Computers in Industry 80 (2016) 30–42

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Framework for modelling and simulating the supply process monitoring to detect and predict disruptive events

Erica Fernández[a,b,*], Verónica Bogado[c], Enrique Salomone[d], Omar Chiotti[d,e]

[a] Departamento Ingeniería Industrial, FRRA, UTN, M. Acuña 49, Rafaela 5900, Argentina
[b] GEMPRO – UTN FRSF, Lavaisse 610, Santa Fe 3000, Argentina
[c] Departamento Ingeniería en Sistemas de Información, FRVM, UTN, Av. Universidad 450, Villa María 5900, Argentina
[d] INGAR – CONICET, Avellaneda 3657, Santa Fe 3000, Argentina
[e] CIDISI – UTN FRSF, Lavaisse 610, Santa Fe 3000, Argentina

ARTICLE INFO

Article history:
Received 15 December 2015
Accepted 11 April 2016
Available online 26 April 2016

Keywords:
SCEM system
Monitoring system
Disruptive event
Simulation

ABSTRACT

Disruptive events that take place during supply process execution produce negative effects that propagate throughout a supply chain. Event management systems for supply chains have emerged to provide functionality for monitoring schedules, managing disruption, and repairing schedules affected by a disruptive event. A Web service that provides a schedule monitoring functionality for supply chain event management was developed. This paper provides a framework to allow enterprises that hire this service to develop simulation models of monitoring processes and evaluate their ability to detect and anticipate disruptive events. The framework, based on discrete event simulation, is implemented in a library that can be used for developing and testing monitoring processes by means of a friendly interface. A marine freight transport process was used as a case study to show how a supply process and its environment can be modelled and simulated by using the library. Simulation results show the ability of this approach to anticipate disruptive events and identify critical stages of a supply process in order to prevent disruptive events.

1. Introduction

In an integrated supply chain, the overall performance largely depends on keeping the coordination of schedules for producing and distributing goods. These schedules are generated by the planning subsystem of the Enterprise Resource Planning (ERP) system of each enterprise and are typically represented by production and distribution orders, where each order represents a particular instance of a generic supply process.

During the execution of scheduled orders, significant changes may occur either in the specification of orders or in the availability of involved resources. These unplanned changes, called disruptive events, can produce negative effects that propagate throughout the supply chain, affecting schedules and their coordination [1,2].

The paradigm of robust planning proposes to define buffers (material, capacity, and time) to absorb changes that may occur during the execution of scheduled orders [3]. These buffers allow achieving a schedule most likely to remain stable during its execution. This avoids re-planning tasks, which can be costly and time-consuming, since all enterprises involved in the supply chain should agree on a new collaborative plan. However, buffers cannot usually absorb all changes due to the impossibility of forecasting with certainty the time and place in which disruptive events could occur and their magnitude.

Under this scenario, Supply Chain Event Management (SCEM) Systems have emerged [4,5]. SCEM systems should provide functionality for: *monitoring schedules* during its execution to detect disruptive events (*reactive monitoring*) or to prevent disruptions before they occur (*predictive monitoring*); *managing disruption* after a disruptive event to check if schedules are still feasible; and *repairing schedules* affected by a disruptive event considering the distributed nature of a supply chain.

Fernández et al. [6], present an Agent-based Monitoring Service for Management of Disruptive Events in Supply Chains (MSMDE), which is a Web service that provides the *schedule monitoring* functionality of a SCEM system named collaborative management of disruptive events in supply chains presented in [7]. Enterprises that hire monitoring Web service MSMDE must provide a monitoring model of the supply process to be monitored. For

* Corresponding author at: Departamento Ingeniería Industrial, FRRA, UTN, M. Acuña 49, Rafaela 5900, Argentina.
E-mail addresses: ericafernandez@santafe-conicet.gov.ar (E. Fernández), vbogado@frvm.utn.edu.ar (V. Bogado), salomone@santafe-conicet.gov.ar (E. Salomone), chiotti@santafe-conicet.gov.ar (O. Chiotti).

http://dx.doi.org/10.1016/j.compind.2016.04.002
0166-3615/© 2016 Elsevier B.V. All rights reserved.

that, enterprises must develop the monitoring model by using the abstract modelling language provided by the reference model for supply process monitoring presented in [8]. This reference model is a meta-model that specifies the abstract syntax of a modelling language to represent the static part of the monitoring model of a supply process [8]. The monitoring model represented in terms of the reference model is automatically transformed by the Web service MSMDE into a monitoring process, which is used by the service for monitoring the supply process.

So, enterprises that hire this Web service must understand the abstract modelling language provided by the reference model, which is not easy to understand and use. As Moody states [9] using an abstract language for model building and testing may be a difficult task to perform and prone to mistake if a suitable tool is not available.

The objective of this paper is to provide a framework to allow enterprises that hire Web service MSMDE to develop simulation models of monitoring processes and evaluate their ability to detect and anticipate disruptive events without the need of knowing the abstract modelling language provided by the reference model. The framework, based on discrete event simulation, is implemented in a library that contains a set of simulation elements related to concepts of the reference model. The library can be used for developing monitoring processes through a friendly interface, hiding the abstract modelling language provided by the reference model. Different monitoring models tailored to supply processes can be developed and tested by setting input parameters.

The remainder of this paper is structured as follows: Section 2 discusses related works. Section 3 briefly defines the concepts of the reference model. Section 4 presents the framework and its implementation in a library. Section 5 describes a case study and Section 6 presents conclusions and future work.

2. Related works

2.1. Approaches for supply process monitoring

Approaches for reactive monitoring of schedules are based on capturing information about material resources and/or order specifications during execution for assessing performance indicators or rule-based arithmetic ratios in order to detect disruptive events. Indicators are also used to assess the impact of disruption along the supply chain. Among the proposals for reactive monitoring, those presented by Bansal et al. [10], Liu et al. [11], and Winkelmann et al. [12] can be mentioned.

The approach presented by Bansal et al. [10] uses performance indicators assessed at regular intervals, cause-effect relationship models to identify the root cause of a disruptive event and rectification strategies to repair the schedule. Liu et al. [11] present a methodology that uses Petri nets to formulate supply chain event rules and analyse cause-effect relationships among events. Events are classified in the following types: task status-related events, events produced by a task, and external events. Based on interactions between partners in the supply chain, events are identified and rules relating them are defined to represent a supply chain in order to propagate events, analyse them, and suggest a solution when a disruption is detected. Winkelmann et al. [12] present an approach for conceptual modelling of SCEM systems. The main tasks are supply chain process definition and identification of relevant logistical objects where disruptive events could occur. These objects are used as reference points for monitoring activities. For each object, arithmetic ratios are defined and combined with a rule-based expression to detect disruptive events. Once detected, disruptions are notified to applications or people for corrective actions.

The main technologies used for tracking and tracing orders and/or resources during the schedule execution are RFID and GPS [13]. RFID is generally used for monitoring products, pallets, and machines, and GPS for vehicle location. The massive data stream coming from a supply chain when each object provides its current status requires the use of data flow processing technologies. The main technologies commonly used for this purpose are: complex processing [14], agents [15], and event-condition-action [16]. A complex processing technology is a pattern-based event processing. Relevant events are selected by data stream filtering and matched to predefined patterns to derive complex events that allow detecting disruptive events. Agents' technology is a software application including complex algorithms able to collect up-to-date data about orders and/or resources and filter them for capturing relevant information, and with knowledge and reasoning capability for identifying disruptive events. The event-condition-action technology extends traditional databases with a layer of rules and event detection mechanisms for event processing. Among the proposals based on data flow processing technologies, those presented by Meyer et al. [15], and Ko et al. [17] should be mentioned. Meyer et al. [15] present an agent-based architecture for collecting up-to-date information of products and comparing their current status with that planned in orders so as detect disruptive events. Agents determine the planned status of products by analysing the schedule information (such as order due dates and planned transactions and operations that will affect the product). If a disruption is captured, agents propose solutions or suggest how to reduce the severity of the problem. Ko et al. [17], present an agent-based system for monitoring product locations. A monitoring agent checks product arrivals at nodes specified in the monitoring plan. If product arrivals are detected within the planned time period, the monitoring agent visits the next node. Otherwise, it suspends monitoring and searches for products deviated from their planned path.

Approaches for predictive monitoring of schedules are based on capturing information about resources, order specifications, and environment variables during execution to prevent disruptions before they occur. Some proposals for predictive monitoring should be mentioned: those presented by Kim et al. [16], Fernández et al. [8], and Vlachakis and Apostolou [18]. Kim et al. [16] propose a rule-based language to develop monitoring models to predict and prevent business process disruptions before they occur or detect and repair them once occurred. The rules defined to capture information of business processes have an event-condition-action structure extended with other components such as contexts in which rules are applicable, preferences that specify rule priorities to be triggered by events, and frequency that specifies periods in which rules are checked. Fernández et al. [8], present an abstract language to develop monitoring models of supply processes. It is based on cause-effect relationships among variables that represent features about order specifications, resources, and the supply process environment. These variables are monitored in different milestones related to supply process stages to anticipate or detect disruptive events. To this aim, predictive or reactive evaluation functions must be defined. These can be simple mathematical functions or more complex ones such as Bayesian Networks, Petri Nets, Complex Processing, or rule-based. Vlachakis and Apostolou [18] define a supply chain event management framework able to capture events related to resources, orders, or environment and process them to detect undesired deviations when the predefined threshold is exceeded. After a disruptive event is detected, the damaged schedule is repaired according to predefined rules or decision models.

