Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

COMBINING AGENT-BASED SUPPLY NET SIMULATION AND CONSTRAINT TECHNOLOGY FOR HIGHLY EFFICIENT SIMULATION OF SUPPLY NETWORKS USING APS SYSTEMS

Hartwig Baumgaertel
Ulrich John

Department for Manufacturing and Supply Networks
Research Information and Communication
DaimlerChrysler AG
Alt-Moabit 96a
10559 Berlin, GERMANY

ABSTRACT

This paper introduces an approach for highly efficient simulation of supply networks in which several nodes use advanced planning and scheduling (APS) systems. APS systems increasingly become part of bigger companies IT landscape. Today, APS systems communicate only with the ERP system they sit on top of. If APS systems would be able to communicate with each other, qualitatively new processes for planning collaboration in supply nets could emerge. Simulation is accepted to be an useful approach for support of business process design. Design of planning processes which should exploit APS systems requires simulation systems with integrated APS functionality.

We augmented an existing, agent-based simulation system by an APS component which we developed based on finite domain (FD) constraint technology. In this paper, we present our simulation system with special focus on the APS component, and results of a simulation experiment which was used for the proof of concept.

1 INTRODUCTION

In many industries, companies tend to reduce their own operation to fields of their core competence. The depth of value added processes decreases, as outsourcing of many operations is performed. Closely coupled supply networks emerge. In such networks, management of the relationship with suppliers becomes more important than ever.

The key concept shaped out in last year is collaboration. It comprises the fact that the most critical factor for successful management of relationships in supply nets is that companies work together. Fields of collaboration range from sharing knowledge, data and IT systems over common process design and implementation up to common supply net controlling and assessment.

Another trend is that APS systems increasingly become part of bigger companies IT landscape. As of today, APS systems communicate only with the ERP system they sit on top of. If APS systems would be able to communicate with each other, qualitatively new processes for planning collaboration in supply nets could emerge.

Most projects on collaboration reported in the last year act mainly on the operational level. Sharing knowledge and data, e.g. production program plans, plans for new products and their production start, inventories, key performance indicators (KPIs) and even calculations increasingly become reality in industries like automotive, high-tech and consumer packaged goods. Commonly used systems which act above transaction level arise. The systems make data and knowledge visible for partners, perform simple calculations, e.g. comparisons of demands, inventories and capacities, create and send alerts in case of mismatches and support the workflow for solving problems. These systems are labeled as demand visibility, demand-capacity-comparison, and collaborative capacity management systems.

Although they help to solve essentially operational problems, these systems are not the high end of supply net management, for the following reasons:

- Despite the fact that most approaches are supported by web-based IT systems, they heavily rely on human action. The systems are usually not closely coupled with the ERP systems of the partners. Solving an alerted problem, e.g. a capacity shortage, requires from the user to interpret the data, understand the problem, and solving it by use of his primary planning system.
- The systems reported so far are mainly proprietary solutions for particular supply chains, not seldom only between two companies. Extending these approaches to much more partners, e.g. to the automotive supply network in it's broadness and

deepness, will induce new questions. Due to network effects of information flow and parallel decision making in different branches of the network, it is not clear if stable states can be reached: a solution for one sub-problem, i.e. a new production plan, can cause new problems in other sub-networks in which demand and capacity were aligned before. If such problems start to oscillate, the user will be confronted with tons of alerts in short time. This would inevitably de-motivate the user. She would start to ignore the alerts or work only on that they assume to be important.

- The systems provide no concepts for collaboration on tactical or even strategic level. But the foundation for supply network management comes from these levels: the better the partners in a supply net are aligned with regard to their strategy and their process designs, the better the operational work can be performed. Alerting operational problems is often not more than allaying symptoms of bad process design or alignment. Sustainable repair can be reached only by work on the higher levels.
- The most common approach for design of collaborative processes is to execute a pilot project with a software which is roughly specified in advance or chosen from the software market. This is again a very operational approach, which has several shortcomings: (1) implementation of software pilot creates effort for software installation and IT support, data gathering, software adaptation, user teaching, etc. Process design, which is the original aim, tends to be crowded out of the project focus. (2) there are often no unique causal relationships between processes and KPIs. Other changes in the environment can effect the same KPIs, i.e effects of the piloted processes are difficult to measure. (3) supply net management processes mainly have mid- and long term effects. Pilot projects will be performed usually over periods of three to six month. The effects of process changes can often not be shown within the pilot project.

Two essential problems can be derived from these observations. The first is to reduce the need of human action in planning collaboration. The second problem is to shift the work on design of collaborative approaches from the operational to at least the tactical level.

