

December 2006

Managing Supply Chain Events to Build Sense-and-Respond Capability

Akhil Kumar
Pennsylvania State University

Wil van der Aalst
Eindhoven Technical University

Follow this and additional works at: <http://aisel.aisnet.org/icis2006>

Recommended Citation

Kumar, Akhil and van der Aalst, Wil, "Managing Supply Chain Events to Build Sense-and-Respond Capability" (2006). *ICIS 2006 Proceedings*. 10.
<http://aisel.aisnet.org/icis2006/10>

This material is brought to you by the International Conference on Information Systems (ICIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICIS 2006 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

MANAGING SUPPLY CHAIN EVENTS TO BUILD SENSE-AND-RESPOND CAPABILITY

Design Science

Rong Liu¹
IBM T.J. Watson Research Center
Hawthorne, NY
rliu@us.ibm.com

Akhil Kumar
Penn State University
University Park, PA
akhilkumar@psu.edu

WMP van der Aalst
Eindhoven Technical University
Eindhoven, Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract

As supply chains become more dynamic, there is a need for a sense-and-respond capability to react to events in a real-time manner. In this paper, we propose Petri nets extended with time and color (for case data) as a formalism for doing so. Hence, we describe seven basic patterns that are used to capture modeling concepts that arise commonly in supply chains. These basic patterns may be used by themselves and also be combined to create new patterns. Next, we show how to use the patterns as building blocks to model a complete supply chain and analyze it using dependency graphs and simulation. Dependency graphs can be used to analyze the various events and their causes. Simulation was, in addition, used to analyze various performance indicators (e.g. fill rates, replenishment times, and lead times) under different supply chain strategies. We performed sensitivity analysis to study the effect of changing parameter values on the performance indicators. In the experiments, by cutting resolution time for production delays in half (strategy 1), we were able to increase order fill rate from 89% to 95%. Similarly, upon raising the probability of successful alternative sourcing (strategy 2) from 0.5 to 0.7 the order fill rate again increased from 89% to 95%. We show that by modeling timing and causality issues accurately, it is possible to improve supply chain performance.

Keywords: Supply chain event management (SCEM), event causality, Petri nets, time Petri nets, colored Petri nets, dependency graph, sense-and-respond capability.

Introduction

The environment for supply chains has been transformed completely with the advent of sophisticated information technology in a global and highly interconnected economy. Modern supply chains must be tightly integrated across several partners such as customers, suppliers, vendors, shippers, financiers, etc. Even a minor unexpected event such as a late arrival of a shipment or a machine breakdown can propagate in such collaborative supply chains across partners and have far-reaching effects, and even disrupt the supply chain. The well-known bullwhip effect (Lee et al. 1997) may also result from such propagation.

In an integrated supply chain with tight delivery times and low tolerances, unexpected events or exceptions occur almost regularly because of gaps between planning and actual execution in a dynamic environment (Asgekar 2003). Therefore, a supply chain must be able to handle such events not only responsively but also proactively. This

¹ This author's work on this paper was done at Penn State University.

reflects a need for *sense-and-respond* capability (Kapoor et al. 2005). With sense-and-respond capability, a supply chain can identify changing customer needs and new business challenges as they happen and respond to them quickly and appropriately (Haeckel 1999). In this paper, we develop a formal modeling approach to managing supply chain events and building sense-and-respond capability by capturing timing and causality issues accurately.

A supply chain event is "any individual outcome (or non-outcome) of a supply chain cycle, (sub) process, activity, or task" (Alvarenga and Schoenthaler 2003). Events are correlated with each other to form a network of events; some events have significant consequences, and therefore they must be monitored closely, while others are of lesser importance. The critical problem lies in extracting the significant events and responding to them in real-time. Doing so requires an ability to monitor them proactively, simulate them to help decision-making, and use them to control and measure business processes (Montgomery and Waheed 2001, Strozniak 2002). In this paper, we present a methodology that uses a Petri net approach for formulating supply chain event rules and analyzing the cause-effect relationships between events.

Petri nets are a powerful modeling technique for problems involving coordination in a variety of domains. A variant of Petri nets called time Petri nets allows us to model temporal intervals also. Considering the dynamic characteristic of supply chain events, such Petri nets are useful for describing the time constraints associated with events. Examples of temporal constraints are: "*event e_1 follows event e_2 after time T* " and " *N occurrences of event e_1 within time T lead to event e_2* ". These temporal constraints are important for proper correlation between events; otherwise, the management could be unable to anticipate events or track causes of events. To deal with variety in case data (e.g. order ids, order quantities, rush orders versus normal orders, etc.) we extend the model with "token colors".

Thus, using *time colored* Petri nets, we can model event patterns common in Supply Chain Management (SCM). We have proposed seven event patterns as the basic building blocks for Petri nets (Liu et al. 2004). As an extension to the previous work, in this paper we show how the seven basic event patterns can be used for event causality analysis through an extensive example and simulation experiments. Further, we show how sensitivity analysis can help to improve supply chain performance by simulating the effect of various strategies. In our example, we compose these event patterns to build a complete Petri net model for a Vendor Managed Inventory (VMI) case situation. Using this Petri net as input, dependency graphs are constructed to analyze cause-effect relationships between events.

A variety of Supply Chain Event Management (SCEM) systems is offered by companies such as SAP, i2, and Manugistics (Strozniak 2002). Most systems mainly perform monitoring and provide "early warning," rather than analyzing events and suggesting solutions (Asgekar 2003, Bodendorf and Zimmermann 2005, Marabotti 2002, McCrea 2005, Montgomery and Waheed 2001, Strozniak 2002). Actually, the more powerful part of SCEM would be the capability of "aggregating data from key business systems at a high level and presenting the ramifications of exceptions and the possibilities of solutions" (Marabotti 2002). Therefore, this research can contribute to the research area of SCEM in three ways. First, this work introduces a formal and general approach to modeling events and event rules, and the approach provides flexibility in associating occurrence counts and temporal constraints with events, avoiding the customization problem that often poses an obstacle to the implementation of the existing SCEM systems (Bodendorf and Zimmermann 2005). Second, this approach allows excellent event analysis, including event forecasting with temporal information and causality analysis, which provide real-time visibility about the implications of events and traceability to the root causes for events. Third, it offers a way to track supply chain performance metrics by events, and shows how through simulation, decision makers can compare different strategy alternatives or fine-tune a solution in terms of key performance indicators.

The paper is structured as follows. The next section on "Background" reviews related literature and gives an overview of events, event rules, event aggregation, event causality, and our notion of a dynamic supply chain. The section on "Petri net preliminaries" describes Petri nets briefly and serves as an introduction for the lay reader. Next, in the section "Event formulation", we review event semantics and give seven event patterns or building blocks of event rules based on previous work. A complete event Petri net can be constructed easily by using these blocks. Then, the section on "Example case" models an extensive supply chain using our ideas. Later the section on "Simulation" gives simulation results of the example to illustrate the practical value of our approach. The last section concludes the paper with a brief description of future work.

Background and Overview of Supply Chain Events

Literature Review

Related research for detailed modeling of supply chain events is still limited. One area where events related to data manipulation operations have been systematically studied is active databases, which rely on event-condition-action (ECA) rules (McCarthy and Dayal 1989). Such rules make databases "active" by allowing them to react to events, i.e., when an event occurs, if some conditions hold, an action (such as database update, insert, query) is taken. For example a supply chain rule would be expressed as:

Event: shipment arrival

Condition: actual arrival time - expected arrival time > 1 day

Action: notify manager about late arrival

This ECA rule states that upon the arrival of a shipment, if the shipment is late by one day, then the manger should be notified. While ECA rules are useful, their drawback is that they cannot do event chaining in a natural way, and hence cannot easily facilitate the analysis of cause-effect relationships between events. Thus, it is difficult to trace back the causes of events or forecast future events based on ECA rules. Moreover, typically temporal attributes cannot be modeled explicitly with ECA rules. Notably, Chakravarthy et al (1994) proposed an approach to modeling primitive events with certain temporal attributes and then to detecting immediate composite events by constructing event trees, but this approach does not consider composite events that occur within certain time intervals as the delayed consequences of primitive events.