Languages to develop models for supply process monitoring in the supply chain context as that proposed by Kim et al. [16] and Fernández et al. [8] have the advantage of an abstract syntax that allows representing all type of monitoring models for different domains with independence of an implementation technology. However, as it was highlighted in the Introduction Section, building and testing models by using an abstract language may be a difficult task to perform and prone to mistake if a suitable tool is not available. Then, these proposals should consider the development of suitable tools that allow using abstract modelling languages easily.

2.2. Simulation of monitoring process

Simulation is a powerful tool to study systems behaviour at specified points (milestones). Particularly, discrete event simulation is a suitable tool for simulating supply processes as a network of stages, operations, resources, and external entities with the aim of evaluating the ability of a monitoring process to capture and predict disruptive events. The main reasons for using simulation to study the supply process monitoring are: (i) availability to represent the supply process and its dynamics; (ii) availability to specify the monitoring process by using appropriate simulation elements that were developed to exempt users from having to know the abstract modelling language provided by the reference model; and (iii) availability to test the monitoring process ability to detect and anticipate disruptive events.

Discrete event simulation is a modelling approach widely used as a decision support tool in the supply chain context. Tako and Robinson [19] present an extensive review of approaches that use simulation to study different supply chain problems. Despite the large list of contributions, with the exception of the work by Heinecke et al. [20], the supply process monitoring problem does not appear as studied. Heinecke et al. [20] analyse the material flow in a just in time manufacturing system by using a dynamic model that anticipates material delivery delay through a cause-effect relationship among operational performance indicators. Discrete event simulation and dynamic system tools are frequently used to analyse the impact of disruptions in supply chains but not to test the monitoring process ability to detect and anticipate disruptions.

3. Reference model for supply process monitoring

The reference model for supply process monitoring (Fig. 1) is defined by the tuple *RfMo* = (*Et*; *Rt*) where: *Et* is a set of concepts; and *Rt* is a set of allowed relationships between these concepts. The static view of a Monitoring Model (*MoMo*) of a supply process is an instance of the reference model *RfMo* [6].

A *Schedule* is a sorted set of orders and resources that specifies the time period during which each *Resource* is required by each *Order*, and its required capacity and states. It is defined by the tuple *Sch* = (*R*, *O*, *S*, *E*, *C*) where: *R* is a set of resources *r*; *O* is a set of orders *o*; *S* is a set of order specifications $s_o = <o, quantity, startTime, endTime>$ that states the quantity to produce/supply and times in which the order starts and ends; *E* is a set of states $e_{o,r,t} = <o, r, e_r, t>$ that specify the required state $e_r \in E_r$ of resource $r \in R$ for order $o \in O$ at time *t*; and *C* is a set of capacities $c_{o,r,t} = <o, r, c_r, t>$ that specify the required capacity $c_r \in C_r$ of resource $r \in R$ for order $o \in O$ at time t. E_r is the set of state of resources and C_r is the set of capacities of resources.

For being fulfilled, each order $o \in O$ requires the execution of a production or distribution process. These processes are generically referred to as supply processes.

A supply process (*SupplyProcess*) is a sorted set of stages *SP*, each stage $ps \in SP$ involving operations such as production, load, unload, or transport.

A supply process could be associated with a set of features *F*, which are aspects about order specifications, resources required by

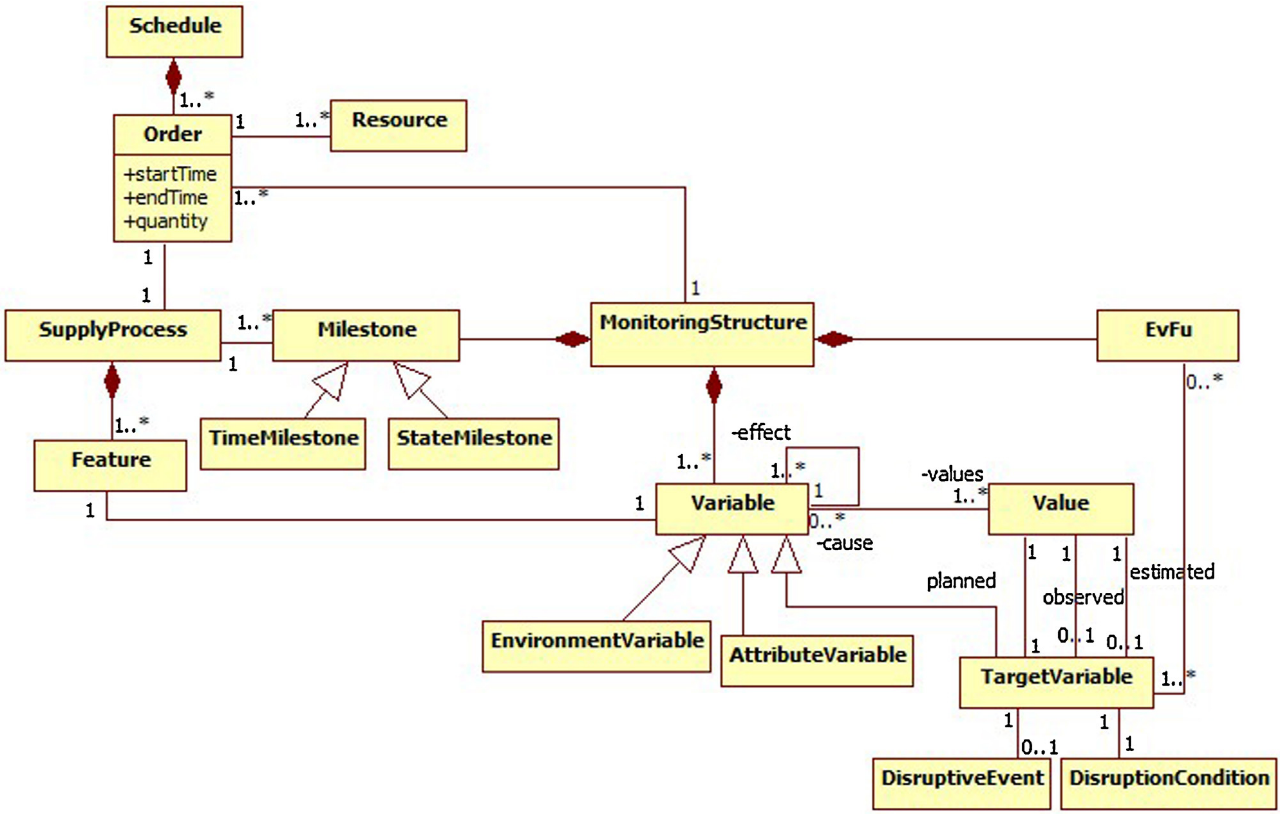

Fig. 1. Reference model for monitoring supply process.

the supply process (port operation time, ship speed), and the supply process environment (weather conditions, port congestion) [21]. Each *Feature* $f \in F$ is associated with a *Variable* $v \in V$ that must be observed for evaluating the progress of an order execution with the aim of detecting or anticipating disruptive events. $V = S \cup E_r \cup C_r \cup Z$ is the set of all variables that can be observed by a supply process monitoring, where Z is a set of environment variable z.

A *Milestone* $m \in M$ is a control point that defines a state *(StateMilestone)* or time *(TimeMilestone)* in which a set of variables will be observed. It is defined by the tuple $m = (mt, stv, OV_m)$ where: mt is the milestone type (state or time); stv is the state or time value; and $OV_m \in V$ is the set of variables to be observed in this milestone. They can be order specifications $s_o \in S$, capacities $c_r \in C_r$ of resources $r \in R$, states $e_r \in E_r$ of resources $r \in R$, and environment variables $z \in Z$.

Each supply process stage $ps \in SP$ that has to be monitored during execution has a defined milestone. So, a supply process could be associated with a set of milestones M.

The monitoring structure *(MonitoringStruture)* is defined by the tuple $MS = (M, V, EvFv)$ where: M is a set of milestones; V is a set of variables, which can be attribute variables *(AttributeVariable)* such as order specifications $s_o \in S$, capacities $c_r \in C_r$ of resources $r \in R$, states $e_r \in E_r$ of resources $r \in R$, environment variables $z \in Z$ (EnvironmentVariable) that may affect a supply process, or target variables (TargetVariable); and EvFv is a set of evaluation functions, which can be reactive evaluation functions reacEvFu or predictive evaluation functions predEvFu.

A target variable *(TargetVariable)* is used to assess the possible occurrence of a disruptive event. It is defined by $tv = <pv, ov, ev, dc, de>$ where: pv is the planned value; ov is the observed value; ev is an estimated value calculated by using a predictive evaluation function $predEvFu \in EvFu$; dc is the disruption condition *(DisruptionCondition)* that defines a threshold value to assess a disruption; and de is a Boolean variable that specifies if a significant change in order specifications $s_o \in S$, capacities $c_r \in C_r$ of resources $r \in R$, or states $e_r \in E_r$ of resources $r \in R$ generates a disruptive event *(DisruptiveEvent)*.