A solution approach for the first problem is the proper use of APS systems in collaboration, the second problem can be addressed by use of simulation for process design, as described in Wilke (2002). A pioneering approach, addressing both topics in combination, was presented by Lendermann, Gan, and McGinnis (2001). The approach presented in this paper also addresses the combination of supply chain simulation and APS systems, but on a higher level of abstraction. Instead of using an event-based simulator and an commercial APS system, we use an highly efficient special purpose supply net simulator and an APS component which was specifically developed for this simulator. With this approach, we can simulate supply networks of real size where several APS systems are in use. Simulations have runtimes of few minutes, so that large scenario spaces can be explored and interactive use of the simulation is possible.

We would like to present our approach in the remainder of this paper in the following way. After discussing the motivating questions more detailed in the following two sub-sections, we introduce in Section 2 the ingredients of the technical system SNS-APS, the agent-based supply net simulator SNS and Finite Domain constraint solving. In Section 3 we present our APS system and it's integration into SNS. One sample scenario we investigated will be presented in Section 4.

1.1 Collaborative Planning by Interacting, Intelligent Planning Systems

An extension to the human-centric collaboration approach is to develop automated, intelligent information exchange and processing tools for supply net planning. Of course, this does not mean to get the human completely out of the process: the planning systems will need interaction with their users too, but only in cases when it is really necessary, and then, by presenting solution proposals instead of problem descriptions. We claim that there is a lot of routine activity in demand visibility and capacity comparison processes from which human could be unburdened.

It is also absolutely out of question that central planning approaches cannot work. Some software vendors provide a central planning engine on a web server with the possibility to connect many clients via Internet and call it "collaborative planning tool". These approaches do neither understand the importance of planning and decision autonomy of independent companies nor the effects of computational complexity which make this approach infeasible for large networks both from the organizational and from the technical point of view.

We started to explore the possibility of distributed, collaborative supply net planning based on interacting planning systems in a research project in 2002. Since nobody knew in advance whether this kind of collaborative planning is a useful approach or not, we developed a proof of concept under laboratory conditions. The central questions shaped out first in the project were:

1. Which information is appropriate to be sent at which time from which node to which node in the supply network?
2. How can this information be created and how should it be processed, as far as possible automatically?

The work was mainly influenced by our previous research on distributed, coordinated, constraint-based planning for DaimlerChrysler's internal production network (Baumgaertel 2000) and by our recent activities on supply net simulation which are described in the next sub-section.

1.2 Simulation-Based Process Design

An alternative to the mainly IT driven pilot projects is to design collaboration processes first, check them and make them robust, and finally define supporting IT tools. Today, business and information processes are usually defined by process description languages and tools, e.g. ARIS, alfabet, OMEGA, UML etc. These tools are based on static process descriptions and provide hardly support for the analysis of the dynamic behavior of processes. Simulation is the logical way to solve this problem.

Together with ERIM CEC (which migrated to Altarum meanwhile) we developed a specific supply net simulation approach. It meets the requirements to the right modeling level of abstraction, high performance, and focus on investigation of information and decision processes, e.g. production planning, demand forecasting, material procurement and dispatching, and capacity management. It is based on agent-based modeling and inspired from system dynamics. The origins of the approach can be found in Parunak, Riolo, Savit, and Clark (1999), the result of our common work of ERIM CEC in 2000 is presented in Baumgaertel, Brueckner, Parunak, Vanderbok, and Wilke (2003). This system was completely redesigned and implemented in Java in 2002 and is now called SNS. It is shortly described in Section 2.1.

This simulation system was augmented by supply chain planning capabilities. The operation of a sample supply net was simulated for different kinds of information processing and context settings.

2 INGREDIENTS OF SNS-APS

In this section we introduce the two basics of the technical part of the APS-augmented supply net simulator. The first ingredient is obviously the supply net simulator, SNS, the other is Finite Domain constraint solving.

2.1 SNS

SNS is a supply net simulation tool which combines agent-oriented modeling with state transitions over time (time step based simulation).

The SNS modeling architecture consists of two hierarchy levels. The upper level provides agents for the inter-organizational view, the lower level for the intra-organizational model of manufacturing companies. The agent types of the upper level are Producer, Supplier, Consumer, Mailer and Shipper. The intra-organizational level follows the structure of the SCOR model (Supply-Chain Council 2003). The main elements of this model are SOURCE, MAKE, DELIVER and PLAN. While SOURCE, MAKE and DELIVER comprise information as well as material flow, PLAN is a pure information flow level. PLAN spreads into three sub topics: PLAN DELIVER, PLAN MAKE and PLAN SOURCE. The corresponding agent types are Part Buffer (SOURCE), Production Resource Manager (MAKE), Finished Goods Buffer (DELIVER), Part Procurer (PLAN SOURCE), Demand and Production Planner (PLAN MAKE) and Customer Order Center (PLAN DELIVER). Additionally, a Capacity Manager exists which is responsible for adjusting production capacity to the demand.

