Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

SIMULATION ASSISTED MATCH-UP RESCHEDULING OF FLEXIBLE PRODUCTION SYSTEMS SUBJECT TO EXECUTION EXCEPTIONS

Wilhelm Dangelmaier
Kiran R. Mahajan
Mark Aufenanger
Thomas Seeger

Heinz Nixdorf Institute
Business Computing, esp. CIM
University of Paderborn
Fuerstenallee 11, 33102 Paderborn, GERMANY

ABSTRACT

An immense amount of research work has been done in the areas of scheduling and re-scheduling of various types of manufacturing systems. In this paper we present a simulation assisted approach to rescheduling complex production system configurations subject to execution exceptions. Issues like how to bring the deviation of a schedule due to exceptions back to its original trajectory and how to do this in real-time without affecting co-ordination problems on the shop floor are addressed. Our results show that combining simulation and optimization for rescheduling indeed helps to achieve both these objectives and that this approach proves to be promising to help reduce chaos in today's dynamic manufacturing environments.

1 INTRODUCTION

These days, manufacturing systems are getting more complex. This complexity is due to several factors like complex production systems, product variety and uncertainty in business processes. Efficient techniques for planning and re-planning the entire supply chains to cope with the complexity and business dynamics are more and more apparent. Within the manufacturing and distribution supply chain, manufacturing re-scheduling is one area, which will be vital to the success of the manufacturing organization, and which has attracted an increasing volume of research. Especially, the day-to-day operations and the exceptions arising out of uncertain characteristics of the system have to be handled efficiently. It is a well known fact that ERP systems do not have the capability to provide detailed scheduling and rescheduling systems. Moreover, the simulation assisted replanning of such systems becomes important as they offer the possibility to interactively analyze the system by testing the effect of exceptions and the resulting solutions before implementing them on the actual production system.

In the last decade, several configurations of production systems were researched in these areas. In this paper, we seek to address the rescheduling of a complex system configuration – the flexible flow shop with parallel and identical machines on one or more stages. For such kind of system configuration, there do not yet exist rescheduling and real-time control methods. Here the rescheduling methods should especially answer questions like how to reschedule (change the original or predictive plan) as little as possible? Is rescheduling really required? If yes, how can we get the deviation back to its predictive trajectory? If we decide to reschedule, how can we make sure, future execution is not affected due to the rescheduling step, or if it is affected, how can we estimate the impact and solve these problems now, as a result of the rescheduling action? Real-time control should be strongly coupled with a rescheduling system because a rescheduling system should be able to compute, present and implement a solution before the real-world evolves way too ahead, leaving the solution infeasible to implement. In other words, a rescheduling system should address co-ordination problems that may occur on the shop floor, and simultaneously also address the above questions.

In the areas of rescheduling, three methods (simulation and non-simulation based) have been widely researched namely completely reactive approaches (Wu and Wysk 1989, Harmonosky et al. 1997), robust scheduling approaches (Wu et al. 1999) and predictive-reactive approaches (Haldun et al. 2005), with the last being well studied. In the predictive-reactive approaches (where the predictive part generates a schedule for the next planning period and the reactive part modifies (reschedules) this schedule in the real world as exceptions happen with low frequencies), two main questions have been addressed in literature like when to initiate a rescheduling action and what the rescheduling action should be. The first question is answered by developing approaches like event-driven rescheduling, periodic rescheduling and continuous rescheduling (Haldun et al. 2005 – the reader may refer for a thorough treatment).

In the areas of real-time control, (Manivannan and Banks 1991), describe a method of providing near optimal rescheduling solutions instantaneously. They however do not consider that some parts of a real-system can/may nevertheless continue to evolve despite exceptions in other parts of the system. Evidently, all the above approaches do not deal with the questions addressed by our research.

In this paper, we describe a simulation assisted rescheduling system for managing a flexible production system environment which is subject to execution exceptions with low frequencies. We think it is best to use a combination of simulation and optimization for this purpose. The optimization algorithm can reduce the computational burden on simulation by presenting to it fewer good candidate solutions and compute solutions feasible for real-time control (considering the evolving real-world), whereas simulation can consider the effects of wider production system elements and be used for providing a problem free execution of the rescheduling solution by predicting accurately in the future. Details and further information are given in the next section.

We prove here how simulation is a vital tool in organizing the production system on deviations and exceptions. We also show encouragingly that simulation provides answers to some difficult system configurations, said to be at least N-P hard (Brah and Hunsucker 1991, Cheng et al. 2001, Phadnis and Irani 2003), for the kind of problems discussed earlier. In the next section we describe the overall framework of the system, followed in the next sections by describing the implementation, results with discussions and conclusions.

2 THE OVERALL FRAMEWORK

Our overall framework is preceded by a simulation assisted predictive scheduling system not described in this paper. This predictive scheduling system generates a feasible schedule to implement in the real-world for the next planning period. The system considers a complex production system with several decision points, each providing several alternative policies or routes on each production stage, limited buffer spaces, and product variety (different processing times) with and without product routings and delivery time requirements, which add to the complexity of the problem. As the execution of this schedule proceeds in the real-world, exceptions occur with different consequences on the processing of jobs. These exceptions cause deviations to the schedule in terms of job starting time deviations, and deviations from the overall makespan. Exceptions considered here are generalized with the results of the exceptions being more important than what the exceptions are. As an example, let the exception be anything, the result is that a job is stopped, or a job is damaged.