Event management has also been studied with considerable interest and success in the area of network management. Here the objective is to manage a large number of low-level events that may be related and to extract high-level events that require management attention while ignoring the unimportant ones. Hasan et al. (1999) provide a conceptual framework for describing causal and temporal relationships between network events. Gruschke (1998) gives a dependency graph based algorithm for event correlation in networks. This algorithm is used to map raw events in the network to faulty objects based on the links in the graph. These approaches are relevant in supply chains also, but they lack a precise representation of temporal constraints. In an approach proposed by Casati et al. (1998), time Petri nets are integrated into databases and used for semantic mapping of events in computer networks. The transitions are associated with guard conditions expressed as database constraints. It is an interesting approach with possible applications in supply chains, but harder to implement and verify. In particular, there is no standard approach to transform an event rule to a Petri net, and temporal constraints are captured in an ad-hoc way.

A case-based approach for event correlation in networks is given by Lewis (1993). This method compares a new case against a database of cases and looks for stored solutions; however, it requires an application-specific model and is computationally complex. Pattern discovery and specification techniques for alarm correlation are discussed in (Gardner and Harle 1998, Wu et al. 1998) in the context of networks. For instance, in Gardner and Harle (1998), a series of events is considered as a stream, and a stream language is defined for describing various erroneous patterns of event occurrences. The language supports a rich vocabulary of predicates using sequence operators (*and*, *or*), and modifiers (*last*, *nth*, *any*, *none*). An example of stream processing is as follows:

Stream B = block(Stream A) if type is_duplicate within 10 sec

This statement requires stream A to be scanned and duplicate alarms (or events) that occur within 10 seconds to be blocked, thus producing Stream B. These approaches can be applied in supply chains also, but they are tailored more toward network management. Similarly, our approach has a comparable expressive power and is more suitable for supply chain management, although it can be applied in other areas also.

Finally, Petri nets have been used to model rules in knowledge bases in previous research (Liu and Dillon 1991, Meseguer 1990, Zhang and Nguyen 1994), but these approaches do not consider color and time. In general, most event modeling approaches either do not consider temporal constraints or they capture such constraints in an ad-hoc way, making them hardly applicable to other domains. Supply chain events and event rules are typically associated with temporal constraints. Without capturing these constraints properly, a supply chain may not be able to detect events, analyze the consequences of these events, and respond to them in a timely manner, as showed next.

Events and Causality in a Supply Chain

When supply chain partners are integrated, events at one partner impact other partners, and their responses to these events may cause a storm of events. Therefore, causality analysis is the key to controlling such a storm. Our analysis begins with events and event rules.

There are two types of events: *simple* and *composite*. *Simple* or *primitive events* are captured directly during a process. For example, event "out of stock" is the result of the "check availability" task, and "order arrival" arises from other supply chain partners' ordering process. *Composite events* are derived from simple events by event aggregation. A composite event is deduced when a group of simple events occurs (Luckham 2002). A group of simple events may together reveal potential problems. For example, if a product is out of stock only once in a month, perhaps it is quite normal and an alarm should not be generated, but *if this stock out happens two times in a week, then it may reflect some underlying problems in the product supply chain, and this should be recognized by generating an event*. As another example, a group of stock trading events, related by accounts, timing, and other data, taken together, may constitute a violation of a policy or a regulation (Luckham 2002). Event aggregation is a mechanism to filter simple events and extract meaningful information from them by setting up alarms in advance.

Thus, event aggregation extracts value from a management point of view out of trivial and unorganized simple events. In order to achieve this objective, it is important to recognize event patterns and set up *event aggregation rules*. Besides aggregation rules, business rules must also be considered. Business rules capture the causal relationships between events. For example, if an order is delayed for more than time T , then it is automatically cancelled. Therefore, a rule is needed to express that the event "order delayed by T " is a cause of event "order cancelled".

Moreover, a supply chain is viewed as a series of synchronous and asynchronous interactions among trading partners. Usually when an event, particularly an exception, happens, the trading partner responsible for it may react to this event within a reasonable resolution time to resolve it. For instance, suppose an order is delayed for delivery. If the delay is within an acceptable range specified by the customer, the customer is notified of the delay and the order is processed. However, if the delay exceeds the acceptable tolerance (also called expiration time), the order should be automatically cancelled, and hence, the event "order delay" is not relevant in this case. On the other hand, a series of new actions arises because of this new event, such as canceling the order, removing any reservations made, refunding any payments, etc. Therefore, to model events and event rules precisely, our modeling approach should be able to capture such temporal constraints correctly. In our analysis, each event is associated with two time values: resolution time and expiration time. In most cases, event resolution takes an unpredictable amount of time because of complexities of various business situations, and it is more realistic to set up a resolution time interval. Therefore, in order to model such temporal constraints, we choose to use time Petri nets, where these constraints can be modeled explicitly as transition waiting times. Next, we will briefly introduce the concepts of time colored Petri nets and then show how to capture the dynamic aspect of events.

Petri Net Preliminaries – A Brief Introduction

A Petri net is a directed graph consisting of two kinds of nodes called *places* and *transitions*. In general, places are drawn as circles and transitions as boxes or bars. Directed arcs connect transitions and places either from a transition to a place or from a place to a transition. Arcs are labeled with positive integers as their weight (the default weight is 1). Places may contain tokens. In Figure 1, one token is represented by a black dot in place $p1$. A marking is denoted by a vector M , where its p th element $M(p)$ is the number of tokens in place p . The firing rules of Petri nets are (Murata 1989):

- (1) A transition t is *enabled* if each input place of t contains at least $w(p,t)$ tokens, where $w(p,t)$ is the weight of the arc from p to t . (By default, $w(p,t)$ is 1.)
- (2) The *firing* of an enabled transition t removes $w(p,t)$ tokens from each input place p of t , and adds $w(t,p)$ tokens to each output place p of t , where $w(t,p)$ is the weight on the arc from t to p .

There is another special type of arc called the inhibitor arc with a small circle rather than arrow at the end. An inhibitor from a place to a transition prohibits the transition from being enabled and, thus, firing, if there is a token in the place. An example of an inhibitor arc is given later

In this paper, we use *Time Colored Petri Nets* (TCPN), i.e., Petri nets extended with *time intervals* and *token values*. First, the above classical Petri nets can be extended by associating a time interval $[I_1, I_2]$ with each transition, where I_1 (I_2) is the *minimum* (*maximum*) time the transition must wait for before firing *after* it is enabled. Such a Petri net is known as Time Petri net (TPN) (Wang 1998). If $I_1 = I_2$, we just associate one time value with each transition, while if the interval is not specified, then $I_1 = I_2 = 0$. Analysis techniques for TPNs are discussed in (Berthomieu and Diaz 1991, Wang 1998). Second, tokens can be tagged with data values (or a color) to create a colored Petri net (CPN) (Jensen 1996, Jensen 1998). For example, we use tokens of different colors (or values) for each order or product. For a given place, all tokens must be from one color set.

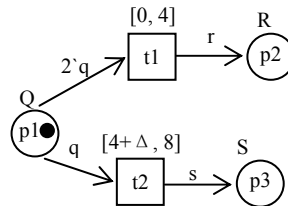


Figure 1: Colored Time Petri Net

In Figure 1, Q , R , and S represent different color sets. q , r , and s are variables, such that $q \in Q$, $r \in R$, and $s \in S$. In a Time colored Petri net (TCPN), the arcs are also labeled with colors. For example, in Figure 1, two tokens colored "q" are consumed if transition $t1$ fires. The fired transition $t1$ will put one token colored "r" in place $p2$. Moreover, if there are two tokens colored "q" continuously existing in place $p1$, transition $t1$ will fire no later than time 4. If there is still a token colored "q" remaining in place $p1$ after time 4 (relative to arrival of this token), transition $t2$ will fire shortly after time 4 (denoted as $4+\Delta$, where Δ is a very short time period, close to 0) and before or at time 8.