When the execution of an order $o \in O$ starts, an initial milestone $m \in M$ is activated and current values of variables $v \in OV_m$ are monitored. These values are used by the target variable for analyzing the possible occurrence of a disruptive event. If a disruptive event occurs, the monitoring process ends. On the other hand, the next milestone $m_+ \in M$ is selected to continue the supply process monitoring.

If monitoring is reactive, the target variable uses a reactive evaluation function $reacEvFu \in EvFu$ to compare the current observed value ov of each observed variable $v \in OV_m$ of the current milestone $m \in M$ with its planned value pv for calculating its variation $\Delta v = (ov - pv)$, and then compare this variation with the threshold value dc to detect if a disruptive event de has occurred by using the comparison function: *if* $(\Delta v \le dc \ \forall \ v \in OV_m)$ *then* $\{de = False\}$ otherwise $\{de = True\}$.

If monitoring is predictive, the target variable uses a predictive evaluation function $predEvFu \in EvFu$ able to infer through cause-effect relationships an estimated value ev of order specifications $s_o \in OV_{m+}$ in upcoming milestones $m_+ \in M$ from changes in the planned values pv of any observed variable $v \in OV_m$ of the current milestone $m \in M$. This function is defined as: $ev = f(ov, \Delta c_r, \Delta e_r, \Delta c_r(z \ \forall \ z \in OV_m), \Delta e_r(z \ \forall \ z \in OV_m) \ \forall \ r \in OV_m) \ \forall s_o \in OV_{m+}$ where $\Delta c_r = (ov - c_{o,r,t})$, $\Delta e_r = (ov - e_{o,r,t})$. After this, the target variable compares the estimated value ev with its planned values pv for calculating the variation of order specifications $\Delta s_o = (ev - pv) \ \forall s_o \in OV_{m+}$ in upcoming milestones $m_+ \in M$, and then it compares these variations with a threshold value dc to predict if a disruptive event de could occur by using the comparison function: *if* $(\Delta s_o \le dc \ \forall s_o \in OV_{m+})$ *then* $\{de = False\}$ otherwise $\{de = True\}$.

4. Framework for modelling and simulating the supply process monitoring

The framework is composed of four main levels: *Schedule Execution*, *Monitoring*, *External*, and *Decision-Making* (Fig. 2).

The *Schedule Execution level* consists of two layers: *Supply Process*, to represent the supply process *SP*, and *Supply Process Features*, to represent the set of relevant features *F* involved in the supply process *SP*. These features define the set of variables *V* that drive the supply process monitoring.

The *Monitoring level* consists of the *Monitoring Process* layer, which represents the monitoring workflow *MWf* and evaluation functions *EvFs* of a monitoring process *MoPr* [6].

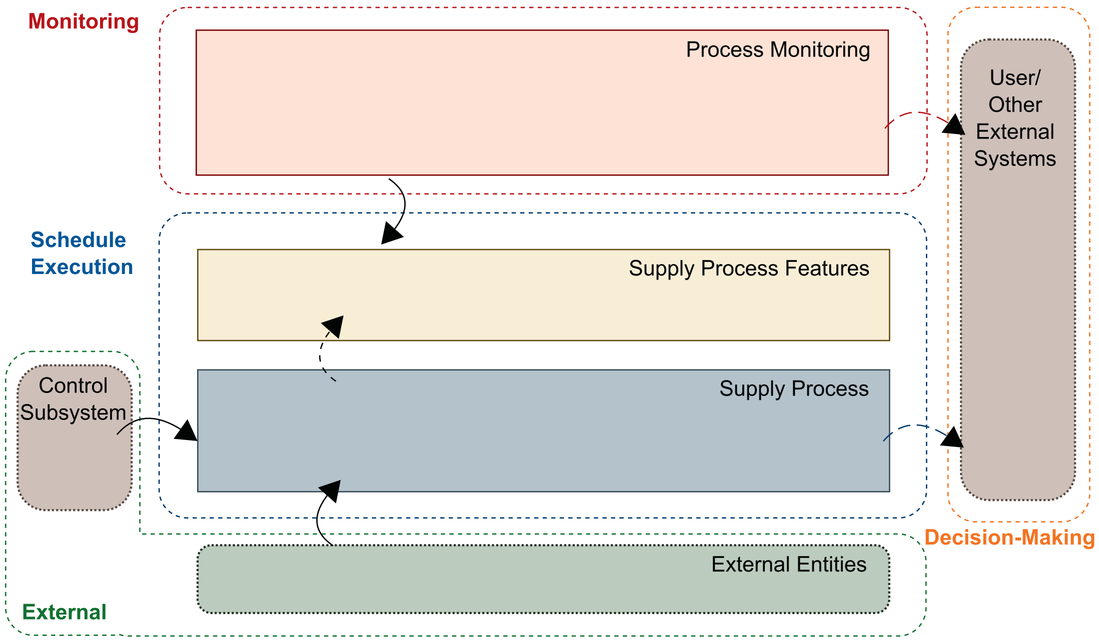

Fig. 2. Simulation schema of the supply process monitoring.

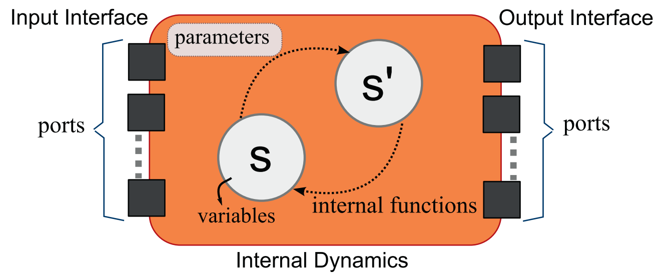

Fig. 3. Simulation element, its internal structure, and its dynamic.

The *External level* is composed of two layers: *Planning System*, to represent the planning subsystem of the ERP system that defines the schedule to be executed, *External Entities*, to represent the set of external entities that can affect the supply process and the order fulfilment.

The *Decision-Making level* consists of the *User/Other External System* layer, which represents users or external systems that receive output information of the simulation model for deciding if the monitoring process *MoPr* behaves as expected. This layer is transversal since it involves supply process information, which consists of executed orders, delay times and process steps, and monitoring information, which consists of disruptive events and other information useful to verify and validate the monitoring process *MoPr* under development.

4.1. Simulation elements for modelling the monitoring process

The framework includes a set of simulation elements related to concepts of the reference model *RfMo* (Fig. 1). They were developed following fundamentals of discrete event simulation [22] and the object-oriented paradigm that provides a modular, hierarchical, and incremental construction of large-scale models [23,24]. Each element has interfaces to communicate with other simulation elements and an internal dynamic that defines its behaviour (Fig. 3). Input and output interfaces are defined by *ports* and internal behaviour is specified by *internal functions* that define state changes and internal calculus, *parameters* to configure its behaviour, and internal *variables* that perform as buffers and define the element state.

Simulation elements and interactions between them are illustrated in Fig. 4. At *Schedule Execution level*, in the *Supply*

