Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

A METHOD FOR SIMULATION STATE MAPPING BETWEEN DISCRETE EVENT MATERIAL FLOW MODELS OF DIFFERENT LEVEL OF DETAIL

Daniel Huber
Wilhelm Dangelmaier

Heinz Nixdorf Institut
Universität Paderborn
33102 Paderborn, Germany

ABSTRACT

In this paper a method is presented for mapping the simulation state between models of different level of detail in dynamic multiresolution modeling of discrete event material flow systems. In dynamic multiresolution modeling it is possible to inactivate a model part of certain detail and activate a model of higher or lower detail representing the same part of the system during simulation. To allow a consistent simulation, the state of the inactivated model has to be mapped to the activated model. The mapping is done by first mapping the jobs, then mapping the breakdown status and statistical variables. Experiments show that consistent states are generated, such that the simulation continues after state mapping. The errors generated by mapping are small and caused by the loss of information in the models of lower detail.

1 INTRODUCTION

Multiresolution modeling is the generation of a composed model by coupling highly detailed model parts with model parts of lower detail. All coupled model parts have to interact during simulation to enable a composed simulation of the whole system. In dynamic multiresolution modeling the composition of model parts can change during simulation. One part of the system modeled and simulated in low detail, for instance, gets more significant – by internal or external events or by user interaction – and thus the low detailed model is exchanged by a model of higher detail. One model gets activated – gets part of the composed model – and one model gets inactivated.

Generally multiresolution modeling is used when the scope and the level of detail – the computational complexity and thus simulation runtime – of a model is to high for an efficient simulation. Model parts which are not so important for the current experiment or simulation phase are simulated in low detail, the rest in high detail. Many works in multiresolution modeling are in military application, i.e. (Davis 1992, Davis 2000, Natrajan, Reynolds, and Srinivasan 1997), and only few are in manufacturing applications, i.e. (Dangelmaier and Mueck 2004).

The benefit of dynamic multiresolution modeling (DMRM) lies in adjusting the level of detail of the composed model – and with detail also simulation runtime – to the computational resources available. With dropping detail not only runtime decreases, but also the accuracy of the model is. Thus with DMRM not only runtime can be adjusted to the resources available, but also model accuracy. Using DMRM in a world with ever rising computing power is reasonable because synchronously, models become more detailed in the same order of magnitude.

When exchanging model parts in DMRM a severe problem arises because the state of the newly activated model must be defined. Namely in a way that the state of the composed model does not change significantly. Thus the state of the inactivated model has to be mapped to the activated model.

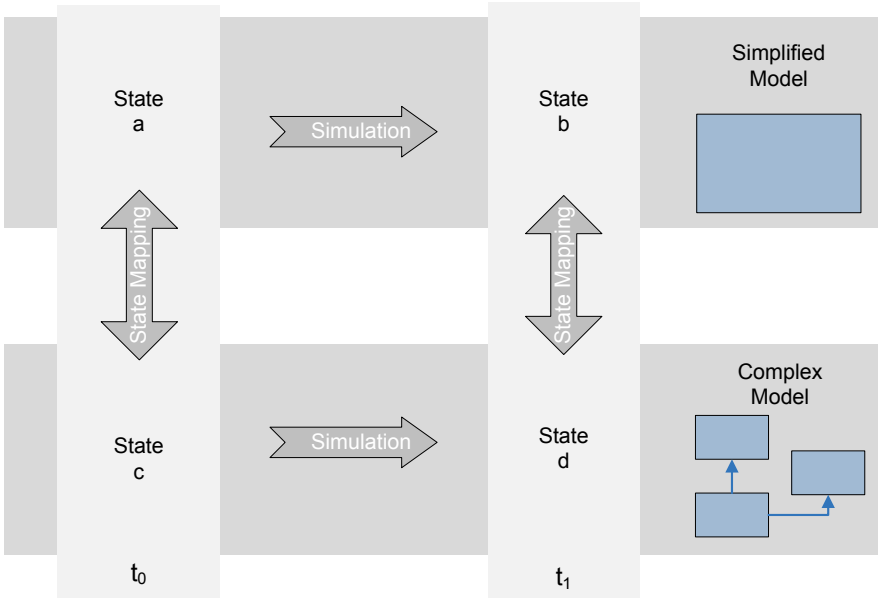

Figure 1: Consistency Requirements for State Mapping.