Agents are characterized by internal state variables, for example inventory levels, work-in-process level, order amounts, orders in pipeline, production plans etc.

Since the system is time step based, the user has to define the interpretation of a basic time unit, called basic tact. Useful tact interpretations range from one week over one day, one shift down to few hours.

For each state variable s of the agents exists a transition function F which describes the computation of the value of s at a time step t from the values of arbitrary state variables at time step t-1 and from values of particular state variables at t. The computation of the value of a state variable is also called simulation action. They belong to the following kinds of activities:

- Calculation of demand for the current time step and creation of a demand forecast
- Receiving incoming information
- Receiving incoming material
- Processing information
- Sending out information to supplying nodes
- Processing of material
- Sending out material according to received orders.

The classes of simulation actions will be performed in sequence, while each particular action is performed in parallel for all agents of the corresponding agent type. Whether a state variable is computed from other state variables values at t-1 or at t depends on the sequence of calculations of the action classes.

Production planning is part of the information processing. State variables exist for the demand forecast, the production output plan, the production release plan and the part demand plan. They belong to the Demand and Production Planner (DPP). It calculates the demand forecast from customer forecast or from demand history, the production output with respect to desired finished goods inventory levels, initial inventory, work in process, production capacity, and production batch size, the release plan from output plan and production cycle times, and the part demands from release plan and bill of materials. The planning algorithm is an adapted MRP algorithm, extended to production capacity treatment. The user can choose between sev-

eral forecasting and planning modes, e.g. forecast from weighted average of past demand or using customers forecast, and planning with or without regard to production capacity. When capacity has to be treated and demand peaks over capacity are recognized, DPP tries to shift production orders to earlier time steps (production in advance) to avoid delivery shortages.

But even the best planning procedure implemented in SNS cannot take finished goods buffer capacity or part availability into account. In practice, these kinds of restrictions are usually treated, most often by human planners during sequential planning procedures, supported by specific software tools. Further, companies optimize their plans to keep the finished goods inventory near an optimal level or to keep the production rate constant. Real APS systems support these kinds of planning optimization.

The aim of the APS system was consequently to extend the set of available planning methods of the DPP by methods which provide these planning capabilities. This leads to solving of complex, constrained integer optimization problems. Writing an algorithm for that in Java from scratch would obviously not be appropriate. The appropriate tools for solving this kind of problems are either mathematical optimization technology, i.e. mixed integer programming (MIP), or finite domain (FD) constraint solving. In our case, FD constraint solving was the better alternative since execution efficiency was more important than optimization. The overall performance of the simulator should be kept as far as possible. This could better be reached by employing the explicit control capabilities for solving the non-deterministic search problem in the FD approach.

2.2 FD Constraint Technology

This chapter gives a short introduction to constraint programming. For a deeper understanding we refer to Marriot and Stuckey (1998) and Bartak (2002). Starting in the early nineties, Constraint Programming was developed more and more as a new programming paradigm for the solving of complex planning and scheduling problems.

In Constraint Programming, a problem is modeled as a variable set and functions/ relations over the variables. Generally, those relations specify conditions (also often named restrictions or constraints) on valid variables assignments. In the classical programming paradigm, the fulfillment of those conditions (sometimes also named constraints) can be checked only after the variable assignment. In Constraint Programming, the computation of the specified constraints will be done substantially before starting the generation of variable values (labeling). That means, the constraint programming paradigm turns upside down the sequence between variable assignment and computation which is necessary in classical programming.

In the case of constraint programming over finite domains, each variable is annotated with a finite set of values (domain) that is initially a superset for all possible assignments of the variable. A variable with annotated domain is called constraint variable.

Constraint variables can be connected by a constraint which defines a relation between them, i.e. it describes permissible combinations of values. A constraint net is a graphical interpretation of these relations, consisting of constraint variables as nodes and constraints as edges. If the domain of a variable CV is reduced, the constraint solver (working as a daemon) computes immediately the consequences for all variables which are connected with CV by constraints. This process is called propagation. For each connected constraint it deletes values from variable domains which cannot fulfill the constraint.

If the domain of a variable becomes empty – no valid value exists for the variable – an inconsistency is detected. If this happens during the search phase (see below), backtracking to an alternative reduction of CV has to be performed. The crucial advantage of constraint programming is the ability to recognize those situations before assigning values to all variables. This property results in a very efficient processing, in particular if the number of relevant constraints is high.