Event Formulation and Event Patterns

Event Semantics

Having given a preliminary introduction to Petri nets, now we turn to develop the techniques to formulate event related rules as Petri net structures. In most cases, events are not only the triggers but also consequences of supply chain tasks, i.e., one event causes another event. Therefore, it is quite natural to model events as places that represent pre-conditions or post-conditions of transitions. Thus, events and places will be used interchangeably while modeling events. Moreover, time Petri nets offer an attractive choice for modeling the dynamic aspect in supply chains. To make such models, we first formulate events and event rules as follows:

Event rule R : $e_1(n_1x_1, I_0) \xrightarrow{[I_1, I_2]} e_2(x_2)$, where

e_1 : input event class

n_1 : number of event instances (for simplicity, we just say events), i.e., number of tokens (by default, $n_1 = 1$).

x_i : data value of event i for $i = 1, 2$. In other words, the color of tokens, $x_i \in$ color set X_i .

I_0 : expiration time of e_1 .

\rightarrow : "imply" or "lead to", which establishes a cause-effect relationship between the left side and right side of the rule.

$[I_1, I_2]$: an optional time interval which corresponds to the event resolution time. In order not to make the problem trivial, we require $I_2 < I_0$. If this interval is not specified, we assume $I_1 = I_2 = 0$

e_2 : output event class. For every rule, only *one* instance of e_2 is generated because it is not necessary to repeat supply chain events.

This event rule shows the semantics of event e_1 succinctly. Suppose e_1 continues to arrive at a system. If the number of its occurrences reaches a threshold, say n_1 , and these events persist in the system long enough, event rule R can be triggered during interval $[I_1, I_2]$, and e_2 is then generated. I_0 is the expiration time of e_1 . If rule R does not fire within the $[I_1, I_2]$ interval, then e_1 expires. After e_2 occurs, e_1 may normally be consumed by rule R . However, if e_1 is

required by another rule, then a token should be returned to e_1 . Hence, two representations are possible for event rules:

Representation 1 (consumption case - event e_1 is consumed): This case can be modeled as a Petri net shown in Figure 2. This representation is useful when an event is not required again, say by another rule.

Representation 2 (non-consumption case - event e_1 is not consumed): Event e_1 is not consumed because it may be required by another rule. Nevertheless, event e_2 must not be generated multiple times from these occurrences of e_1 . This case can be accurately modeled as a Petri net as shown in Figure 3.

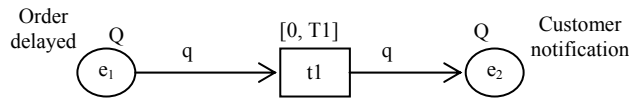


Figure 2: Petri Net of Example 1 Showing a Rule R

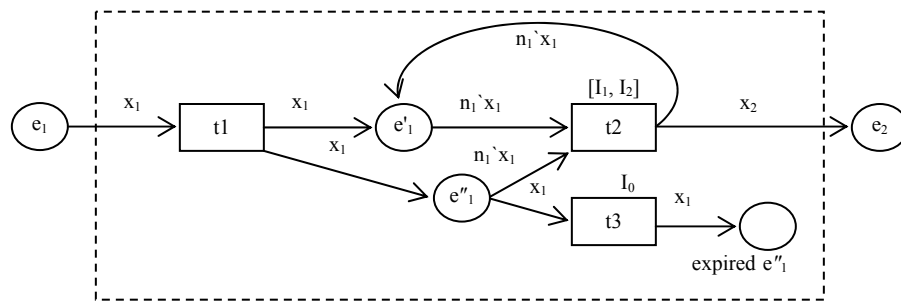


Figure 3: Petri Net Representation for Non-consumption Case

When comparing Figure 2 and Figure 3, several differences should be noted. First, events are not consumed in Figure 2. Second, the representation chosen in Figure 3 abstracts from color sets and focuses on timing issues and causalities. Third, it allows for multiple, say n_1 , events to occur to trigger another event. Since event e_1 is not consumed by rule R , we need a special mechanism to prevent event e_2 from being generated repeatedly. Therefore, as Figure 3 shows, place e_1 is first transformed into two auxiliary places, e'_1 and e''_1 , by a transition $t1$. Tokens in e''_1 are consumed if transition $t2$ fires, while $n_1 x_1$ tokens (denoted as $n_1 \cdot x_1$) are brought back to place e'_1 . (Recall that here n_1 events are needed to enable transition $t2$.) Therefore, after the first firing, although there are $n_1 x_1$ tokens in place e'_1 , transition $t2$ cannot fire, and, thus, at most one e_2 event is generated (with respect to $n_1 x_1$ tokens). If transition $t2$ does not fire (because of insufficient tokens in e'_1), e''_1 expires at the end of expiration time by the firing of transition $t3$.

These two general representations are employed in our various patterns in the next section.

Event Patterns to Model Supply Chain Rules

Next we will develop several patterns for constructing complex temporal event relationships and also give equivalent logical expressions for these patterns. In general, three logic connectives, OR (\vee), AND (\wedge), Negation (\neg), can be used on either the left or the right side of an event rule. Since modeling of time is crucial in understanding the behavior of our Petri net models, we call these patterns *temporal event patterns*.

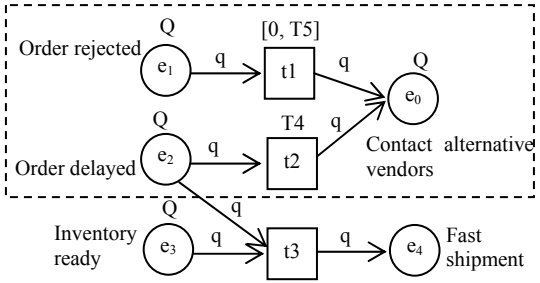
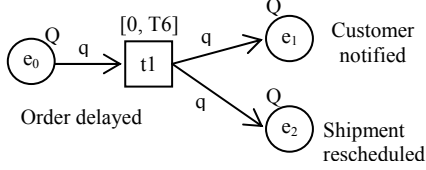
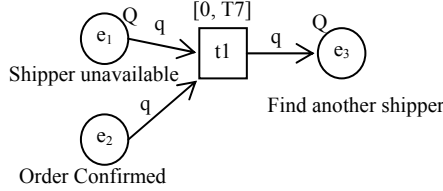
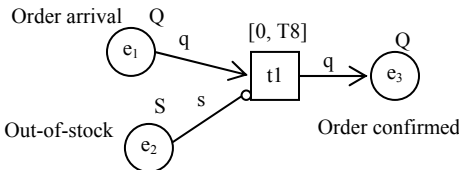
Seven basic event patterns were proposed in previous work (Liu et al. 2004) as a means to model a large variety of typical supply chain events. Table 1 summarizes these patterns and gives illustrative examples. Note that in Table 1, the non-consumption case is illustrated by Example 2-3 with explicit expiration times; other examples assume that events are consumed after firing rules. In general, the notion of expiration times can be applied to any pattern.

