Model-Based Performance Engineering of General Motors' Vehicle Supply Chain

Armin Zimmermann, Michael Knoke, Shang-Tae Yee, and Jeffrey D. Tew

Abstract— **We present results of a collaboration project, during which parts of General Motors' north American vehicle supply chain have been modeled and evaluated. A variant of colored stochastic Petri nets has been developed for this task, and support for model creation and performance evaluation was implemented in our tool TimeNET. Variations of the model and their stepwise evaluation lead to suggestions about how the supply chain can be reorganized in order to improve the time between customer order and vehicle delivery (order-to-delivery time).**

I. INTRODUCTION

Supply chains and logistic networks play a major role in today's businesses because of the growing importance of external suppliers to final products and partition of work between distributed plants. Material buffer levels are kept small to decrease the bound investments. Timely deliveries of intermediate parts are thus necessary to avoid shipment delays while keeping the amount of work in process small.

Model-based quantitative evaluation of logistic networks is a vital tool to aid in the decision-making at various stages of planning, design, and operation of supply chain operation. A robust control and management of supply chain loops can be achieved by analyzing performance measures such as the throughput rate, average resource utilization, expected number of parts in a buffer, setup costs, work-in-process inventory, mean order queue time, etc.

Typical decisions during the planning and design stages include the number of containers or transport facilities, buffer storage capacity in a logistic center, scheduling of material and parts, number of links in inbound and outbound logistics, and location of distribution centers. During the operational phase, performance modeling and analysis can help in making decisions related to predicting the probability of a material shortage.

The work presented here is the result of an industrial project carried out by General Motors Research and Development together with groups of Stanford University and Technische Universität Berlin. A specific variant of colored stochastic Petri nets has been developed and implemented in the software tool TimeNET [1], [2] during the project. The overall goal was to model and analyze supply chain issues of GM. Previous results have been presented in [3].

This work was supported by General Motors Research and Development.

A. Zimmermann and M. Knoke are with the Faculty of Electrical Engineering and Computer Science, Real-Time Systems and Robotics Group, Technische Universität Berlin, 10587 Berlin, Germany {zimmermann, knoke}@cs.tu-berlin.de

S.-T. Yee and J. D. Tew are with the GM Research and Development Center, 48090-9055 Warren, MI, USA {shang-tae.yee, jeffrey.tew}@gm.com

The presented supply chain model is an adapted version of the actual one. Details have been altered in order to keep the original information confidential. Model and problems are however characteristic. The main issue considered here (thus driving the modeling and evaluation process) is the question how long customers have to wait for their vehicle from the day of the purchase decision. This delay between order and delivery (or order-to-delivery time) is denoted by *OTD time* in the sequel.

The background is the way how the U.S. vehicle business (in contrast to European car makers) works today. Vehicle dealerships keep a high number of cars in their lot to have a large selection available for prospective customers. Buyers usually select a vehicle directly from the lot and thus do not need to worry about the time it would take to get an individually ordered vehicle. There are however big drawbacks for the car making companies and dealers. The high amount of available cars binds money and makes fast reactions to changing customer needs impossible. In fact, the selection of vehicles and the installed options are guessed based on past sales patterns. What thus often happens is that many of the vehicles can only be sold by giving large rebates, decreasing the earnings substantially. Another issue is that individually manufactured vehicles could be sold with a higher price considering the different options.

The current way of operating the supply chains, plants and logistics implies a very long OTD time for an individually ordered vehicle. Car makers have realized that an agile supply chain and a short OTD time can be a substantial marketing factor and lead to higher earnings. Even in Europe, where individually ordered vehicles are sold by tradition, OTD times have become an issue to attract more customers lately. The example considered in this paper shows how OTD times of a certain operation style can be evaluated and how possible improvements are quantifiable to aid in strategic design decisions.

An adequately complex and flexible model class is necessary to capture the detailed behavior of a logistic network and the individual nodes, such as manufacturing plants (assembly, fabrication), logistics centers, suppliers and dealerships. We use a variant of colored stochastic Petri nets for the modeling. There is a large quantity of literature available on Petri nets with individual tokens. Different variants have evolved, namely colored Petri nets [4], [5], algebraic high-level nets [6] and Predicate/Transition nets [7].