The application of constraint programming results in substantial reductions of the respective search space without cutting off potential solutions. A unique solution is identified if all variable domains have exactly one value left.

In general we have more than one value left after setting the constraints (propagation by the solver, see above). In those cases, search is necessary, i.e. cycles of variable selection, domain reduction and propagation will be processed as long as a solution is found or a variable domain becomes empty. In such cases, backtracking is needed. Usually, the application of problem-independent and problem-specific heuristics is needed during search if the initial problem is hard.

A state-of-the-art FD constraint programming tool is ILOG Solver from ILOG S.A., France (www.ilog.com). It is a C++ library which provides methods for defining constraint variables with their domains, constraints, and search strategies, as well as for initiating of solving and for returning results. ILOG provides also a modeling language called OPL (Optimization Programming Language), which is described in van Hentenryck (1999). OPL allows specifying constraint models which can be solved by ILOG Solver. Two software components based on OPL and Solver are the integrated development environment OPLStudio and a Java API for loading and executing OPL models from Java applications. This set of tools allows rapid prototyping and fast, simple integration of the constraint model into a Java application. They were used in this way for creation of the SNS APS node.

3 THE SNS-APS SYSTEM

The task of the APS system is to provide a production planning algorithm which can simultaneously plan production output, release and part demand in a way that demands are fulfilled, production capacity is not exceeded and finished goods and part inventories neither exceed buffer capacities nor fall below a specified safety stock level. It has to consider part availability in terms of parts in transit and supplier capacity and shall minimize the distance between planned and optimal finished goods inventory. This planning algorithm had to be integrated into SNS as one production planning method of each DPP agent.

IBM's Supply Chain Analyzer (SCA), presented in 1999, was already a supply net simulation system with an integrated advanced planning algorithm (IBM 1999). But the system architecture was only able to model central planning: the planning algorithm could be instantiated only once in a supply net model, and it computed a production plan for all production nodes which were connected to it. Effects of forecast horizon length, information delay, forecast falsification etc. could not be addressed with this approach. One novelty of SNS-APS is that the APS node can occur in multiple instances, up to one instance for each producer node.

3.1 APS for SNS

As described above, the main task of the APS system in SNS is to create plans of production output, production release and part demand with regard to several constraints and an optimization goal. This task is called production program planning. The time horizon of program planning in practice ranges between six and twelve months, beginning with the current date. This is one important task of today's ERP or APS systems, but not the only one. That is, the APS of SNS covers not the whole range of subsystems of commercial APS systems. It focuses on the program planning aspect which is the most important one for investigations of distributed planning concepts.

Since the APS system of SNS appears as one among other planning methods of the DPP agent, it needs to cope with the whole parameter set of the planning methods. Especially, the planning horizon and granularity can be customized by the user to map the specific situation of each simulated manufacturing company. This requires the APS system to be generic with respect to horizon and granularity of it's planning.

3.1.1 Handling of Over-Constraint Problems

Since the planning problems are constrained problems, it is possible that over-constrained problems occur. That is, some planning problems which occur during a simulation run may have no solution, e.g. since demand exceeds the current capacity, or since a supplier delivers not the requested amount in time. In those cases, a classical MIP or FD program identifies the unfeasibility and returns the fact that no solution exists. In practice, trouble shooting will be initiated when such situations are recognized. Such situations are basically the drivers for all work on collaborative planning, which is aimed to minimize trouble shooting effort by introducing pro-active and preventive processes. Both prevention and trouble shooting rely on the analysis of the problem. The reason for an unfeasibility needs to be identified before actions for repair can be taken.

Consequently, the SNS-APS system has to have an unfeasibility analysis capability too. New processes for prevention or intelligent reaction to problems are the main subject of it's application scenario, process design for collaborative planning.

Analysis of unfeasibility is performed by so called conflict tests. These tests check the input data for several possible conflict situations. A conflict situation exists, for example, if demand exceeds capacity for a time period so that even with production in advance and use of inventory not all orders can be fulfilled in time. Another conflict occurs if parts for particular product variants are not available in sufficient amount and time. If the reason for the unfeasibility could be analyzed, it will be returned in terms of an error code together with additional information, e.g. on affected variants, parts, and time steps. Based on that information, processes for repair can be invoked. If the analysis was not successful, a relaxation of the planning problem will be solved and the result will be accepted as a plan. In such cases, it is possible that the simulated material flow does not meet the simulated plans. This situation is said to be recognized in practice too. Nevertheless, it is a subject of further work to reduce the possibility of this situation to a minimum. That is, more conflict tests need to be designed and implemented to analyze nearly all possible failure causes.