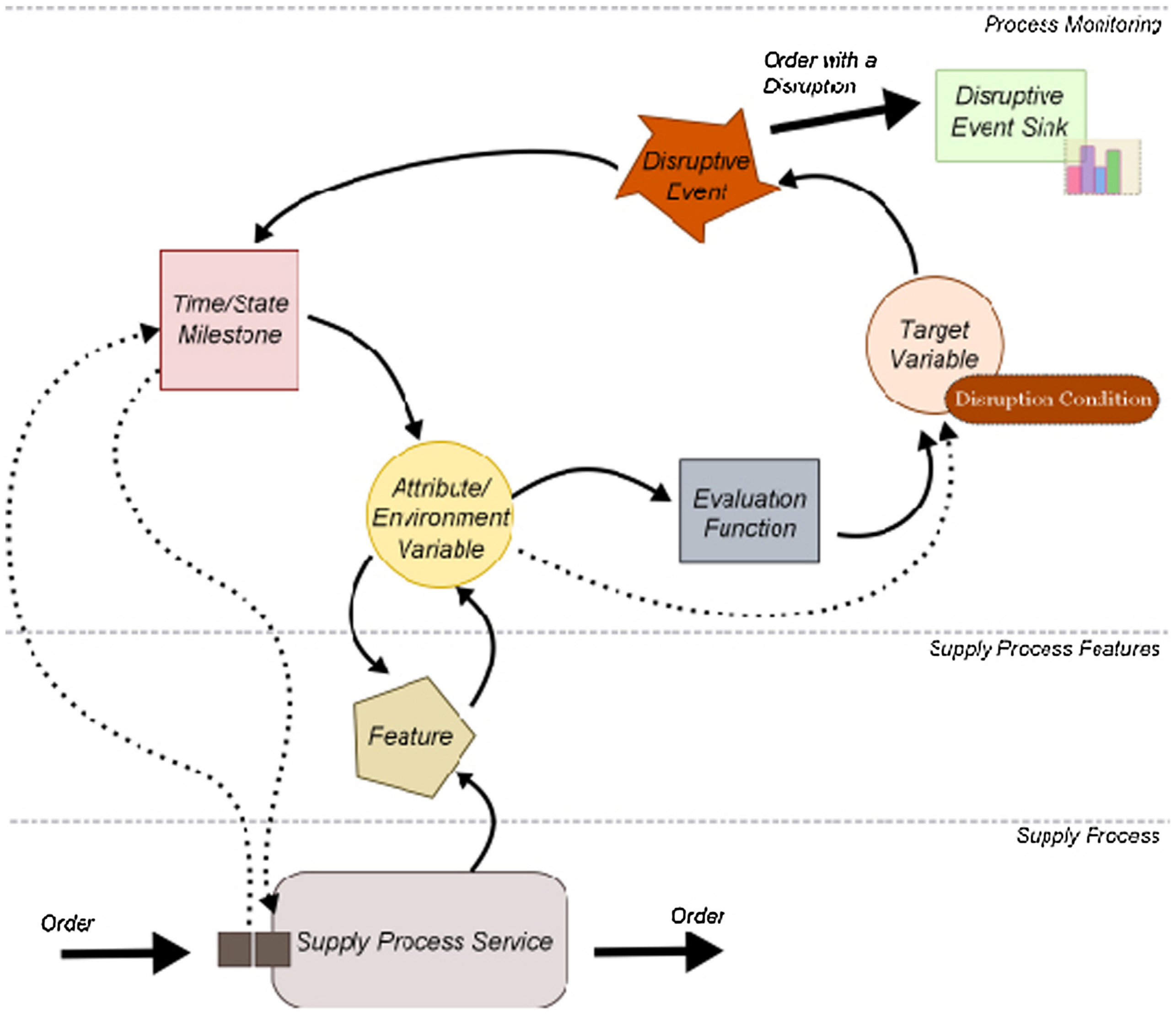

Fig. 4. Simulation elements and their interactions.

Process layer, the specialized *Supply Process Service* element for capturing the behaviour of each supply process stage $ps \in SP$ was defined. So, this simulation element receives an order $o \in O$, processes it, and sends it to the next supply process stage $ps^{+} \in SP$ through input/output interfaces. A supply process stage $ps \in SP$ can be monitored or not. If a supply process stage $ps \in SP$ must be monitored, the supply process service has interfaces to plug in features that define the set of variables to be observed $OV_m \in V$ in its associated milestone $m \in M$. Furthermore, the service has an interface that allows it to communicate with the related *Time/State Milestone* element for evaluating the possible occurrence of a disruptive event. If a disruptive event *de* has occurred, the monitoring process ends. But depending on the nature of the supply process, it could be stopped and the order must not be forwarded; otherwise, the order is sent to the next supply process stage $ps^{+} \in SP$ to continue its execution. When an order $o \in O$ is informed, the corresponding milestone $m \in M$ is activated; then, the monitoring of order processing takes place. In case a supply process stage $ps \in SP$ must not be monitored, the milestone activation is not verified. Internally, this simulation element keeps an orders queue while it is processing an order. The time needed to process an order is given by a probabilistic distribution set.

In the Supply Process Features layer, the *Feature* element is specified to capture aspects about order specifications, resources, and the supply process environment that must be observed for the monitored supply process stage $ps \in SP$. These features define the set of variables to be observed $OV_m \in V$ in milestone $m \in M$ associated to the supply process stage. This simulation element has an interface to get, from supply process stage $ps \in SP$, information of order $o \in O$ that is being executed. The element also has an interface to interact with the corresponding *Attribute/Environment Variable* elements that keep observed values of each variable $v \in OV_m$ in timestamps. The feature behaviour responds to a probabilistic distribution that can be adjusted before simulation.

At the *Monitoring level*, in the *Monitoring Process* layer, the main simulation elements are:

Attribute Variable keeps observed values of a feature that can be an order specification $s_o \in OV_m$, capacity $c_r \in OV_m$ *or* state $e_r \in OV_m$ of a resource $r \in R$. This simulation element has an interface to receive, from *Time/State Milestone* elements, signals to update the feature value, an interface for making queries to the *Feature* element under observation, and an interface to send the observed current value of the variable $v \in OV_m$ to the *Evaluation Function* element, or directly to the *Target Variable* element if the monitoring is reactive. The internal behaviour is driven by messages sent by the *Time/State Milestone* element, which activates this element and, in consequence, this element sends messages to the feature to be observed, saves its current value, and sends it through a message to either the *Evaluation Function* or the *Target Variable* elements, depending on the kind of monitoring.

Environment Variable: it keeps observed values of an environment variable $z \in OV_m$ that could affect supply process stage $ps \in SP$. Interfaces and internal operations are the same as those of the *Attribute Variable* element.

Time Milestone: it indicates a time point where attribute or environment variables $v \in OV_m$ must be observed. So, this simulation element generates a signal to notify the updating of observed variables. It has a parameter to define the time in which this signal has to be generated. This simulation element has an interface to dispatch an updating signal to the observed variable and another one to communicate with the *Supply Process Service* element that represents supply process stage $ps \in SP$ related to this milestone $m \in M$.

State Milestone: it indicates a condition after which attribute or environment variables $v \in OV_m$ must be observed. This simulation element checks a condition and, if it is true, dispatches a signal to observe the value of related variables; otherwise, it continues waiting. The condition is set as a parameter and can be a value (simple condition) or a function that returns a Boolean value (complex condition). Interfaces and internal operations are the same as those of the *Time Milestone* element.

Evaluation Function: it uses a predictive evaluation function $predEvFu \in EvFu$ for calculating an estimated value *ev* of order specifications $s_o \in OV_{m+}$ in upcoming milestones $m_{+} \in M$ from changes in the planned values *pv* of any observed variable $v \in OV_m$ of the current milestone $m \in M$. This simulation element has an interface to receive from *Attribute/Environment Variable* elements the observed current value of variable $v \in OV_m$ that activates this element, and an interface to send messages to the *Target Variable* element informing the estimated value *ev*.

Target Variable: For reactive monitoring, it uses a reactive evaluation function $reacEvFu \in EvFu$ to compare the current observed value *ov* of each observed variable $v \in OV_m$ of the current milestone $m \in M$ with its planned value *pv* for calculating its variation $\Delta v = (ov - pv)$ and then compares this variation with the threshold value *dc* to detect if a disruptive event *de* has occurred. To this aim, it uses the comparison function: *if* $(\Delta v \leq dc \ \forall \ v \in OV_m)$ *then* $\{de = False\}$, otherwise $\{de = True\}$. For predictive monitoring, it compares the estimated value *ev* received from the *Evaluation Function* element with its planned values *pv* for calculating the variation of order specifications $\Delta s_o = (ev - pv) \forall s_o \in OV_{m+}$ in upcoming milestones $m_{+} \in M$, and then compares these variations with a threshold value *dc* to predict if a disruptive event *de* could occur. To this aim, it uses the comparison function: *if* $(\Delta s_o \leq dc \ \forall s_o \in OV_{m+})$ *then* $\{de = False\}$, otherwise $\{de = True\}$. This simulation element has an interface to receive the observed value from *Attribute/Environment Variable* elements (if monitoring is reactive) or the estimated value from the *Evaluation Function* element (if monitoring is predictive), and an interface to send a message to the *Disruptive Event* element when the disruption condition is verified. Parameters that can be set are threshold values *dc*, the reactive or predictive monitoring choice, and planed values *pv*.

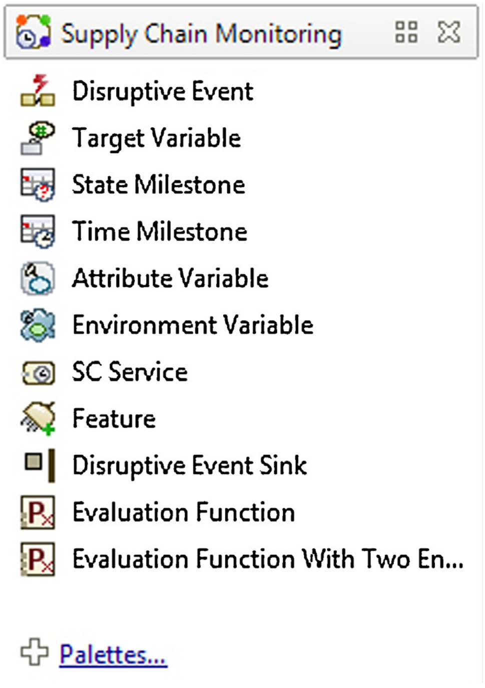

Fig. 5. Supply chain monitoring palette.

Disruptive Event: it sends an event notification to all connected milestones when the *Target Variable* element has indicated that the disruption condition has returned "True" for an order $o \in O$. This simulation element has an interface to receive the message sent by the *Target Variable* element when a disruptive event affecting an order $o \in O$ has occurred, and an interface to notify all *Time/State Milestone* elements of this disruption.