Figure 1 shows the overall concept of our system. The system consists of four modules namely the Real-Time Monitoring and Control (RTMC) module, the Simulation Evaluation component, the Optimization Algorithm and the Scheduling Controller. The RTMC receives the system status and events on a regular basis. Based on certain conditions set by the user, the Exception and performance analysis component activates the Simulation Evaluation component as soon as an exception occurs. This runs the simulation with the exception to compute the deviations and upper bounds of the makespan. Thereafter, the Optimization Algorithm is invoked which computes the possible rescheduling alternative solutions along with guidelines (Adaptation Synchrony Analysis - ASA) on co-ordination problems and passes back the control to the Simulation Evaluation component. This ASA computes the evolution of the real production system and decides if and how the simulation should evaluate the rescheduling solution based on whether co-ordination problems will arise in the real world. This Simulation Evaluation component starts evaluating the solutions and presents the final solution based on user criteria to the Scheduling Controller which passes it on to the RTMC to be finally implemented in the real-world. In the next section, more details about the algorithms are presented as a combined system of simulation and optimization.

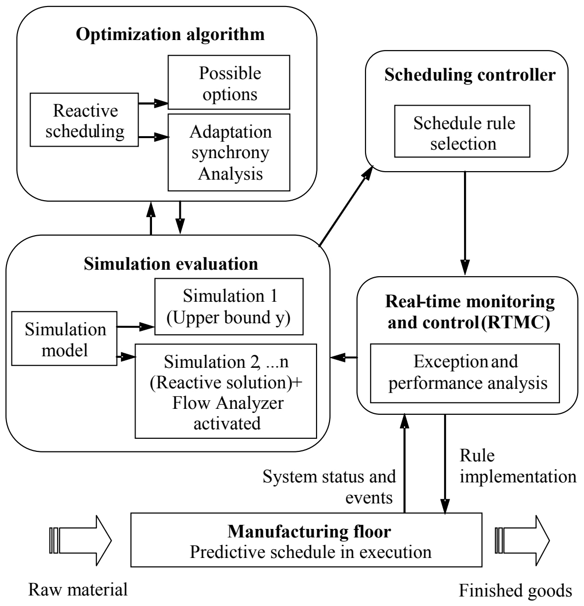


Figure 1: Overall concept of the simulation assisted rescheduling system.

The benefits and reasons for using this system are as follows. It is a well-known hypothesis that optimization algorithms (heuristics, genetic algorithms, etc) consider the number of jobs, number of machines and processing times on machines to compute a schedule. They cannot predict in the future and it is harder to consider the effect of system elements like buffer sizes, material handling equipment, etc in its computations due to the complex dynamics in the system. Due to this it is difficult to assess accurately the impact of the exception on the schedule. Simulation on the other hand, cannot by itself optimize (using one simulation run) but can predict accurately into the future and take into consideration all the additional elements of a production system in addition to the ones an optimization algorithm considers. However, simulation can be used to estimate accurately the impact of the exception on the schedule and to predict if the rescheduling solution can be implemented in the real world without causing further execution problems.

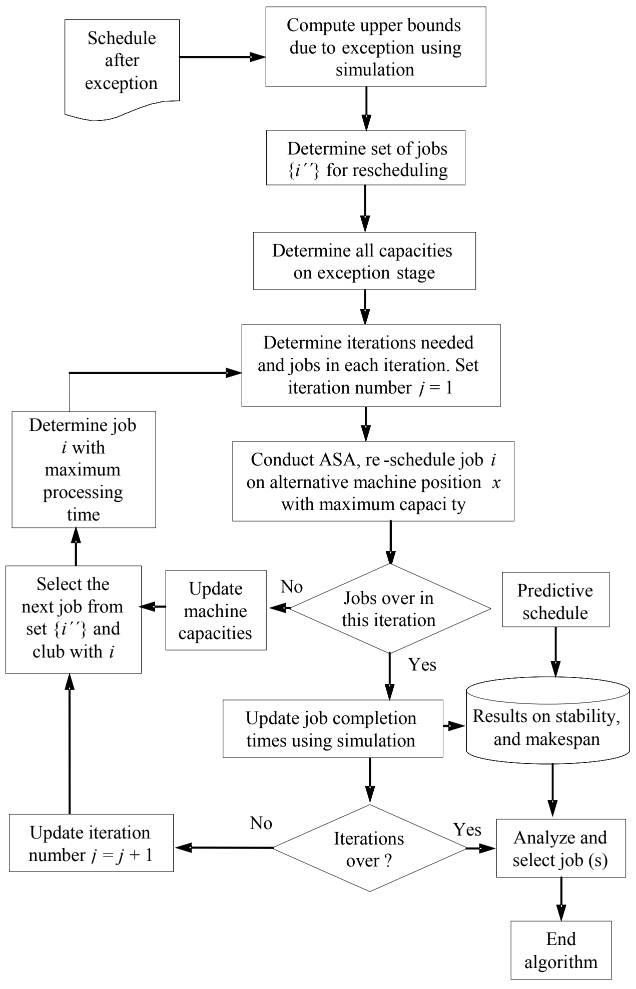

Schedule after exception
Compute upper bounds due to exception using simulation
Determine set of jobs {i´´} for rescheduling
Determine all capacities on exception stage
Determine iterations needed and jobs in each iteration. Set iteration number j = 1
Determine job i with maximum processing time
Conduct ASA, re-schedule job i on alternative machine position x with maximum capacity
Select the next job from set {i´´} and club with i
Update machine capacities
No
Jobs over in this iteration
Predictive schedule
Yes
Update job completion times using simulation
Results on stability, and makespan
Update iteration number j = j + 1
No
Iterations over ?
Yes
Analyze and select job (s)
End algorithm

Figure 2: Simulation assisted rescheduling system.