The following sub-sections describe the input and output data, the constraints, and the search strategy of the APS system.

3.1.2 Input Data

The input data for the APS system can be divided into the classes static, dynamic, and planning control data. They comprise the following single information:

Static data are the product types or variants the company produces, their production batch sizes, bill of materials (BOM) and times for assembly (the time step after production start when a part is really needed for a product's assembly (time of part assembly, TOP).

Dynamic data are orders and forecasts from customers, inventories of finished goods and parts and amount of work in process, production capacity, production cycle times, and order cycle times for parts. (Production capacity

is dynamic since the user can specify the nominal capacity as a curve over time and since the capacity manager agent can change the capacity by invoking capacity adjustment actions if demand requires this.) Finally, also the order pipeline, i.e. the list of open orders of all parts, and supplier capacity (which can either be specified by user or communicated by the supplier node during simulation) belong to dynamic data.

Planning control data comprise the desired inventory levels and/or safety stocks for both finished goods and parts, the planning horizon and the planning granularity for the resulting production plans.

3.1.3 Output Data

The main output data of the APS system are the plans, namely **production output plan, production release plan** and **part demand plan**. All of these plans contain amounts of products or parts for each period within the planning horizon. The production output plan specifies how much products will be finished during the periods, the release plan specifies the amount of products which will be released to the production system during the periods, and the part demand plan specifies the amounts of parts needed for production, also during the time periods. The plans are determined if the APS found a regular solution for the problem stated by the inputs.

In the case of unfeasibility, but successful failure analysis, an error code and information on affected products, parts and time steps is returned. Otherwise, only the fact that no solution could be found is the answer.

It is possible to get two more kinds of output data from the system which will be filled during the planning process: the **estimated finished goods inventory** and the **estimated maximal part inventory**. These data can be used for assessment of solutions, e.g. for comparison with stated optimal inventory levels.

3.1.4 Constraint Variables and Basic Constraints

While all input data are handled as constants in the constraint model, the output data define the most important constraint variables.

The basic constraints are identically with the basic equations of the so called manufacturing resource planning (MRP) algorithm. This algorithm calculates production output and estimated product inventory in an iterative manner. The iteration starts with the current time step and the current inventory and work in process amounts. The iteration runs over the planning periods.

Let *i* be the index of the planning periods and *p* the product index. Let further be *fc(p,i)* the demand forecast for product *p* at (during) period *i*, *op(p,i)* the planned production output for p at (during) i, and *esi(p,i)* the estimated product inventory of p at the end of period i. Let finally be *ci(p)* the current inventory level, $dsi_{min}(p)$ and $dsi_{max}(p)$ the minimal and maximal desired inventory level and *wip(p)* the current work in process amount of product p. Then the basic constraints are

$$esi(p,0) = ci(p) + wip(p) - fc(p,0),$$

$$\forall\ i: 1 \le i \le i_{max}: op(p,i) \ge fc(p,i) + dsi_{min}(p) - esi(p,i-1),$$

$$\forall\ i: 1 \le i \le i_{max}: op(p,i) \le fc(p,i) + dsi_{max}(p) - esi(p,i-1),$$

$$\forall\ i: 1 \le i \le i_{max}: esi(p,i) = esi(p,i-1) + op(p,i) - fc(p,i).$$

Since we assume that each product consumes one capacity unit of the manufacturing resource, the capacity constraints can be specified as follows:

$$\forall\ i: 1 \le i \le i_{max}: \sum_p op(p,i) \le cap(i),$$

where *cap(i)* is the production capacity at period i. The complete constraint model contains numerous further constraints which cannot all be presented here. Basically they are concerned with production batch sizes, time relations between production output and release, bill of material relationship between production release and part demand, and estimated part availability and inventory, respectively.

The constraint definition given above demonstrates the difference between classical programming paradigm and constraint programming. The inequations define a range of possible values for the constraint variables on the left-hand side instead of giving a calculation instruction. This allows to specify a range of allowed values explicitly, for example the interval between minimally and maximally allowed inventory levels. This simplifies the problem specification, but requires strong mechanisms inside the solver to create a concrete solution. The general approach of constraint programming which consists of iteration cycles of propagation, variable choice and labeling, provides the framework for such mechanisms. Efficient behavior heavily rely on good strategies for variable choice and labeling, as well as of additional (redundant) constraints which prune the search space by reducing the domains of as much as possible constraint variables as strong as possible after each labeling step.

3.1.5 Cumulative Constraints and Search Strategy

This paper cannot present the search strategy and search space pruning mechanisms of SNS-APS in detail. They will be subject of a forthcoming publication. We will only present the basic ideas and some results here.