Davis (Davis 1992) and Palmore (Palmore 1992) defined requirements for the consistency of these exchange operations. According to Figure 1 they demand for full consistency that the following four equations are fulfilled at time t_1 (where a, b, c and d are states and α and β name different paths of simulation phases and state mapping operations to get to that state):

No.	State equality at time t_1	Paths to archive these states
1.	$d_\alpha = d_\beta$ with	$d_\alpha : a \to c \to d$ and $d_\beta : a \to b \to d$
2.	$b_\alpha = b_\beta$ with	$b_\alpha : a \to b$ and $b_\beta : a \to c \to d \to b$
3.	$b_\alpha = b_\beta$ with	$b_\alpha : c \to a \to b$ and $b_\beta : c \to d \to b$
4.	$d_\alpha = d_\beta$ with	$d_\alpha : c \to a \to b \to d$ and $d_\beta : c \to d$

In any realistic case none of these equations will be satisfied, because it would render the simulation of the highly detailed model pointless. It would be pointless, because any state of the highly detailed model could be generated by mapping the state of the less detailed model to it – generating accuracy from nothingness. More realistically the equations should read $\text{state}_\alpha = \text{state}_\beta + \varepsilon$ and shall be called relaxed consistency. The ε in the relaxed consistency should be as small as possible and has to determined considering the simulation objectives / the experimental frame (cf. (Davis 2000)). To the best of the authors knowledge, there exists no paper in the field of multiresolution modeling and state mapping which actually shows quantitative analyses of state mapping operations regarding consistency of or generated error in simulation states.

The work presented in this paper focuses on the use of DMRM in general discrete event material flow models. The models concerned M(C, L) have a graph structure, where nodes C are model components representing machines, buffers, sources, sinks, etc. and the directed edges L define the flow of jobs from component to component. Jobs are objects representing production jobs, raw material, etc. The nodes contain variables, parameters as well as event routines, peaces of programming code to change the state of a component and to schedule further events. Event routines are executed when the associated event occurs. The state of a model in the context of DMRM is defined as the set of all states of the active model parts. The state of a model part is defined as the set of all states of its model components. Finally the state of a model component is defined by the values of it's state variables, it's contained jobs and it's set of events in the scheduler of the simulation kernel.

Mueck and Dangelmaier (Dangelmaier and Mueck 2004) developed a method for dynamic multiresolution modeling for material flow simulation. In their work a multi-level hierarchy of models of different detail is created by simplification based on one original model of very high detail. A sketch of one possible part of such a hierarchy is shown in Figure 2, where rectangles represent model components and arrows are logic links between these. The original model is partitioned into model parts $b_i(h)$ on level h. These model parts are simplified either in a one to one relation, i.e. one model part $b_i(h)$ has exactly one associated simplified model part $b_j(h-1)$ which represents the same part of the system as $b_i(h)$. A n to one relation is also possible, in this case the simplified model $b_j(h-1)$ represents the same part of the system as the set of models $B(b_j(h-1)) = \{b_i(h) | m \leq i \leq n;\ m,n,i \in \mathbb{N}\}$ with $B(b_j(h-1)) \cap B(b_l(h-1)) = \emptyset$ and $j \neq l$. In Figure 2 $b_1(h-1)$ is the simplification of the composition of $b_1(h)$ and $b_2(h)$.