Extensions of colored Petri nets by stochastic firing times, as they are necessary for a performance evaluation, have been introduced in [8], [9]. Applications are e.g. demonstrated

in [10], [11].

The variant of stochastic colored Petri nets used here was mainly influenced by the definition given in [5]. The main differences are easier specification of arc inscriptions, allowing an automated generation of efficient analysis algorithm code, and a true stochastic timing semantics which is in accordance to the usual understanding of stochastic timed Petri nets. Type definitions are simplified, and arc variables do not need to be declared as they have to be in CPNs using the specification language Standard ML.

Model-based evaluations of supply chains have been reported in the literature, including the following results. Generalized stochastic Petri nets are used for modeling and analyzing supply chain networks in [12]. Make-to-stock and assemble-to-order systems are compared in terms of total cost. A toolset for modeling and analysis of logistic networks is presented in [13]. Another software tool which can be used to model and analyze logistic systems is ExSpect [14], which uses hierarchical colored Petri nets. The discrete behavior of logistic systems is modeled by timed Petri-Nets with individual tokens in [15]. The application of timed colored Petri nets to logistics is also covered in [16].

The paper is structured as follows. The subsequent section briefly introduces the variant of colored stochastic Petri nets used throughout the paper and project. Section III describes the developed model of General Motors' supply chain. Order-to-delivery time for customers is evaluated based on the model in Section IV, and different setups are checked if they improve it. The results are summarized in the conclusion.

II. COLORED STOCHASTIC PETRI NETS

This section briefly describes *stochastic colored Petri nets* (SCPNs), which are especially useful to describe complex stochastic discrete event systems and thus appropriate for logistic problems. Places and transitions of a Petri net naturally map to buffers and activities or similar entities. Objects which are created, changed and moved through a system are usually described by tokens in places. The application of classic Petri nets to examples in which these objects carry some significant attributes leads to cluttered models in which places and transitions need to be unfolded to keep track of the individual attributes. These problems motivated the development of *high-level Petri nets*, a set of net classes with distinguishable tokens. The main difference between simple Petri nets and colored models is that tokens may have arbitrarily defined attributes. It is thus possible to identify different tokens in contrast to the identical tokens of simple Petri nets.

The introduction of individual tokens leads to some questions with respect to the Petri net syntax and semantics. Attributes of tokens need to be structured and specified, resulting in *colors* (or *types*). Numbers as arc information are no longer sufficient as in simple Petri nets. Transition firings may depend on token attribute values and change them at firing time. A transition might have different modes of enabling and firing depending on its input tokens. Our class of SCPNs uses arc variables to describe these alternatives.

In the following we mostly point out differences to uncolored Petri nets. The syntax of textual model inscriptions is chosen similar to programming languages like C++ or Java.

A. Token Types or Colors

Tokens belong to a specific *type* or *color*, which specifies their range of attribute values as well as the applicable operations just like a type of a variable does in a programming language. Types are either *base types* or *structured types*, the latter being user-defined. Available base types in the software tool TimeNET, which has been extended with SCPNs recently [1], [2], include Integer, Real, Boolean, String, and DateTime. Structured types are user-defined and may contain any number of base types or other structured types just like a Pascal `record` or a C `struct`.

Types and variables are textually specified in a declarational part of the model. This is done with type objects in the graphical user interface of TimeNET, but is omitted in the model figures. Variable definitions are not necessary in difference to standard colored Petri nets because they are always implicitly clear from the context (place or arc variable).

B. Places

Places are similar to those in simple Petri nets in that they are drawn as circles and serve as containers of tokens. By doing so they represent passive elements of the model and their contents correspond to the local state of the model. As tokens have types in a colored Petri net, it is useful to restrict the type of tokens that may exist in one place to one type, which is then also the type or color of the place. This type is shown in italics near the place in figures. The place marking is a multiset of tokens.

The unique name of a place is written close to it together with the type. The *initial marking* of a place is a collection of individual tokens of the correct type. It describes the contents of the place at the beginning of an evaluation. A useful extension that is valuable for many real-life applications is the specification of a place *capacity*. This maximum number of tokens that may exist in the place is shown in square brackets near the place in a figure, but omitted if the capacity is unlimited (the default).