The *Evaluation Function* element can be defined as a simple mathematical function or can be supported by techniques such as Bayesian Networks and machine learning, among others. Finally, the *Disruptive Event Sink* element is defined to resume the information of disruptive events related to orders.

4.2. Metrics

In order to develop a simulation model of a monitoring process *MoPr* and assess its ability to detect and anticipate disruptive events, metrics providing quantitative performance measures for schedule execution, supply process monitoring and prediction ability of a monitoring process *MoPr* must be used.

The schedule execution can be assessed by using the following two metrics usually employed for this purpose [25–27].

On-Time Delivery: it is the percentage of orders delivered by the requested delivery date, as indicated in the order specification, related to all orders delivered during a time period (Eq. (1)).

$$On\_Time\_Delivery = \left(\frac{number\_of\_orders\_delivered\_by\_requested\_date}{total\_number\_of\_delivered\_orders}\right) \times 100$$

Order Turnaround Time: it is the average amount of time taken by a facility to fulfil an order, from the date on which each order is received by the supply chain until the date on which the order is shipped to customer. This metric defines the processing efficiency of a set of orders by adding the elapsed time between the reception date and the shipping date of each order (Eq. (2)).

$$Order\_Turnaround\_Time = \frac{\sum_{i=1}^{n} time\_to\_process\_order_i}{total\_number\_of\_processed\_orders} \tag{2}$$

where n = is the total number of processed orders.

The supply process monitoring can be assessed by the following two metrics [24,26]:

Orders affected by Disruptive Events: it is the amount of delayed or incorrectly fulfilled orders due to a disruptive event in a time period (Eq. (3)).

$$Orders\_withDE = total\_number\_of\_processed\_orders \tag{3}$$

Delivered Orders: it is the amount of correctly fulfilled orders (Eq. (4)).

$$Delivered\_Orders = total\_number\_of\_filled\_orders \tag{4}$$

The prediction ability of a monitoring process *MoPr* can be assessed by the following three metrics based on the contingency table:

Exactitude: it is the percentage of orders that have been correctly predicted in relation to all orders delivered during a time period (Eq. (5)).

$$Exactitude = \frac{(TP + TN)}{(TP + TN + FP + FN)} \times 100 \tag{5}$$

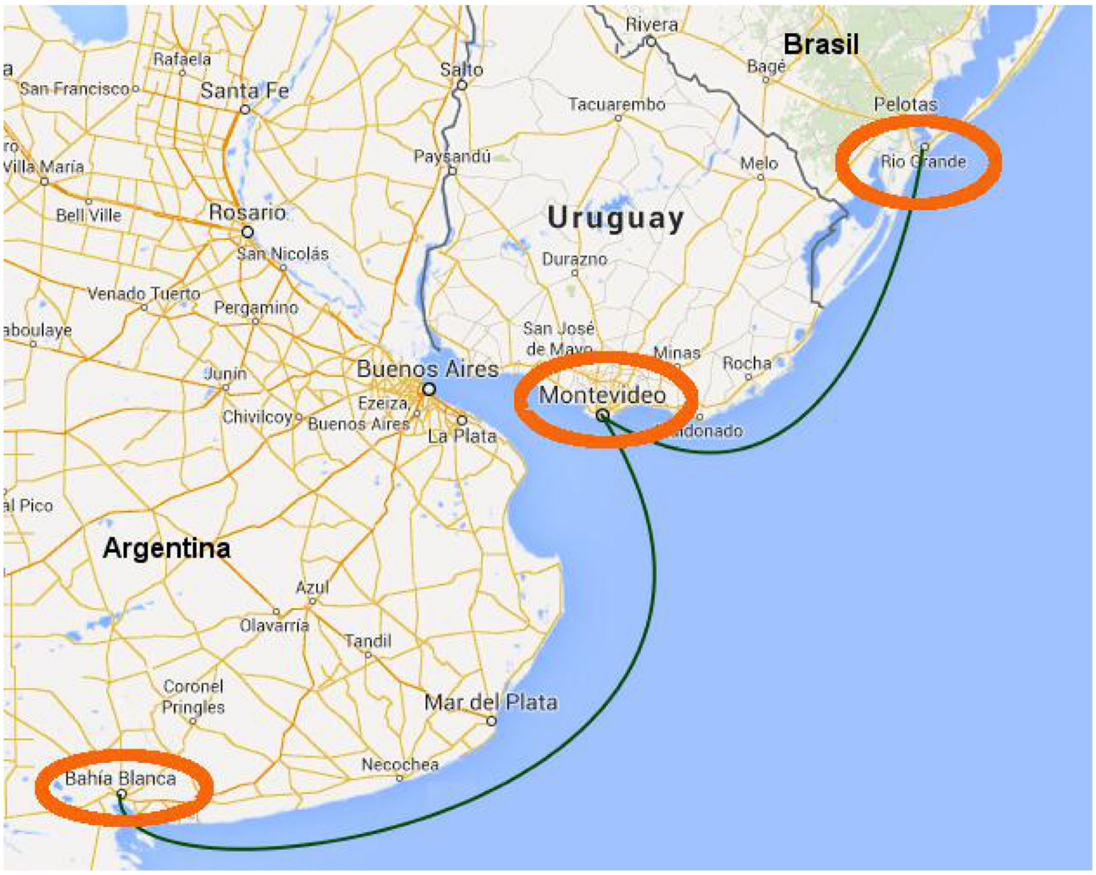

Fig. 6. Sea route.Reduce this figure.

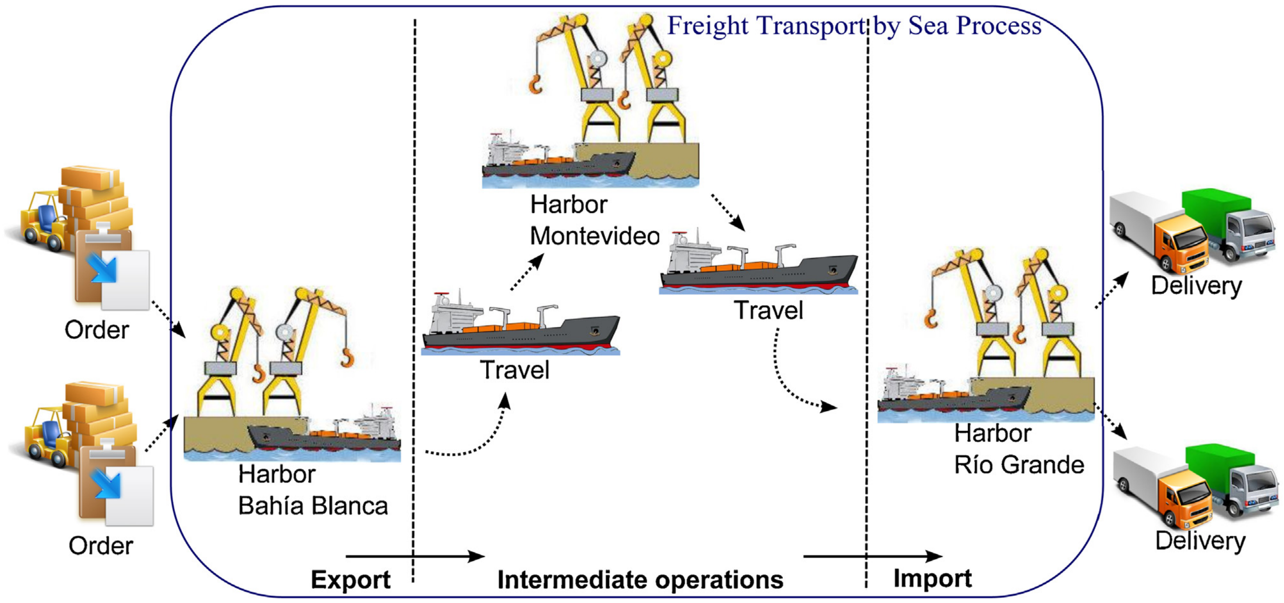

Fig. 7. Supply process MFTPr: route Argentina-Uruguay-Brazil.

Accuracy: it is the percentage of orders affected by disruptive events that have been correctly predicted in relation to all orders that have been predicted as affected by disruptive events (Eq. (6)).

$$Accuracy = \frac{TP}{(TP + FP)} \times 100 \tag{6}$$

Prediction Recall: it is the percentage of orders affected by disruptive events that have been correctly predicted in relation to all orders affected by disruptive events (Eq. (7)).

$$Recall = \frac{TP}{(TP + FN)} \times 100 \tag{7}$$

where:

TP (True Positive) is the amount of orders affected by disruptive events that have been correctly predicted.

TN (True Negative) is the amount of orders not affected by disruptive events that have been correctly predicted.

FP (False positive) is the amount of orders not affected by disruptive events that have been wrongly predicted as disrupted.

FN (False negative) is the amount of orders affected by disruptive events that have been wrongly predicted as not disrupted.

These metrics allow understanding the dissemination of disruptions and tracing the operational performance of a monitoring process *MoPr* for early decision-making.