These patterns allow us to capture sophisticated relationships involving multiple event instances, event expiration times, and resolution times. To illustrate, we briefly discuss Example 3 (Inclusive choice pattern). When an order is confirmed (see Figure 5), a token is placed in place e'_0 and e''_0 as well. Transitions $t1$, $t2$, $t3$, and $t4$ are enabled but do not fire at that moment. If this token is consumed by the shipment transition $t5$ before time $L2$ (relative to its arrival), transitions $t1$, $t3$, and $t4$ are disabled, but transition $t2$ will fire at time $L2+T3+2\Delta$ after the token arrival. Otherwise, if during the time interval $[L2, L2+T3]$ this token remains in place e'_0 , transition $t1$ will fire. After transition $t1$ fires, this token is immediately brought back to e'_0 because some other rules (like $t4$) may use it later. If there is still a token in e'_0 after $L2+T3$, transition $t4$ fires and produces event "order cancelled". Thus, the token in e'_0 is consumed. In general, if this rule is triggered, it can produce two possible results: order delayed and cancelled, or only order delayed, depending upon the temporal relationships. One can see this rule actually has complex semantics, yet its Petri net model can precisely describe such temporal relationships. Note that in Figure 5, transition $t3$ never fires and it can be removed. We keep this transition in the figure for consistency with event semantics.

Table 1: Event Patterns and Examples

Patterns and Examples	Petri net Representations
<p><u>Pattern 1</u> (<i>simple cause-result</i>) shows that event e_1 can cause event e_2 within a time period $[I_1, I_2]$.</p> <p><u>Example 1:</u> If an order is delayed (e_1), contact customer (e_2) before time $T1$:</p> $e_1(q) \xrightarrow{[0, T1]} e_2(q)$ (Note: q is order number)	<p>See Figure 2</p>
<p><u>Pattern 2</u> (<i>Repeat cause-one effect</i>) concerns the case where multiple occurrences of one event within a certain time period cause another single event to occur.</p> <p><u>Example 2:</u> If product s is out of stock (event e_1) more than once within period $T2$, contact the supply chain manager (event e_2):</p> $e_1(2s, T2) \xrightarrow{[0, \Delta]} e_2(s)$ (Note, s is the product ID).	<p>Figure 4: Petri Net of Example 2 (Pattern 2)</p>
<p><u>Pattern 3</u> (<i>Inclusive choice</i>) describes the scenario where multiple, alternative events can occur based on temporal conditions.</p> <p><u>Example 3:</u> If an order, with lead time $L2$, has not been shipped (i.e., not consumed by some other rule) within time $L2$ after it is confirmed (e_0), the order is treated as delayed (e_1) (but e_0 is not consumed yet); however, if an order is delayed by more than time $T3$, it is treated as undeliverable and cancelled (e_2). (Perhaps the customer does not want it if the delay is more than $T3$. So e_0 is consumed at this time.). This rule can be formulated as:</p> $e_0(q, L2+T3+2\Delta) \{ [-\frac{[L2, L2+T3]}{\rightarrow e_1(q)}] \vee [-\frac{L2+T3+\Delta}{\rightarrow e_2(q)}] \}$	<p>Figure 5: Petri Net of an Order Process (Pattern 3)</p>

Table 1: Event Patterns and Examples

Patterns and Examples	Petri net Representations
<p>Pattern 4 (<i>1 of N causes – single result</i>): A result can have multiple alternative (combination of one or more) causes.</p> <p><u>Example 4</u>: Contact alternative vendors (e_0) in $T5$ when a rush replenishment order is rejected (e_1), or after it is delayed (e_2) by more than time $T4$ (if the delay is less than $T4$, the delayed time can be compensated by faster shipment), i.e., $\{[e_1(q) \xrightarrow{[0, T5]}] \vee [e_2(q) \xrightarrow{T4}] \}$; e_0</p>	 <p>Figure 6: Petri Net of Example 4 (Pattern 4)</p>
<p>Pattern 5 (<i>1 cause – N results</i>): This pattern recognizes that a cause may have multiple consequences and captures all <i>concurrent</i> consequences of a particular event.</p> <p><u>Example 5</u>: If an order is delayed (e_0), notify customer (e_1) and reschedule the shipment (e_2) immediately (say, in a short time $T6$), i.e., $e_0(q) \xrightarrow{[0, T6]} [e_1(q) \wedge e_2(q)]$</p>	 <p>Figure 7: Petri Net of Example 5 (Pattern 5)</p>
<p>Pattern 6 (<i>N causes – 1 result</i>): This pattern is the reverse of the above pattern, and it is used to model the concurrent causes of a particular event.</p> <p><u>Example 6</u>: When the shipper of a confirmed order (e_2) is not available (e_1), find another shipper (e_3) in a short time $T7$, i.e., $[e_1(q) \wedge e_2(q)] \xrightarrow{[0, T7]} e_3(q)$</p>	 <p>Figure 8: Petri Net of Example 6 (Pattern 6)</p>
<p>Pattern 7 (<i>non-occurrence of an event</i>): Non-occurrence of an event can also signal valuable information. Typically, non-occurrence of an event and occurrence of some other events may, in conjunction, cause some other significant events to happen.</p> <p><u>Example 7</u>: When an order arrives (e_1), if there is no out-of-stock (e_2) situation, the order is confirmed (e_3) in time $T8$, i.e., $\{e_1(n_1x_1, I_{10}) \wedge [\neg e_2(n_2x_2, I_{20})]\}$ $\xrightarrow{[I_{31}, I_{32}]} e_3(x_3)$</p>	 <p>Figure 9: Petri Net of Example 7 (Pattern 7)</p>

In this section, we have developed seven basic patterns that capture cause-effect relationships in Petri nets. These seven canonical patterns were chosen because they capture a variety of scenarios, and they can also be combined to create new patterns. Next, we show how these patterns can be used as building blocks to model a complex supply chain.

An Example Case: Modeling and Event Causality Analysis of a Supply Chain

In this section, we will first show a Petri net that is built using the above seven patterns in the context of a realistic supply chain scenario. Subsequently, we will analyze event causality by simulation and dependency graphs.

Scenario of Events and Rules for a Complete Petri Net

First, we will give an example scenario description. Suppose there is a Vendor Managed Inventory (VMI) arrangement between a distributor and a vendor. In this arrangement, the vendor manages the inventory level for the distributor, proposes the new supply orders to the distributor, and ships them after the distributor's approval. The distributor sells products to its customers, and normally ships its customers' orders (for simplicity, we just call them orders) from stock, but whenever there is an out-of-stock situation, a rush supply order is placed with the vendor. When there is more than one out-of-stock event in a week at the distributor, this situation should be considered as a supply chain exception and reported to the supply chain manager immediately. The vendor would usually respond to rush supply orders as soon as possible, but they may be rejected if there is a serious production delay. Moreover, in case of production delay, all supply orders may be delayed. The distributor can contact an alternative vendor for replenishment in case that its normal or rush supply order is delayed or rejected. Figure 10 shows all the trading partners in such a supply chain. *For simplicity, in this figure we assume there is one product, but one can similarly model multiple products also, as we show in the next section.* First, we need to identify the events and then write the rules that connect them together. The events of interest are summarized in Figure 11, and each event corresponds to a place in the Petri net.

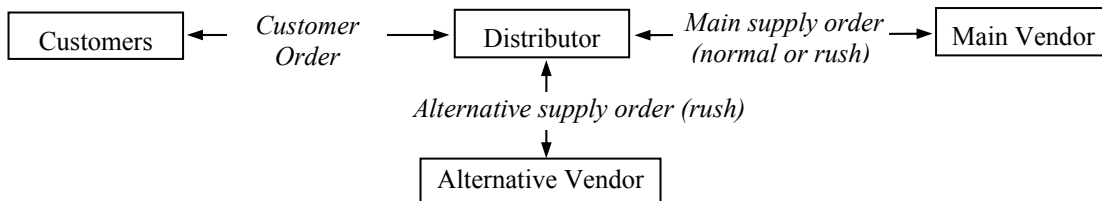


Figure 10: Interactions between Trading Partners in the Example Supply Chain

Place (or event) description	
p1: Customer order arrival	p2: Out-of-Stock
p3: Back order	p4: Rush supply order
p5: Rush supply order confirmed	p6: Customer order confirmed
p7: Customer order delayed	p8: Notify customer of order delay
p9: Customer order cancelled	p10: Customer order shipped
p11: Out-of-Stock event expires	p12: Notify supply chain manager
p13: Rush supply order rejected	p14: Production delay
p15: Supply order delayed	p16: Contact alternative vendors
p17: Stock unavailable when delivery is due	p18: Rush supply order shipped
p19: Alternative sourcing failed	p20: Customer order rejected
p21: Production delay (p14) resolved	p25: Back order cancelled
p22: p2' expired	p23: p3' expired
p26: p5'' expired	p24: p5' expired
	p27: p6' expired
	p28: p2'' expired

Figure 11: Possible Events in the Supply Chain

Next, we consider the rules that relate these events to one another, and also refer to the corresponding patterns used for modeling these rules (in parentheses).