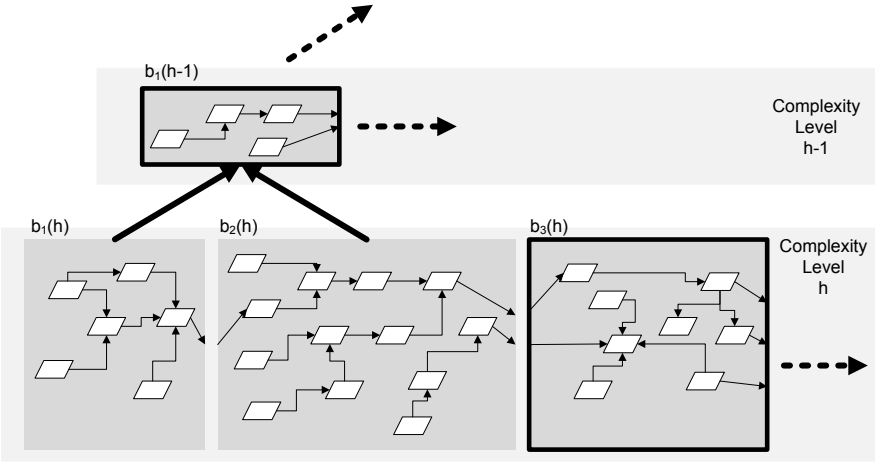

Figure 2: Hierarchy of Model Parts.

In the work of Mueck and Dangelmaier model parts are activated or inactivated based on user focus and amongst other things also based on geometrical and logistical distance to this user focus. As sketched in Figure 3 where a three-level hierarchy (pentagon, octagon and 24-gon) is assumed, the user focus or position is represented by the eye symbol and the level of detail for the model parts is determined by the above mentioned indicators.

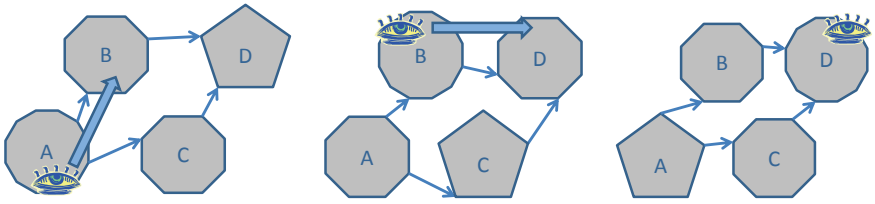

Figure 3: Sketch of Model Composition by Model Parts of different Detail.

Huber and Dangelmaier (Huber and Dangelmaier 2009) developed an automatic simplification method to improve the applicability of the dynamic multiresolution modeling. In the method of Mueck and Dangelmaier all simplified model parts in the hierarchy have to be created manually – a lot of work, especially if there have to be several levels of detail. Using the method of Huber and Dangelmaier it is possible to create a hierarchy automatically, by only providing the original model.

The DMRM created by Mueck, Dangelmaier and Huber meet all the requirements for MRM design defined by Yilmaz et al. (Yilmaz, Lim, Bowen, and Ören 2007), except for the first one, which states that the MRM should be decoupled from the knowledge regarding how its model parts are composed, created and represented. By using automatic simplification this requirement is broken, but also creates

the requirement for automatic sate mapping. The works of Mueck, Dangelmaier and Huber are used as a basis of the work presented in this paper, because an open simulation framework was created by them which allows manipulation of all of it's components and delivers an optimal test setup. In the following sections a method is presented for mapping the states of automatically simplified model parts in dynamic multiresolution modeling in a way that relaxed consistency is archived.

2 BASICS AND REQUIREMENTS OF AUTOMATIC STATE MAPPING

Without defining requirements, simulation states and models of different level of detail are too unspecific for an algorithm to analyze and synthesize. Therefore several requirements have to be made.

Based on the modeling methodology described in the last section, the first requirement for automatic state mapping is that there is read as well as a write access to all elements of the simulation state. Referring to the definition of a state, there has to be access to the variables of model components, to the contained jobs and to the event scheduler of the simulation kernel.

When reading the state, it has to be possible to assign each event in the scheduler to one model component and one job, if the event is job related – start processing event, move job to next component event etc. This is necessary because the scheduler only exists once for a composed model and events have to be assigned to the scheduling component to get the simulation state on component level. The assignment to a specific job is necessary, to calculate at which time a specific job is scheduled to leave a component.

