATION(L_{BN})
13: $L_C = \{C_{eff,rel}(v)\}$
14: **end while**
15: Component k = ANALYZEOMISSION(q_{Thres})
16: **while** $\mathscr{C} > \mathscr{C}_{Target}$ and $k \neq null$ **do**
17: SYNTHESISOMISSION(k)
18: Calculate $\mathscr{C}$
19: k = ANALYZEOMISSION(q_{Thres})
20: **end while**
21: **if** $\mathscr{C} > \mathscr{C}_{Target}$ **then**
22: L_{BN}.remove(last element)
23: $q_{Thres} += \vartheta_q$
24: **end if**
25: **end while**
26: **if** $\mathscr{C} \leq \mathscr{C}_{Target}$ **then**
27: Calculate $\mathscr{V}$
28: **if** $\mathscr{V} \leq \mathscr{V}_{Target}$ **then**
29: Simplification successful
30: **else**
31: Switch off setup times consideration
32: Calculate $\mathscr{V}$
33: **if** $\mathscr{V} \leq \mathscr{V}_{Target}$ **then**
34: Simplification successful

```
35:         else
36:             Simplification not successful
37:         end if
38:     end if
39: else
40:     Simplification not successful
41: end if
```

The requirements for this algorithm are the target values for complexity $\mathscr{C}_{Target}$ and behavioral deviation $\mathscr{V}_{Target}$, as well as a bottleneck-preservation-factor ϑ_{BN} and a q-increment ϑ_q. ϑ_{BN} defines how many components are treated as preservable bottlenecks (Line 5, L_{BN} as the list of preservable bottlenecks). With a increasing number of preservable bottlenecks the behavioral deviation will decrease, but the possible maximum complexity reduction as well. q_{Thres}, the threshold for deletable components, is defined by ϑ_q. To allow higher complexity reduction and higher behavioral deviation L_{BN} is shortened (Line 22) and q_{Thres} is raised by ϑ_q (Line 23) if no further simplification is possible. The algorithm terminates, if the target complexity is reached or no further simplifications are possible. This is the case if all but one bottlenecks are removed from the list L_{BN}. If the complexity is reached, the behavioral deviation $\mathscr{V}$ is measured. If the deviation is not small enough, the compensation of set-up times can be switched of and the behavioral deviation $\mathscr{V}$ is measured once again. If this did not reduce the behavioral deviation below the target value, no simplified model is created.

8 RESULTS

The models, simulations as well as all algorithms and metrics presented in this paper were implemented by using the simulation tool d^{3}FACT insight (Dangelmaier, Huber, Laroque, and Mueck 2005), which uses a Java-based, discrete event simulation kernel. Models are stored in a XML-file. The computer used to generate data had a Intel E8400 CPU with 4GB RAM running Windows Vista and Java 1.5.

To test the developed methods three test-models where created, Model A, B and C. Model A is a big model with 151 components, containing medium long lines and many parallel lines and five assembly machines. Model B has 133 components, including six assembly machines, quite short lines and some parallel lines. Model C has 61 components and contains only three lines and two parallel lines.

The complexity metric in (2) was used to calculate the complexity of these three models and three simplifications each. Additionally the runtime of all these models was measured as well, whereas the computer used is unimportant, because only the proportions of the values are used. In Table 4 the results are presented for model A exemplary. The correlation coefficient

Table 4: Complexity Metric and Runtime Model A

Step	$\mathscr{C}$	runtime in s			
		M1	M2	M3	∅
11	317.6	135.0	131.0	130.0	132.0
10	275.5	113.0	106.0	106.0	108.3
9	259.0	107.1	103.3	101.9	104.1
8	245.2	97	95.1	100.1	97.4
7	208.9	77.1	77.4	78.0	77.5
6	200.1	76.3	75.8	75.8	76.0
5	197.6	80.2	78.1	79.3	79.2
4	186.6	78.6	75.5	74.5	76.2
3	181.6	73.3	70.3	68.7	70.8
2	166.9	64.3	64.3	64.9	64.5
1	162.9	67.2	65.1	62.6	65.0

Table 5: Quality of Bottleneck Identification

Model	Correlation Coefficient
A	0.63
B	0.04
C	1.00

of complexity and runtime is 0.99 for model A, 0.83 for model B and 0.98 for model C. These correlation coefficients are quite high, thus the complexity metric is able to predict the runtime of simplified models and can be used for controlled model simplification.