1. When a customer order arrives ($p1$) and there is no out of stock (not $p2$), the order is confirmed ($p6$). (Pattern 7: *Non-occurrence*)
2. When a customer order arrives ($p1$) but there is an out-of-stock ($p2$), a back order ($p3$) is generated. (Pattern 6: *N causes – 1 result*)
3. When a back order occurs, a rush supply order with lead time $L1$ is sent to the vendor ($p4$). (Pattern 1: *Simple cause-effect*)
4. When the rush supply order is confirmed by the vendor ($p5$), the back order is also confirmed to the customer ($p6$). A back order must be confirmed within $L2+T3$, where $L2$ is the lead time of the back order, and $T3$ is the maximum allowed delay time; otherwise, it expires and is cancelled ($p25$) (Pattern 6: *N causes – 1 result*)
5. If there is a production delay ($p14$), any incoming rush supply order is rejected ($p13$), because there is no production capacity left to fulfill any rush supply order in a short time. Otherwise, the rush supply order is confirmed. A production delay can be resolved in time interval $[a, b]$. (Pattern 6: *N causes – 1 result*; Pattern 7: *Non-occurrence*)
6. A rush supply order is shipped during time $[0, L1]$ if there is no production delay (not $p14$). (Pattern 7: *Non-occurrence*)
7. A production delay ($p14$) can cause a supply order delay for more than time $T4$ ($p15$) and unavailable inventory when customer order delivery is due ($p17$). (Pattern 5: *1 cause – N results*)
8. If a rush supply order is rejected ($p13$) or delayed for more than time $T4$ ($p15$), contact alternative vendors for alternative sourcing ($p16$). (Pattern 4: *1 of N causes-single result*)
9. When a rush supply order is shipped ($p18$) by one of alternative vendors, the corresponding back order can be confirmed ($p6$) and shipped ($p10$); otherwise, the customer order can be rejected ($p20$). (Pattern 6: *N causes – 1 results*; Pattern 1: *Simple cause-effect*)
10. When a supply order is shipped from a vendor ($p18$), inventory is available for delivery (so if there is a token in $p17$, it is removed). (Pattern 6: *N causes – 1 result*)
11. When delivery is due, if inventory is available (not $p17$), the order is shipped ($p10$). (Pattern 7: *Non-occurrence*)
12. a. If an order (with lead time $L2$) has not been shipped in time $L2$ after it is confirmed ($p6$), the order is delayed ($p7$).
b. If an order is delayed ($p7$) more than time $T3$, then the order is cancelled ($p9$). (Pattern 3: *Inclusive choice*)
13. If there are two unresolved out-of-stock events ($p2$) during time $T2$, the supply chain manager is contacted immediately ($p12$). (Pattern 2: *Repeat_cause-one_effect*)
14. If the order is delayed ($p7$), notify the customer at time $T1$ ($p8$). (Pattern 1: *Simple cause-effect*)

The above 14 rules can be easily formulated in terms of colored time Petri nets as shown in Figure 12. The transitions are labeled by the corresponding rule number for ease of reference. For example, the transition with label "2-1" is a part of the formulation of Rule 2. The darkened places in the figure are input events of this net. Place $p1$ contains two different tokens representing the two order arrivals. Events that are not consumed by event rules are transformed into multiple places, such as $p2$, $p2'$, and $p2''$, where $p2$ holds tokens for events, and the others are special mechanism for preventing repetitive firing of transitions. This point was explained in the section on "Event Semantic" in the non-consumption case.

The Petri net was implemented using CPN Tools (Ratzer et al. 2003), a graphical computer tool supporting colored Petri nets. In addition, to modeling complex supply chain scenarios, we can also use hierarchical Petri nets. For example, event rules can be represented as sub Petri nets, and those sub Petri nets are composed to achieve a model at a higher level. Thus, hierarchical Petri nets allow good scalability. The details of hierarchical Petri nets can be found in Jensen (1996). Next, we describe dependency graph analysis based on the Petri net model of Figure 12.

Dependency Graph Analysis

The Petri net shown in Figure 12 can be considered as an "event machine," i.e., when fed with input events, it will generate a set of composite events (both intermediate and final), and show the causal relationships between them. The behavior of this "machine" for the life of a particular instance or for a given time period can be represented by a simple dependency graph (Gruschke 1998). A *dependency graph* is a cause-effect graph of events produced from one or more Petri net instances (say, one or more orders) over a time period. The dependency graph is created from the Petri net by using the rule that *the output event(s) of a transition depends upon its input event(s)*.

By executing the Petri net model with actual case data, we can create dependency graphs to show causal relationships that actually transpired between events. Table 2 describes the sequence of event occurrences and the transitions that fire when the events take place. The relationships are reflected in Figure 13 that shows an event dependency graph generated based on the Petri net of Figure 12. Moreover, it also gives the correspondence between place numbers and event numbers. (Note that the events and the corresponding place numbers are not always the same.) The table also gives time values in the last column. These times are based on assigning suitable values for a hypothetical case to the parameters of Figure 12 as follows in time units (say, days):