The basic idea of the search strategy follows the iteration approach of the MRP algorithm. The iteration runs along the planning periods, output plans and estimated part inventories for a period i are computed based on corresponding values at period i-1. Our variable choice works in the same way: it chooses the plan and estimated inventory vari-

ables along the planning periods. If values for early chosen variables cause a conflict in later steps, backtracking to the earlier choice points will occur. Since backtracking requires a heavy computational effort, it should be avoided if possible. This can be done by deleting values from the domains of early chosen variables in advance and by detecting forthcoming problems as soon as possible. Both mechanisms help to prune the search space and make the solution process efficient. We use the concept of so called cumulative constraints to reach this: in addition to the basic constraints, relationships of variables can be analyzed and stated as constraints. One class of such relationships exists between sums of demands and capacities over all periods between period 1 and each period i. These sum constraints are called cumulative constraints. An example for a cumulative constraint is given in the following equation.

$$\forall\, i\colon 1 \le i \le i_{max}\colon \sum_{k\colon 1 \le k \le i} \sum_{p} op(p,k) \le \sum_{k\colon 1 \le k \le i} cap(k)$$

The same basic principle underlies the production and inventory control concept called progression numbers. These number cumulate inflow and outflow of material by enumerating continuously.

Another important class of additional constraints is defined over the minimal remaining production demand for each product p at a period i. With this, values can be deleted from the domains of plan variables which would be sufficient to fulfill the demand at the current period but not at further periods. This mechanism mainly controls the "production in advance", i.e. production to stock if capacity shortages in later periods are recognized.

Use of these mechanisms helped us to dramatically increase the computational efficiency of our APS system. We illustrate this for a particular, but representative sample planning problem with 25 planning periods and 6 product types. Table 1 and Figure 1 compare the efficiency of the search by performance numbers and by search tree visualization, respectively.

Table 1: Search Efficiency of APS

Search efficiency comparison		
	Standard search	Specific search
Constraints	1779	1779
Variables	1016	1116
Choice points	16537	246
Failures	16431	116
Solving time	59,72 sec	0,63 sec

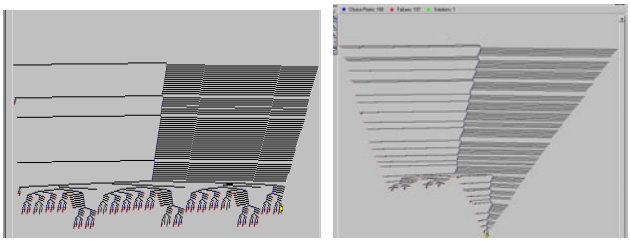

Figure 1: Search Trees of Standard and Specific Search

In the search tree visualization, leaf nodes correspond to failures (red marked nodes) or to success (solution, green marked node). The left picture shows a part of the tree of the standard search procedure (500 of 16.500 choice points), the right picture shows the complete tree of our search procedure. While the standard procedure performs many backtracking steps, the customized search procedure comes rather straight to a solution.

A runtime of only few seconds was needed to keep the overall simulation system at a high performance. For one singular planning problem, a runtime of one minute may be acceptable. But in supply net simulation, several producer nodes may use an APS with a frequency of one week. For a simulation experiment with a simulation time of one year and with 10 producers using the APS weekly, 520 planning problems need to be solved. With an average runtime of one minute, the APS runtimes would sum up to nine hours. APS execution in one second per problem would lead to an overall APS time of only nine minutes.

3.2 Integration of FD Solver and Java Application

The APS constraint model was developed in the language OPL by use of the integrated development environment ILOG OPLStudio. OPLStudio allows to compile OPL models after development and tuning. Compiled OPL models can be used from Java applications via the advanced programming interface (API) ILOG provides for it's library ILOG Solver.

The solver appears as a Java class. This class provides mainly the following methods:

- **Solver.init:** create an instance of the class Solver
- **Solver.loadCompiledModel:** feed a compiled OPL model to the solver object. The compiled model can be read from a file or a string.
- **Solver.loadData**: feed data initializations for the compiled model to the solver object. The data can also be read from a file or a string.
- **Solver**.**solve**: trigger the solving mechanism for the previously fed problem specification.
- **Solver**.**getVariableValue**: returns the value of a constraint variable at the current solution.

The APS system is integrated to SNS by use of this API. A very critical detail which made this approach possible is the distinction between model and data in the problem specification. It allows to use the same constraint model for all Solver instances. The problem specifications are customized only by the data initializations. This required the constraint model to be generic with regard to all input data, even to planning time line and products. Since OPL supports such generic constraint specifications this requirement could be easily met.