A combination of simulation and optimization algorithms can hence solve the problem of not being able to encompass the entire system, whilst still providing a better and feasible solution in a limited amount of computational efforts with good accuracy. In the next section, this combined simulation and optimization system is presented with its detailed description.

3 DETAILED SYSTEM DESCRIPTION

In this approach, the fundamental idea is to try as much as possible to reduce starting time deviations for all directly and indirectly affected jobs and to come back the original schedule upon the occurrence of a exception by making selective local changes to the schedule in such a way that performance indicators such as starting time deviations, sequence deviations for all jobs or the makespan are as close as possible to the original schedule. Figure 2 shows the rescheduling system.

As soon as an exception happens, an upper bound is computed. The upper bound computation can be performed using an algorithm or a simulation system. In our case, since the system is quite complex (customer centric parts with varying part processing times with multiple routings between stages and buffer elements), it is required to use simulation for the computation of the upper bound. The most important reason to use simulation is to be able to help in the computation of the capacity that exists where rescheduling could possibly take place. Another reason to use simulation for the computation of the upper bound is due to the system configuration – it is mostly the case that the upper bound is not equal to (but in fact is less than) the original predictive makespan plus the exception duration!

This is due to the fact that with the exception some jobs can advance to the next stage on the parallel machine configuration, thus reducing the makespan further. This upper bound represents the worst case scenario as a result of doing nothing, or in other words, the result of not rescheduling. Then, on the same stage where the exception happened, we determine the amount of capacity that is available on all the alternative machines using a heuristic algorithm with input from the simulation run.

A number of jobs are then created which are candidates for re-scheduling (the rescheduling set i''). These jobs are the ones which should have started on the machine during the duration of the exception, according to the predictive schedule. For these candidate jobs, the total number of simulation iterations and the number of jobs within each iteration is computed and set.

Jobs are rescheduled using a newly introduced "Updating Capacity Principle". In the first iteration, Job 1 is selected (from the set i'') and is planned to be re-scheduled by the algorithm at a certain position P1 at time t1 on the alternative machine where maximum capacity exists. An Adaptation Synchrony Analysis (ASA) is also conducted at this step, which attempts to solve co-ordination problems, which is described separately in the next section. Since there is only one job in the first iteration, the system simulates this constellation and saves the results in a database. Following this, the algorithm updates the iteration number to the second iteration and gets the associated jobs with this iteration (jobs 1 and 2 from set i''). From these two jobs, the one with maximum processing time on the stage where the exception occurred is selected. This job is then placed at a certain position at a machine with highest capacity. Note that at the beginning of each iteration, the inputs about the highest capacity are obtained from the upper bound simulation run. Again, just before conceptually rescheduling the ASA is conducted – explained in the next section. If there are more jobs in the iteration (true – second iteration has two jobs), the machine capacities are updated using a heuristic, new information on rough capacities are obtained and the second job is rescheduled on the newly calculated highest capacity machine. This is shown as an example in Figure 3.

At the end of this iteration, the jobs are placed at the calculated positions and a simulation run is carried out to save results on performance indicators on this run. If there are more iterations, the same process is repeated, where within each iteration, jobs are selected in the order of maximum processing times and machine capacities are updated intermittently. This method of updating capacity in our experience gives much better results than other methods we have developed and tested. A reason for this is that rescheduling jobs with maximum processing times first on machines with highest capacities seem to prioritize jobs which results in faster processing for jobs and balancing of loads on machines. This method of combining simulation and optimization results in fewer simulation runs (we do not simulate each and every constellation), whilst still providing good results (shown later).

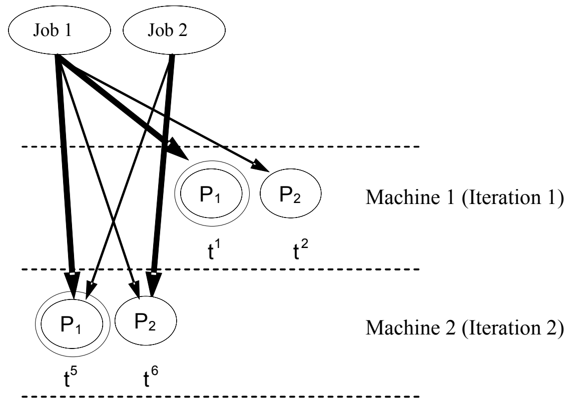


Figure 3: Tree of the iterations for positions and job rescheduling.

Simulation is used to compute the exact maximum capacity for a system that can contain anything from varying processing times on machine, buffers and transportation elements. The optimization algorithm then takes this as an input at the start of each iteration to keep the number of simulation iterations to a bare minimum. As mentioned earlier, just before each sub-iteration starts within each iteration, an Adaptation Synchrony Analysis (ASA) is conducted. This may alter the exact position (as calculated by the initial computations of the optimization algorithm) where a job may be rescheduled considering the future system state. This is explained in the next section.