$L1 = 20, L2 = 50, T1 = 1, T2 = 50, T3 = 20, T4 = 10, a = 60, b = 80.$

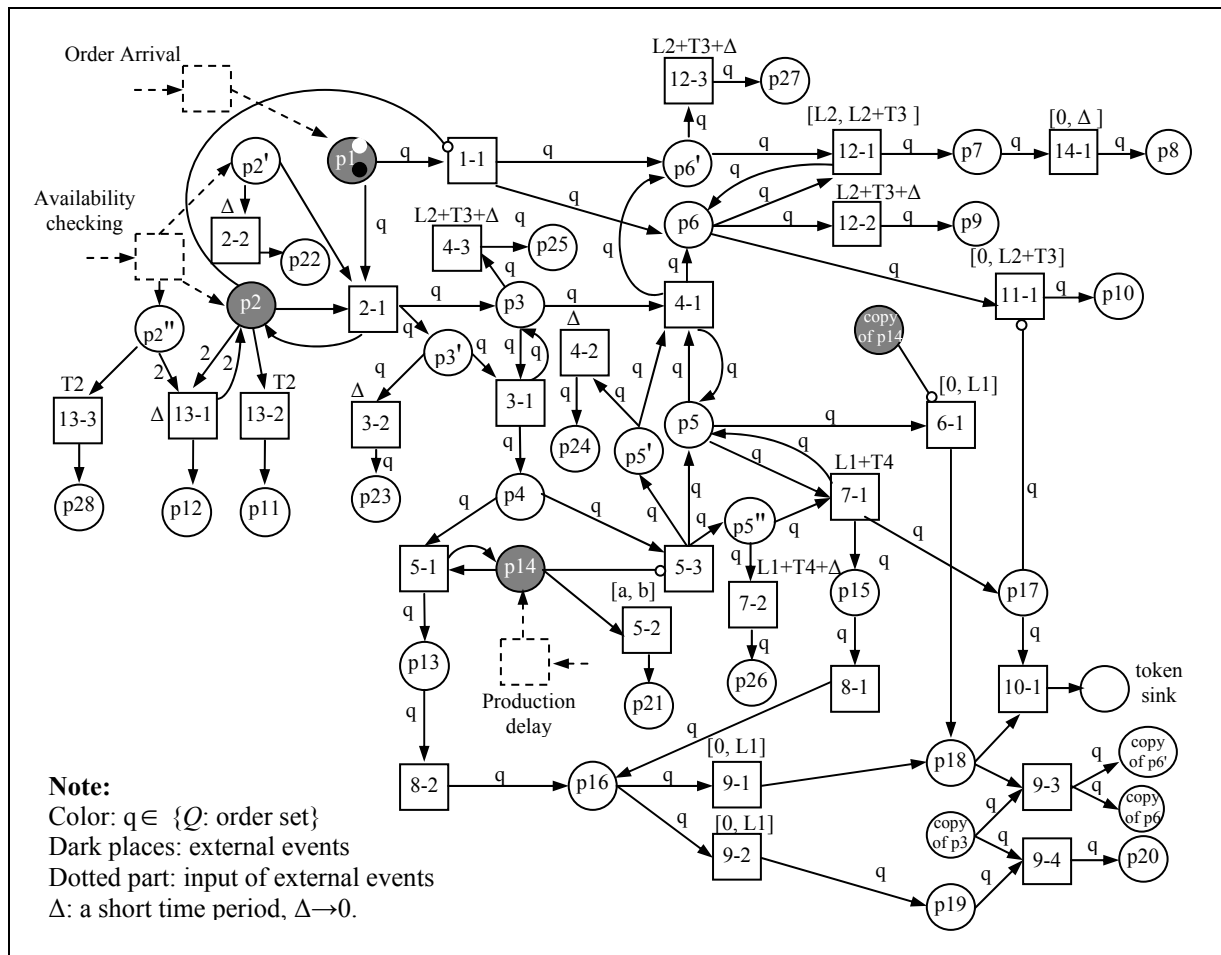


Figure 12: A Supply Chain Event Petri Net

Figure 13 enables us to analyze the various events and their causes. The events that represent *exceptions* are shaded in this figure. The consequences of a particular event can be traced forward along this directed graph, while the causes of it should be traced backwards until one or more root nodes are reached. For example, it is not difficult to see that $E8$ and $E12$ are the main causes of exception $E13$, i.e., product A was out of stock with the distributor and a

rush supply order $R2$ was issued, but this rush supply order was rejected by the vendor because of a production delay. Similarly, the graph shows that the ultimate exceptions resulting from $E12$ are $E21$, $E22$, and $E24$. The sequence of main events is as follows:

Production is delayed ($E12$) → rush supply order $R1$ is also delayed ($E15$) → another vendor is contacted ($E17$) → alternative sourcing failed ($E22$) → Order $O1$ cancelled ($E24$)

This sequence of events actually suggests ways to avoid exception $E24$ (Order $O1$ cancelled). For example, one strategy could be reducing the production delay time (i.e., resolving $E12$ quickly) so that rush supply orders can still be fulfilled within their lead times. Another strategy could be raising the level of safety stock to counter the production delay. Also, we can improve the probability of successful alternative sourcing to reduce event $E22$. In general, we can introduce different strategies to resolve each exception in this sequence and then avoid the final undesirable consequence. As an illustrating example, we will simulate some strategies and compare their effectiveness shortly.

Moreover, notice that the exception $E11$ (notify supply chain manager) happens because of two stock-out events of product A within 50 time units as denoted by events $E2$ and $E8$. Thus:

Stock out of A for order $O1$ ($E2$) & Stock out of A for order $O2$ ($E8$) → Notify supply manager ($E11$)

Table 2: A Trace of Possible Event Sequence Generated from Figure 12

Event	Description	Trans. fired	Place	Time
E1	Order $O1$ arrival	-	p1	0
E2	Out-of-Stock of product A (for $O1$)	-	p2	0
E3	$O1$ is on back order	1-1	p3	0
E4	Rush supply order $R1$ is placed for $O1$	3-1	p4	0
E5	Supply order $R1$ is confirmed to customer	5-3	p5	0
E6	Order $O1$ is confirmed	4-1	p6	0
E7	Order $O2$ is received	-	p1	10
E8	Product A is out-of-stock (for $O2$)	-	p2	10
E9	$O2$ is placed on back order	1-1	p3	10
E10	Rush supply order $R2$ is placed for $O2$	3-1	p4	10
E11	Contact supply chain manager	13-1	p12	10
E12	Product A production is delayed	-	p14	10
E13	Rush supply order $R2$ is rejected	5-1	p13	10
E14	Alternative vendor is contacted for $R2$	8-2	p16	10
E15	Rush supply order $R1$ is delayed for time $T4$	7-1	p15	30
E16	Product A is unavailable when $O1$ is due	7-1	p17	30
E17	Alternative vendor is contacted for $R1$	8-1	p16	30
E18	Rush supply order $R2$ is shipped from the alternative vendor (i.e., non-occurrence of event “product unavailable when $O2$ due”)	9-1	p18	30
E19	Order $O2$ is confirmed	9-3	p6	30
E20	Order $O2$ is shipped	11-1	p10	31
E21	Order $O1$ is delayed	12-1	p7	50
E22	Alternative sourcing attempt for $R1$ failed	9-2	p19	50
E23	Notify customer about order $O1$ delay	14-1	p8	50
E24	Order $O1$ is cancelled	12-1	p9	71

Actually, Figure 13 only shows one possible scenario and gives the ultimate disposition of orders $O1$ and $O2$ ($O1$ was cancelled, while $O2$ was fulfilled). Figure 14 shows another out-of-stock situation during order fulfillment; however, now the outcome is different. Here, $E16$ (Product A unavailable when $O1$ is due) is resolved by $E18$ (Rush supply order for $R2$ shipped from the alternative vendor). Therefore, order $O1$ is shipped ($E20$) within its lead time. Later on, in spite of $E21$ (alternative sourcing for $R1$ fails), rush supply order $R1$ is shipped ($E22$) from the main vendor after some delay. Eventually order $O2$ is also fulfilled ($E24$) by the incoming inventory from rush order $R1$.

The modified events for this scenario are shown in Table 3 (events *E1* through *E17* are the same as in Table 2). Figure 14 shows the new dependency graph for these events. Nevertheless, *E11* still happens as before.

These two dependency graphs show only two of many possible scenarios and serve to illustrate our approach. The advantage of this approach is that using a Petri net model as an event machine, we can generate dependency graphs to predict and analyze different "interesting" scenarios. Moreover, by playing "token games", supply chain managers can explore a large number of possible event dependency graphs that lead to desirable results (e.g. order fulfilled successfully) or significant exceptions (e.g. order cancellation). The design of an algorithm or heuristic that can automatically generate dependency graphs containing such events of interest is left as a future exercise. In this context, it should be noted that it is possible to analyze all possible dependency graphs using reachability analysis techniques (Berthomieu and Diaz 1991) for time colored Petri nets. However, as discussed there, this is not very feasible for large problems for complexity reasons, and heuristic techniques are required. Next, we provide a summary of simulation results and analyze their implications for supply chain management.

