International Journal of Innovative Computing, Information and Control
Volume **8**, Number **7(B)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 5341–5368

A SERVICE-ORIENTED APPROACH TO COLLABORATIVE MANAGEMENT OF DISRUPTIVE EVENTS IN SUPPLY CHAINS

Armando Guarnaschelli[1], Erica Fernandez[1], Omar Chiotti[2] and Héctor E. Salomone[1]

[1]INGAR, Instituto de Desarrollo y Diseño
CONICET-UTN
Avellaneda 3657-3000, Santa Fe, Argentina
{ guarnaschelli; ericafernandez; salomone }@santafe-conicet.gov.ar

[2]CIDISI, I&D Information System Engineering Centre
FRSF-UTN
Lavaisse 610, Santa Fe 3000, Argentina
chiotti@santafe-conicet.gov.ar

Received March 2011; revised July 2011

Abstract. *This work presents a comprehensive proposal to systematically address the problem of Collaborative Management of Disruptive Events in Supply Chains (CMDE SC). The problem is defined as a collaborative business process that specifies a set of decision making activities that require complex models to systematize the activities of capturing information about internal and external changes to predict disruptive events that can affect the schedule execution, and to systematize the activities of feasibility checking and schedule repairing considering the distributed nature of supply chains. A service-oriented approach implementing this collaborative business process is presented. Reference Models developed for automatically deriving appropriate executable models for feasibility checking and schedule repairing, and for monitoring orders and resources, are described. Examples of models validation are also described.*
Keywords: Business processes, SOA, Supply chain, Event management

1. **Introduction.** The ultimate goal of integrated management of Supply Chains (SCs) is to improve the competitive advantages resulting from the integration. To this aim, enterprises in a supply chain perform collaborative business processes [1]. Particularly, collaborative planning processes allow each enterprise to obtain production and/or distribution schedules synchronized with the schedules of other SC members [2].

During the execution of scheduled operations, significant changes with regard to planned values of order parameters and resource availability may occur. These changes may affect the schedules and their synchronization.

In this work, a *schedule* is defined as a set of orders, where each order represents a supply process (production or distribution) that assigns materials to a place, states the required resources, the time period during which each resource is required and its required capacity and state. It is assumed that during the schedule definition, certain buffers (material, resource capacity and time) have been provided to absorb future changes. A supply chain schedule may be composed of different synchronized schedules. A *disruptive event* is defined as any significant change during the schedule execution either in the specification of an order or in the planned values in resource availability that may affect the schedules and their synchronization. They can produce negative effects that propagate throughout the supply chain [3, 4, 5]. The importance of disruption management has been theoretically shown in [6, 7].

An *exception* is defined as a deviation from the schedule that prevents the fulfillment of one or more supply process orders.

This uncertainty is recognized by the robust planning paradigm [8] that proposes to define allocations of buffers (material, resource capacity and time) to achieve robust schedules most likely to remain stable during execution. The objective is to avoid re-planning tasks, which can be costly and time consuming, since all enterprises involved in the SC should agree on a new collaborative plan. However, due to the impossibility of predicting with certainty the time and place of occurrence of disruptive events and their magnitude, usually the provided buffers cannot absorb all changes. Then, after a disruptive event has occurred it is necessary to check if schedules are still feasible and if not, define modifications to them. In practice the decision process that takes place after the disruption is loosely structured. Managers are seldom supported by systematic methodologies to cope with the problem caused by the disruptive event, and when they do, the solution is usually a re-planning task. This is a weakness that needs to be addressed to preserve a SC competitive in the future landscape. Future SCs have to be more adaptive reacting quickly and correctly to changes, and disruptive events have to be managed and contained on site making re-planning activities less frequent [5, 9].

The benefits of having a robust schedule are indisputable, but despite the effort in terms of resource buffers done to provide robustness, operation managers know that is not easy to effectively use these buffers in a systematic way maintaining the execution adherence to planned targets. The objective is to repair a schedule through limited and localized modifications. Most models to repair schedules we have found in the literature do not consider the distributed nature of SCs [10, 11, 12, 13]. For instance in [14] the authors develop a disruption management decision support system for logistics scheduling, but do not consider the important factor that in a supply chain many business partners can coexist and their schedules and scheduling systems are different in general. Within an SC, the schedules of different enterprises are synchronized, so they all have to be properly repaired. To this aim, appropriate models to check feasibility and repair schedules considering the distributed nature of an SC have to be used.