4.3. Implementation: monitoring process library

The implemented framework includes all simulation elements needed to represent the monitoring process of a supply process. That is, elements to represent the monitoring workflow *MWf* and evaluation functions *EvFs* of a monitoring process *MoPr*. The framework was implemented by using AnyLogic [28] since it provides a flexible modelling language that allows capturing the complexity and dynamics of supply processes and the detection of disruptive events at different detail levels. Furthermore, AnyLogic provides graphical interface, tools, library objects, and support for the object-oriented model design paradigm.

The framework includes the *Supply Chain Monitoring* library (Fig. 5), which requires the Enterprise Library of AnyLogic. This library implements previously defined simulation elements, their interfaces and internal behaviour, and specialized classes that were implemented by using the JAVA programming language.

The library is composed of six elements to represent the static view *StaticView* of a monitoring model *MoMo* as an instance of the reference model *RfMo* (Fig. 1). These elements, which define the core of the library, are: *Disruptive Event*, *Target Variable*, *State Milestone*, *Time Milestone*, *Attribute Variable*, and *Environment Variable*. The library also includes the support elements: *SC Service,* for modelling the supply process; *Feature,* for modelling supply process features under monitoring; *Disruptive Event Sink,* for modelling statistic disruptive events; and *Evaluation Function* and *Evaluation Function with Two Entries,* for prediction, which could be linked to external prediction tools or techniques. All elements were implemented as *Active Object Class* provided by AnyLogic. Two additional classes were internally developed: *Order,* which models the order sent by the planning subsystem of the ERP; and *MonitoringMsg,* which represents messages sent to and received by the developed simulation elements. The library is a package that can be added as a tool bar providing users with a graphical interface.

5. Case study: marine freight transport process

This section presents a case study consisting of the monitoring process of a marine freight transport process for predicting disruptive events. The simulation model is composed by two main parts: Marine Freight Transport Process (*MFTPr*), which is the supply process under monitoring, and Marine Freight Transport Monitoring Process (MFTMoPr).

Table 1
Correspondence between supply process and simulation elements.

Element of the Supply Process MFTPr	Simulation Element
Port operation	Service
Travel (seen as process)	Service
Roadstead waiting	Delay
Port queue	Queue
Access channel	Conveyor

5.1. *Marine freight transport process* MFTPr

The case study is a traditional marine freight transport process carried out by South American industries. The sea route *Bahia Blanca-Montevideo-Rio Grande Do Sul* that links Argentina, Uruguay, and Brazil, is used for transporting a variety of goods, raw materials, and other merchandises among these countries (Fig. 6).

The port of Bahía Blanca (Argentina) gathers the major part of exports coming from Patagonia and the major part of the cereal grain production from Buenos Aires and La Pampa. This port has two main foreign-trade zones, which have several terminals specialized in different products: cereal grain, general load, containers, oil, and fuel. The access channel is 97 km long in which the maximum navigation speed of ships is 10 knots. The port of Montevideo (Uruguay) has a main terminal prepared to receive general cargo ships, container ships, bulk carriers, and oil tankers, among other kinds of ships. The access channel is 42 km long in which the maximum navigation speed of ships is 10 knots. The port of Rio Grande do Sul (Brazil) is prepared to handle container ships and bulk carriers. It has several terminals for general cargo and other kinds of ships. The access channel is 0.2 km long in which the maximum navigation speed of ships is 8 knots.

Supply process *MFTPr* involves ship loading at the port of Bahía Blanca. After 38 h of navigation, an intermediate stop in the port of Montevideo is planned, where the ship might be loaded or unloaded, depending on operations defined in the order or only proceed with the formality required to continue the travel. After leaving the port of Montevideo, the ship travels 28 h to the port of Rio Grande do Sul. This is the last stop, where the ship is unloaded (Fig. 7). In this supply process, a disruptive event can occur due to delays in port operations or along the journey, which cause a delay in the arrival of goods required by the distribution processes taking place in different locations.

Elements of the supply process *MFTPr* and their corresponding simulation elements are detailed in Table 1. The set of elements are connected for modelling the ports of Bahia Blanca, Montevideo, and Rio Grande do Sul, and the travels required by the planned route. These elements are illustrated by the simulation tool in Fig. 8. In the port of Bahia Blanca (prefix BB), only ship departure is modelled, so operation time before leaving (*BBHarborOperation*) is included. In the ports of Montevideo (prefix M) and Rio Grande do Sul (prefix RG), the modelled elements are: roadstead (*MRoadsteadWaiting* and *RGRoadsteadWaiting*), the port queue (*MHarborQueue*, and *RGHarborQueue*), the access channel (*accessChannelToM*, and *accessChannelToRG*), and the port operation itself (*MHarborOperation*, and *RGHarborOperation*).

5.2. *Monitoring process MFTMoPr*

A monitoring process *MoPr* is defined according to the complexity and features related to the supply process to be monitored. Features of the supply process *MFTPr* are described in Table 2. Features *BBOperationTime*, *MOperationTime*, and *RGOperationTime* are defined to capture the operation time at the port. Wind speed can affect the journey from Bahia Blanca to Montevideo by reducing ship navigation speed or even forcing the ship to stop. So, the *windSpeedSeaTravelToM* feature is specified for monitoring wind speed. The general condition of the sea between Montevideo and Río Grande do Sul can be changeable due to the variability of maritime currents in the first part of the route. Thus, *shipSpeedSeaTravelToRG* feature for monitoring ship speed and *nauticalMileSeaTravelToRG* feature for monitoring nautical miles travelled by the ship during the trip are specified.

The monitoring process *MFTMoPr* is defined by associating features of supply process *MFTPr* to variables that keep observed values in a time stamp (Table 2). The *observedWindSpeedSeaTravelToM* environment variable is specified for monitoring wind speed; and attribute variables *observedBBOperationTime*, *observedOperationTimeInM*, *observedOperationTimeInRG*, *observedShipSpeedSeaTravelToRG*, and *observedNauticalMileSeaTravelToRG* are specified for monitoring the operation time in ports, ship speed, and nautical miles travelled respectively.

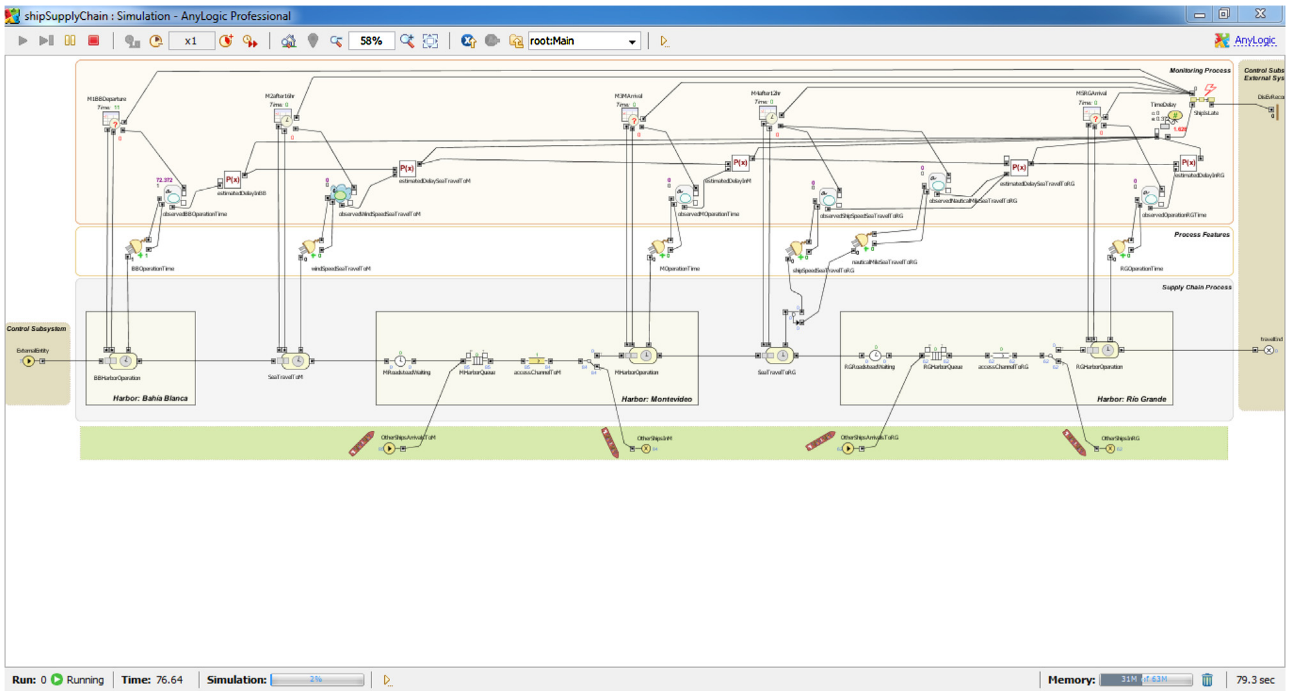

Fig. 8. Simulation Environment: monitoring process MFTMoPr.

Milestones that define the time in which the monitoring is executed (Table 2) are defined. State milestones *M1BBDeparture*, *M3MArrival,* and *M5RGArrival* are defined in each port for monitoring load, unload, customs process, and other specific operations, except for ships' arrival to the port. These milestones are activated when a ship starts its operation. Two time milestones are defined to monitor the travel: *M2after16hr* for the stretch from Bahia Blanca to Montevideo, being activated 16 h after the time at which the ship sets sail; and *M4after12hr*, for the stretch from Montevideo to Río Grande do Sul, being activated 12 h after the time at which the ship sets sail.