C. Arcs and Arc Inscriptions

Places and transitions are connected by directed arcs as in any other type of Petri net. An arc going from a place to a transition is called input arc of that transition, and the connected place is also called input place (and vice versa for output places and output arcs). In contrast to simple Petri nets, where a number is the only attribute of an arc, the modeler must be able to specify what kinds of tokens should be affected and what operations on the token attributes are carried out when a transition fires. This is done with *arc inscriptions*, which are enclosed in angle brackets <> here.

Input arcs of transitions and their inscriptions describe how many tokens are removed during a transition firing,

and attach a name to these tokens under which they may be referenced in output arc and guard expressions. They carry a variable name in pointed brackets for the latter task, optionally extended by a leading integer specifying the number of tokens to be removed from the place. A token from the input place is given to the variable as its current value, and removed from the place during firing.

A transition's *output arcs* define what tokens are added to the connected place at the time of transition firing. There are two general possibilities for this: either existing tokens are transferred, or new tokens are created. In the transfer/copy case the name of the chosen input token is used at the output arc. The multiplicity of tokens must be the same to avoid ambiguities.

Fig. 1 depicts a small example. Transition `Assembly` has two input places `InputA` and `Stock`, from which one product `p1` and three products `p2` are removed respectively. The firing of the transition transfers the `p1`-token to the output place `OutputA`, and changes the attribute `step` to the new value `5`. Tokens bound to input variables which are not used on output arcs (`p2`-tokens) are destroyed at the end of the firing, modeling an assembly operation here.

New tokens of the output place type are created if no input variable is specified at an output arc. The attributes of a new token are set to their default values initially, or can be set to specific values. Operators are allowed in expressions as long as their resulting type corresponds to the required one.

The type of the variables contained in the input and output arc inscriptions is implicitly given by the type of the connected place and is thus not defined by the modeler. Restrictions on the input tokens are modeled using transition guards as described below.

D. Transitions

Transitions (drawn as rectangles with different shapes) model activities of the system. They can be activated (enabled) when all necessary input tokens are available and an optional *guard function* is true. Their firing models the occurrence of the activity and changes the marking of places with tokens (the state of the system). There are different transition types with their corresponding shapes: *immediate transitions* firing without delay are drawn as thin rectangles, *timed transitions* bigger and empty, while *substitution transitions* have black rectangles at the top and bottom.

Transitions have several attributes besides their name. The *firing delay* (timed transitions only) describes the probability distribution of the delay that needs to elapse between the transition enabling and firing. Immediate transitions have a *firing weight* (a real number) and a *priority* (integer greater than 0) just like in simple Petri nets, with the same function. Their default value is one in both cases.

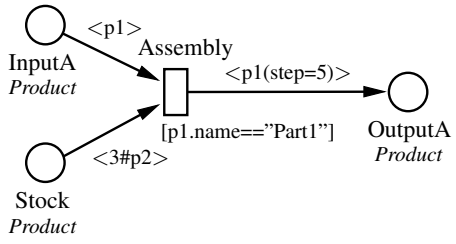

Fig. 1. Arc inscription example

The set of tokens bound to a transitions' arc variables determines its mode of firing in a specific marking, and is usually coined *binding* in the literature. In a state of a SCPN all possible assignments of input tokens to their respective arc variables may be valid firing modes. A *guard function* can be used to restrict the tokens for which a transition may be enabled. The guard is a boolean function that may depend on the model state and the input arc variables. It is shown in square brackets close to the transition in figures. The transition is only enabled with a certain binding of tokens to variables in a model state if the guard function evaluates to True for this setting. In the example of Fig. 1, transition `Assembly` is only enabled for tokens in place `InputA`, which have their attribute `name` set to `Part1`.

E. Model Hierarchy

A SCPN model consists of *pages* in a hierarchical tree. There is exactly one *prime page* (or main model), which forms the base of the tree structure, and to which other pages are subordinated on different levels of hierarchy via *substitution transitions*. Hierarchical refinement and modular description of complex systems is thus possible.

Substitution transitions act as a placeholder or association to a refining subpage. They have no firing semantic as other normal transitions do. The associated subpage is a place-bordered subnet, i.e. no transitions of the subnet may have direct interactions with elements outside the substitution transition. Interaction of submodels with the surrounding model only happens via the places that are connected to the substitution transition. All of these places are known in the submodel and depicted there as a dotted circle.