To perform this task, disruptive event information in advance can help to make better decisions. As stated earlier, disruptive events are significant changes with regards to the planned values that can affect the schedule execution. Some of these changes can take place into the enterprise, e.g., the availability of an enterprise resource, and therefore easy to capture by the enterprise; but other changes can take place outside the enterprise, e.g., the availability of an external resource, which may not be as easy to obtain. A model to capture disruptive events has to be able to get information about internal and external changes that can affect the schedule execution. By capturing them, it is possible to infer changes that will affect specification of scheduled orders or the availability of a resource and predict a disruption when it has enough evidence that this will occur. To this aim, appropriate prediction models of disruptive events have to be used.

The business environment is subject to frequent changes so the SC structure can also change with some frequency. That is, new enterprises become part of the SC while others may leave it, defining a dynamic SC structure.

Under this scenario a new concept called Supply Chain Event Management (SCEM) has emerged [15, 16]. It proposes to track and trace the flow of events along the SC, *monitoring* planned and unplanned events, detecting relevant changes and *notifying* them to decision makers in real time. Also, it gives the decision maker *simulation* support for analyzing alternatives to find a response to unforeseen events, *control* support for analyzing and documenting the effect for subsequent SC processes, and allows decision makers

to introduce proactive changes into the established conditions, and provides support for measures assessing, analyzing and evaluating historical data [17].

Academic and industrial researchers [5, 16] have recognized the systematic repairing of the schedules using the planed buffers as an outstanding issue to be considered.

This work, defines the *Collaborative Management of Disruptive Events in Supply Chains (CMDESC)* as a *collaborative business process* that specifies a set of decision making activities that require complex models to systematize the capture of information about internal and external changes, the prediction of disruptive events that can affect the schedule execution, and the activities of feasibility checking and schedule repair considering the distributed nature of a SC.

Because of the dynamic SC structure and the complexity of decision support models required, the main requirements to be satisfied by an information system implementing this collaborative business are as follows: ability to join or withdraw an enterprise from the business process; ability to describe ongoing synchronized schedules of any kind from different enterprises coexisting in the same SC; ability to capture the planned buffers in a way that it is suitable for analyzing feasibility in presence of disruptive events; ability to automatically derive an appropriate executable model for feasibility check and automated repair of disrupted schedules searching for a feasible solution using just the planned buffers and finally, ability to automatically derive appropriate executable models for capturing and analyzing significant changes in environmental variables and/or resources that allow the inference that a disruptive event affecting the specifications of an order or the future availability of a resource will occur.

In the literature, several proposals of SCEM systems architectures can be found [16, 18, 19, 20, 21, 22, 23, 24], but none of them satisfy all requirements listed before, particularly the requirements for collaborative automated repair of disrupted schedules and/or ability for predicting disruptive events on orders and resources. Another set of related works found in the literature is focused on error recovery problem [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35], which can be considered similar to the SCEM problem, but all of them are focused on the intra-enterprise context, and therefore do not satisfy the required ability for collaboration.

The objective of this work is to present a business process for the collaborative management of disruptive events in the supply chain execution, to describe its functional and nonfunctional requirements, and to propose an approach based on a service-oriented architecture (SOA) [36] that supports the requirements. The work is organized as follows. Section 2 presents the functional and nonfunctional requirements of the CMDESC business process. Section 3 describes the CMDESC business process. Section 4 presents a SOA proposal for the CMDESC system. Section 5 describes the reference models used to represent the semantics of the proposed SOA solution and to derive the associated models for implementing the internal structure of the services. Section 6 presents the service implementation, describing the software components that provide the services. Section 7 describes the case studies we have used for validation. Section 8 presents discussions and future work.

2. **Requirements Analysis.**

In Section 1, an initial elicitation of requirements for a CMDESC system solution was introduced, and in the next sections a more elaborated description of these requirements is given by aggregating and typifying them into two categories functional and non-functional requirements. Functional requirements are the ones that establish required functionalities of an information system. Non-functional requirements specify criteria that can be used to judge the operation of a system, rather than specific functions in a system.

2.1. **Functional requirements.** In our analysis of requirements for the CMDESC system the following three main functionalities have been identified:

2.1.1. *Execution control of a schedule.* This implies to coordinate the activities for tracking the progress of a schedule under execution and implementing strategies to restore feasibility to a damaged schedule; when an exception is detected, notify it to the decision maker in real time. A Control subsystem is proposed to provide this functionality.