The third requirement is, that there is a relation between the components of exchangeable model parts in the hierarchy. When using the common simplification method to substitute a set of model components by a buffer with delay (further referred to as delay) (for instance in (Rose 2007, Johnson, Fowler, and Mackulak 2005, Huber and Dangelmaier 2009)), this relation assigns the set of components in the more detailed model to the delay in the simplification. If the simplification is done correctly the delay is able to contain the same number of jobs as the substituted set of components and also has a similar average throughput time. This relation between components of different levels of detail shall be called 'relation of creation' and it generates a forest $F(C,E)$ of directed trees, where C is the set of all components in the whole hierarchy and E is the set of all 'relations of creation'. All trees of the forest are disjoint and are rooted in the components of the model on the lowest level of detail. Every component on a higher level of detail $c \in B(b_j(h-1))$ is connected to exactly one component $k \in b_j(h-1)$ on a lower level of detail and every k has incoming edges from $K \subseteq \{k|k \in B(b_j(h-1))\}$. Note that in this notation the level in the hierarchy representing the lowest level of detail has the index 0, which is the lowest index number, but this level is the topmost level of the hierarchy (cp. Figure 2).

When using a multi-level hierarchy, where the difference in detail between two adjacent levels is not very high, there is a set of unchanged components $K_u = \{k_u||\{k|(c,k) \in E\ \forall c\}| = 1\}$ which have only one incoming edge. The states of these unchanged components can simply be 'copied'. Without loss of generality, it is assumed that only model parts of adjacent levels of detail are exchanged. If a model part of a more distant level shall be activated the state mapping have to be done repeatedly.

The requirements described above are quite extensive and cannot be met by all simulation frameworks or applications of material flow simulation. In particular these are, the broad access to the scheduler and the necessity of 'relations of creation'. Firstly, read and write access of variables and jobs is possible in basically all simulation frameworks, broad access to the scheduler is not possible in any commercial framework the authors know. But the list of scheduled future events is an essential part of the simulation state and without it, the error generated by state mapping would increase significantly. In this work the authors had and used the liberty to implement the state mapping algorithm directly into the used simulation kernel, thus having rightful access to all objects (in the terms of Object Oriented Programming) of the simulation kernel and the model. To what extend automatic state mapping is possible in commercial simulation frameworks, which kernel normally cannot be modified and thus the state mapping algorithm must be implemented on the model level, is an interesting topic for further research. Secondly, without having something like the 'relation of creation', thus having a standardized and complete relation between all model components

on all levels of detail, automatic state mapping seems impossible. If automatic state mapping should be implemented in an other application in manufacturing, similar relations have to be created, but if doing so, automatic state mapping should be applicable.

To do the actual state mapping the process is structured into three steps. At first, jobs of the inactivated model will be 'put on' the related model components in the activated model. After this step the activated model already has a state sufficient for a continuing simulation of the composed model. To reduce the ε in the consistency equation, the breakdown status of components in the activated model are mapped as well as statistical variables – throughput counters, utilization indicators and so on. Jobs, breakdown status and statistical variables are part of basically every material flow simulation model, making this three step process applicable for general material flow models. In the following these three steps are explained in detail.

3 MAPPING OF JOBS

When mapping jobs two aspects have to be considered: All jobs from the inactivated model have to be in the activated one and – in an virtual what-if analysis comparing the inactivated and activated model parts – the throughput times of mapped jobs have to be similar.

The mapping of jobs is done using the 'relations of creation'. As shown in Figure 4 the jobs of two delays (D1 and D2) and one machine (M1) in the component set of higher detail are put on the delay D4 in the component set of lower detail. D1, D2 and M1 are connected to D4 by gray arrows representing the 'relations of creation'. This seems relatively straight forward in case a model of lower detail is activated.