III. A SUPPLY CHAIN MODEL

In this section we present a simplified model of General Motors' supply chain operation. It is altered to keep business information confidential, but the main issues that the modeling and evaluation aims at are maintained. Due to space limitations we can only show some selected parts of the model, which in reality comprises many more subpages than explained here.

Fig. 2 shows a rough overview of the covered entities. Customer, dealership and (assembly) plant as well as the logistic network that transports vehicles from the plant to the dealership are considered. Internal and external suppliers which deliver intermediate parts to the assembly plant (*inbound logistic*) are not considered here. We assume that enough material is available at the plant to avoid downtimes. How this can be efficiently achieved has been considered in the project as well.

A pure on-demand production setup would work as follows. When a customer comes to a dealership, he has a certain vehicle in mind that he or she wants to buy. This

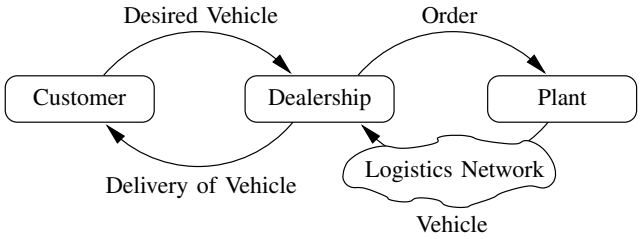

Fig. 2. Sketch of supply chain entities

Color	Element	Element Type
`Config`	`model`	`string`
	`drive`	`string`
	`interior`	`string`
	`color`	`string`
`Order`	`Conf`	`Config`
	`OrderTime`	`DateTime`
	`Origin`	`string`
`Vehicle`	`Order`	`Order`
	`ProdTime`	`DateTime`

TABLE I
TOKEN TYPES (COLORS) OF THE SUPPLY CHAIN MODEL

specification of a desired vehicle is ordered by the dealer from the plant. The order is stored in an order queue, which is after some time processed by the plant, resulting in a new vehicle in the plant yard. From there it is transported by the logistics network to the dealer where the customer can pick it up.

The supply chain model and its hierarchical decomposition into submodels is covered in the following. Each model describes the actions and decisions of one typical entity, for example a dealer or customer. Communication takes place by exchanging tokens via interface places. Behavior and decisions based on rules are specified using transition firing attributes.

Fig. 3 shows the topmost level of hierarchy of the supply chain model. It describes the main entities and their interaction. Transitions with thick bars depict substitution transitions, that are refined with a submodel. Types of tokens in places are depicted in italics (e.g. *`Config`* or *`Vehicle`*).

Tokens model complex entities and thus have a set of corresponding attributes. The set of token types used in the supply chain model are listed in Table I. `String` and `DateTime` are the only necessary base types. The latter is a convenient way of handling a point in time by specifying day and time. Configuration of a vehicle and the type of car that a customer wants to buy are described by the type `Config`. It specifies the model, type of drive, interior and color. This is a simplification of the actual configuration attributes for the sake of readability. An order for a vehicle production is described by color `Order`, describing the order time as well as the ordered configuration. An actual vehicle has the attributes `Order` — the order that initiated the production of the vehicle — and the time when its production was finished.

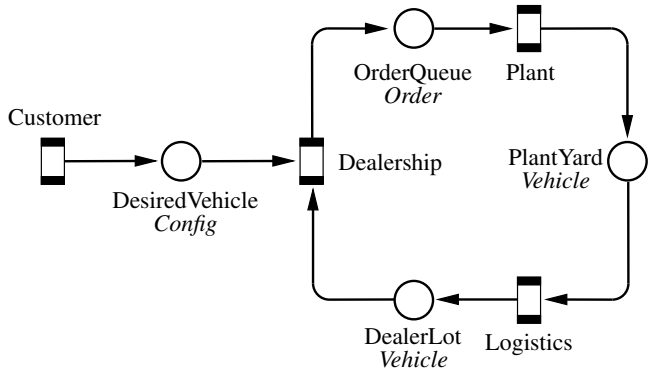

Fig. 3. Main colored Petri net model of the example

Transition `Customer` models the customer behavior. The selected vehicle type and configuration information is transferred to the dealership model (transition `Dealership`) through place `DesiredVehicle`. The queue of waiting orders at the plant is modeled by place `OrderQueue`. After production in the `Plant`, new vehicles arrive in the `PlantYard` and are transported by `Logistics` to the `DealerLot`. The model considered here only takes into account one dealership and one plant in contrast to the original detailed model.