2.1.2. *Capture disruptive events.* These may affect the schedule. A Monitoring subsystem is proposed to provide this functionality. It must be able to predict and detect disruptive events.

2.1.3. *Feasibility management.* This implies analyzing the impact of a disruptive event on a schedule and searching for strategies to restore feasibility to a damaged schedule. A Feasibility Management subsystem is proposed to provide this functionality.

A CMDESC solution should provide these three functionalities. The subsystems defined previously might be defined as independent systems one from the other, or as parts of a centralized system.

2.2. **Actors.** The actors requiring the functionalities offered by these subsystems are:

2.2.1. *Controller.* This actor represents the entity controlling the execution of a schedule.

2.2.2. *Monitor.* This actor represents the entity detecting and predicting disruptive events that may affect a schedule execution.

2.2.3. *Feasibility manager.* This actor represents the entity in charge of analyzing schedule feasibility and repair schedules.

2.3. **Use cases.** The following UML use cases [37] have been defined.

2.3.1. *Use case: predict and detect disruptive events.* This use case is realized by the Monitoring subsystem. The monitoring function implies detecting relevant disruptive events in real time. To perform this functionality four main monitoring activities have to be carried out during the execution of a schedule.

Activity 2.3.1.1: *Monitoring order specification changes.* The objective is to capture independent changes of the order's requirement values, i.e., start time, quantity or end time. By independent, we mean original modifications to the order specification, not as derived consequence of adjusting the order in response to other disruptive events.

Activity 2.3.1.2: *Monitoring current status of resource feasibility.* The objective is to capture significant changes on the current value of any attribute of a resource that is critical to grant its feasibility.

Activity 2.3.1.3: *Monitoring order progress.* The objective is to monitor on going execution orders to proactively predict if a disruptive event affects the order expected completion. This implies to capture significant changes on any variable measuring the order progress or having a predictive relationship with this progress. Typically, they are variables in the execution environment that are used as predictors of potential disruptions.

Activity 2.3.1.4: *Monitoring changes on resource's future expected availability.* The objective is to capture significant changes of the planed value of the future availability of a resource.

Precondition 2.3.1.1: The schedule should be expressed in terms of a schedule representation understandable by both parties (Controller and Monitor).

Precondition 2.3.1.2: To realize this use case information from other systems is needed

(for example, manufacturing execution system, warehousing management system and transportation management system).
Post-condition 2.3.1.1: The disruptive event should be expressed in a common schedule representation.

2.3.2. *Use case: check schedule feasibility.* In this use case the feasibility analysis of a schedule is realized by the Feasibility Management subsystem.
Precondition 2.3.2.1: The Schedule should be expressed in terms of a Schedule representation understandable by both parties.

2.3.3. *Use case: repair schedule.* In this use case the repair of a damaged schedule is required. This use case includes the use case *Check Schedule Feasibility*, because before introducing changes to a schedule, the impact of the disruptive event should be assessed and it is also realized by the Feasibility Management subsystem.
Precondition 2.3.3.1: The schedule is expressed in a common representation.
Precondition 2.3.3.2: The schedule information sent by the Controller should include information on orders requirements and resources availability to the extent of allowing the assessment of the execution feasibility and searching for schedule repair solutions.
Post-condition 2.3.3.1: The solution to a schedule disruption, if found, must consider all schedule synchronizations and should be executable and expressed in a common representation. The solution to the schedule disruption only introduces changes within planned buffers.

2.3.4. *Use case: collaborate in a monitoring process.* In order to carry out the monitoring task (performed by the Monitoring subsystem) it is necessary to obtain execution data and environment data. This information should be available in any CMDESC solution. We assign this responsibility to the Control subsystem which will provide the necessary functionality to collect the information as requested by the Monitoring subsystem.

2.3.5. *Use case: collaborate in a repair process.* Sometimes, in order to restore feasibility in a damaged schedule it is necessary to propagate changes towards either customer or providers orders. These changes are feasible only if the customer's and provider's schedules are analyzed together with the damaged schedule, to ensure synchronicity and execution feasibility. In this use case the Feasibility Manager requires collaboration from a Controller (in charge of controlling provider's or customer's schedule).

2.4. **Non-functional requirements.** The main non-functional requirements detected are described as follows.