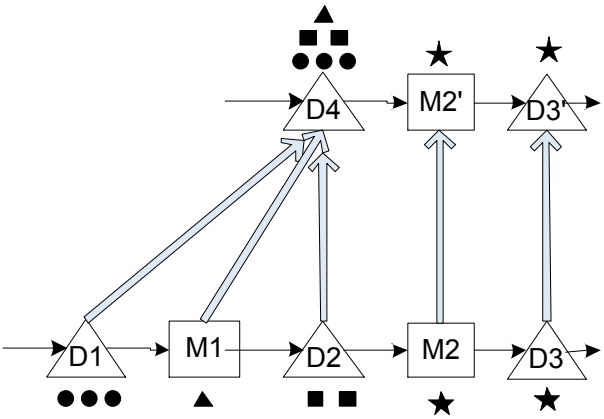

Figure 4: Raw Mapping of Jobs in a Chain of Components.

But in the other case, the question arises, on which of the connected three components the jobs should be put on. To solve this problem, the list of scheduled events of the inactivated model is analyzed. Events of interest are the output-events – move job to next component event – of these jobs. Each of these events has a scheduled time $0 \leq \Delta t_i$. If there is no such event for one job, the job is waiting because the allocated component or the successive component is blocked or the allocated component waiting for further input. In this case Δt_i for this job is set to 0. If there is an event scheduled for a job, but it is no output-event, Δt_i has to be approximated by analyzing the event structure of the component. After calculating the Δt_i in the inactivated components the mean throughput times t_e of components in the activated components are considered. All components have estimated throughput times, for instance 0 for buffers or the mean service time for machines.

While working backwards in the chain of components in the activated model ($D2 \rightarrow M1 \rightarrow D1$), the jobs are put on that component, where they can have the same output time (for instance $\Delta t_i \leq t_e(D2) + t_e(M1)$ and $\Delta t_i > t_e(D2)$) and no capacity restrictions are broken. After placing the jobs, appropriate events have

to be scheduled to archive a consistent state. For instance there have to be a output-event for every job on a delay, otherwise nothing would happen to these jobs in the future.

For each component class and allocation state the future behavior in a consistent state can be defined. The allocation states are as follows:

Allocation state	Future behavior
empty	Allow further input
allocated, waiting for input	Allow further input
allocated, waiting for output	Allow further input Output in $\Delta t_a = 0$
allocated, processing	Allow further input Output event in Δt_a
full, waiting for output	Disallow further input Output event in $\Delta t_a = 0$
full, processing	Disallow further input Output event in Δt_a

If components are not full, further incoming jobs are allowed, if they are full none are allowed. A component is full if a buffer capacity is reached or a batch of jobs is inserted into a machine and the next cannot be inserted until this batch has finished processing. The three states empty, allocated and full can be derived directly by comparing the allocation to the buffer / batch size. For a consistent state it is necessary to identify if components are waiting for further input before they can start processing, if they are waiting for output, because the successive component was blocked or if they are processing. To identify if a component is waiting for further input, it has to be determined if the current allocation allows it to start processing. This is especially important if the component is a assembly process. If a component is waiting for output, a component class specific output-event has to be scheduled for immediate execution ($\Delta t_a = 0$). If a component is processing, a processing-finished or output-event has to be scheduled in Δt_a time units.

Δt_a can be calculated by analyzing the event list of the inactivated model. Analogical to the job allocation problem above, the time until output out of the whole chain in the inactivated chain $t_c = \Delta t_i + \sum_{next\ c}^{end\ of\ chain} t_e(c)$ is approximated (for instance: $t_c(D4) = \Delta t_i(D4) + 0$). This time shall be met in the activated model, thus Δt_a is calculated as $\Delta t_a = t_c - \sum_{next\ c}^{end\ of\ chain} t_e(c)$ in the activated chain (for instance: $\Delta t_a(D1) = t_c(D4) - t_e(D2) - t_e(M1)$).

An other specialization of the described job mapping method so far is necessary, if the logistical 'chain' in the model of higher detail is not a chain but a more complex structure. A structure of parallel lines is possible, where several chains consist of similar components and all chains have specific common start and end components. In this case going backwards in the 'chain' in the activated model for job allocation results in multiple possible locations. To solve this problem the dispatching rule of the dispatcher in front of the parallel structure have to be analyzed. The job is then placed on the component with the highest probability regarding how the dispatching rule would have handled the job in question. No adaption of the method to calculate Δt_a is necessary, because the path from the end of the 'chain' to the chosen component is definite.

