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

A PROTOTYPE SIMULATION TOOL FOR A FRAMEWORK FOR SIMULATION-BASED OPTIMIZATION OF ASSEMBLY LINES

Evangelos Angelidis
Falk Stefan Pappert
Oliver Rose

Dresden University of Technology
Institute for Applied Computer Science
01062 Dresden, GERMANY

ABSTRACT

General purpose simulators offer an easy way to create simulations for a large variety of scenarios, although they are prepackaged with some drawbacks. To achieve their usefulness for most cases they need additional overhead and custom made extensions for special behavior which comes at enormous runtime and development cost. Especially when working with simulation-based scheduling, these are severe issues since runtime is precious and automated generation of the models is a necessity. Another approach to simulation challenges is the creation of a very specific custom-built simulator which focuses on a chosen domain where it excels compared to other simulators. In our paper, we introduce a simulator designed specifically for the simulation of complex assembly lines with their common characteristics including thousands of activities, realistic schedules, priority rules, resources and model restrictions. It furthermore allows the creation of new strategies for different aspects of scheduling in this environment.

1 INTRODUCTION

Last year, the authors presented a paper (Pappert, Angelidis, and Rose 2010) about a framework for simulation-based scheduling for assembly lines. This framework aims to offer the possibility to create various solution strategies for assembly line scheduling problems. The basic idea of simulation-based scheduling is to simulate a scenario, analyze it and create a new one utilizing the results of the current and the previously simulated scenarios. This step is repeated until an acceptable scenario is found. The framework is designed to support most simulation tools as long as it is possible to write a model generator for them.

Over the last years we worked with several simulation tools and with most of them we successfully created simulation models of production lines and a suitable simulation plugin for the framework. Although the framework can cooperate with most of the tools, we realized that the transformation of our current meta-model into the specific model of the simulation tool is not always effective.

General purpose simulators are usually powerful tools which allow the users to create models of most scenarios they can imagine. Models are easily created under most conditions. Functionality which is not provided out of the box can be added using scripts or own libraries which contain user created behavior. The downside of these easy to enhance designs is usually the need to follow this path early in most cases. Numerous hours are spent to enhance the capabilities of these simulators to a point where it almost seems like re-implementing the simulator's core. Furthermore to provide a large spectrum of features the overhead is slowing down the simulation resulting in a solution which is costly due to the working hours spent and slow due to the interpretation of scripts, graphic routines and support for unused functions.

In simulation-based optimization (April et al. 2003; Majohr 2008), the solution time depends on the runtime of the optimization tool and the runtime of the simulation tool in every step of the iteration loop.

Usually the duration of the optimization depends on many parameters (complexity of chosen algorithms, database connection, difficulty of the current decision, etc.) and varies a lot. The simulation runs to evaluate a generated scenario are very stable in their runtime behavior. Furthermore the time spent waiting for the results of simulation runs is far larger than the time the optimizer prepares new scenarios. So the time of the solution search mostly depends on simulation speed and thereby the choice of the right simulation tool. A fast simulator which allows the reduction of simulation runtime will have a significant benefit on the overall runtime of a simulation-based optimization.

Another important issue for the choice of a suitable tool is the development cost. Furthermore there is a certain effort to familiarize oneself with a new tool and gain the ability to build real world scenarios with it. It is not always easy to compare or validate models from different tools (black box behavior). The models we are usually working with contain special characteristics which take some effort to be implemented with a new tool. Finally, in some cases it is quite difficult to automate the start of the simulator or to control a simulator from outside.

An alternate way to the general purpose simulators is to design a tool specifically for the simulation of complex assembly lines with their common characteristics including thousands of activities, realistic schedules, priority rules, resources and model restrictions. Furthermore the tool needs to be able to support extensions on the model and has to offer the opportunity to add new scheduling strategies easily.

The paper is organized as follows. In the next section we describe the problem that can be solved with simulation-based optimization. In Section 3 we provide our modeling approach. Section 4 will present the simulation tool for this problem and the last section, results and future work.

2 PROBLEM DESCRIPTION

The focus areas of our research are complex assembly lines for planes, turbines, printing manufacturing or other heavy machinery. In this kind of facilities the scheduling problem for large groups of highly specialized workers is very hard to solve. There are numerous workers with different skills, dozens of work centers, and hundreds of jobs with a considerable number of processing steps. Apart from that, there is a large variety of scheduling constraints like time bounds and the restriction for resources for different tasks in the production systems such as operating resources (machines, means of transportation, buffers as well as jigs and tools) and personnel resources.