2.4.1. Each of the functionalities provided by the subsystems of Monitoring, Feasibility Management and Control requires the ability to express and understand schedules. The use of a common schedule representation is required. This representation should be able to express schedules, disruptive events and schedule repair solutions, across all the collaborating subsystems.

2.4.2. SC members must be able to join or withdraw from the CMDESC business process performed by these subsystems dynamically, without affecting the performance and execution of this process.

2.4.3. Supply Chains implementing this CMDESC business process should be able to utilize one or more of the functionalities as desired and independently. For instance, an SC may need to utilize the Monitoring subsystem only.

The last two of these non-functional requirements impose a condition on the design of a CMDESC solution supporting its business process. This condition is that each of the main functionalities required for CMDESC should be provided by an independent system. Therefore, CMDESC needs to emerge from the collaboration of these systems.

3. **The CMDESC Business Process.** To satisfy the functional requirements above specified, following the methodology for SOA development [38], we define the Business Modeling phase proposing these functionalities to be provided by three participants: Controller, Monitor and Feasibility Manager. The collaborative CMDESC business process is graphically represented using BPMN [39] in Figure 1. This business process defines the necessary collaboration among participants and the tasks each them has to perform to provide the CMDESC functionalities to any SC implementing it.

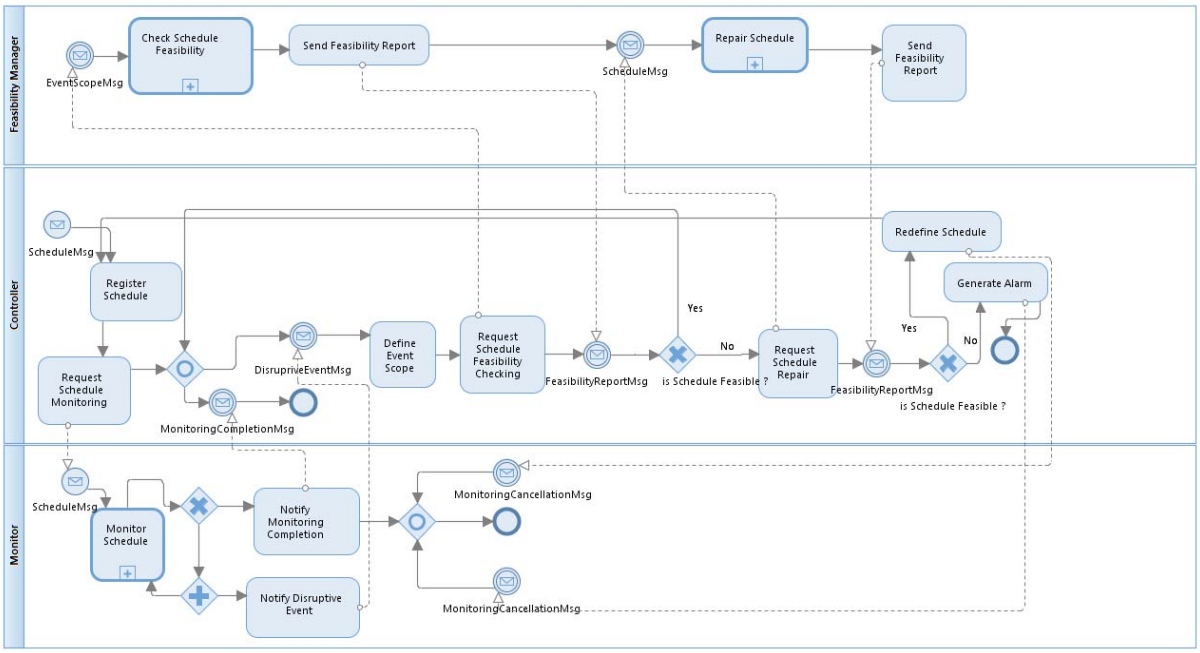

FIGURE 1. CMDESC collaborative business process

3.1. **Participants.** The actors identified in the requirement analysis are the participants of this business process. Each participant of this business process is described next.

3.1.1. *Controller.* This participant (Actors 2.2.1) is responsible for providing the functionality to control a production and/or distribution schedule (Functional Requirement 2.1.1). Each member of an SC may have one or more schedules. In this proposal each schedule is controlled by one Controller. The Controller is responsible for requesting monitoring function to the Monitor and for providing the access to updated data from the execution systems. It also interacts with the Feasibility Manager requesting the feasibility verification and schedule repair when a disruptive event is detected and engaging in collaborative repair when requested.