4 MAPPING OF BREAKDOWN STATUS

Some component classes have breakdown properties, most probable defined by MTTF and MTTR as random variables. With these properties the simulation kernel schedules events of breakdown start times and breakdown end times. In the following we assume that the simulation kernel only schedules the events for the next breakdown after the last breakdown has ended.

Analogous to the mapping of jobs the mapping of breakdown status is trivial for unchanged components – components with one incoming creation relation edge. The relevant events are simply 'copied'.

In all other cases a component on level $h-1,2,..$ has two or more incoming edges in the tree of 'relations of creation'. Thus the breakdown status of multiple components have to be mapped to one in case that the model of lower detail is activated (Case 1) or the status of one have to mapped to multiple components in the other case (Case 2). Hence there are several problems to solve to get a good mapping.

If in Case 1 the activated component does not have breakdown properties – when for instance as usual a delay without breakdown properties as a simplification is used – the solution is trivial, because nothing can be done. In all other cases the breakdown start event of the first component to break down is 'copied' and the breakdown end event is scheduled using the activated components breakdown properties. If one component is broken down during inactivation, the breakdown start event of the activated component is scheduled instantly and the breakdown end event is scheduled such that the breakdown ends as in the inactivated model. This is only a relatively good approximation of future behavior if the connected structure in the inactivated model is a simple chain. To get a better approximation in more complex structures, calculations as in reliability analysis (for instance (Birolini 1994)) are necessary, but were not implemented nor tested in this work.

If in Case 2 the inactivated component does not have breakdown properties the next breakdown of the connected components in the activated model is generated by normal means – the simulation kernel schedules breakdown start and end events according to the components breakdown properties. In the other case all connected components with breakdown properties get a breakdown start event scheduled at the same time as the component in the inactivated model. The breakdown end events are scheduled by normal means. This is a rough worst-case approximation in lack of a better solution especially designed for breakdown start events in the near future. Because if all breakdown events of the activated components would be scheduled by normal means the next breakdown could be in the far future and the 'future-behavior' of the inactivated model would be very different to the actual future behavior of the activated model.

5 MAPPING OF STATISTICAL VARIABLES

Most component classes have several statistical evaluations which current values are stored in variables. Common are for instance a throughput counter, a utilization calculation, sums of work, blocked and idle times. To evaluate the performance of the dynamic multiresolution model these values shall be mapped from inactivated model parts to activated model parts.

The mapping of these variables for unchanged components is trivial. For all other components this problem is only universally feasible for basic statistics like throughput counters.

When mapping the throughput, the value of the last component in the inactivated chain is mapped to the last component in the activated chain. Every present predecessor of this component gets the value of its successor plus the number of jobs on this successor. The length of the chain in the model of lower detail is always 1, such that no predecessor oder successors are present. Analogously to the mapping of jobs the dispatch rules of a dispatcher must be considered if activating a more detailed model where the connected components do not form a simple chain, but parallel lines.

There are several problems concerning the mapping of more complex statistics. The first problem is, that the 'same' statistical value, for instance the utilization, is possibly defined differently in different component classes. The utilization of machines is defined by calculating the ratio of work, breakdown and blocked time to the simulation duration. The utilization of conveyor belts or delays is defined by the ratio of actual throughput to maximal throughput. Taking into consideration that the connected component by 'relation of creation' in the less detailed model is a delay, a mapping to a machine in the model of higher detail is not possible, because the times necessary to calculate the utilization are not present and cannot be generated. When mapping the highest utilization of the connected components in the model of higher detail to the delay of the less detailed model it is most likely that this value is not equal to the calculation of the ratio of actual throughput and maximal throughput.

As a result only throughput counters are mapped on changed components to avoid a suggestion of more precision or information than actually present.