For efficiency reasons both the constraint model and the data are fed from strings. The compiled OPL model will be read into a global string variable during initialization of the simulation. The DPP takes the model from this global string when it has to perform an APS run. Further, the DPP collects all input data from the agents of it's producer site and glues it to a data string. After that, the DPP feeds model and data to it's solver object.

After solving, the DPP gathers the values of the output variables and fills it's own corresponding variables.

4 SAMPLE SCENARIO

The simulation experiments for the proof of concept were based on an artificial, but realistic scenario. Figure 2 shows the structure of the supply net and the product graph.

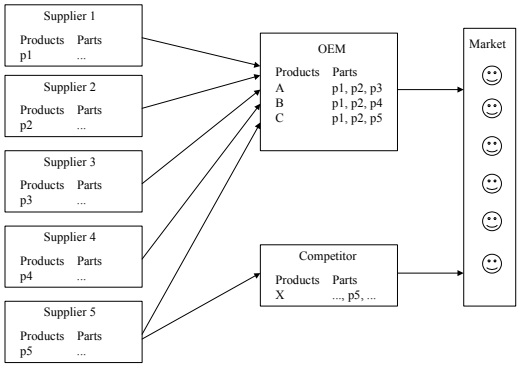

Figure 2: Structure of the Sample Supply Network

An OEM produces 3 products, A, B, and C. Each product consists of three parts. While parts p1 and p2 are contained in each product type, parts p3, p4 and p5 occur only in products A, B, and C, respectively. They differentiate the products. The OEM has 5 suppliers, one for each part type. Further, a competitor of the OEM belongs to the supply network. This competitor produces a product X which also contains part p5. Unfortunately, the competitor also obtains p5 from supplier 5.

Supplier 5 has aligned it's capacity with the demand from it's customers in a way that it's production system is almost completely utilized. To increase capacity, Supplier 5 would need to build a new production line. Supplier 5 would not hesitate to do this if it would recognize a stable demand increase. But installing a new production facility lasts several weeks.

For arbitrary reason, competitors demand for p5 increases rapidly and suddenly. Supplier 5 starts to build the new production line, but the demand increase starts six weeks before the capacity is adjusted. During this period, Supplier 5 cannot fulfill all orders. Supplier 5 has to distribute it's production amount between it's two customers. A usual policy is to calculate delivery amounts in proportion of demand. Since the competitor increased demand, the proportion of OEMs demand decreased. The delivery amount decreases too. As a consequence, the inventory level of p5 at OEM falls, and in case of constant production rate released production orders for product C cannot start due to missing parts. This may lead to unused production capacity and to falling product inventories. In worst case, customer orders cannot be fulfilled in time.

We investigated many different simulation experiments based on that scenario. Mainly we compared the question if use of an APS could help the OEM to mitigate the effects of this situation. We assumed that a supplier would inform the OEM in case of such demand increase, or any other kind of capacity shortage. This could be realized either by setting the simulation parameters accordingly or by using the APS at supplier 5 and communicating the recognized capacity shortage automatically. Since APS communication is not implemented completely now, we used the first alternative. That is, we defined the supplier capacity parameter of the OEM according to the expected supply situation. This parameter is used by the OEM's APS system to calculate part availability. Figures 3 and 4 show the result charts of two simulation experiments.

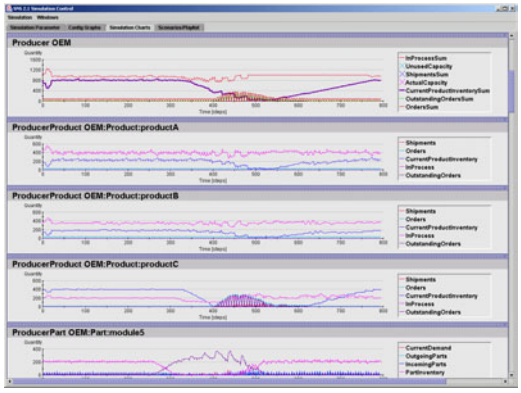

Figure 3: Scenario Results without APS Use

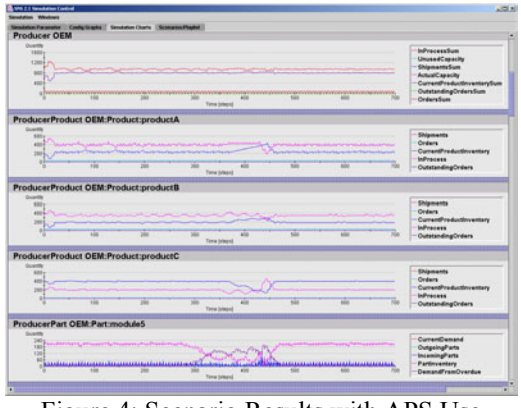

Figure 4: Scenario Results with APS Use

The main result of the experiments is that the APS system was able to cope with the situation in a way that no customer order became late, no capacity was lost and part