The three test-models where simplified with the controlled simplification in several steps. The diagram in Figure 6 shows the results of these simplifications. The simplification-trajectories are strongly dependent on the used model. Model C could be simplified very well with a remaining complexity of 20%. The measured curve has very similar characteristics

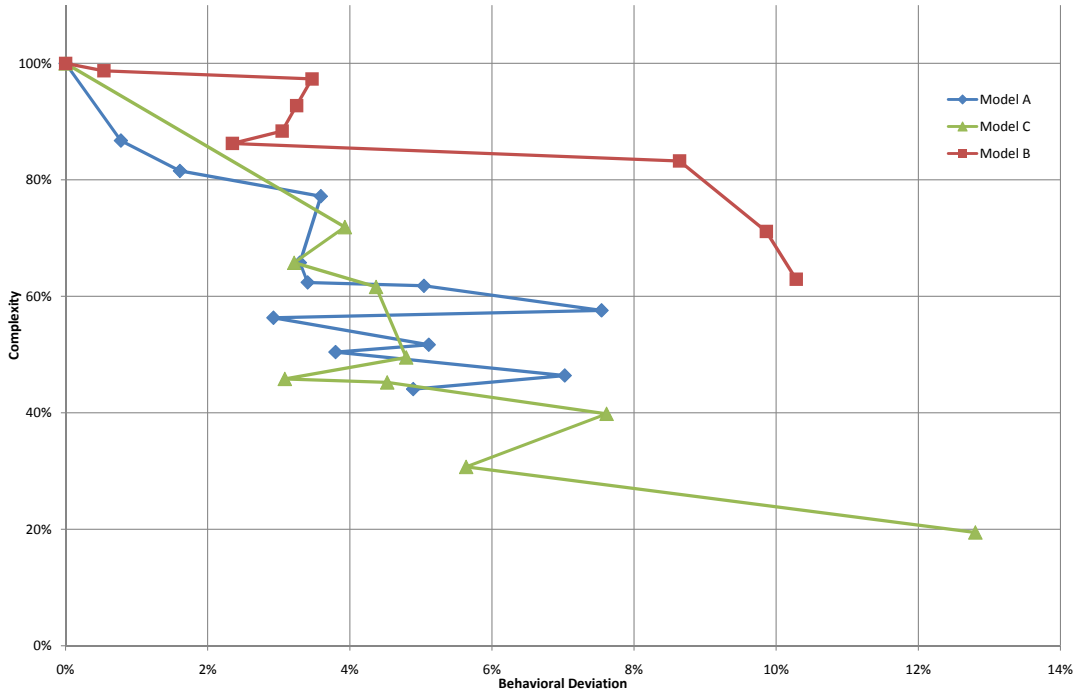

Figure 6: Simplification Trajectory (Real)

as the desired ideal in Figure 5. The remaining complexity of model A is 45% and the one of model B is 60%. The behavioral deviation of model A is 5%, model B 10% and model C 13%. The difference in maximum simplification is caused by the structure of the original model. If many assembly machines (model A) and many distributors or joints are used without creating parallel lines (model B), the maximal possible simplification is decreased. The highest simplification of model A shows a relatively small behavioral deviation, so further simplification seems possible, but was not, because for further simplification q_{Thres} would have to be raised to a level where the resulting model was not valid any more.

If the results are compared to other simplification attempts, like Hung and Leachman, they does not seem so good. Hung and Leachman created simplifications of 20% and 9% complexity with a error in cycle times of 0.7% and 5.2%. But without further knowledge of the model used by Hung and Leachman and considering the strong model dependency of the designed method in this paper and the difference in behavior measurement, a comparison is of limited significance. A remaining complexity of 20% in model C means, that of the 61 components in the original model only seven remain (1 source, 1 sink, 2 sequencer, 2 conveyor belts and 1 machine).