In the monitoring process *MFTMoPr*, the objective is to monitor the cumulative delay that can affect the order fulfilment. So, *TimeDelay* target variable and *ShipIsLate* disruptive event were deined (Table 2). Parameters of *TimeDelay* target variable were set in 2 h for the *PlannedValue* and 15 h for the *Threshold*. In this case, tolerance is up to 5% of the total hours [29]. This percentage represents the total cumulative delay tolerated by this kind of supply process and was defined according to the total time required for the whole itinerary from Bahía Blanca to Río Grande do Sul. The *ShipIsLate* disruptive event sends a signal and records the disruption when it is anticipated by the predictive evaluation function *PredEvFu*.

The cumulative delay is defined according to evaluation functions *EvFs* programmed for this example (Table 2). In this case, the predictive evaluation function *PredEvFu* was programmed as a simple mathematical function.

The simulation objective is the cumulative delay that can affect the order fulfilment. To this aim, the simulation contrasts the order fulfilment under normal conditions with the disruption caused by the cumulative delay of delivery.

5.3. The environment simulation model

External systems such as the planning subsystem of ERP systems that provide the schedule of orders to be processed are modelled as sources that generate the "load" on the supply process under monitoring (Table 3). Ships that can arrive to a port but are not under monitoring are modelled as sources that affect the time of entry into the port (Table 3). Ships that can depart from a port but are not under monitoring are modelled as sink that affect the departure time from the port, since these departures are not counted into the operation time of ports (Table 3). Those responsible of either production or logistic areas who make decisions according to the information of the supply process and disruptive events are also modelled as sink (Table 3).

5.4. Experimental setup

Input parameters allow configuring experiments based on the domain. Parameters values can be adjusted to the needs and estimated from previous experiences, historical data, or information of the data base of involved systems [30–33]. For the case study, the following parameters were set:

- *Order*: start date, end date (302 h in average after start date, considering a threshold setting by the user), description, quantity (load), and additional control information.
- *Interarrival time*: the arrival of orders responds to uniform distribution functions because scheduled orders were simulated to generate the load. So, to generate a significant load for validating the monitoring process, the time between orders was low, receiving an order each ten hours.
- *Services behaviour*: a normal distribution is set up to model the service time with mean and standard deviation calculated from statistics of port operations. Travel time in each stretch of the route involved in this supply process responds to a normal distribution, also with mean and standard deviation taken from estimated travel times. The internal queue capacity was limited according to the operation capacity of ports.
- *Roadsteads waiting*: a normal distribution is defined with mean and standard deviation calculated following statistical waiting times in the roadstead of ports.
- *Port queues*: queues in ports (Montevideo and Río Grande do Sul) were configured with limit capacity according to characteristics of ports entry.
- *Access channels*: length of access channels and maximum ship speed are defined according to that previously detailed.
- *Features behaviour*: the operation time in each port includes loading and unloading, customs process, and other specific operations. This time was set as a normal distribution with a mean and standard deviation calculated from historical data of each port. Wind speed has been represented with a triangular distribution with the major probability in a defined value corresponding to the Beaufort scale. Ship speed was configured following a normal distribution with a mean value calculated by a function that divides the total miles and the travel time (time in which the milestone activates). Nautical miles travelled by the ship during the trip is defined with a normal distribution, where the mean value is dynamically calculated from the speed at which the ship is coming multiplied by the time in which the milestone indicates the measurement (12 h after the time the ship sets sail).
- *Threshold:* maximum time delay to consider a disruptive event.
- *Planned value:* in this example, the planned time delay was 2 h.
- *Milestones conditions/time frequencies*: state milestones were configured by defining a conditional function that returns the value "true" each time the ship enters a port for starting operations. The time of time milestones is defined after the ship sets sail. In the stretch Bahía Blanca-Montevideo the time was set as the departure date plus 16 h and in the stretch Montevideo-Río Grande do Sul the time was set as the departure date plus 12 h.

Orders were sequentially processed to have a major control over each of them that arrived to be processed. Orders were monitored over six months. The simulation was run several times by using different seeds to set the generator elements.

5.5. Simulation results

Five simulation runs with several replications for each of the four *Threshold* values were performed. Results of average values of simulation outputs for the monitoring process *MFTMoPr* are compared with simulation outputs of the supply process *MFTPr* to

Table 2
Correspondence between *MFTMoPr* and simulation elements.

MFTMoPr Elements	Simulation Element
Port operation time	Feature
Wind speed	Feature
Ship speed	Feature
Nautical mile	Feature
Observed port operation time	Attribute Variable
Observed wind speed	Environment Variable
Observed ship speed	Attribute Variable
Observed nautical mile	Attribute Variable
Control point	State/Time Milestone
Target and disruption condition	Target Variable
Ship delay that could imply delay of delivery	Disruptive Event
Delay estimation	Evaluation Function

Table 3
Correspondence between environment and simulation elements.

Environment Element	Simulation Element
External entities from system	Source
Other ships arrival	Source
Travel end (supply process)	Sink
Disruption (disruptive events)	Disruptive Event Sink
Other ships departure	Sink

Table 5
Prediction errors of the monitoring process.

Threshold	TP	TN	FP	FN
3	22.6	17.0	13.6	0.8
6	20.2	21.0	11.6	1.2
15	13.8	32.4	6.6	1.2
18	12.2	35.4	5.2	1.2

validate monitoring predictions by using the metrics defined in Subsection 4.2.

On average, 432 orders per simulation run were received. From them, 54 orders in average were delivered; other orders were in process when simulation finished, i.e. after an elapsed time of 180 days.

Average values of the metrics of the analysed supply process *MFTPr* were *On-Time Delivery = 52.85%* (Eq. (1)) and *Delivered Orders = 54 orders* (Eq. (4)). Table 4 shows average values of *Order Turnaround Time* (Eq. (2)), *Orders affected by Disruptive Events* (Eq. (3)), and the simulation time for four different threshold values. Table 4 shows that the amount of orders affected by disruptive events decreased from 23.4 to 13.4 as the specified time delay to consider a disruptive event (threshold) was increased from 3 to 18 h.

From simulation results, the contingency Table with average values of prediction errors of the monitoring process *MFTMoPr* for each threshold value was generated (Table 5).

Based on the contingency table, the average values of the metrics of monitoring process MFTMoPr were assessed. The ability of the monitoring process to predict orders affected by disruptive events can be analysed from the average values of *Exactitude* (Eq. (5)), *Accuracy* (Eq. (6)), and *Recall* (Eq. (7)) summarized in Table 6. Table 6 shows that the monitoring process predicted the orders affected by disruptive events with an exactitude that increased from 0.73 to 0.88 as the specified time delay to consider a disruptive event (threshold) was increased from 3 to 18 h. For threshold = 3, 62% of the orders predicted as affected by disruptive events by the monitoring process were actually affected by disruptive events. This percentage increased to 70% for threshold = 18; while 97% of orders affected by disruptive events were correctly predicted by the monitoring process for threshold = 3, and 91% for threshold = 18.

Such accuracy indicates that the monitoring process will issue alerts about orders that might be potentially affected by a disruptive event, when in fact between 30 and 38% of them (depending of the threshold value) will end without disruption. This imprecision is not of significant concern since they result in warnings that are not confirmed.

The recall, however, implies that the monitoring process will not issue alerts about orders that will be affected by disruptive events. This fact is more disturbing but, for the present case, this failure will vary between 3 and 9% (depending on the threshold value). Namely, between 91 and 97% of the orders affected by a disruptive event will be alerted, which is a range of values that can be considered as acceptable.

The ability of the monitoring process to predict orders affected by disruptive events depends on the prediction function being used. Prediction functions based on models with greater predictive power than that used in this study, for example Bayesian Networks, could improve the monitoring process performance (this issue being out of the scope of this proposal).

Simulations were run on a notebook computer with a Pentium 5i processor and 8 GB of RAM. The simulation time was 441 min on average (Table 4). It is important to highlight that the simulation involves three levels of complexity, which imply a lot of components and variables as previously explained.

5.5.1. Analysis objectives

The framework for modelling and simulating the supply process monitoring based on the reference model *RfMo* allows modelling, simulating, and validating a monitoring process *MoPr* for each order $o \in O$ involved in a schedule *Sch*. It is able to detect disruptive events affecting any order specifications $s_o \in S$, capacities $c_r \in C_r$, *or* states $e_r \in E_r$ of resources $r \in R$, or to anticipate disruptive events affecting any order specifications $s_o \in S$.

5.5.2. Scalability

The framework was implemented in a library which provides a set of simulation elements following fundamentals of discrete event simulation and the object-oriented paradigm [22,23]. Each element is an embedded object that may embed other objects from simple to complex elements, encapsulating the internal behaviour (dynamics) and communicating through its interfaces. This type of modelling allows evolving into modelling and simulation of complex monitoring process *MoPr*. Evaluation functions used in the case study involves simple mathematical calculus, but other prediction techniques could be supported such as Bayesian Networks, Petri Nets, CEP, or a rule language.