For our purposes it is important to specify *when* a customer comes to a dealership to purchase a car and *what configuration* he wants to buy. This customer behavior is captured in a customer submodel (that refines transition `Customer`). The configuration of the customer's desired vehicle is generated in a stepwise fashion. A newly created token of type *`Config`* without set attributes is generated. The token then follows through a series of places, in which one attribute of the configuration is set one after another.

Firing probabilities that correspond to known customer choices are associated to the transitions. The fully specified configuration token finally arrives in the interface place `DesiredVehicle` to the upper model level.

When a customer wants to purchase a vehicle, the corresponding token of type *`Config`* thus arrives in place `DesiredVehicle` in the dealership model shown in Fig. 4. The dealer sends a new order to the order queue (by creating a token of type *`Order`* in place `OrderQueue`). The configuration of the order is copied from the customer's choice, and the order time is set to the current model time (`NOW`). A similar copy of the order is kept in place `WaitingOrders` at the dealership, to identify waiting orders later on when new vehicles are available in the dealer lot.

If a vehicle arrives at the dealer lot (place `DealerLot`) that matches a waiting order, transition `DeliverOrdered` fires. The guard function of that transition ensures that only vehicles are delivered that match exactly.

Details of the remaining submodels are omitted due to space limitations. The plant submodel describes how orders in the order queue are processed. New vehicles arrive in place `PlantYard` finally.

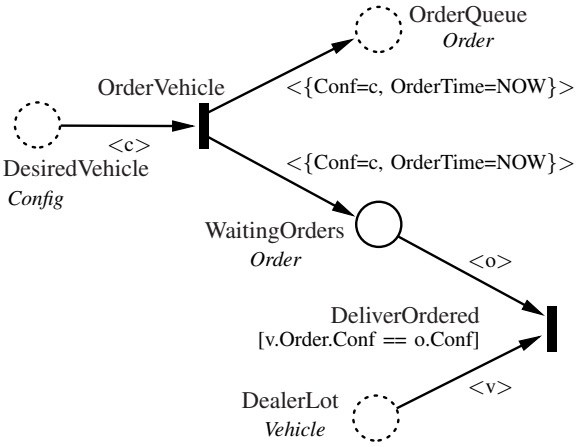

Fig. 4. Dealership model

New vehicles need to be transported from there to the dealership, which is done by the logistics network (substitution transition `Logistics`). Vehicles are picked up from the plant yard and transported to the dealer lot, at first considering only train transport. Vehicle transport by train has the advantage of being very cost-efficient. The downside is its long delay and high variance. Moreover, train transport is externally managed by the train companies, and thus cannot be influenced significantly by the car making company.

IV. MODEL-BASED PERFORMANCE ENGINEERING

The order-to-delivery time for customers is evaluated based on the model explained above, and different setups are checked if they improve it. Results of the quantitative model evaluation lead to ideas how the design and operation of the supply chain can be changed in order to decrease the OTD time. We demonstrate with the example how a series of model changes and quantitative evaluations can be successfully exploited to improve the performance of a supply chain significantly.

Discrete event simulation is used for the quantitative evaluation of the model. Numerical analysis techniques are not applicable because of several transitions with non-exponentially distributed firing delay distributions. A simulation run of six years of model time typically took 45 seconds. All evaluations have been carried out on a PC with Intel Pentium III Mobile processor running at 1 GHz under Windows XP. Measure samples for the first year of model time are discarded to avoid influences of the initial transient phase. Statistical analysis shows that the remaining simulation length leads to sufficiently accurate results for our purposes. A typical evaluation shows that the number of considered samples results in a maximum relative error of only 2% for a confidence level of 99%.

The initial setup of the model is evaluated first. There are obviously no vehicles available in the dealer lot, because every one is produced on demand and immediately delivered. The mean OTD time is computed as 25.08 days.

The first change in the model includes details of how vehicles can be produced and stored without prior customer order. A rough estimate of the probability of vehicle configurations of customer purchase decisions is known from past sales numbers. The dealership can thus order a restricted set of popular configurations, keep them available at the dealer lot, and sell them to customers with matching vehicle desire immediately. The dealership can check the influence of a certain selection of popular configurations and the number of available vehicles at the lot using the performance evaluation of the model.

