Compound Web Service for Supply Processes Monitoring to Anticipate Disruptive Event

Erica Fernández[1], Enrique Salomone[1], Omar Chiotti[1,2]

[1] INGAR – CONICET, Avellaneda 3657, Santa Fe, Argentina, 3000
[2] CIDISI- UTN FRSF, Lavaisse 610, Santa Fe, Argentina, 3000
{ericafernandez, salomone, chiotti}@santafe-conicet.gov.ar

Abstract. The execution of supply process orders in a supply chain is conditioned by different types of disruptive events that must be detected and solved in real time. In this work we present a compound web service that performs the monitoring and notification functions of a supply chain event management system. This web service is designed based on a reference model that we have proposed to improve the event management activity through a deeper analysis of the occurrence and causality of events, leading to anticipate an exception during the execution of a supply process order. The web service composition is defined based on business processes. The ability to proactively detect, analyze and notify disruptive events is given through of a Bayesian network with decision nodes.

Keywords: **s**upply chain, event management, compound web service, Bayesian Networks.

1 Introduction

The execution of supply process orders usually deviates from original plans due to unexpected events. Interdependent processes are affected negatively by these events, and ripple effects in inter-organizational networks are common. These unexpected events (cancellation of an order, failure in a process, change in a process capacity, etc.) must be detected and solved in real time because they can propagate across many levels in the supply chain. Introducing a control mechanism for managing events and responding to them dynamically requires ability to proactively detect, analyze and notify disruptive events. In this scenario, Supply Chain Event Management Systems (SCEM) [1], [2] have been proposed. SCEM is defined as a business process in which significant events are timely recognized, reactive actions are suddenly executed, flows of information and material are adjusted, and the key employees are notified. In other words, SCEM can be seen as a complex control problem. SCEM systems emphasize the need of managing the exceptions by means of short term logistics decisions, avoiding frequent re-planning processes.

Montgomery [3] defines 5 functions that a SCEM system should perform. These are: Monitoring (to provide data in real time about supply chain processes, events and current states of the orders, shipments, production, and supply); Notification (to help to support real-time exception management through alert messaging, which proactively warns a decision-maker if an action must be taken); Simulation (to evaluate the effect of actions to be taken); Control (if an exception takes source, to evaluate the changes in the processes proactively); Measurement (to evaluate the supply chain performance).

In this paper, we present a compound web service [4] [5] that performs the monitoring and notification functions of a SCEM system. This web service is designed based on a reference model that we have proposed to improve the event management activity through a deeper analysis of the occurrence and causality of events, leading to anticipate an exception during the execution of a supply process order. The web service composition is defined based on two business processes. The ability to proactively detect, analyze and notify disruptive events is given through of a Bayesian Network with decision nodes.

This paper is organized in the following way. Section 2 discusses related works. Section 3 presents a reference model for monitoring, analysis and notification of events. Section 4 presents the compound web service design. Section 5 presents an example and section 6 presents conclusions.

2 Related Works

In order to provide a SCEM solution, two contributions closely related with this work have been proposed: Speyerer [6] developed a prototype based on web services to detect the root cause of a disruption. After that the disruption has been detected and classified it is compared in the data base to identify the root cause. This prototype also ranks exceptions in order to provide better accessibility and evaluation for operations managers; Maheshwari [7] designed a methodology to analyze business processes with the handling of exceptions suggesting web services to monitor and notify. Other contributions, in the SCEM scenario, have been proposed however, these works have not used web service. Following, some of them are mentioned: In the work of Zimmerman [2] the orders to be monitored are initialized by different triggers: queries from customers, alerts from suppliers and critical profiles. The critical profiles are the orders with high probability of being affected by disruptive events. This SCEM solution is based on a multi-agents architecture to proactively monitoring orders, integrating and interpreting several data gathered from the supply chain members to evaluate and distribute the results. Kurbel [8] proposed a mobile agent-based SCEM system to collect and analyze data provided by a supply chain monitoring system. To detect exceptions, his approach is not only based on target-state comparison, but it includes statistical analysis as well. Tai-Lang [9] proposed a method to analyze and manage the impact of an exception during order execution; Chin-Hung [10] developed a model based on cause-effect relationship to represent the disruptive exception caused by unexpected events in a supply chain. Liu [11] presented a methodology that uses Petri nets to formulate supply chain event rules and analyze the cause-effect relationships among events.