Table 3: An Alternative Scenario of Events Generated from Figure 12

Event	Description	Trans. fired	Place	Time
... Events <i>E1</i> thru <i>E17</i> are same as in Table 2...				
E18	Rush supply order for <i>R2</i> shipped from the alternative vendor.	9-1	p18	30
E19	Product available for <i>O1</i> (token in p17 removed) (i.e., non-occurrence of event "product unavailable when <i>O1</i> due")	10-1	p17	30
E20	Order <i>O1</i> shipped	11-1	p10	30
E21	Alternative sourcing for <i>R1</i> fails	9-2	p19	50
E22	<i>R1</i> Shipped from the main vendor	6-1	p18	80
E23	Order <i>O2</i> confirmed	9-3	p6	80
E24	Order <i>O2</i> shipped	11-1	p10	80

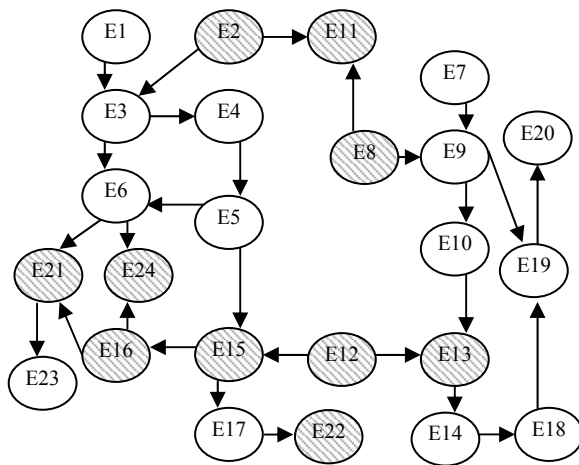


Figure 13: Dependency Graph of Table 2 (exceptions are shaded)

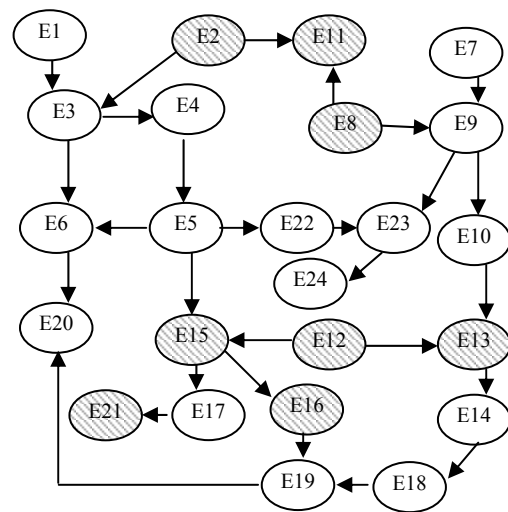


Figure 14: Dependency Graph of Table 3

Simulation Results and Analysis

To demonstrate the practical value of our approach, a detailed simulation experiment was conducted. In this simulation, we generated a large number of customer order arrival events and traced the order fulfillment process in terms of times of occurrence of each event. To make the simulation realistic, we assume there are three products, say

A, B, and C. In general, more products can also be supported. Table 4 shows the parameters of our simulation experiment.

Table 4: Simulation Parameter Settings

Parameter Name	Value or Distribution
Set of items in a customer order	Random selection from three products: A, B, and C
Customer order inter-arrival time	Exponential distribution with mean of 7 time units
Prob. of successful alternative sourcing (PSAS)	0.5
Inter-arrival time between production delayed events	Exponential distribution with mean of 100
Resolution time for production delay (RT)	Uniform distribution range [60, 80]
Normal supply arrival schedule	2 arrivals for each item every 30 time units

The simulation runs for a period from 0 to 3500 time units. 500 customer orders are generated and processed. Among them, 445 orders were successfully shipped, and the other 55 orders were cancelled or rejected because of out-of-stock events and failures to find alternative sourcing. In Table 5, the "baseline case" column summarizes the number of main events generated during the simulation interval. Table 5 also shows that although about one-quarter of customer orders (135 out of 500) occur in stock out situations, yet most of them (84 out of 135) can still be successfully fulfilled through rush supply orders. In addition, about 10% of customer orders (48 out of 500) are fulfilled by alternative sourcing, which shows that alternative sourcing is important.

Table 6 shows the detailed distribution of out-of-stock events by product. Each product accounts for about one-third of these 182 out-of-stock events. In practice, it may be difficult for a supply chain manager to trace each one of these 182 events individually. Using Rule 13 (see the subsection "Scenario of Events and Rules for a Complete Petri Net") we can filter these events and reduce the number of events sent to the manager. Thus, the supply chain manager may be notified only when there are *two out-of-stock events* within a 50 time unit interval. Therefore, the number of events that need management attention is reduced to 80, about 40% of the original number of events. Moreover, the manager can adjust Rule 13 to further reduce this number suitably.

In addition, the events in Table 5 can be used to calculate key performance indexes of the supply chain. As Table 5 shows, the fill rate of customer orders is 89%, and the average time between an order arrival and the shipment of the order is 28 time units. In addition, on average, it takes 54 time units for the main vendor to replenish rush supply orders, because production delays occur frequently (35 delay events), and they last a while before being resolved. In contrast, it takes a shorter average time (10 time units) to get supplies from alternative vendors. In general, since the customer order fill rate is somewhat low, the performance of this system may need to be improved. We show next how this can be done with our approach.

An important aspect of our approach is the ability to do sensitivity analysis. To show how such analysis can help to improve the performance of this supply chain, we alternately considered the effect on performance of changing two parameters: reducing the resolution time of production delays (Strategy 1), and increasing the probability of finding alternative sourcing (Strategy 2). Strategy 1 considers the possibility that a production delay can be resolved in a time interval [30, 50] instead of [60, 80]. For Strategy 2, another alternative vendor is introduced into the supply chain so that the probability of finding alternative sourcing is increased to 0.7. The simulation results of these two strategies are also shown in Table 5. Using Strategy 1, although there is a large number of back orders, more than a half of them (77 out of 135) are still delivered through successful rush supply orders from the main vendor, while only 34 back orders are replenished by alternative vendors. For the second strategy, 70% of back orders (78 out of 111) are fulfilled by alternative vendors. Both strategies lead to an increase in the fill rate of customer orders. Thus, compared with the baseline strategy, Strategy 1 and Strategy 2 can increase the fill rate to 95%. Similarly, other scenarios can be explored and analyzed in detail with this technique.

Table 5: Comparing Different Strategies in Terms of Events

Events	Baseline case	Strategy 1	Strategy 2
	RT = [60, 80] PSAS = 0.5	RT = [30, 50]	PSAS = 0.7
Order arrivals (p1)*	500	500	500
-- Customer order shipped (p10)	445	473	475
-- Customer order cancelled (p9)	4	3	1
-- Customer order rejected (p20)	47	21	20
-- Back order cancelled (p25)	4	3	4
Out-of-stock events (p2) *	182	182	182
Production delay (p14) *	35	35	35
Customer order delayed (p7)	4	4	1
Back order (p3)	135	135	135
rush supply order (p4)	135	135	135
rush supply order fulfilled	84	111	111
-- by main vendor	36	77	33
-- by alternative vendors	48	34	78
Rush supply order rejected by main vendor (p13)	102	60	102
Supply order delayed (p15)	8	16	6
Contact alternative vendors (p16)	110	76	108
Alternative sourcing failed (p19)	62	42	30
Performance Indexes			
Customer order fill rate	89%	95%	95%
Average customer order fulfillment time	28	27	28
Average replenishment time of rush supply orders (main vendor)	54	18	27
Average replenishment time of supply orders (alternative vendors)	10	10	11

*: These are input events. The three strategies have the same input events.

Table 6: Numbers of Out-of-Stock Events

Products	A	B	C	Total
Out-of-stock events	53	66	63	182
Notify supply chain manager of out-of-stock events	24	29	27	80

Conclusions

We developed an approach for modeling event relationships in a supply chain through Petri nets. The formalism consists of seven basic patterns that capture cause-effect relationships in Petri nets. These patterns can be combined together as building blocks to create other patterns and also more complex Petri nets. We used a very extensive example to illustrate this approach and showed in detail how dependency graph analysis can be used to determine causal relationships between events in a dynamic supply chain. It should be noted that these cause-effect

relationships are complex and depend upon the exact timing of events. We demonstrated that slight changes in temporal relationships can result in a very different dependency graph and also final outcome.