5.5.3. Reusability

The supply process, monitoring process *MoPr* and its environment, as well as, prediction algorithms that have been previously defined and tested could be reused by others enterprises that hire monitoring web service MSMDE. By setting parameters of simulation elements, it is possible to add or delete some of them to achieve a new simulation model of supply process monitoring.

5.5.4. Usability

The framework allows generating simulation models of monitoring processes *MoPr* through a user-friendly interface without the need of knowing the abstract language provided by

Table 4
Execution schedule and supply process monitoring metrics.

Threshold [h]	Orders affected by Disruptive Events	Order Turnaround Time [h]	Simulation Time [min]
3	23.4	299.87	453.04
6	21.4	303.98	454.84
15	15.0	301.81	358.00
18	13.4	301.98	500.22

Table 6
Exactitude, precision, and recall of the monitoring process.

Threshold	Disruptive Event Prediction		
	Exactitude	Accuracy	Recall
3	0.73	0.62	0.97
6	0.76	0.64	0.94
15	0.86	0.68	0.92
18	0.88	0.70	0.91

the reference model *RfMo*. The developed simulation elements are graphical representation tools. They make this framework easy to be learnt and used by those who are in charge of hiring monitoring web service MSMDE for an enterprise.

5.5.5. Input parameters

The simulation model requires the configuration of input parameters about supply processes, planned values of orders and resources, and environment data. Data about supply processes are obtained from historical data available in the domain. Data about environment may be obtained from web services that provide weather forecasts and road traffic conditions. Planned orders and resources are obtained from production and distribution schedules in the data base of the planning subsystem of ERP systems. And the execution values are obtained from the data base of the execution subsystem of ERP systems, in which disruptive events are registered.

6. Conclusions

The presented framework allows for the responsibility of enterprises that hire the monitoring web service MSMDE to generate simulation models of monitoring processes through a user-friendly interface, without the need of knowing the abstract language provided by the reference model *RfMo*. The library implemented by the framework allows representing static and dynamic parts of a supply process monitoring model through concepts of the reference model RfMo.

Metrics providing quantitative performance measures enable users to develop and test different monitoring processes until reaching one that best represents their supply process. So, monitoring models to be provided to the web service MSMDE could be tested suitably.

Simulation models generated with the library are encoded in JAVA. So, models can be exported, allowing users to employ other simulation tools or embed simulation models.

The marine freight transport process allowed showing how a supply process and its environment can be modelled and simulated by using the library. Simulation results show the approach ability to anticipate disruptive events. Also, critical stages of a supply process could be identified in order to prevent disruptive events. For example, it is possible to identify if a port operation is requiring more time than planned and if this can produce significant ship delays.

While for the case study a simple function was used for predicting disruptive events, more complex functions based on the best prediction techniques or probabilistic models could be developed to be included in the library so as to improve the attained results. Future work will be aimed at this direction.

Acknowledgements

Financial support for this research study was granted by "Desarrollo de Herramientas de Software para Implementar procesos Colaborativos" PIP 11220110100705 of Consejo Nacional de Investigaciones Científicas y Técnicas, "Modelos de negociacióṅn de holguras para reparar un programa de abastecimiento en ejecución" 25/o146 and "Evaluación dinámica de la calidad de procesos y productos para la toma de decisiones tempranas en industrias de base tecnológica" PID EIUTNVM0003581 of Universidad Tecnológica Nacional.

References

[1] P.R. Kleindorfer, G.H. Saad, Managing disruption risks in supply chains, Prod. Oper. Manage. 14 (2005) 53–68.
[2] N. Radjou, L. Orlov, Adapting to Supply Network Change, The TechStrategy Report, 2002.
[3] H. Van Landeghem, H. Vanmaele, Robust planning: a new paradigm for demand chain planning, J. Oper. Manage. 20 (2002) 769–783.
[4] R. Zimmermann, Agent-based supply network event management, Whitestein Ser. Softw. Agent Technol. (2006).
[5] L.A. Bearzotti, E. Salomone, O.J. Chiotti, An autonomous multi-agent approach to supply chain event management, Int. J. Prod. Econ. 135 (2012) 468–478.
[6] E. Fernández, C.M. Toledo, M.R. Galli, E. Salomone, O. Chiotti, Agent-based monitoring service for management of disruptive events in supply chains, Comput. Ind. 70 (2015) 89–101.
[7] A. Guarnaschelli, E. Fernandez, O. Chiotti, E. Salomone, A service-oriented approach to collaborative management of disruptive events in supply chains international journal of innovative computing, Inf. Control 8 (2012) 5341–5368.
[8] E. Fernández, E. Salomone, O. Chiotti, A model driven development approach based on a reference model for predicting disruptive events in a supply process, Comput. Ind. 63 (2012) 482–499.
[9] D.L. Moody, The 'Physics' of notations: toward a scientific basis for constructing visual notations in software engineering, IEEE Trans. Softw. Eng. 35 (2009) 756–779.
[10] M. Bansal, A. Adhitya, R. Srinivasan, I.A. Karimi, An online decision support framework for managing abnormal supply chain events, Comput. Aided Chem. Eng. 20 (2005) 985–990.
[11] R. Liu, A. Kumar, W. van der Aalst, A formal modeling approach for supply chain event management, Decis. Support Syst. 43 (2007) 761–778.
[12] A. Winkelmann, S. Fleischer, S. Herwig, J. Becker, A conceptual modeling approach for supply chain event management (SCEM) ECIS, Proceedings of the 17th European Conference on Information System (2009) 2009.
[13] Y. Huang, B.C. Williams, L. Zheng, Reactive, model-based monitoring in RFID-enabled manufacturing, Comput. Ind. 62 (2011) 811–819.
[14] D.C. Luckham, The power of events: an introduction to complex event processing, Distributed Enterprise Systems, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
[15] G. Meyer, J.C. Wortmann, Robust planning and control using intelligent products. agent-Mediated electronic commerce. designing trading strategies and mechanisms for electronic markets, Lect. Notes Bus. Inf. Process. 59 (2010).
[16] K. Kim, I. Choi, C. Park, A rule-based approach to proactive exception handling in business processes, Expert Syst. Appl. 38 (2011) 394–409.
[17] J.M. Ko, C. Kwak, Y. Cho, C.O. Kim, Adaptive product tracking in RFID-enabled large-scale supply chain, Expert Syst. Appl. 38 (2011) 1583–1590.
[18] G. Vlachakis, D. Apostolou, An event-based framework for supply chain management, The 5th International Conference on Information, Intelligence, Systems and Applications, IISA 2014, 2014, pp. 440–445.
[19] A.A. Tako, S. Robinson, The application of discrete event simulation and system dynamics in the logistics and supply chain context, Decis. Support Syst. 52 (2012) 802–815.
[20] G. Heinecke, J. Köber, A. Kunz, S. Lamparter, Modeling the basic cause-Effect relationship between supply chain events and performance, 3rd International Conference on Dynamics in Logistics (2013) 163–174.
[21] E. Genc, N. Duffie, G. Reinhart, Event-based supply chain early warning system for an adaptive production control, Robust Manufacturing Conference (RoMaC 2014), Procedia CIRP 19, 2014, pp. 39–44.
[22] C.G. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, 2nd edition, Springer, US, 2008 978-0-387-33332-8.
[23] E.C.V. Alexander, Verbraeck Design guidelines for simulation building blocks, Proceedings – Winter Simulation Conference (2008) 923–932.
[24] V. Bogado, S. Gonnet, H. Leone, Modeling and simulation of software architecture in discrete event system specification for quality evaluation, Simulation 90 (2014) 290–319.
[25] A. Gunasekaran, C. Patel, R.E. McGaughey, A framework for supply chain performance measurement, Int. J. Prod. Econ. 87 (2004) 333–347.
[26] G. P.Kurien, M.N. Qureshi, Study of performance measurement practices in supply chain management, Int. J. Bus. Manage. Social Sci. 2 (2011) 19–34.
[27] Usaid – Deliver Project, Task Order 1, Measuring Supply Chain Performance Guide to Key Performance Indicators for Public Health Managers (2010).
[28] Anylogic, Multimethod Simulation Software (2016) Available from: http://www.anylogic.com/ (accessed 15.09.15).
[29] J. Palombarini, E. Martínez, SmartGantt—an intelligent system for real time rescheduling based on relational reinforcement learning, Expert Syst. Appl. 39 (2012) 10251–10268.

[30] Puerto de Bahía Blanca, Consorcio De Gestión Del Puerto De Bahía Blanca, Puerto de Bahía Blanca, 2016 http://puertobahiablanca.com/ (accessed 06.11.15).
[31] ANP Administración Nacional de Puertos, República Oriental Del Uruguay, Montevideo Departamento, 2016.
[32] Porto do Rio Grande, Superintendência Do Porto De Rio Grande (2016). www.portoriogrande.com.br (accessed 06.11.15).
[33] Sea-Distances. http://www.sea-distances.org/ (accessed 06.11.15).