A common feature of these contributions is that the proposed SCEM models do not have the ability to anticipate a disruptive event during the execution of a supply process order. These models detect event in a reactive way and are not predictive.

In the field of event management, Complex Event Processing (CEP) has evolved into the paradigm of choice for the development of monitoring and reactive applications. CEP already plays an important role in many application areas like logistics. CEP addresses two crucial prerequisites to built highly scalable and dynamic systems. CEP-systems support the detection of relationships among events that can be specified by defining models of causal relations [12]. Different types of model can be defined, for instance, they can be defined using statistics and probabilities. Probabilistic models of causality should be supersets of computational causality in the sense that if any two events are causally related by the computation then they must be related by any probabilistic model of cause, but the probabilistic model may also relate other events as well [13]. In the context of this paradigm, this paper presents a reference model for the supply chain domain that contributes to systematizing the generation of models of causal relations among events.

The main difference between Business Activity Monitoring (BAM) and operational intelligence appears to be in the implementation details, real-time situation detection is a feature that only appears in operational intelligence and is often implemented using CEP. Furthermore, BAM focuses on formally modeled processes whereas operational intelligence instead relies on correlation to infer a relationship between different events.

3 Reference Model

We have defined a reference model for monitoring, analysis and notification of events in the supply chain domain. It can be used for monitoring the execution of any supply process order using a monitoring structure based on causal relations, when the total supply process time is of long duration. That is to say, when during the progress of realization of a supply process order is possible to define intermediate check points, so that it can be proactively detected the occurrence of disruptive events.

The reference model is represented as an UML class diagram in Figure 1, it shows that the *Schedule* (provided by the planning system) defines the execution timetable of a set of *Orders* and determines the resources that are linked to a supply process. Each *Resource* has a set of attributes. Associated with each *SupplyProcess*, there are milestones. Each milestone defines a measurement point which is used to assess the progress during its fulfillment. A *Milestone* is a *TimeMilestone* or *StateMilestone* and it has a *PlannedValue* (planned date of achievement of milestone, planned/ordered quantities). The last milestone (in the class diagram is the *Milestone* with *final* role) associated with an order is used by the control structure to anticipate disruptive events. To this purpose, the *ControlStructure* has an assigned *Monitor*. During the plan execution, the *Monitor* determines the *Milestone* to assess where one or more values of the resource *Attributes* or *EnvironmentVariables* can be observed. The *ControlStructureAnalizer* detects the occurrence of an event comparing the *ObservedValue* with the *PlanedValue* of resource *Attributes*. The impact of this event on the following *Milestones* is analyzed by the *ControlStructureAnalizer* using a cause-effect relation network defined by *Nodes* and *CausalRelationArcs*. An *EstimatedValue* of each *Milestone* is obtained. The *Comparator* takes the *EstimatedValue* associated with the last *Milestone* and compares it with its *PlannedValue*. The *Comparator,* based on decision criteria, estimates if a disruption will occur on the last *Milestone*. Upon the occurrence of a *DisruptiveEvent*, the *Notifier* reports it and the monitoring process finishes. Else, the *Monitor* determines the next *Milestone* to assess where one or more values of the resource *Attributes* or *EnvironmentVariables* can be observed. The monitoring process can also finish if the supply process order is completed without problems.

The *DisruptiveEvent* can be: changes in the requirements of an order (quantity of material, deadline), and/or changes in the parameters of a resource (transition time between states, available capacity).

An innovative characteristic of this model is that the network is dynamically constructed in each milestone. That is to say, the monitor, depending on the *EstimatedValues* generated by the *ControlStructureAnalizer*, can extend its monitoring strategy to other milestone including other observed values or eliminating those unnecessary variables.

In this paper, the *ControlStructureAnalizer* has been implemented by means of a Bayesian network with decision nodes. Bayesian networks [14] are a method to represent uncertain knowledge, which allows reasoning based on probability theory.