3.1 Adaptation Synchrony Analysis (ASA) During Rescheduling

Adaptation synchrony defines how the actual process continues during its computation and modification. In this analysis, the problem that the real world production system will continue to progress and evolve over time, while we calculate and implement a rescheduling solution is undertaken. As seen in Figure 4, when the rescheduling computation starts, the data at the time of the exception was considered for computation. When the rescheduling computations and implementations would be finished (assuming we know how much time is needed to implement a change – not an unrealistic assumption), the real system state has evolved to a new state (corresponding to data 2).

In other words, the data used for the computation of a rescheduling solution corresponds to the wrong state of the system as marked by the solid grey arrow. Still in other words, the position where we wish to reschedule a particular job may not be available, since a job around that position has already progressed ahead in time, making it infeasible to attempt the rescheduling step.

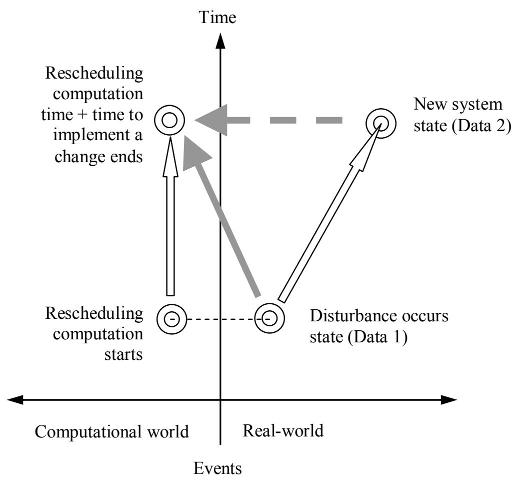

Time
Rescheduling computation time + time to implement a change ends
New system state (Data 2)
Rescheduling computation starts
Disturbance occurs state (Data 1)
Computational world
Real-world
Events

Figure 4: Real-world state evolution and interaction with the rescheduling system.

The following is the pseudo-code of the ASA system and the description of each point where necessary:

1. Let i'' = number of jobs selected for rescheduling,
2. Let j be the machine selected for rescheduling,
3. Let P_1 at time t_1 be the position calculated by the optimization algorithm for job i,
4. Get the job change\implementation time t_{ch} from the "Change chart(s)" for job i.

The idea here is to include the computational times for the simulation runs, the computational times of the optimization algorithm, and most importantly the time required to actually make actual changes on the shop floor which would include times required for taking jobs out of their current position, times required for transportation and times required for setting the jobs in their new positions. Other actual times required for changes can be included in the system. The computation times (iteration time using simulation) duration can be estimated to a high degree of accuracy as we have already carried out simulation runs earlier (the upper bound simulation for instance) from which we can determine the time required for a particular iteration or iterations. The times to actually reschedule a job from one machine to other on the shop floor can be estimated a priori (line operators can fill up a "Change chart(s)" with relative ease for each job change on other machines). This time will include additional times like setup times, transportation times, etc. All these times are added to result in a *total time* to be considered in this analysis.

5. Get the times required for the future simulation iterations as t_{Sim} and compute *total time* as $t_{total} = t_{ch} + t_{Sim}$.
6. Compute the time t_{free} machine j (the one where job i is going to be rescheduled) will become free after machining one or more jobs to take in job i. t_{free} = remaining processing times for the job currently being processed on machine j + processing times for some or all of the remaining jobs on previous buffer of machine j.
7. If $t_{total} > t_1$, then Compute new position $P_{1\ new}$ with $t_{1\ new}$ in such a way that $t_{1\ new} > t_{total}$ and t_{free} but with $P_{1\ new}$ (position between one of the jobs that machine j has to machine) just after a job that machine j will machine at a later time period. Decide to proceed with simulation iteration on the new position.

An investigation is carried out as to whether the position, where we can seek rescheduling on other machines is available *after* the expected duration of *total time* or not. When there are events where factors cannot be considered in the total times, a provision is made in the simulation model to account for such events by scheduling them at a later – appropriate time. The ASA system will determine the time when the machine on the same stage becomes free (ready to take in another job which is part of the positions). A comparison is then made between the time the machines become free and the *total time*. If the *total time* exceeds the rescheduling position (each position also has a time), then obviously, we may not reschedule the job at that position. This is because by the time we reschedule this job physically the real system is ahead in time, consequently meaning that the position calculated by the optimization algorithm is no longer available. In such a situation the ASA system then determines if the next position is available. If it is available, it reschedules the job at the new position, thus overriding the position calculated by the optimization algorithm. The simulation iteration is then performed with the new position at the end of the iteration with the new positions calculated.

8. Else, decide to proceed with simulation iteration with position P_1.
9. If more jobs in set i'', select next job $i+1$ and go to step 2, else, end iteration and simulate all jobs at the decided positions.

To put the above in different words, the ASA system places a job at an appropriate position by computing in the future and by considering the worst case scenario that the current (iterated) job(s) will be finally selected for rescheduling in the end after going through all iterations! After each iteration, the results on the performance indicators are saved in each iteration as explained earlier.