6 RESULTS

The method of dynamic multiresolution modeling, the automatic simplification of model parts and the state mapping where implemented into the simulation tool d^{3}FACT insight (Dangelmaier, Huber, Laroque, and Mueck 2005, Dangelmaier, Mueck, Laroque, and Mahajan 2004).

For testing, three different original models were used (A, B and C). Model A is a big model with 151 components, containing medium long lines and many parallel lines. Model B has 133 components, composed of quite short lines and some parallel lines. Model C is composed of 61 components in three lines and two parallel lines. In these models lines are shorter, if there are more branching points in the model.

Each model was simplified in two different levels of detail, $b(1)$ is a model with slight simplification and $b(0)$ a strong simplification. The evaluation of the mapping method was done analogous to paths 2 and 4 defined in Section 1. For path 2 the experiment is as follows: The original model is simulated starting at t_0 until t_1, then the original model is inactivated and $b(1)$ or $b(0)$ is activated and simulated until t_2. At t_2 this model is inactivated again and the original model is reactivated. The state at t_2 reached after two mapping operations is now compared to the state reached if the original model is simulated from t_0 to t_2 without any mapping operations. The approach for path 4 is analogous.

In a first set of experiments the mapping times were chosen such that $t_1 - t_0 = t_2 - t_1$ and equal for all three models. When doing so there is a transient phase in which the model is filling with jobs – when starting empty – but also a phase between mappings when the states diverge because there is a difference in behavior between the original model and the simplification. Hence a state deviation at t_2 is not only caused by the mapping operation but also by the simulation period from t_1 to t_2. To eliminate this error, a second set of experiments were done by setting $t_1 = t_2$ and $t_1 > t_0$.

In the first set of experiments the state comparison is done by comparing the statistical variables of all components. For unchanged components these are all statistical variables of that component and for changed components its only the throughput counter. The relative errors of throughput are shown in Table 1. The errors of the other statistical variables of unchanged components have similar values.

Table 1: Relative Error of Throughput if $t_1 < t_2$.

Original Model	Rel. Error [%]			
	Path 2		Path 4	
	$b(1)$	$b(0)$	$b(1)$	$b(0)$
A	0.70	4.70	1.00	6.11
B	1.84	3.04	6.16	0.19
C	3.48	3.51	3.03	2.61

In the second set of experiments the activation is done only according to path 2. Additionally to comparing the throughput, also the error in allocation is calculated. The error of allocation is not regarding the number or set of jobs – there is no error in this –, but the number of jobs which are not on the same component after the two mapping operations in comparison to where they were before the mapping in the original model at t_1. The results are shown in Table 2.

Table 2: Relative Error in Throughput and Allocation if $t_1 = t_2$.

Modell	Relative Error [%]			
	$b(1)$		$b(0)$	
	Throughput	Allocation	Throughput	Allocation
A	0.04	16.0	1.3	40.0
B	0.03	4.7	2.7	24.8
C	0.25	28.0	1.6	49.0

The first and basic conclusion of Table 1 is that the state mapping creates a consistent state and the simulation of the activated model continues. Secondly, the relative error after two mappings is between 0.2 and 6.2 % , with no real significant difference between $b(1)$ and $b(2)$ regarding these measurements. The behavioral deviation of $b(1)$ and $b(2)$ in comparison to the original model is around 4 % (taken from separate experiments), such that no conclusion about the error caused by the mapping is possible.

The results presented in Table 2 show a more significant difference between the error of mapping to a less simplified model and a more simplified model. When activating and inactivating $b(1)$ the relative error is between 0.03 and 0.25 % and between 1.3 and 2.7 % when using $b(0)$. The error is relatively small and mainly caused by mapping throughput values to components in parallel lines, as detailed analysis revealed. The error of job allocation is caused by the same effect. If activating parallel lines, the jobs are set to random lines – if there is a uniform dispatching rule as in the test models –, such that there is a very big error created, which is partly compensated by the correct allocation of unchanged components and components in chains. The bigger error when using the model of higher simplification is caused by the reduced number of unchanged components and higher number of simplified parallel lines.