Mapping the reference model to a Bayesian network requires defining the following nodes in the net: *Chance nodes,* which describe the set of attributes associated with a resource and a set of environment variables. This type of node can be observed to detect the occurrence of an event. Each node is represented by X: (xi, P(xi)), where xi is a particular value that X (resource attribute or environmental variable) can take, and P(xi) is a conditional probability distribution for each combination of the variables of the parent nodes or a marginal probability distribution. A chance node can be: *Estimated* (its value is calculated from the inference process) or *Observed* (the evidence is captured and propagated through the network, affecting the overall joint distribution). Initially, all chance nodes of a network have likelihoods, which are calculated with a priori information. *Decision nodes* define the chance node that has to be observed to incorporate evidence; *Value nodes* represent utility functions.

The chance nodes and decision nodes are connected in the *ControlStructure* by directed arcs, encoding dependence relations and information precedence.

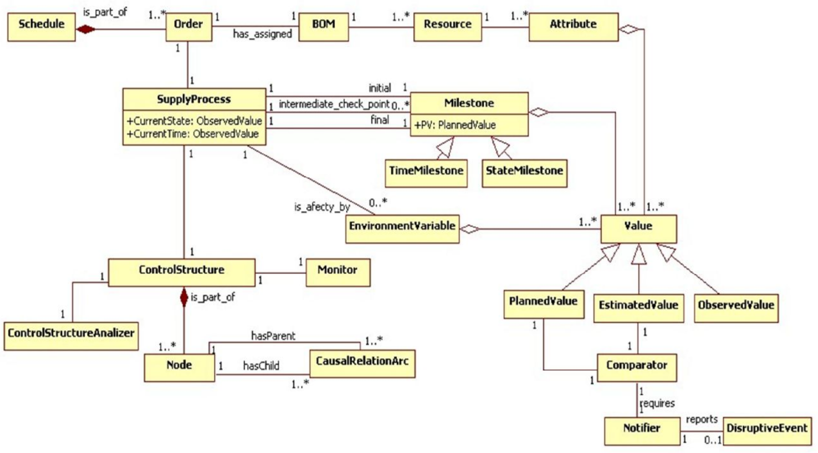

Fig. 1. Reference Model for monitoring, analysis and notification of events.

4 Compound Web Service Design

We have designed a compound web service that has the ability of proactively detect, analyze and notify disruptive events that take source in a supply chain. The composition is defined based on two business processes: Monitoring Service Contract and Order Monitoring Service. A conceptual model of each business process has been developed using the language BPMN (Business Process Modeling Notation) [15]. Theses business processes are described following:

Monitoring Service Contract (Figure 2) defines a contract to use the service. This business process defines the monitoring structure associated to a supply process and the sources from where observed values will be obtained. The process starts with a message reception event where the request service for potential orders monitoring is received. The supply processes are gotten from the model base and are offered in two modes: with standard a prior probability or requestor-defined. The requestor can cancel the request service or can select a supply process. In this case, its monitoring structure is gotten from the model base. Based on the reference model above described, it is determined when and what observed values will be required. The requestor is informed about of these requests and it sends the source from where observed values will can be gotten.

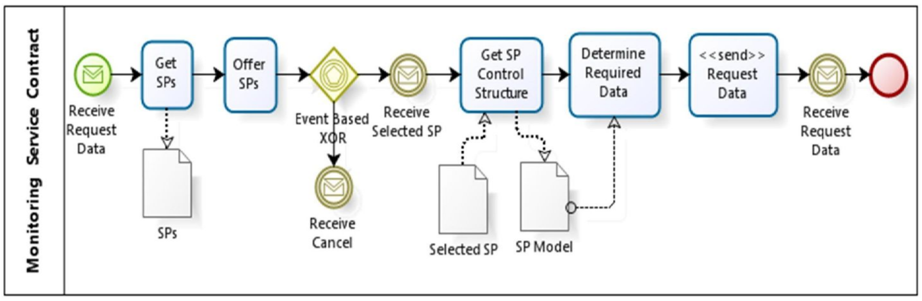

Fig. 2. Business Process Monitoring Service Contract.

Order Monitoring Service (Figure 3) monitors orders based on the contract defined. The process starts with a message reception event where the order to be monitored is received. The monitoring structure is instantiated based on the reference model and completed with order parameters (planned values). During the order execution, request of observed values are sent. After receiving the observed values, the inference model is executed. If a disruptive event is detected, the result is notified. Otherwise, the monitoring process continues. Detail about the network dynamics were described in section 2.