Especially the scheduling problem of personnel intensive production systems is difficult and the dynamic of the production system increases the complexity of operative daily scheduling (Sauer 2002). Different possible shift plans, breaks, technical breakdowns and organizational problems cause immediate change in the production environment.

This environment can be classified as a Job Shop Scheduling Problem (JSSP) with several assembly specific characteristics. A set of jobs and a set of resources are given in the JSSP. Each job consists of a set of activities that must be processed in a given order. Furthermore a deterministic processing time is given for each activity as well as at least one limited resource on which it has to be processed. Once an activity is started it is processed without interruption (except for breakdowns and problems with resources). Each machine resource can process only one single job at a time. The following scheduling conditions result from the production environment:

- The jobs are represented by project networks with precedence constraints.
- All jobs have fixed release dates defined by material delivery dates and due dates defined by product delivery dates as well as submission dates of the following production system.
- The production system consists of multiple work centers with parallel machines.
- Work centers can have finite buffers. There are buffers that can be used by more than one work center.
- Some consecutive activities have maximum and minimum time bound requirements.
- For certain activities, jobs have to use a limited number of jigs or tools. This can lead to additional waiting times if the jigs or tools are already in use.

- A few activity groups are allowed to lock some machines, jigs, tools or personnel for a certain period of time. During that time the resources are only available for that group. It is used under special circumstances to avoid waiting times for valuable limited units or unnecessary transportation between parallel machines.
- Some activities end but must keep allocating some of their resources until some of the successors activities start.

The optimization of this scheduling problem (JSSP) is not only depended on the constraints of the project networks. It furthermore depends on resource restrictions. The complexity of scheduling these systems is further increased by the mode based nature of operations. Most operations do not have a fixed processing time. The amount of time necessary to complete a certain task depends on the amount and qualification of the workers assigned to it. Thus, the allocation of an additional worker reduces the process time by a defined value. This value is discrete. It is displayed in a characteristic curve of the personnel-activity-relation (Figure 1). These mode based operation durations come as an addition to the JSSP which usually has only fixed processing times.

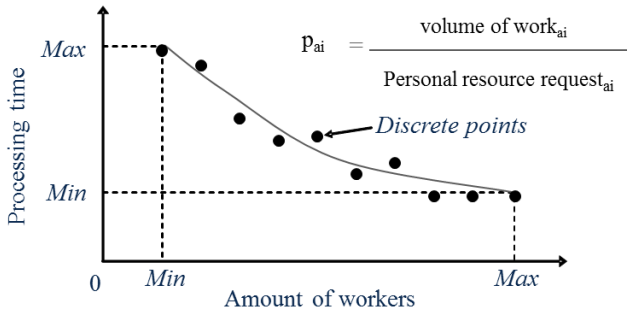

Figure 1: Example of a characteristic curve of the personnel-activity-relation, translated from Majohr (2008)

To optimize within this environment, both the Job Shop Scheduling and the Personnel Allocation Problem have to be solved simultaneously. The key of the personnel allocation problem is the right amount of workers at the right places at the right time. Due to the consideration of limited personnel capacities the personnel allocation is a knapsack problem that represents the allocation of limited personnel capacities to the activities at the work centers. This means the shift-based personnel resource supply has to be allocated to the activities regarding the personnel structure (for example working groups per work center). The combined personnel assignment and scheduling problem is very complex, in particular, for realistic problem sizes. Each part of the problem is NP-hard by itself. With regard to the short target solution times, an optimal solution of these NP-hard problems does not seem to be possible. Thus, it is more practical to search for good solutions based on heuristic approaches (Majohr 2008).

The intention of this paper is to present a simulation tool for a simulation-based optimization application which solves the combined problems using a heuristic approach. The benefit of using this optimization method is the solution of large scale problems. The number of solution candidates increases the total search time but do not slow down the search process itself considerably. The reason is that we consider every step just once for a new scenario and store it. Then, we utilize these results for all future scenarios.

In summary, the presented problem is a Multi-Mode Resource-Constrained Multi-Project Scheduling Problem (MMRCMPSP) or MMRCMPSP with activity splitting. As already mentioned, the goal is to create a scenario, where the modes the activities run in are chosen by an optimization as are the earliest-dates and due dates of the jobs, priority rules for the jobs and shift, break and breakdown for the resources. The scenario handed to the simulator can be classified as a Single-Mode Resource-Constrained Multi-Project Scheduling Problem.