3.1.2. *Monitor.* This participant (Actors 2.2.2) is responsible for providing functionalities for monitoring a schedule (Functional Requirement 2.1.2). The execution of a schedule implies performing the operations defined in the supply process each order represents and is feasible if both the order requirements and the availability of the resources take their planned values. Therefore, to capture disruptive events, the Monitor should provide all

the four activities in the Use Case 2.3.1. Whenever possible, it is desirable to extend their proactive capabilities by using predictive models that allow anticipating the disruptive event through cause-effect relationships. The causal relationship can be modeled using Bayesian Networks, Petri Nets, Decision Trees, or similar approaches.

3.1.3. *Feasibility manager.* This participant (Actors 2.2.3) is responsible for providing functionalities for verifying the feasibility of a schedule when a disruptive event has occurred, and to repair a disrupted schedule requesting the collaboration of other Controllers if it is necessary (Functional Requirement 2.1.3). To perform these functionalities appropriate decision making models, such as mathematical programming and heuristic programming, are required.

3.2. **Tasks.** To carry out its functionalities each participant has to perform a set of tasks and sub-processes, which are described as follows. A detailed description of the messages involved by these tasks is presented in Appendix A.

3.2.1. *Register schedule.* This task represents the registration of the schedule that is going to be supervised by the Controller. The message *ScheduleMsg* contains a complete description of an ongoing execution schedule including the information needed to assess the feasibility regarding the scheduled requirements and resources.

3.2.2. *Request schedule monitoring.* A message to the Monitor participant is sent to request monitoring the Schedule. The message sent contains the previously registered Schedule.

3.2.3. *Notify monitoring completion.* Whenever the monitoring task completes the monitoring of a schedule, that is when all Supply Process Orders have been completed, a notification is sent by the Monitor to the Controller.

3.2.4. *Notify disruptive event.* Whenever the Monitor Schedule sub-process detects a disruptive event, the Monitor notifies this event to the Controller with a Disruptive Event Message.

3.2.5. *Define event scope.* An event scope contains all the required information necessary to evaluate its impact on an Execution Schedule. Given a *DisruptiveEventMsg* the scope is defined by all the assigned resources of every supply process affected by the disruptive event, plus all the supply processes scheduled in each resource affected by the disruptive event. This set of resources and supply process is the minimum required set to assess the impact of the disruptive event in the schedule. Any Controller can include this minimum set as the event scope, or augment the size of information shared accordingly to an internal policy up to defining the event scope as the entire execution schedule being supervised.

3.2.6. *Request schedule feasibility checking.* Once the scope of the event has been defined, a request is placed to Feasibility Manager to assess the impact on the schedule by determining whether it affects its execution feasibility or not.

3.2.7. *Send feasibility report.* After performing the feasibility verification, this task implies the sending of a Feasibility Report message back to the Controller. This report contains information regarding a successful search for feasibility and the changes introduced in the schedule to obtain it.

3.2.8. *Redefine schedule.* Upon the reception of a feasibility report stating the modifications to the current schedule, this task implements this changes and start the process of supervising a schedule with this redefined one, all over again.