Additionally the quality of the bottleneck calculations in (4) was tested. To do so, all components of a model were sorted by its $C_{eff,rel}$ and numbered according to its position in this sorted list. Then the utilizations of all components were acquired by executing three simulation runs for each model. The components then were sorted and numbered by utilization. Then the correlation coefficients of these two lists per model were calculated (Table 5). There is a perfect positive correlation in model C and none in model B, model A is somewhere in between.

9 CONCLUSION

In this paper a method for controlled simplification was presented, which is able to create simplified models with specific properties concerning complexity and behavioral deviation automatically. The method requires a finite set of model component classes, of which instances a user-defined model can be created. A set of simplification rules concerning the component classes and the model structures line and parallel line was defined. These rules are used by a simplification algorithm which is embedded in a control loop of complexity measurement and behavior measurement. Results show, that the designed method works and good simplifications are created. To optimize results further research is necessary, especially to improve the bottleneck identification calculation and to fully understand and to decrease the strong model dependency of used methods.

For testing only three models were used; to get really significant results and more insight, more models have to be created and tested.

REFERENCES

Birolini, A. 1991. *Qualität und Zuverlässigkeit technischer Systeme*. Springer.

Brooks, R. J., and A. M. Tobias. 2000. Simplification in the simulation of manufacturing systems. *International Journal of Production Research* 38 (5): 1009–1027.

Chwif, L., M. R. Pereira Barretto, and R. J. Paul. 2000. On simulation model complexity. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 449–455. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Dangelmaier, W., D. Huber, C. Laroque, and B. Mueck. 2005. d^3fact insight - an immersive material flow simulator with multi-user support. In *Proceedings of the 2005 Summer Computer Simulation Conference*, 239–242: SCS Press.

Dangelmaier, W., and B. Mueck. 2004. Using dynamic multiresolution modelling to analyse large material flow system. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 1720–1727. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Frantz, F. K. 1995. A taxonomy of model abstraction techniques. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos and K. Kang, 1413–1420. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Hung, Y.-F., and R. C. Leachman. 1999. Reduced simulation models of wafer fabrication facilities. *International Journal of Production Research* 37 (12): 2685–2701.

Johnson, R. T., J. W. Fowler, and G. T. Mackulak. 2005. A discrete event simulation model simplification technique. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 2172–2176. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rose, O. 2000. Why do simple wafer fab models fail in certain scenarios? In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1481–1490. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rose, O. 2007. Improved simple simulation models for semiconductor wafer factories. In *Proceedings of the 2007 Winter Simulation Conference*, ed. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1708–1712. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Schruben, L., and E. Yücesan. 1993. Complexity of simulation models: a graph theoretic approach. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. Evans, M. Mollaghasemi, E. Russell, and W. Biles, 641–649. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Völker, S., and P. Gmilkowsky. 2003. Reduced discrete-event simulation models for medium-term production scheduling. *Systems Analysis Modelling Simulation* 43:867–883.

Wallace, J. C. 1987. The control and transformation metric: toward the measurement of simulation model complexity. In *Proceedings of the 1987 Winter Simulation Conference*, ed. L. Thesen, H. Grant, and W. D. Kelton, 597–603. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation*. 2nd Edition ed. Academic Press.

AUTHOR BIOGRAPHIES

DANIEL HUBER studied industrial engineering at the University of Paderborn, Germany. Since 2005 he is a research assistant at the group of Prof. Dangelmaier, Business Computing, esp. CIM at the HEINZ NIXDORF INSTITUT in Paderborn Germany. His main interests are material flow simulation, modeling methodology and automatic model simplification. His website is <http://wwwhni.upb.de/cim/> and his email address is <huber@upb.de>.

WILHELM DANGELMAIER was director and head of the Department for Cooperate Planning and Control at the Fraunhofer-Institute for Manufacturing. In 1990 he became Professor for Facility Planning and Production Scheduling at the University of Stuttgart. In 1991, Dr. Dangelmaier became Professor for Business Computing at the HEINZ NIXDORF INSTITUT; University of Paderborn, Germany. His website is <http://wwwhni.upb.de/cim/> and his email address is <whd@hni.upb.de>.