After considering each iteration for re-scheduling, comparison is made between the predictive schedule and the result of re-scheduling the jobs at various iterations. Some job iterations may result in values of makespan higher than the predictive schedule, but result in lower starting time deviations with some sequence deviations. As a result, the user is presented with these results, in a consolidated form. The user then selects a particular rescheduling solution and does the post rescheduling analysis using the simulation based Flow Analyzer Module (FAM), which is described in the next section. This post rescheduling analysis will reveal if the rescheduling solution will have problems in the future execution when actually implemented in the real-world. In this analysis, the FAM system will solve the potential future problems by over-riding as less jobs as possible and present a final solution. The final solution deviations and performance data are recorded and presented to the user. The final rescheduling solution is compared to the upper bound simulation run to find out if it's worth to implement this solution in the real-world or not. The final solution is then given back to the scheduling controller and on to the RTMC module as seen and explained in Figure 1.

3.2 Post Rescheduling Solution Analysis

Once a solution has been computed using the system, the user can carry out post rescheduling solution analysis. This analysis basically checks if the rescheduling solution will cause disruptions in the real-world if implemented as calculated. In other words, it may suggest further changes to the rescheduling solution to avoid problems in the real-world due to the rescheduling action calculated by the system (described in the previous section). Hence, if the user selected to do this analysis, the system proceeds with a simulation run, and analyses the directly and indirectly affected jobs. If required, it over-rides the jobs from the original predictive schedule. A Flow Analyzer Module (FAM) is implemented to conduct this analysis. This FAM system consists of decision points within the simulation model placed where a job can have an alternative path. Within these decision points, rule generators are placed. These rule generators have the code which analyze the situation locally for problems, and provide an alternative rule if certain conditions are fulfilled. During the run-time of the simulation, as and when a job reaches a decision point, the rule generator checks for the local conditions. If problems such as bottleneck occur, i.e. conditions are satisfied, it solves the problems by providing alternative rules (routes) to the job. Note that during this post rescheduling analysis, if it is revealed that additional rescheduling may have to be carried out, the ASA need not be carried out here because the jobs further rescheduled by the FAM are only rerouted from decision points (not from their original destinations).

The user can decide between a rescheduled plan with his performance measures perhaps with problems which will occur in the future or a rescheduled plan with performance measures without problems in the future. Both these alternatives have implications on make-span, and optimization criteria.

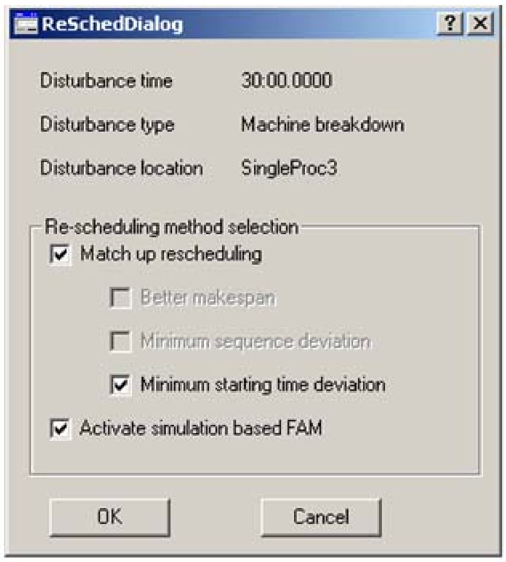
4 IMPLEMENTATION AND RESULTS

The entire system was implemented within the simulation software *Technomatix eM-Plant* as an add-on. The system was developed and tested on an Intel Celeron Processor, 1.06 GHz, and a 256 MB RAM machine run using Windows XP operating systems.

Figure 5 shows the screen shot of the rescheduling window for selection of options. We assume that we receive events from the real-production system on a continuous basis. This window comes up when the exception is detected by the real production system (in our work the real-production system is substituted by the simulation system). In this window, the user can select which method of rescheduling he wishes to use and settings like which performance indicators he wishes to achieve like minimum starting time deviations or minimum sequence deviations or perhaps a better make-span.

He can also select the option that will solve the problems that may occur in the future due to a particular rescheduling action by selecting "Activate simulation based FAM". The user or line operators can separately fill up the change charts as shown in Figure 6, 7 and 8. These charts are respectively for the time required to remove a job from its current position, transport it to the new location and set it up on the new location. This chart is implemented for each stage and the times shown are required to change the job from its current position to a new position. So for instance, job 1 requires 20 minutes to be changed from its current machine to machine 1, 2 and 3. Figure 9 and 10 shows the screen shot that is activated at the end of the analysis, giving out the complete results of the analysis.

A test result is shown to demonstrate the effectiveness of the developed approaches and concepts. The match-up rescheduling system along with the ASA and post rescheduling analysis were tested. The test is done with the ASA and with the FAM considering variation of job processing times on all stages. The data for the first test is shown in Table 1.