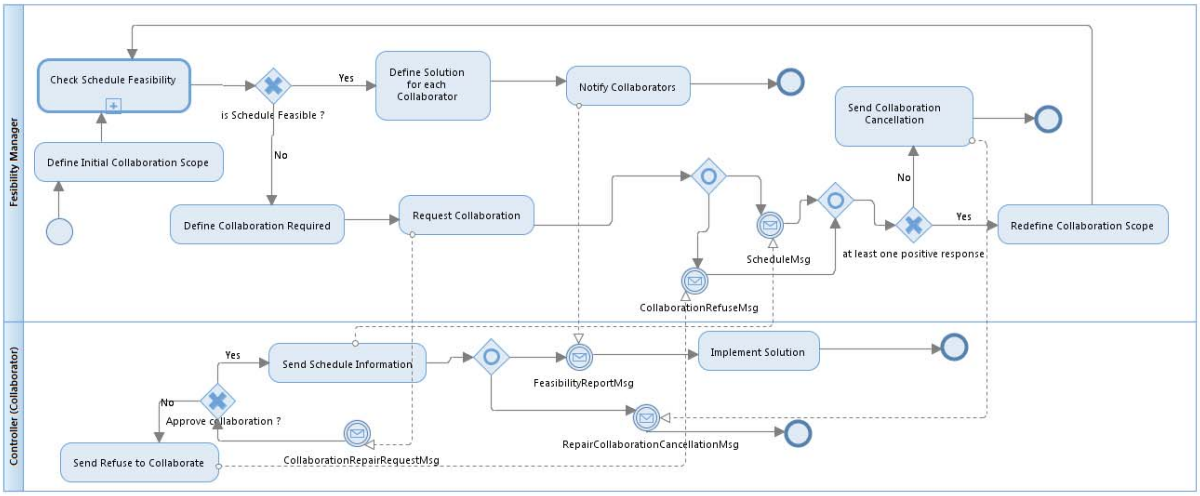

FIGURE 2. The repair schedule sub-process

3.2.9. *Generate alarm.* This task is only executed whenever a disruptive event could not be resolved by the Feasibility Manager. It implies sending a message to the Monitor in order to cancel the monitoring of the schedule and may include any other procedure, e.g., notify the Planning Systems.

3.3. **The repair schedule sub-process.** The Feasibility Manager participant has to perform the Repair Schedule sub-process, which is graphically represented in Figure 2. When a disruptive event occurs, this sub-process has to support the use cases (Use Cases 2.3.2 and 2.3.3) defined in the requirement analysis.

A key concept in this business process is the *Collaboration Scope*, which defines the set of supply process orders which can be modified during the search of feasibility and which cannot.

3.3.1. *Tasks.* The tasks that compose this business sub-process are described following and detailed descriptions of the messages involved by these tasks are presented in Appendix A.

Task 3.3.1.1. Define Initial Collaboration Scope. The initial collaboration scope establishes that all supply process orders are first considered as not-modifiable and therefore the feasibility is checked only considering whether the disruptive event can be absorbed by scheduled buffers in the resources.

Task 3.3.1.2. Define Collaboration Required. This task implies augmenting the analysis capabilities of the Feasibility Manager participant. In the process of searching for feasibility, if the repair cannot be accomplished with the current scope of collaboration, this task will attempt to expand the scope by adding new supply process orders and resources. This expansion procedure may eventually require the inclusion of supply processes orders and resources belonging to different schedules and owned by even different enterprises of the SC. Therefore, this scope expansion may trigger the need for new Controllers be invited to collaborate is the feasibility search.

Task 3.3.1.3. Request Collaboration. As a result of the collaboration scope expansion, new Controllers are requested to participate. This task will request this collaboration by sending a set of supply process orders and resources that the Controller should accept to become part of the repair process.

Task 3.3.1.4. Redefine Collaboration Scope. After receiving the acceptance and refusals of all the invited Controllers, this task will compose the new Collaboration Scope

agreed and will trigger a new feasibility search.

Task 3.3.1.5. Send Collaboration Cancellation. This task cancels the collaborative Repair Schedule sub-process with all Collaborators except with the one who originally sent the event Scope. This is produced when the repair process cannot reach a feasible solution and there is no possibility of further augmenting the Collaboration Scope.

Task 3.3.1.6. Implement Solution. This task includes everything a Controller in the role of Collaborator needs to do in order to accept a modified schedule defined by the Feasibility Manager during the collaboration process. For instance, the Controller should re-define the schedule, register the new schedule and request the monitoring with the updated schedule.

Task 3.3.1.7. Send Schedule Information. This task implies sharing the schedule information requested in the Collaboration Request Message.

Task 3.3.1.8. Send Refuse to Collaborate. This task represents the Collaborator declining the proposal to collaborate.