7 SUMMARY

By using the method described in this paper it is possible to run a simulation of dynamic multiresolution models where model parts of different level of detail are activated or inactivated during simulation. The necessary state mapping of models of different level of detail is possible if relaxed consistency, i.e. an error in simulation state, is allowed otherwise it is not possible, because information is lost when using models of a lower levels of detail which can not be generated when activating a model of higher detail. The developed method requires that there is a component based 'relation of creation' connecting components of different level of detail. In this work, the relation is created by the simplification operations necessary to create the models of lower detail out of the original model with highest detail. The state mapping is done in three consecutive steps, the mapping of jobs, the mapping of breakdown status and the mapping of statistical variables. In experiments with three original models and two different levels of simplified models each, it is shown that consistent states are generated such that the simulation continues after state mapping. The errors generated in throughput are small and comparable to the behavioral deviation of the simplified models. The error in job allocation to specific components is bigger and caused by the loss of information in the simplified models.

REFERENCES

Birolini, A. 1994. *Quality and Reliability of Technical Systems: Theory, Practice and Management*. Springer.

Dangelmaier, W., D. Huber, C. Laroque, and B. Mueck. 2005. "d^{3}FACT insight - An immersive material flow simulator with multi-user support". In *Proceedings of the 2005 Summer Computer Simulation Conference*, edited by A. Bruzzone and E. Williams, 239–242.

Dangelmaier, W., and B. Mueck. 2004, December. "Using dynamic multiresolution modelling to analyse large material flow system". In *Proceedings of the 2004 Winter Simulation Conference*, edited by 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.

Dangelmaier, W., B. Mueck, C. Laroque, and K. Mahajan. 2004. "d^{3}FACT insight: A Simulation-Tool for multiresolution material flow modells". In *Simulation in Industry - 16th European Simulation Symposium*, edited by G. Lipovszki and I. Molnar, 17–22.

Davis, P. K. 1992. "An Introduction in Variable-Resolution Modeling and Cross-Resolution Model Connection". In *Proceedings of the Conference on Variable-Resolution Modeling*, edited by P. K. Davis and R. Hillestad, 1–43.

Davis, P. K. 2000, December. "Exploratory Analysis enabled by Multiresolution, Multiperspective Modeling". In *Proceedings of the 2000 Winter Simulation Conference*, edited by J. A. Joines, R. R. Barton,

K. Kang, and P. A. Fishwick, 293–302. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Huber, D., and W. Dangelmaier. 2009, December. "Controlled Simplification of Material Flow Simulation Models". In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 839–850. Piscataway, New Jersey: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Johnson, R. T., J. W. Fowler, and G. T. Mackulak. 2005, December. "A Discrete Event Simulation Model Simplification Technique". In *Proceedings of the 2005 Winter Simulation Conference*, edited by 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.

Natrajan, A., P. F. Reynolds, and S. Srinivasan. 1997. "Consistency Maintenance in Multiresolution Simulations". *ACM Transsctions on Modeling and Computer Simulation* 7:368–392.

Palmore, J. 1992. "Variable Resolution Modeling in Mathematics". In *Proceedings of the Conference on Variable-Resolution Modeling*, edited by P. K. Davis and R. Hillestad, 362–376.

Rose, O. 2007, December. "Improved Simple Simulation Models for Semiconductor Wafer Factories". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton. Piscataway, New Jersey: 1709-1712: Institute of Electrical and Electronics Engineers, Inc.

Yilmaz, L., A. Lim, S. Bowen, and T. Ören. 2007, December. "Requirements and Design Principles for Multisimulation with Multiresolution, Multistage Multimodels". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 823–832. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

DANIEL HUBER studied industrial engineering at the University of Paderborn, Germany. From 2005 to 2011 he was a research assistant at the group of Prof. Dangelmaier, Business Computing, esp. CIM at the HEINZ NIXDORF INSTITUT in Paderborn Germany. He received his doctorate in 2009. His main interests are material flow simulation, modeling methodology and automatic model simplification. His email address is d@nielhuber.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.