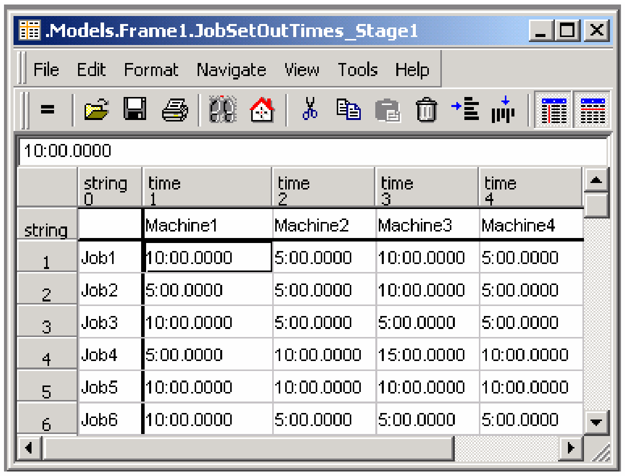


Disturbance time	30:00.0000
Disturbance type	Machine breakdown
Disturbance location	SingleProc3

Re-scheduling method selection

- [✓] Match up rescheduling
  - [ ] Better makespan
  - [ ] Minimum sequence deviation
  - [✓] Minimum starting time deviation
- [✓] Activate simulation based FAM

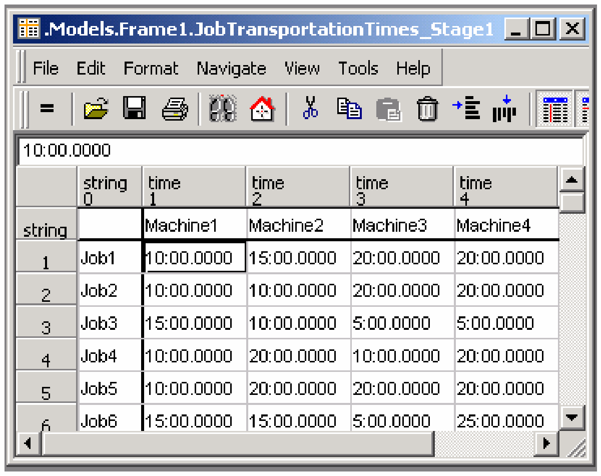
Figure 5: Rescheduling dialog activated on user selected criterion.



10:00.0000

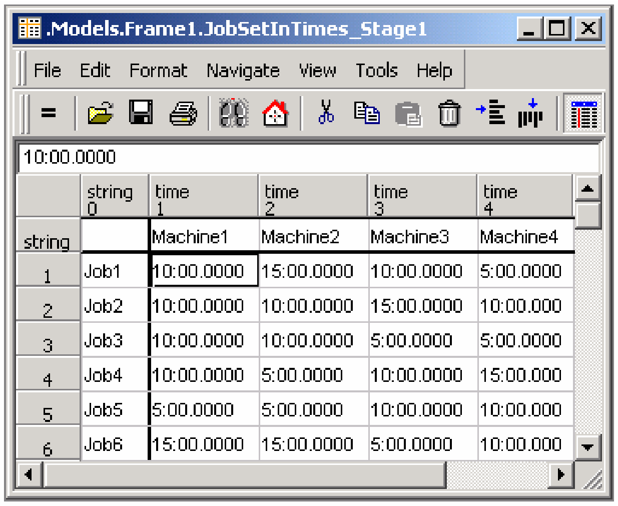
	string 0	time 1	time 2	time 3	time 4
string		Machine1	Machine2	Machine3	Machine4
1	Job1	10:00.0000	5:00.0000	10:00.0000	5:00.0000
2	Job2	5:00.0000	5:00.0000	10:00.0000	5:00.0000
3	Job3	10:00.0000	5:00.0000	5:00.0000	5:00.0000
4	Job4	5:00.0000	10:00.0000	15:00.0000	10:00.0000
5	Job5	10:00.0000	10:00.0000	10:00.0000	10:00.0000
6	Job6	10:00.0000	5:00.0000	5:00.0000	5:00.0000

Figure 6: Change chart – set out factor


.Models.Frame1.JobTransportationTimes_Stage1
File Edit Format Navigate View Tools Help
10:00.0000

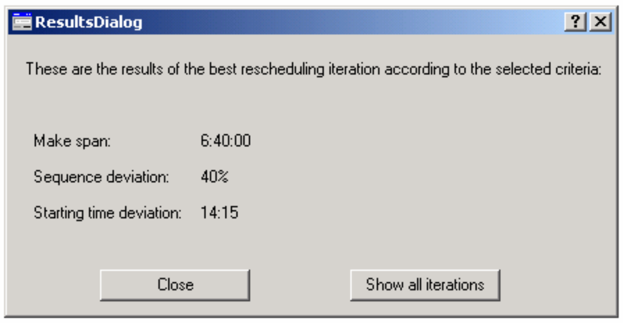
	string 0	time 1	time 2	time 3	time 4
string		Machine1	Machine2	Machine3	Machine4
1	Job1	10:00.0000	15:00.0000	20:00.0000	20:00.0000
2	Job2	10:00.0000	10:00.0000	20:00.0000	20:00.0000
3	Job3	15:00.0000	10:00.0000	5:00.0000	5:00.0000
4	Job4	10:00.0000	20:00.0000	10:00.0000	20:00.0000
5	Job5	10:00.0000	20:00.0000	20:00.0000	20:00.0000
6	Job6	15:00.0000	15:00.0000	5:00.0000	25:00.0000

Figure 7: Change chart – transportation factor


.Models.Frame1.JobSetInTimes_Stage1
File Edit Format Navigate View Tools Help
10:00.0000

	string 0	time 1	time 2	time 3	time 4
string		Machine1	Machine2	Machine3	Machine4
1	Job1	10:00.0000	15:00.0000	10:00.0000	5:00.0000
2	Job2	10:00.0000	10:00.0000	15:00.0000	10:00.000
3	Job3	10:00.0000	10:00.0000	5:00.0000	5:00.0000
4	Job4	10:00.0000	5:00.0000	10:00.0000	15:00.000
5	Job5	5:00.0000	5:00.0000	10:00.0000	10:00.000
6	Job6	15:00.0000	15:00.0000	5:00.0000	10:00.000