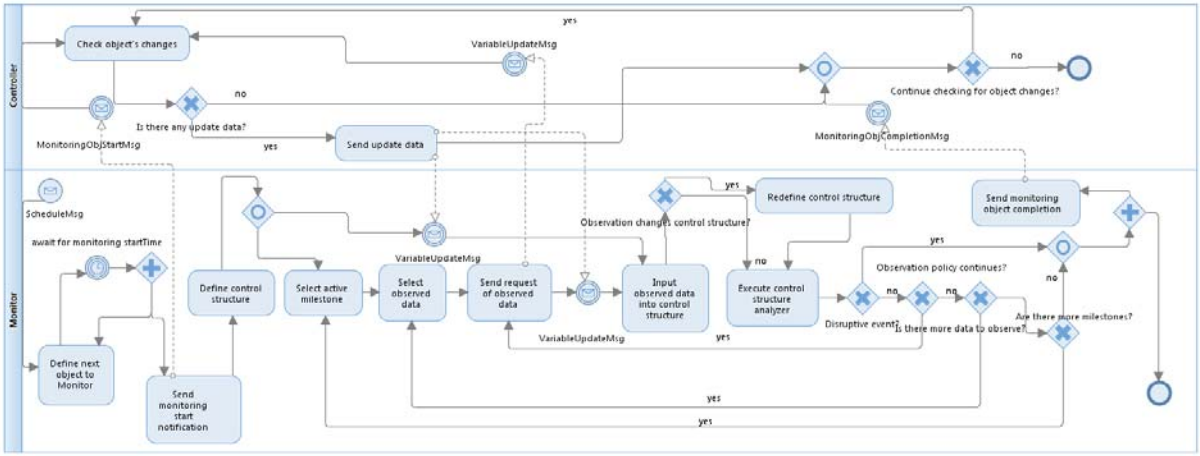

FIGURE 3. Monitoring schedule sub-process

3.4. **The monitoring schedule sub-process.** The Monitor has to perform the Monitoring Schedule sub-process, which is graphically represented in Figure 3. During the schedule execution, this sub-process has to support the four main monitoring activities identified in the requirement analysis (Use Case 2.3.1). They are grouped into two types: Activities that require monitoring an order (Activities 2.3.1.1 and 2.3.1.3) and that require monitoring a resource (Activities 2.3.1.2 and 2.3.1.4).

3.4.1. *Tasks.* The tasks that compose this business sub-process are described next and detailed descriptions of the messages involved by these tasks are presented in Appendix A.

Task 3.4.1.1. Check Object's Changes. This task determines whether there have been changes in an object (order or in a resource availability profile). In such case the Controller communicates updated object's data to the Monitor.

Task 3.4.1.2. Send Updated Data. The updated value of an environment variable or attribute variable is sent.

Task 3.4.1.3. Define next Object to Monitor. This task iterates over all the objects in the schedule (orders and resources) and defines the corresponding monitoring activity associated to the object. Every activity is assigned with time window which will be used

to activate the monitoring at the proper time.

Task 3.4.1.4. Send Monitoring Start Notification. The Controller is notified that the monitoring process associated with a resource or an order has started.

Task 3.4.1.5. Define Monitoring Structure. For each order or resource its monitoring structure is instantiated and completed with planned values.

The monitoring structure for monitoring an order has variables to capture changes in the requirements of an order (start time, end time and quantity) and variables to predict future modifications of the planned values of the order. The monitoring structure for monitoring a resource has variables to capture changes in the current value of the attributes defining a resource's availability and variables to predict modifications of the planned values of its future availability.

Task 3.4.1.6. Select Active Milestone. A milestone defines a point where a set of variables associated with the monitoring structure will be observed to evaluate the execution progress of an order or the availability of the resource.

Task 3.4.1.7. Select Observed Data. Each variable associated with a milestone is observed according to the relation of precedence. In this task, the variable to be observed is selected.

Task 3.4.1.8. Send Request of Observed Data. The update request of the variable (selected in Task 3.4.1.7) is sent to the Controller.

Task 3.4.1.9. Input Observed Data into Monitoring Structure. The observed value is retrieved and inserted into the monitoring structure to evaluate the impact of this variable.

Task 3.4.1.10. Execute Monitoring Structure Analyzer. The analysis function to assess if a disruptive event may occur is executed.

Task 3.4.1.11. Redefine Monitoring Structure. When a variable is observed, the branch of the monitoring structure which predicted its value is no longer necessary and can be eliminated.

Task 3.4.1.12. Send Monitoring Object Completion. The Controller is notified that the monitoring process associated with a resource or an order has finished. In this case, no disrupted event occurs during the monitoring process.

4. SOA of the CMDESC System.

Service oriented computing and architecture is the technology chosen to enact the CMDESC business process. It provides a way to create software artifacts that supports requirements on heterogeneity and autonomy that arise on current supply chain management practices. Those SOA artifacts are tied to the business process requirements that they capture. The collaborative nature of this proposal for CMDESC is benefited by adopting SOA technologies as it promotes few coarse grained interactions between service providers and consumers. Additionally an SOA architecture definition is platform independent allowing its implementation on business partners with diverse co-existent technologies.