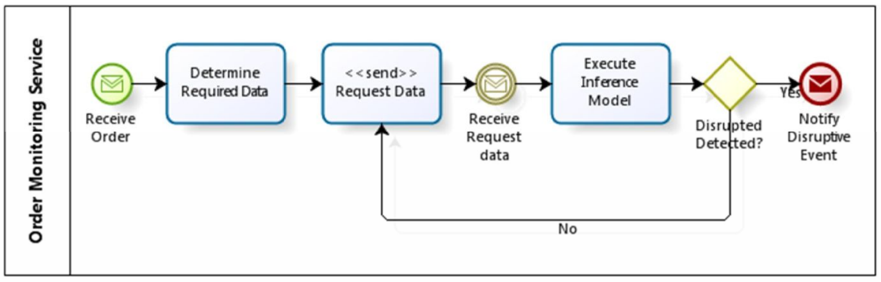

Fig. 3. Business Process Order Monitoring Service.

5 An Application Example

As has been said above, the reference model covers monitoring and notification functions. Based on this model, we have designed a compound web service to proactively detect, analyze and notify disruptive events that take source in a supply chain.

An example of a cheese production plant is described to show the behavior of the compound web service and of the reference model. Through the *Monitoring Service Contract* is defined a contract to monitor the cheese production process. In this process, milk acidity is a parameter that can affect the cheese quality. High acidity can produce sandy cheese, bitter cheese or increase the curdling rate causing surface cracks. Low acidity can produce insipid cheese. Normal acidity produces cheese with required quality. Based on the reference model (Figure 1), is defined the monitoring structure by means of a Bayesian Network with a priori probabilities (Figure 4). These probabilities were provided by the requestor.

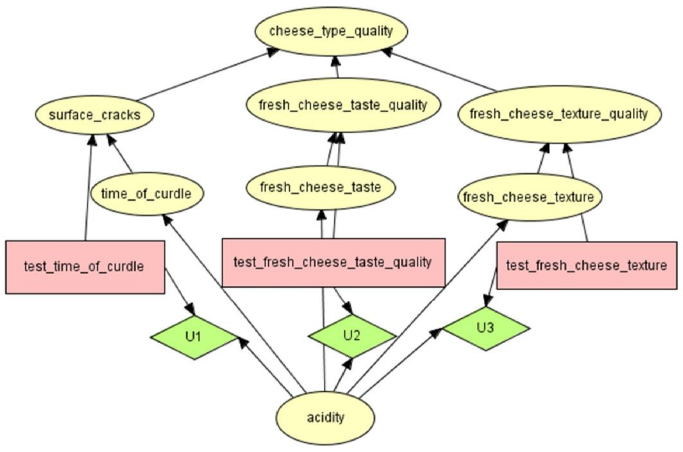

Fig. 4. Monitoring structure based on Bayesian Network

This monitoring structure is composed by the following *nodes*, whose states are represented in braces: *acidity* {normal, low, high}, which is a attribute of resource:*milk; time_of_curdle* {normal, low, high}, *fresh_cheese_texture_quality* {good, bad}, *fresh_cheese _texture* {no_granulated, granulated}, *surface_cracks* {no, yes}, *fresh_cheese_taste* {good, insipid, bitter} and *fresh_cheese_taste_quality* {good, bad}, which are resource attributes of the resource:*cheese_production_plant*; and *cheese_type_quality* {good, bad}, which is a attribute of resource:*cheese_type.* The *decision nodes* are: *test_ time_of_curdle*, *test_ fresh_cheese _texture* and *test_ fresh_cheese _taste.*

The requestor is informed when and what observed values will can be required. In this example the observed values could be the following: *acidity, time_of_curdle, fresh_cheese _texture* and *fresh_cheese _taste*. The requestor sends the source from where observed values will can be gotten.

The Order Monitoring Service starts with a message reception event where an order requires producing a quantity of a cheese type. The monitoring structure is instantiated and adapted to the order parameters (planned values).

The web service notifies changes in the requirement of the order if the probability of the product will be outside the specification is greater than a threshold value (decision criteria). In this example *threshold = 90.0.* The total process time depends on the type of cheese to be produced. In this example *cheese_type* = soft cheese is considered, and the total process time is 240 hours.