3 MODELING APPROACH

Working with several companies of different sizes and assembly approaches we realized the need for a flexible and easy to adapt data model implementation which supports our current data meta model but which will be easily emendable in the future to optimize performance or add features. Our chosen approach is to use model-driven development technologies. In contrast to the research of Weyprecht and Rose (2011), we are not going for a complete model-driven simulator to avoid possible performance problems with larger models. We have chosen to employ a compromise by having a model-driven data design with a conventionally programmed simulation engine. The design of the internal meta-model is created with the Eclipse Modeling Framework (EMF) (Steinberg et al. 2008) and then transformed into Java code which is used by the simulation engine. This approach allows us to dynamically change our simulation tool, adapting to model changes quickly by emending the UML diagrams and regenerating the data management. Through this automation effort time for development is reduced significantly compared to a conventional approach.

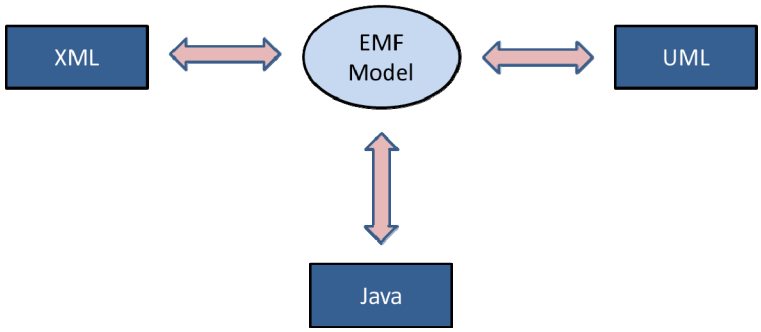

Figure 2: EMF Eclipse Modeling Framework (Steinberg et al. 2008)

3.1 Activities

The set of jobs in the system is represented by project networks which are often used for project planning with complex time, resource, and precedence constraints. The literature is dominated by two types of networks:

- Activity-on-node (AON) network (Patterson et al. 1989; Holzer 1999; Mika, Waligóra and Weglarz 2005)
- Activity-on-arc (AOA) network (Elmaghraby 1977, Neumann and Morlock 1993).

We chose the AOA network type for its more intuitive concept regarding our production environment and the easier way to represent different modes of activities within a single view. Furthermore this representation eases considerations for critical path and complicated branching.

Each single project network represents a single product's path through the production environment with all possible alternative routes and resource configurations. We describe alternate routes and splitting for parallel process steps using node logic. There are four different types of nodes which result from the combination of parallel and alternative paths for incoming and outgoing activities. Using this kind of nodes allows us to represent all possible activity interactions with a maximum of a two stage node cascade while drastically reducing the effort to evaluate complicated precedence constraints. To add the important constraint of job synchronization we further included a constraint to indicate identical start times. This increases the node collection to six nodes.

A set of these project networks can be considered as a description of the amount of work to be done during the time frame of interest. The interaction of different products within the facility is modeled by their competition for the same pools of shared resources. Figure 3 provides an overview of the activities.

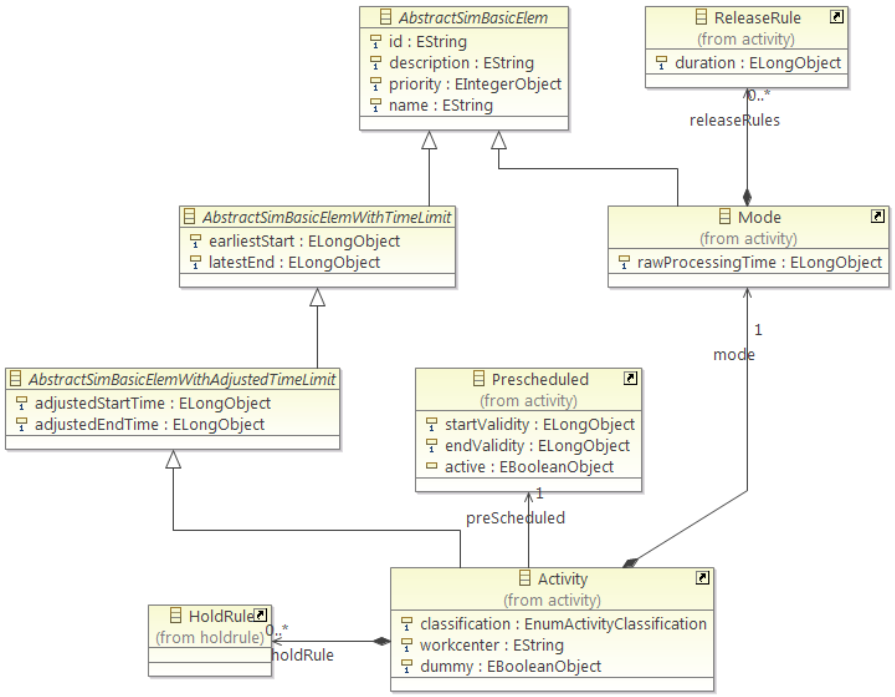

Figure 3: Activity overview

3.2 Resources

The implementation of the model is based on the data meta model described in Pappert, Angelidis and Rose (2010). The data model of the simulator is a subset of the data model of our introduced framework. A *Resource* object represents a resource pool with equivalent elements. The resource capacity is limited by shift, break and breakdown schedules. We currently consider five different types of resources:

- Processing facilities
- Help facilities
- Transport facilities
- Personnel
- Buffers

The model can have many similar resource groups which are independent from each other. To capture and evaluate different resource allocations and to be able to use this data later during automated optimization we are working with the following equipment state model (Figure 4).

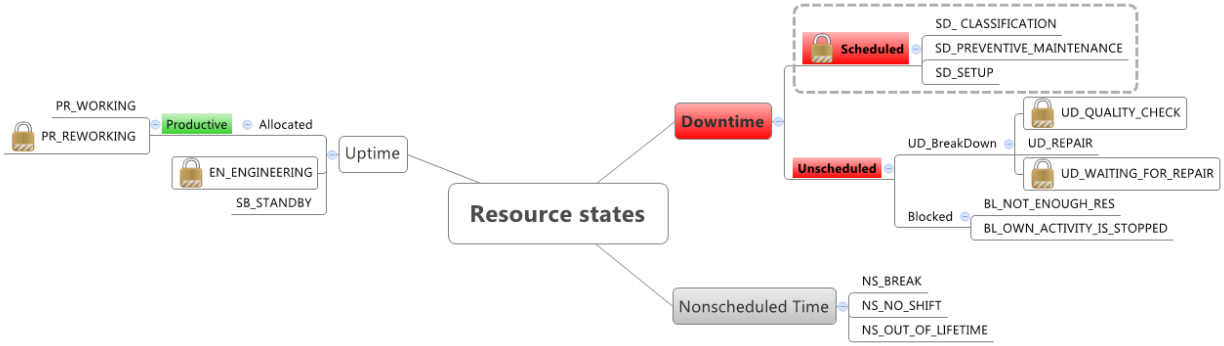

Figure 4: Resource states

The states are based on a specification of the Global Metrics Committee (SEMI 2004) and extended after interviews with our partner companies. The resource states are equal for all types of resources. The lock on certain states indicates states which are not fully incorporated into our simulator yet.

3.3 Constraints

An important constraint to model behavior within a heavy machine assembly is the "remaining rule". We use this rule to describe a way of grouping activities to be able to use resources together. In a real production environment this rule applies for example to the usage of constructions sites. There are basically two approaches used in heavy machine industry. The most common way is to manufacture or assemble the work piece on a specified location and bring all tools and workers needed to this space in the construction area. This approach works well when the equipment needed to produce is easy to move. For these cases a space distribution optimization can be extracted from the problem and solved individually. In aircraft manufacturing on the other hand this is not the case. Here we face large shelling structures which are hard to be moved from one construction site to next site. Furthermore producers try to bring their production closer to a flow line behavior. Thus making construction space a valuable resource whose allocation needs to be reassessed during the production. At each production site a number of different work packages are processed. The order of these work packages may be influenced by precedence constraints, branching alternative work steps, and competition for other resources. Still all of these work packages use the same construction site. This results in an environment where a number of different tasks may ask for the allocation of the resource to the group depending on which of them has all needed resources available first. Then all work packages can be worked on at the same time or based on other constraints. Due to the size of the work piece there is usually almost no interaction between tasks which slow down each other due to space constraints. Defining the time of which the resource can be released is not that easy, too. Due to alternative process steps within work packages or even on factory level, the completion of different activities causes a valid state for leaving the construction site. We model this whole behavior with the "remaining rule".

As mentioned before, our model is based on project networks using logical connections between its nodes. Based on these nodes we can assert a set of all activities which are allowed to allocate a specific resource or resource set for the whole group. We refer to this set as StartActivities.

Once a resource is selected, all other activities within the group of StartActivities are evaluated based on their ability to be performed on this resource. Process routes which depend on the use of restricted activities which cannot be performed with the chosen resource are blocked. To avoid deadlocks we check before the simulation whether there is always a valid path through the project network for each possible resource selection.