Figure 8: Change chart – set in factor

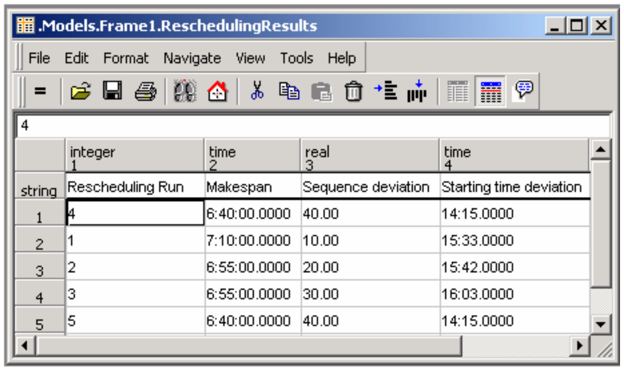


These are the results of the best rescheduling iteration according to the selected criteria:

Make span:	6:40:00
Sequence deviation:	40%
Starting time deviation:	14:15

Figure 9: Results dialog

The exception duration was set to randomly occur for duration of 40 minutes. It is assumed that once an exception occurs, information on its duration can be obtained relatively quickly. The time required to obtain such information can be put into the change chart. The aim was to test the effectiveness of the match-up rescheduling algorithm and the post rescheduling analysis using the simulation based FAM. Table 2 shows the overall results of the system. Table 2 shows the results. As seen it is helpful to reschedule in the real world in this case, because of better results obtained using the system as compared to the upper bounds. There was a reduction in starting time deviations and the make-span using the rescheduling solution as compared to the upper bounds.


.Models.Frame1.ReschedulingResults
File Edit Format Navigate View Tools Help
4

	integer 1	time 2	real 3	time 4
string	Rescheduling Run	Makespan	Sequence deviation	Starting time deviation
1	4	6:40:00.0000	40.00	14:15.0000
2	1	7:10:00.0000	10.00	15:33.0000
3	2	6:55:00.0000	20.00	15:42.0000
4	3	6:55:00.0000	30.00	16:03.0000
5	5	6:40:00.0000	40.00	14:15.0000

Figure 10: Expanded results of the system

Table 1: Data for test case 1

Nr. of jobs	**Nr. of stages**	**Nr. of machines**	**Processing times**	**Buffer capacity**
50	4	Stage 1:4	Varying from 15 to 25 minutes	Varying from 6 to 8 at each stage
		Stage 2:4		
		Stage 3:4		
		Stage 4:4		

Table 2: Summary of results for test 3

Method	**Make-span (hr:min.sec)**	**Starting time deviations (hr:min.sec)**
Best resched. iteration	5:05.00	15.33
Upper bound	5:45.00	19.12
Predictive schedule	4:25.00	
Post resched. Analysis	5:05.00	15.33

The most interesting result obtained using the ASA was that no jobs were selected for rescheduling using the FAM during post rescheduling analysis. The reason why this happened is because the ASA rescheduled the jobs at a later and appropriate point of time, thus, giving a rather realistic result – the result that by the time the changes would actually be ready on the shop floor, there would space to accommodate the jobs in the buffers, thus needing no more

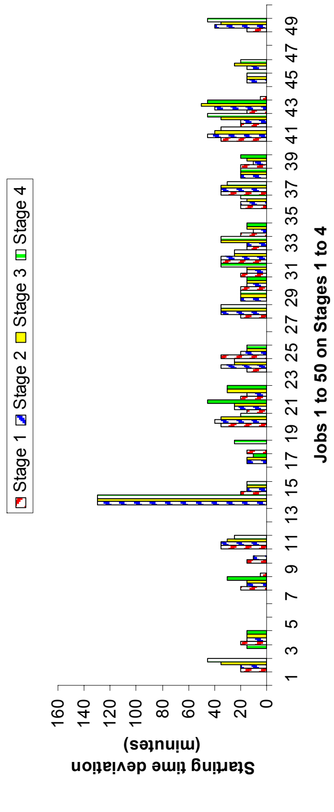


Figure 11: Starting time deviations of the rescheduling solution versus predictive schedule: Test 1

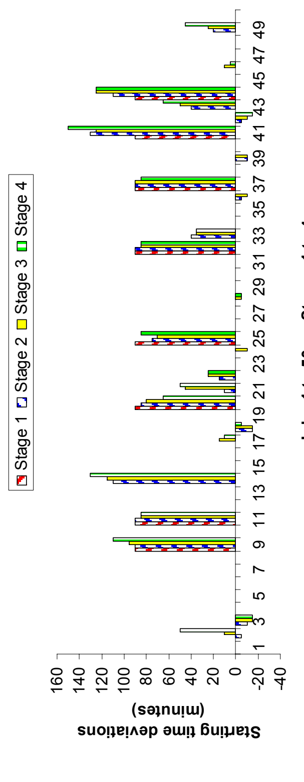


Figure 12: Starting time deviations of the upper bounds versus predictive schedule: Test 1

rescheduling (thus no further problems due to rescheduling) during the post rescheduling analysis phase. Figures 11 and 12 show the comparison of the starting time deviations if rescheduling were to take place, and if no rescheduling were to take place. Several tests more were carried out some suggesting that rescheduling would not at all be recommended. Cases where rescheduling is not possible at all due to lack of alternative capacity is also communicated to the planner. Though computational times for all the tests using the system were limited to several seconds, as imagined in case where system sizes are very big needing excessive computation times, the working principles of the ASA system still facilitate the use of simulation for the real-time control problems discussed here.