The dealership model is adapted accordingly, but not shown here due to space limitations. Different setups have been evaluated after the model change. Results are given in Table II, showing the number of popular configurations that are stored in the dealer lot and the mean number of available vehicles. Both numbers mainly influence the probability with which a customer buys a vehicle from the lot, which is listed as “Immediate Delivery”. The results show that the number of dealer-ordered configurations is more important than a very high number of available vehicles at the lot. The two OTD time values represent mean numbers, taking into account all customers or only the ones that do not purchase an available vehicle. We choose to order nine different configuration types and to keep the amount of available vehicles in the range between 60 and 120 as a consequence of the results. The new OTD time of 9.99 days represents a 60% improvement.

At the plant there are several obvious details which can be changed for a smaller OTD time. We select the following: customer-ordered vehicles should be processed with priority over the ones that will be stored in the dealer lot. This change has the following influence on the performance measures. The OTD time of all customers drops to 8.37 and the time for waiting customers to 18.62 as a result, an improvement of 16%.

In order to reduce the average OTD time we need to concentrate on the OTD time of waiting customers, because the percentage of immediate deliveries could only be increased by storing more vehicles with additional configurations in the dealer lot. Other means of vehicle transport are considered for this reason. Specialized trucks for vehicles will be responsible for the transport between plant yard and dealer lot. Train transport is chosen for dealer-ordered vehicles and in cases where there is no truck available for transport. The submodel describing the logistic network is enhanced with more details describing the truck transport. Model and detailed evaluation

Number of Configurations	Immediate Delivery	`OTDtime` All	`OTDtime` Waiting
2	34%	16.07	24.36
9	53%	10.29	21.87
9	55%	9.99	22.33

TABLE II

INFLUENCE OF VEHICLE STORAGE ON THE OTD TIME IN DAYS

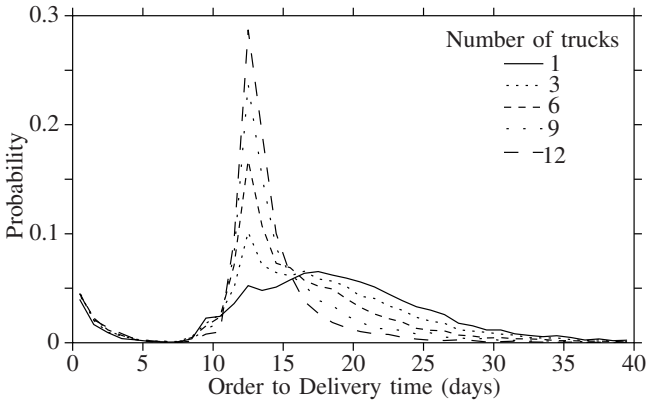

Fig. 5. Order-to-delivery time distributions of waiting customers

results had to be omitted here.

A sensible number of trucks for the example can be chosen based on the resulting OTD times. About 15 trucks are necessary to achieve the minimum OTD time of 5.92 days, which is an improvement of 29%. The mean OTD time of waiting customers is 12.9 days.

We show detailed results for one selected question: how are the OTD times distributed for different numbers of trucks? Fig. 5 shows probability density functions for selected truck numbers, ignoring all zero OTD times. The curves start at the left with the remaining probabilities that a waiting order is fulfilled by an incoming dealer-ordered vehicle which has not been available before. The peak value at 12.5 is due to the mean delays of production and transport, while the stochastic influences lead to the distribution around this value. For smaller number of trucks the transport time is heavily influenced by the long and variating train transport, resulting in a much flatter curve without a peak.

V. CONCLUSIONS

The paper presented results of a collaboration project between General Motors R&D and academic partners. It is obviously much easier to develop and evaluate variants of supply chain systems by using a model-based approach. A variant of colored stochastic Petri nets has been developed for this task, and proper tool support has been implemented in a prototype extension of our tool TimeNET. Colored Petri nets are powerful enough to describe the complex objects and interactions of a supply chain. We considered the order-to-delivery time as a performance issue here, which is attracting high interest in order to fulfill individual customer demands in a timely fashion.