Thus, the initial milestone is a *StateMilestone* (start of the supply process) and the monitoring structure (Figure 4) initially includes the node *acidity*. The *acidity* is monitored at the beginning of the process, and for each of its three possible values, different plans of action can be carried out. Following, the action plan when the acidity is high is described.

1. If *acidity == high*, the monitor gets the evidence (from the source specified by requestor), inserts it in the net assigning *acidity:(high, 100)* and executes the inference model. As a result (Fig. 5), the estimated value of the node, *cheese_type_quality:(bad, 85.1)*. Since, *P(bad) = 85.1 < threshold*, the estimated values analyzed are *surface_cracks:(yes,50.75), fresh_cheese_texture_quality:(bad,45)* and *fresh_cheese_ taste_quality:(bad, 45).* The three values explain the value of the attribute *cheese_type_quality*:

1.1 *surface_cracks:(yes, 50.75)* indicates the probability that the cheese has surface cracks caused by a low time of curdle. Based on this information, the monitor defines the next milestone. *StateMilestone* (finish of the curdle process) and the next process variable to monitor is *surface_cracks*. This is done by assigning to the *decision node test_ time_of_curdle == YES*. This test is made 2 hours after the process has been started. The test has two possible results {*normal*, *low*}.

1.1.1 If *time_of_curdle == low,* the monitor gets the evidence (from the source specified by requestor) inserts it in the net (Fig. 6) assigning *time_of_curdle:(low, 100)* and executes the inference model. The result is *cheese_type_quality:(bad, 91.68)*. Since *P(bad) = 91.68 > threshold*, which implies high probability that product will be outside the specification, The value of the attribute, *cheese_type_quality*, is notified and the monitoring process finishes. This allows predicting the result 238 hours before the process ends.

1.2 If time*_of_curdle == normal,* the monitor gets the evidence (from the source specified by requestor) inserts it in the net assigning *time_of_curdle:(normal, 100)* and executes the inference model. The result is *cheese_type_quality:(bad, 69.7).* Since *P(bad) = 69.7 < threshold*, the variables *surface_cracks:(yes,0.0), fresh_cheese_texture_quality: (bad, 45.0)* and *fresh_cheese_taste_quality: (bad, 45.0)* are analyzed. The first variable indicates that there is no risk that the product will have surface cracks and its branch is pruned of the monitoring structure (Figure 7). The two last variables explain the value of the attribute *cheese_type_quality* and these are the next to monitor. The remainder of the monitoring process is similar to explained.

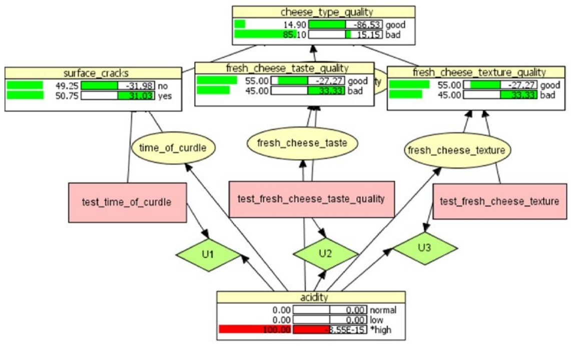

Fig. 5. Monitoring structure based on Bayesian Network with *acidity == high* evidence.

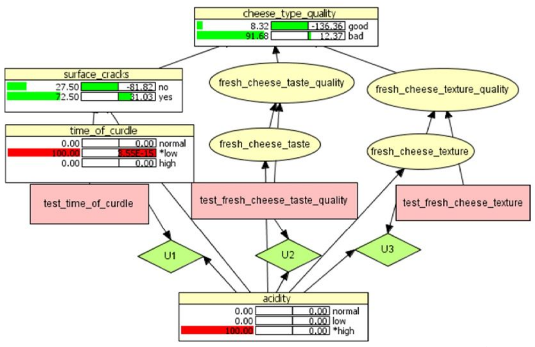

Fig. 6. Monitoring structure based on Bayesian Network with *time_of_curdle==low* evidence.