To define a valid state for releasing the resource we again consult the project networks logical nodes and calculate a logical term which represents all valid situations in which the resource can be idle. All activities which can trigger a check on this condition are part of the EndActivities set, which consists of all activities mentioned in the release condition term.

Other resource allocation oriented rules manage early release of resources which are needed only for a part of an activity. And a hold rule which allows a work piece to block equipment. With this rule it is possible to model situations in which construction sites are still blocked by a work piece while all other resources like workers and smaller equipment can already be redistributed. This is necessary to model the importance of space within the manufacturing environment which is a limited resource in heavy machine assembly.

Time-bound sequences as an essential constraint for some operations are not implemented directly in the simulator itself; in our current models it has been sufficient to check after simulation whether the condition was met. In critical cases the application of additional remaining rules to reserve resources within the starting step to ensure their availability later on is possible. This approach avoids the need of time consuming rescheduling during simulation. Invalid simulation runs which are caused by violated time

bound sequences are still finished to gain more knowledge about the system behavior for later optimization runs.

4 THE SIMULATION TOOL

The main idea is to create a simulation tool that works as fast as possible with the needed characteristics of our model, our special restrictions and the priority rules, we defined in the previous section. The implementation is based on the conventional discrete-event-simulation approaches. A main role in that approach has the event list. The events belong to resources and activities. The event types which are handled by event list are:

- *breakdownResourceEvent,*
- *breakResourceEvent,*
- *shiftResourceEvent*
- *earlierResourceReleaseEvent,*
- *activityEnableEvent* and
- *activityEndEvent.*

4.1 Activities

The behavior within the simulation is mainly influenced by the design of the activities. In this chapter we introduce our activity state model which is seen in (Figure 5). While moving through this state model activities visit a number of states which are internally represented as activity sets. These sets are initially empty but are filled during the initialization of the simulation. Activities without predecessor are moved into the *Initial* state while all other activities are moved to the *Unscheduled* state. In the second initialization step the activities with the earliest release date are selected from the *Initial* state and moved to the *Wait for processing* state. After this initialization the simulation starts at the point of the first release of an activity by starting activities from the *Wait for processing* set. Furthermore all activities add an event for their earliest start date to the event list.

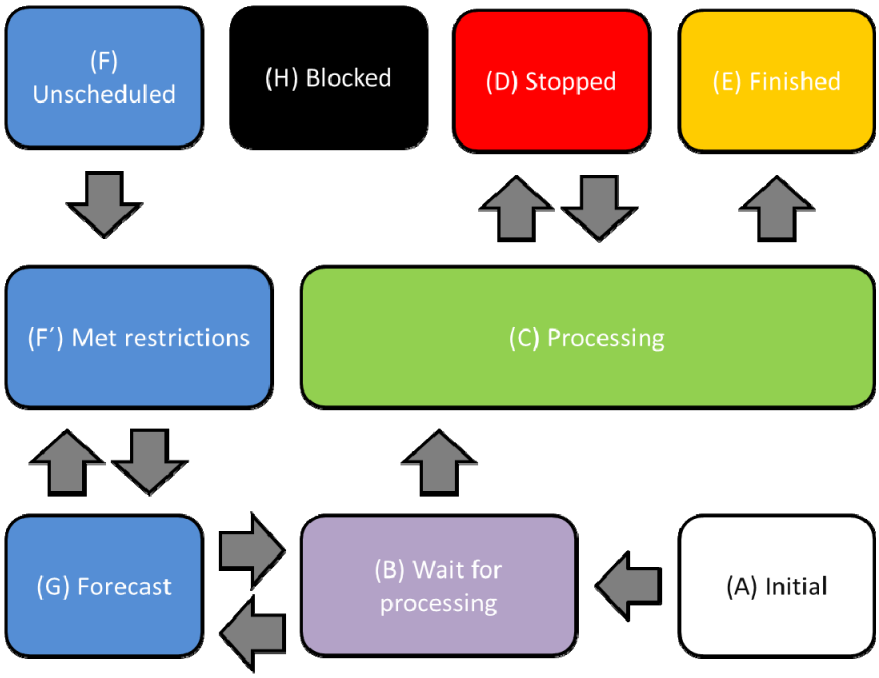

Figure 5: Activities diagram

Initial state. Activities in this state do not have any predecessors. They are the starting points of the production of all jobs. Activities which are still in this state have not yet reached their release date. Once their release date has come and all their restrictions are met, they are moved to the *Wait for processing* state. Otherwise they are moved to the *Met restriction* state.