Results of the quantitative model evaluation lead to ideas how the design and operation of the supply chain can be changed in order to decrease OTD time. The paper demonstrates how a series of model changes and quantitative evaluations can be successfully exploited to improve the performance of a supply chain significantly. The application example has shown that strategic decisions during the design of a supply chain can be efficiently aided using a model and performance evaluation. The mean OTD time for customers as the significant measure for our experiment could be improved from about 25 to less than six days in the model.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support of the cooperation project by General Motors Research and Development. Further thanks go to Peter Glynn at Stanford University, our partner in the described project. We would additionally like to thank the numerous graduate students who implemented the software tool extension.

REFERENCES

[1] A. Zimmermann, M. Knoke, A. Huck, and G. Hommel, "Towards version 4.0 of TimeNET," in *13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2006)*, March 2006, pp. 477–480.

[2] M. Knoke and A. Zimmermann, "Distributed simulation of colored stochastic Petri nets with TimeNET 4.0," in *Proc. 3rd Int. Conf. Quantitative Evaluation of Systems (QEST '06)*, Riverside, CA, USA, Sep. 2006, pp. 117–118.

[3] S.-T. Yee, J. Tew, A. Zimmermann, M. Knoke, and A. Huck, "New methodology for developing supply chain models in support of OTD," General Motors Research and Development Center, Warren, Research Report MSR-121, 2002.

[4] K. Jensen, "Coloured Petri nets and the invariant-method," *Theoretical Computer Science*, vol. 14, pp. 317–336, 1981.

[5] ——, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, ser. EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1992.

[6] J. Vautherin, "Parallel specification with coloured Petri nets and algebraic data types," in *Proc. 7th European Workshop on Application and Theory of Petri Nets*, Oxford, UK, Jul. 1986, pp. 5–23.

[7] H. J. Genrich and K. Lautenbach, "The analysis of distributed systems by means of Predicate / Transition nets," in *Semantics of Concurrent Computation*, ser. Lecture Notes in Computer Science, G. Kahn, Ed. Springer Verlag, 1979, vol. 70, pp. 123–146.

[8] C. Lin and D. C. Marinescu, "On stochastic high-level Petri nets," in *Proc. 2nd Int. Workshop on Petri Nets and Performance Models*, Madison, Wisconsin, 1987, pp. 34–43.

[9] A. Zenie, "Colored stochastic Petri nets," in *Proc. 1st Int. Workshop on Petri Nets and Performance Models*, 1985, pp. 262–271.

[10] G. Balbo, G. Chiola, S. C. Bruell, and P. Z. Chen, "An example of modeling and evaluation of a concurrent program using colored stochastic Petri nets – Lamport's fast mutual exclusion algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 221–240, 1992.

[11] S. M. Koriem and L. M. Patnaik, "A generalized stochastic high-level Petri net model for performance analysis," *The Journal of Systems and Software*, vol. 36, no. 3, pp. 247–266, Mar. 1997.

[12] N. Raghavan, "Performance analysis and design of supply chains: a Petri net approach," *Journal of the Operations Research Society*, vol. 51, no. 10, pp. 1158–1169, 2000.

[13] F. Bause, H. Beilner, M. Fischer, P. Kemper, and M. Völker, "The ProC/B toolset for the modelling and analysis of process chains," in *12th Int. Conf. Computer Performance Evaluation, Modelling Techniques and Tools (TOOLS 2002)*, ser. Lecture Notes in Computer Science, T. Field, P. Harrison, J. Bradley, and U. Harder, Eds., no. 2324. London, UK: Springer Verlag, Apr. 2002, pp. 1–51.

[14] W. van der Aalst and A. Waltmans, "Modelling logistic systems with EXSPECT," in *Dynamic Modelling of Information Systems*, H. Sol and K. v. Hee, Eds. Amsterdam: Elsevier Science Publishers, 1991, pp. 269–288.

[15] K. Lemmer and E. Schnieder, "Modelling and control of complex logistic systems for manufacturing," in *Advances in Petri Nets 1992*, ser. Lecture Notes in Computer Science, K. Jensen, Ed. Springer Verlag, 1992, vol. 616, pp. 373–378.

[16] W. v. d. Aalst, "Timed coloured Petri nets and their application to logistics," PhD Thesis, Eindhoven University of Technology, 1992.