Petri net simulation offers a mature technique for analyzing the Petri net models, and the easy availability of many Petri net software packages is an asset. We implemented Petri net models of a supply chain using CPN tools (Ratzer et al. 2003) and performed sensitivity analysis by simulation. By changing a specific event parameter, such as event resolution time, we can show how supply chain performance is affected. Therefore, by managing events, we can actually manage supply chain performance. We ran comprehensive simulation experiments to illustrate how this approach can help decision makers to improve supply chain performance. The simulated strategies were able to improve the supply chain performance significantly.

In summary, as supply chains become more tightly integrated across partners, it is becoming increasingly important to respond in quickly and appropriately to events (also called sense-and-respond capability). We described a novel approach to model event relationships in a supply chain using Petri net patterns that can be combined to create realistic Petri net models of supply chains. We further implemented a model in a Petri net modeling and simulation tool, and ran simulation experiments with it. A unique feature of the approach is that the Petri nets are constructed from patterns or building blocks that can be composed together and extended to create new user-defined patterns. This approach provides a method to managing supply chain performance through events. It may help research in the area of business performance management. For example, Chowdhary et al. (2006) proposed a model-driven framework for business performance management. In this framework, events are first detected and then aggregated in order to recognize situations warranting business actions. Our approach can help in aggregating events and identifying problematic situations through proper event correlation.

In future work, we would like to develop more formal verification techniques for the supply chain models, and also design heuristics for reachability analysis of dependency graphs to predict "interesting" events.

Acknowledgements

The work of the first two authors was supported in part by a grant from IBM.

References

- Alvarenga, C. A. and Schoenthaler, R. C. "A New Take on Supply Chain Event Management," *Supply Chain Management Review*, March/April, 2003, pp. 29-35.
- Asgekar, V. "Event Management Graduates with Distinction," *Supply Chain Management Review*, September/October, 2003, pp. 15-16.
- Berthomieu, B., and Diaz, M. "Modeling and Verification of Time Dependent Systems Using Time Petri Nets," *IEEE Transactions on Software Engineering* (17:3), 1991, pp. 259-273.
- Bodendorf, F. and Zimmermann, R. "Proactive Supply-Chain Event Management with Agent Technology," *International Journal of Electronic Commerce* (9:4), Summer 2005, pp. 57-89.
- Casati, F., Du, W. and Shan, M. "Semantic Mapping of Events," HP Labs Technical, HPL-98-74 980421.
- Chakravarthy, S., Krishnaprasad, V., Anwar, E. and Kim, S. "Composite Events for Active Databases: Semantics, Contexts and Detection," *In Proceedings of the Intl Conference on Very Large Data Bases (VLDB)*, Bocca, J. B., Jarke, M. and Zaniolo, C. (eds.), 1994, pp. 606-617.
- Chowdhary, P., Bhaskaran, K., Caswell, N. S., Chang, H., Chao, T., Chen, S.-K., Dikun, M, Lei, H., Jeng, J.J., Kapoor, S., Lang, C.A., Mihaila, G., Stanoi, I. and Zeng, L. "Model Driven Development for Business Performance Management," *IBM System Journal* (45:3), 2006, pp. 587-605.
- Gardner, R. and Harle, D. "Pattern discovery and specification translation for alarm correlation," *In proceedings of Network Operations and Management Symposium (NOMS'98)*, New Orleans, USA, February 1998, IEEE.
- Gruschke, B. "Integrated Event Management: Event Correlation Using Dependency Graphs", *In Proceedings of the 9th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 98)*, Sethi, A. S. (ed.), Newark, DE, USA, October 1998.
- Haeckel, S. H., *Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations*, Harvard Business School Press, 1999.

- Hasan, M, Sugla, B., and Viswanathan, R. "A conceptual framework for network management event correlation and filtering systems," In *Proceedings of Integrated Network Management VI*, Sloman, M., Mazumdar, S. and Lupu, E. (eds.), Boston, MA, 1999, pp. 233–246.
- Jensen, K. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Volume 1, Springer-Verlag, Berlin Heidelberg, 1996.
- Jensen, K. "An Introduction to the Practical Use of Coloured Petri Nets," *Lectures on Petri Nets II: Applications*, Reisig, W and Rozenberg, G (eds.), Lecture Notes in Computer Science (1492), Springer-Verlag, 1998, pp. 237-292.
- Kapoor, S., Bhattacharya, K., Buckley, S., Chowdhary, P., Ettl, M., Katircioglu, K., Mauch, E., and Phillips, L. "A technical framework for sense-and-respond business management," *IBM Systems Journal*, March, 2005, pp. 5-24.
- Lee, H., Padmanabhan, V. and Whang, S. "The Bullwhip Effect in Supply Chains," *Sloan Management Review* (38), 1997, pp. 93-102.
- Lewis, L. "A case-based reasoning approach to the resolution of faults in communication networks," In *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management*, Hegering, H. G. and Yemini, Y. (eds.), San Francisco, USA, April 1993, pp. 671–682.
- Liu, N. K. and Dillon, T. "An approach towards the verification of expert systems using numerical Petri net," *International Journal of Intelligent Systems* (6:3), 1991, pp. 255-276.
- Liu, R., Kumar, A., and Aalst, W.M.P. van der. "A Formal Modeling Approach for Supply Chain Event Management," *Proceedings of 14th Workshop on Information Technologies and Systems (WITS 2004)*, Dutta, A. and Goes, P (eds.), Washington D.C., December 2004, pp. 110-115.
- Luckham, D. *The Power of Events*, Addison-Wesley, Boston, 2002.
- Marabotti, D. "Information Technology Insights: Supply Chain Event Management Emerges in Enterprise Software," *Chemical Market Reporter* (262:9), September 2002, pp. 21-22.
- McCarthy, D. R., and Dayal, U. "The Architecture of an Active Database System," *Proceedings of ACM SIGMOD Conference on Management of Data*, J. Clifford, B. G. Lindsay, and D. Maier (eds.). ACM Press, New York, 1989, pp. 215-224.
- McCrea, B. "EMS Completes the Visibility Picture," *Logistics Management* (44:6), June 2005, pp. 57-61.
- Meseguer, P. "A New Method to Checking Rule Bases for Inconsistency: A Petri Net Approach," In *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI-90)*, Stockholm, August 1990.
- Montgomery, N. and Waheed, R. "Supply Chain Event Management Enables Companies to Take Control of Extended Supply Chains," Report on European E-Business, AMR Research, September 2001.
- Murata, T. "Petri Nets: Properties, Analysis and Application," In *Proceedings of the Institute of Electrical and Electronics Engineers* (77:4), April 1989, pp. 541-580.
- Ratzer, V. A., Wells, L., Lassen, M. H, Laursen, M., Qvortrup, F. J., Stissing, S. M., Westergaard, M., Christensen, and S., Jensen, K. "CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets," *Applications and Theory of Petri Nets 2003*, W. van der Aalst and E. Best (eds.), Lecture Notes in Computer Science (2679), Springer-Verlag, GmbH, 2003, pp. 450 - 462.
- Strozniak, P. "Exception Management," *Frontline Solutions* (3:8), August 2002, pp. 16-24.
- Wang, J. *Timed Petri Nets Theory and Application*, Kluwer Academic Publishers, Boston, 1998, pp. 63-123.
- Wu, P., Bhatnagar, R., L. Epshtein, Shi, Z. Alarm correlation engine (ace). In *Proceedings of the 1998 IEEE Network Operations and Management Symposium (NOMS'98)*, New Orleans, Louisiana, USA, 1998, pp. 733-742.
- Zhang, D and Nguyen, D. "PREPARE: A Tool for Knowledge Base Verification," *IEEE Transactions on Knowledge and Data Engineering* (6:6), 1994, pp. 983-989.