5 CONCLUSIONS

In this paper we presented a simulation assisted system for rescheduling complex production systems. Previous attempts to use rescheduling did not include adaptation synchrony analysis as well as post rescheduling analysis. The highlight of the system is the way it combines simulation and optimization to include both the aspects of adaptation synchrony and post rescheduling analysis. The adaptation synchrony analysis can be extended with relative ease to include more time factors of the change management process. Results and discussions showed that the system does indeed help in reducing the deviations from the predictive plans where possible, and answering questions like how much to reschedule, when to reschedule, how to reschedule as less as possible, how to avoid problems due to rescheduling, and how to consider the evolution of the real-world state while computing and implementing rescheduling solutions.

The problem of how to change as late as possible with a view to maintain system stability is being addressed at our research group. As seen from the results here, although the overall deviations were reduced, the number of jobs for which deviations did not exist after the exception had some deviations using the system developed.

This is nevertheless compensated for by performance indicators such as make-span and cycle times which were still better than the upper bounds. In our experience, the rescheduling system along with the ASA and FAM methods provides better results when the job processing time variations are low to no variations at all. In the near future, we will implement and test the rest of the decision support system and prove its applicability under a range of test conditions.

REFERENCES

Brah, S. A. and J. L., Hunsucker. 1991. Branch and Bound algorithm for the flow shop with multiple processors. In *European Journal of Operational Research* 51: 88-99.

Cheng, J., Y., Karuno and H., Kise. 2001. A shifting bottleneck procedure for a parallel machine flow shop scheduling problem. In *Journal of the Operations Research, Society of Japan* 29(2):140-156.

Haldun, H. A., M., Lawley, M. McKay and S. Mohan. 2005. Executing production schedules in the face of exceptions: A review. In *European Journal of Operational Research* 161:86-110.

Harmonosky, C. M., R. H., Farr and M. C., Ni. 1997. Selective rerouting using simulated steady state system data. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson, 1293 - 1298. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Manivannan, S. and J. Banks. 1991. Real-Time Control of a manufacturing cell using knowledge-based simulation. In *Proceedings of the 1991 Winter Simulation Conference*, ed. B. Nelson, W. D. Kelton, G. M. Clark, 251 - 260. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Phadnis, S. and S. Irani. 2003. Development of a new heuristic for scheduling flow-shops with parallel machines by prioritizing bottleneck stages. In *Transactions of the SDPS* 7(1): 87-97.

Wu, D. S. and R. A. Wysk. 1989. An Application of Discrete-Event Simulation to On-Line Control and Scheduling in Flexible Manufacturing. In *International Journal of Production Research* 27: 1603-1624

Wu, D. S., E. Byeon. and R. H. Storer. 1999. A graph-theoretic decomposition of job shop scheduling problems to achieve scheduling robustness. In *Operations Research* 47: 113-124.

AUTHOR BIOGRAPHIES

WILHELM DANGELMAIER studied Mechanical Engineering at the University of Stuttgart, Germany. Since 1981, he was director and head of the Department for Corporate Planning and Control at the Fraunhofer Institute for Manufacturing (Fraunhofer IPA). There he was responsible for the development of SIMPLE++ (today eM-Plant owned by UGS\Siemens). In 1991, Dr. Dangelmaier became Professor for Business Computing at the Heinz Nixdorf Institute; University of Paderborn, Germany. In 1996, he founded the Fraunhofer Center for Applied Logistics. Dr. Dangelmaier has published 510 technical articles in various magazines, journals and conferences. He has written chapters in 73 books, supervised more than 60 PhD thesis' and written 3 books on the subject of Supply Chain Management and Manufacturing. His principal interests today are models and tools for distributed production systems. His e-mail address is <dangelmaier@hni.upb.de>.

KIRAN R MAHAJAN completed his Ph.D. research at the Heinz Nixdorf Institute, group Business Computing, esp. CIM, Germany in 2007. He now works for Toyota Motors in Europe. He studied Mechanical Engineering at the Delft University of Technology (TU Delft) in The Netherlands. He also has a bachelor's degree in Production Engineering from the University of Pune (Govt. College of Engineering, Pune), India. His research interest is development of simulation based planning and scheduling systems for complex production environments. His e-mail address is <kiran@hni.upb.de>.

MARK AUFENANGER studied Business Computing, esp. CIM at the University of Paderborn. Since 2005, he is a research assistant at the group of Prof. Dangelmaier, Business Computing, esp. CIM. He is mainly interested in simulation of logistic systems. His email address is <marka@hni.upb.de>.

THOMAS SEEGER studied Business Computing, esp. CIM at the University of Paderborn. Upon graduation in 2007 he started working for Accenture. His research interests are simulation and optimization of complex logistics and production networks. His email address is <thomas.seeger@acccenture.com>.