Wait for processing state. Activities in this state can already be scheduled and are only waiting for the necessary resources to start. If an activity has to start some remaining rules, then it has to wait for their resources, too. This state is implemented using an ordered list, which allows the application of priority and dispatching rules to influence the order in which activities check for the availability of their resources. An activity can move to the processing state when all needed resources are available. This will assign the resources to this activity. In case the activity is preceded by a node which has a synchronized out, all parallel activities can only move as a bulk, therefore resources for all the activities need to be available to move to the next state.

Processing state. Activities in the processing state are currently in production and have already allocated all the needed resources, which stay assigned until the end of their activities. If the activity starts for the first time there are some steps which have to be done.

- If the activity started a remaining rule, then according to this rule some activities must be blocked. These activities move in the *Blocked* state. Furthermore all activities which share this remaining rule are informed about the rule start.
- If the activity influences hold rules of other activities, then it tries to release the resources.
- In case the activity is preceded by a node which has an alternative out, all alternative activities are moved to the *Blocked* state.
- All successors are informed that the activity started and the simulation event list is updated with the events *activityEndEvent* and all the possible *earlierResourceReleaseEvent*.

If the activity does not start for the first time then only the simulation event list is updated with the new calculated times of the events (*activityEndEvent* and *earlierResourceReleaseEvent*).

If at least one of the assigned resources or the resources of the remaining rules is not available any more (because of a resource event), the activity moves to the *Stopped* state. After the termination of the processing time the activity moves to the *Finished* state.

Finished state. Activities in the *Finished* state are already executed. In this state the activities release all the allocated resources, except for the resources of the hold rules. The resources of remaining rules only release when the required activities of the group are completed. Furthermore all successors are informed about the *Finished* state.

Blocked state. Activities in the *Blocked* state cannot start. They enter this state because of a restriction or a rule and cannot change their state anymore.

Stopped state. The activities in the stopped state are activities, which were at least once in the *Processing* state and entered this state because the availability of their allocated resources changed through a resource event (shift, break or breakdown). At this moment the event list is updated and the *activityEndEvent* and all the *earlierResourceReleaseEvent* of the resources assigned to this activity are released. When a resource event causes an increase in the available capacity of the resource and the activity can start again then the activity enters the set *Processing* again and we update the event list, the *activityEndEvent* and all the *earlierResourceReleaseEvents*.

Unschedule state. The activities in this state cannot be scheduled yet. In the initialization step all the activities which entered it, added their earliest start date in the event list (to be later informed by an event). If at least one of their restrictions is met they enter the next state.

Met restrictions state. The activities wait for their restrictions so they can enter forecast state. Possible met restrictions in the current implementation are: earliest start date is reached, a predecessor is started, a predecessor is ended or a remaining rule, which the activity belongs to did not start yet and this activity cannot start this rule.

Forecast state. This state is similar to the last state. The activity enters only if

- the actual time is higher or equal to the earliest start time,

- all predecessors belong to the *Finished* state except of only one,
- the last predecessor started already and belongs to the *Processing* state at the time of the decision (choosing activities to start), or
- the remaining processing time of the last predecessor is less than a configured forecast value.

Activities in this state will be considered in the next decision with all the other activities of the *Wait for processing* state. Of course, if the last predecessor stopped then the activity enters the *Met restrictions* state. The value of the remaining processing time is chosen before the start of the simulation and changes according the characteristics of the model. The reason why the forecast is important in our modeling approach is because of the long processing times of most of the activities. Activities might allocate resources for a very long time depriving other activities shortly before releasing their resources after a resource event. The forecast is a possibility to prioritize future jobs. The most important and difficult challenges are to choose the correct value for the forecast of all the activities and the decision how many activities can use this opportunity each time.

4.2 Dispatching Rules

Another important characteristic of our tool besides the forecast is the priority rule list on the *Wait for processing* state. The activities are sorted by the first rule of the list. If the values are equal, then the comparison tries the next rule. If all rules within the list cannot prioritize one activity over another we use the unique id of the activities as a last deciding factor. It is very important to communicate implementations like this to avoid unexpected behavior. There are already implementations for several rules. Furthermore we support user specific behavior rules that implement a given interface.

4.3 Resources

In this section we will have a closer look at the implementation of resources. As described in Section 3.2 resources have three main timelines which describe their availability. The resources have shift plans, breakdown plans and break plans. The shift plan defines the amount of a resource which is available (Figure 6: Shift plan).