orders remained relatively stable. The APS computed a production plan like an experienced human planner. As soon as the material for product C was not available completely, the remaining production capacity was planned to be used to produce other products which were not affected by the shortage. These products are produced to stock, i.e. inventory level was increased. As soon as Supplier 5 delivers the missing parts, the production ratio between the products is changed. Production of the stocked products is now reduced and orders are delivered from stock, reducing inventory level to the initial value. The remaining capacity is now used to produce product C and to fill the product inventory up to optimal level. The sum over all products in product inventory remained constant. Figure 3 documents the experiment without APS. The graphs in the upper chart shows the product inventory and work-in-process amount for all products of OEM. The charts below split these information for each single product. The lower chart shows the part inventory for p5. It is not necessary to go into detail to see that the situation in Figure 4 is much smoother. Table 2 provides a comparison of few selected KPIs of both scenarios.

Table 2: KPI Measurement of APS Usage Effects

Simulation result comparison		
	OEM without APS	OEM with APS
Orders backlog (peak)	260	0
Order backlog (total)	18000	0
Product inventory deviation from opt (peak)	800	25
Product inventory deviation (total)	180000	156
Product mix in inventory deviation (total)	180000	27000

The average solving time of the APS was 2.24 seconds, the complete scenario with APS was simulated for a half year with weekly APS run in three minutes.

5 CONCLUSION

In this paper, we have presented an approach to simulate supply chain planning processes which include the use of advanced planning systems. The specific aggregation level of the simulation allows on the one hand concentration to the main processes and effects of supply net collaboration and on the other hand the extremely efficient execution of simulation experiments. Only this efficiency allows to explore large experiment spaces and to use the simulator in an interactive manner.

We have proven that simulation of collaborative supply net processes is possible and can be used to support business process design and assessment for innovative collaboration approaches for supply networks.

REFERENCES

Bartak, R. 2003. *Online Guide to Constraint Programming*. <http://Kzi.ms.mff.cuni.cz/bartak/constraints/>.

Baumgaertel, H. 2000. Distributed Constraint Processing for Production Logistics. *IEEE Intelligent Systems* 15(1): 40-48.

Baumgaertel, H., S. Brueckner, V. Parunak, R. Vanderbok, and J. Wilke. 2003. Agent Models of Supply Networks Dynamics. In *The Practice of Supply Chain Management*, ed. C. Billington, T. Harrison, H. Lee, and J. Neale. International Series on Operations Research and Management Science, Kluwer.

Forrester, J.W. 1961. *Industrial Dynamics.* Cambridge, MA: MIT press.

IBM Research. 1999. *Supply Chain AnalyzerUser's Guide* Version 3.1.

Lendermann, P., B.P. Gan, and L.F. McGinnis. 2001. Distributed Simulation with Incorporated APS Procedures for High-Fidelity Supply Chain Optimization. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, 1138-1145, Arlington, Virginia, USA: Institute of Electrical and Electronics Engineers.

Marriott K., Stuckey P.J. 1998. *Programming with Constraints: An Introduction*. Cambridge, MA: MIT Press.

Parunak, H. V. D., Savit,R., Riolo,R., and Clark,S.J. 1999. *Dynamical Analysis of Supply Chains*. Ann Arbor, MI: Altarum. <www.altarum.org>.

Supply-Chain Council. 2003. *The Supply Chain Operational Reference (SCOR) Model*. <www.supply-chain.org>.

van Hentenryck, P. 1999. *The OPL Optimization Programming Language*. Cambridge, MA: MIT Press.

Wilke, J. 2002. Using Agent-Based Simulation to Analyze Supply Chain Value and Performance. In *Proc. of Supply-Chain World 2002*. New Orleans: The Supply-Chain Council.

AUTHOR BIOGRAPHIES

HARTWIG BAUMGAERTEL is a scientific researcher at DaimlerChrysler Research and Technology in Berlin, Germany. He has a Ph.D. degree in Computer Science from Technical University Berlin. Since 1999, he is responsible for the research on supply net management in the manufacturing and supply net department. His e-mail address is <hartwig.baumgaertel@dcx.com>.

ULRICH JOHN is a scientific researcher at DaimlerChrysler Research and Technology in Berlin, Germany. He holds a PhD in Computer Science from Technical University Berlin. His e-mail address is <ulrich.john@dcx.com>.