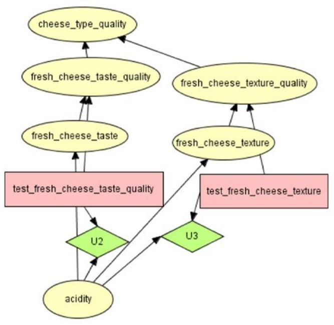
cheese_type_quality
fresh_cheese_taste_quality
fresh_cheese_texture_quality
fresh_cheese_taste
fresh_cheese_texture
test_fresh_cheese_taste_quality
test_fresh_cheese_texture
U2
U3
acidity

Fig. 7. Monitoring structure based on Bayesian network with the surface cracks branch pruned.

6 Conclusions

In this work we have proposed a compound web service for proactively monitor, analyze and notify disruptive events that take source in a supply chain. Monitoring and notification functions are managed through two business processes. The advantage of using web service technology is that it supports the execution of collaborative and distributed processes in the domain SCEM.

The compound web service is developed based on a reference model for monitoring, analysis and notification of events in a supply chain. Mechanisms based only on ad hoc event management are not acceptable. The generated reference model has two advantages: 1) ability to dynamically change the *network of analysis*. That is to say, after an evidence is obtained, the monitoring strategy can be extended including other parameters and increasing its monitoring frequency; 2) the model can be used to monitor the execution of any supply process order using a monitoring structure based on causal relations, when the total process time is of long duration.

The ability of anticipating disruptive event was implemented defining a Bayesian network with decision nodes representing temporal causality. As has been showed through the example, the model proposed has the ability of anticipating, based on evidences, changes in the values of the attributes of the resource or in the requirements of an order.

Different limitations are mentioned to be taken into account for future work: the approach is limited to a conceptual perspective on SCEM, excluding specific implementation aspects. The application is focused on supply process of long duration (when during the progress of its realization is possible to define intermediate check points) and disruptive events only can be detected during the execution of a supply process order.

References

1. Masing, N. SC Event Management as Strategic Perspectiva – Market Study: SCEM Software Performance in the European Market. Master Thesis. Universitié du Québec en Outaouasis (2003)
2. Zimmermann, R. Agent-based Supply Network Event Management. Whitestein Series in Software Agent Techonologies. (eds.) Marius Walliser, Stefan Brantschen, Monique Calisti y Thomas Hempfling (2006)
3. Montgomery, N.; Waheed, R.: Event Management Enables Companies To Take Control of Extended Supply Chains. AMR Research (2001)
4. Alonso, G.; Casati, F.; Kuno, H.; Machiraju, V. Web Services - Concepts, Architecture and Applications.Springer, Berlin (2004)
5. Barros, A., Dumas, M., Oaks, P.: A Critical Overview of the Web Service Choreography Description Language (WS-CDL). BPTrends Newsletter 3 (2005)
6. Speyerer, Jochen K.; Zeller, Andrew J. Managing Supply etworks: Symptom Recognition and Diagnostic Analysis with Web Services, in Proceedings of the 37th Hawaii International Conference on System Sciences (2004)
7. Maheshwari, P, Sharon, S: Events-Based Exception Handling in Supply Chain Management using Web Services, in Proceedings of the Advanced International Conference on Internet and Web Applications and Services (2006)
8. Kurbel K., Schreber D.: Agent-Based Diagnostics in Supply Networks, in Issues in Information Systems, vol. VIII, No. 2 (2007)
9. Tai-Lang, Y.: An Exception Handling Method of Order Fulfillment Process in the i-Hub Supported Extended Supply Chain. Master's Thesis, National Central University, Institute of Industrial Management, Taiwan (2002)
10. Chin-Hung, C.: Assessing Dependability of Order Fulfillment in the i-Hub Supported Extended Supply Chain. Master's Thesis, National Central University, Institute of Industrial Management, Taiwan (2002)
11. Liu, E.; Akhil, K.; Van Der Aalst, W: A formal modeling approach for Supply Chain Event Management. Decision Support Systems, vol. 43, pp. 761-778 (2007)
12. Buchmann A.; Darmstadt TU.; Koldehofe B.: Complex Event Processing, in Information Technology (2009)
13. Luckham, David C. and Frasca, Brian. Complex Event Processing in Distributed Systems. Stanford University Technical Report CSL-TR-98-754 (1998).
14. Jensen, F.: An Introduction to Bayesian Networks, Springer Verlag, New York (1996)
15. OMG/BPMN 1.2: OMG Specification (2009). http://www.omg.org/spec/BPMN/1.2/PDF/.