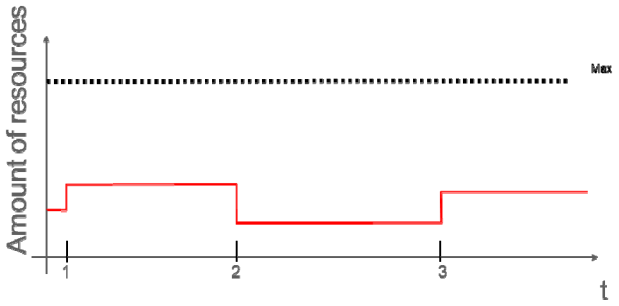

Figure 6: Shift plan

An example of a breakdown or break plan is illustrated in Figure 7. It does not set the amount, but switches on or off all resources which are considered available according to the shift plan at that time.

The breakdown plan controls three states from resource state definition, quality check, repair and waiting for repair. A break plan changes just one state. It is not necessary to define all schedules for all the resources. Just having a shift plan already allows most productions to be modeled. For example many machines have only shift and breakdown plans but to model human resources it is more transparent to model all three plans.

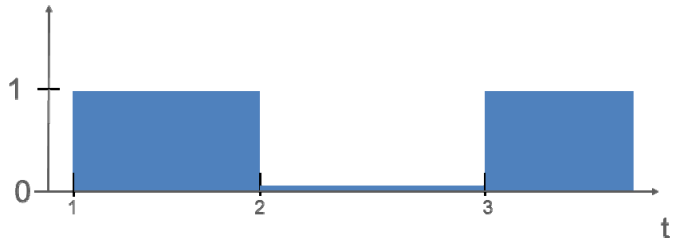

Figure 7: Breakdown and break plan structure

Because all plans influence the state of a resource at the same time, we define a priority for the plans:

Breakdown > Break > Shift

If a breakdown occurs then all the others states will be ignored. If a break takes place then the state of the shift will be ignored. Furthermore it is important to consider all the changes of the three timelines, to calculate the next state. If a shift event occurs the decision is to increase or decrease the available amount of resources. The decision based on the break and breakdown is easy because it occurs for all elements of the resource. On the other hand shifts are difficult. If the resource amount, which has to be switched off, is less than the amount of the resource in standby state, then there is no conflict. Issues arise, however, if the number of the free resources is less than the needed ones. This means we have to switch off allocated resources and activities will stop. The problem is even more complex, because the decision is not just a local one since most of the models have many shift changes at the same time.

The distribution of resources after a shift change is a typical knapsack problem, which is NP-Hard. The same problem exists when we have to increase the availability of many resources. Since there are numerous ways to handle this situation that occurs with most shift changes in a well-utilized facility we offer an interface for the user to implement the approach which is applied typically in their environment. For a comparison of different strategies we implemented the following approaches. Approaches for shift increase include:

- Increase first randomly allocated resources and then the idle.
- Increase first allocated resources with minimum remaining processing time and then the idle.
- Increase first allocated resources by activities, which allocate a minimum number of resources.
- Increase first allocated resources by activities, which are close to their due date.

Approaches for shift decrease include:

- Decrease first standby resources and randomly the allocated.
- Decrease first standby resources and then allocated resources with the maximum remaining processing time.
- Decrease first standby resources and then allocated resources by activities, which use the minimum number of resource units.
- Decrease first standby resources and then allocated resources by activities, which are far away from their due date.

5 CONCLUSION AND FUTURE WORK

We successfully created a simulator prototype designed specifically for the simulation of complex assembly lines (thousands of activities, realistic schedules, priority rules, resources and model restrictions). The tool allows the user to create new priority rules for the activity selection and experiment with the resource allocation problem. This is an appropriate environment for intensive research on the area of the workforce scheduling. At the moment we created test cases of project networks (Figure 8) with up to 5000 activities

and 20 resources with realistic time plans. We validated and simulated successfully the test cases. Right now we are working to accelerate the logging of large models.

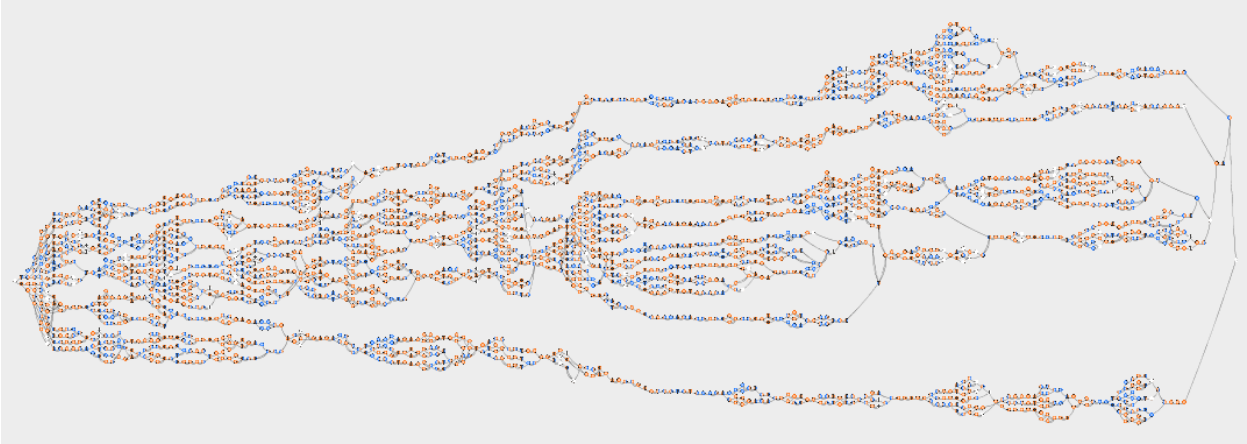

Figure 8: Test case with a self-generated project network with 5000 activities

ACKNOWLEDGMENTS

We would like to thank André Naumann for his implementation work and discussions during the project.

REFERENCES

April, J., F. Glover, P. J. Kelly, and M. Laguna. 2003. "Practical Introduction to Simulation Optimization." In *Proceedings of the 2003 Winter Simulation Conference*, edited by S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 71-78. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Elmaghraby, S. E. 1977. *Activity Networks: Project Planning and Control by Network Models*. New York: Wiley.

Holzer, A. J. 1999. "Prioritätsregelnbasierte Ressourcenplanung für Projekte mit komplexer Ablaufstruktur." PhD Thesis, University of Passau.

Majohr, M. F. 2008. "Heuristik zur personalorienrteirten Steuerung vom komplexen Montagesysteme." PhD Thesis, Dresden University of Technology.

Mika, M., G. Waligóra, and J. Weglarz. 2005. "Simulated Annealing and Tabu Search for Multimode Resource-Constraint Project Scheduling with Positive Cash Flows and Different Payment Models." *European Journal of Operational Research* 164(3):639-668.

Neumann, K., and M. Morlock. 1993. *Operations Research.* 1st ed. Munich: Hanser.

Pappert, F. S., E. Angelidis, and O. Rose. 2010. "Framework For Simulation-based Scheduling of Assembly Lines". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 1690-1698. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Patterson, J. H., R. Slowinski, F. B. Talbot, and J. Weglarz. 1989. "An Algorithm for a General Class of Precedence and Resource-Constrained Scheduling Problems." In: *Advances in Project Scheduling,* edited by R. Slowinski and J. Weglarz, 3-28. Amsterdam: Elsevier.

Sauer, J. 2002. "Multi-Site Scheduling Hierarchisch koordinierte Ablaufplanung auf mehreren Ebenen." PhD Thesis, University of Oldenburg.

SEMI. 2004. *SEMI E10-0304: Specification for Definition And Measurement Equipment Reliability, Availability And Maintainability*. North American Regional Standards Committee, Semiconductor Equipment and Materials International.

Steinberg, D., F. Budinsky, M. Paternostro, and E. Merks. 2008. *Eclipse Modeling Framework*. 2nd Edition. Addison-Wesley Professional.

Weyprecht, P., and O. Rose. 2011. "Model-driven Development of Simulation Solution based on SysML starting with the Simulation Core." In *Proceedings of the Spring Simulation Conference*.

AUTHOR BIOGRAPHIES

EVANGELOS ANGELIDIS is a PhD student at Dresden University of Technology. He is a member of the scientific staff of Prof. Dr. Oliver Rose at the Chair of Modeling and Simulation. He received his M.S. degree in Computer Science from Dresden University of Technology. His e-mail address is evangelos.angelidis@tu-dresden.de.

FALK STEFAN PAPPERT is a Research Engineer at D-SIMLAB Technologies. His focus is on simulation-based scheduling and optimization of production systems. Furthermore he is a PhD student at Dresden University of Technology as a member of the scientific staff of Prof. Dr. Oliver Rose at the Chair of Modeling and Simulation. He has received his M.S. degree in Computer Science from Dresden University of Technology. He is a member of GI. His e-mail address falk.pappert@d-simlab.com.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Institute of Applied Computer Science of the Dresden University of Technology, Germany. He received an M.S. degree in applied mathematics and a Ph.D. degree in Computer Science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of IEEE, INFORMS Simulation Society, ASIM, and GI. His web address is www.simulation-dresden.com and his e-mail address is oliver.rose@tu-dresden.de.