In Section 3, a collaborative business process that meets the requirements realized by the use cases in Section 2 has been defined. In this section, we present a service-oriented architecture that was developed to give systems support to the mentioned business process. To identify all the capabilities and services required to enact the business process we follow a standard modeling technique (SOMA) [38].

The service model is designed starting with the capabilities each participant in the business process has to provide to enact the collaboration. After that, the services exposing these capabilities are defined.

In this proposal we adopted a document-centric approach to support the access to service operations; therefore, for each operation defined in a service there is an associated

document containing all the information required to provide the service. These documents are specified using the *messageType* artifact from the SOAML specification [40] and their structure is detailed in Appendix A.

In order to define the messages and its information content in a consistent and complete way, we have used a Reference Model for Disruptive Event Management. This model provides self-contained descriptions of any on-going execution schedule of supply process orders with all the information required to assess its feasibility.

4.1. **Capabilities and services provided by the controller.** In order to enact the functionalities described in the CMDESC collaborative business process diagram in Figure 1, the controller provides three capabilities: *Execution Control, Collaborative Repair and Collaborative Monitoring* and they are depicted in Figure 4. The operations offered by these capabilities are exposed trough the following service interfaces.

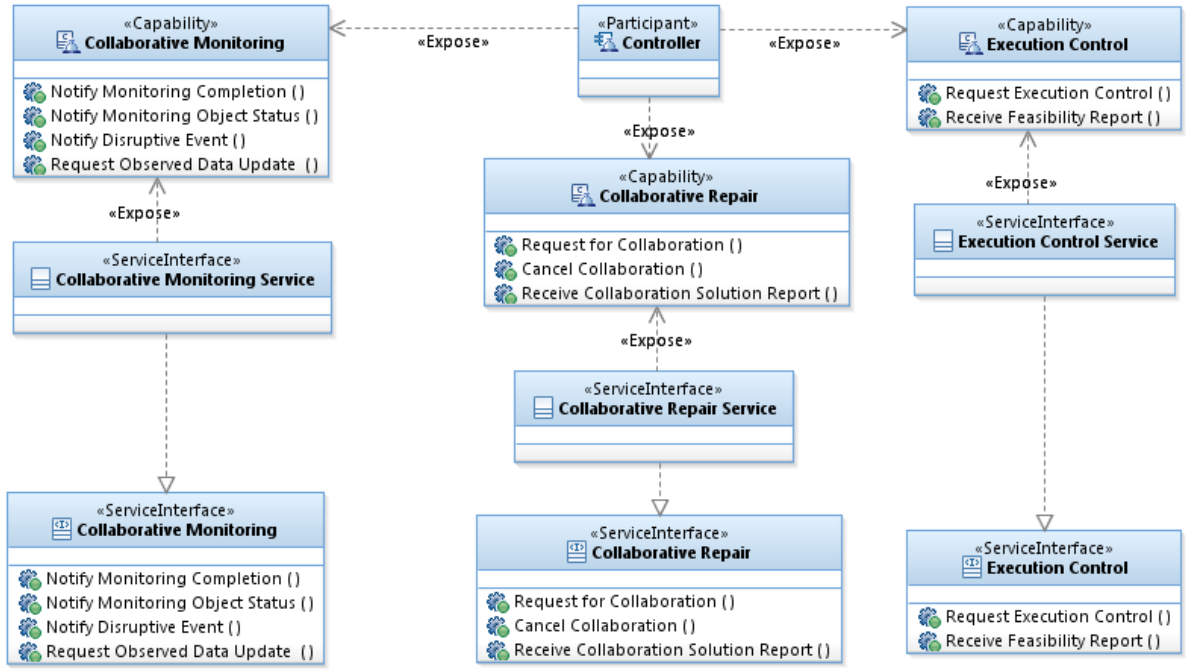

FIGURE 4. Capabilities exposed by the controller

4.1.1. *<<ServiceInterface>> execution control.* This service offers the following operations:

Request Execution Control (executionSchedule: ScheduleMsg): This operation is used to place a request for a schedule execution control. The message associated to this operation contains an execution schedule defined by any planning or scheduling system, which is sent to the Controller as part of an execution control request. Figure 19 shows the ScheduleMsg message structure.

Receive Feasibility Report (feasibilityReport: FeasibilityReportMsg): The message associated to this operation contains all the information related to the feasibility status of a schedule after a collaborative repair process has been performed, or after checking schedule feasibility, in order to communicate the result. Figure 19 shows the *FeasibilityReportMsg* message structure.

4.1.2. *<<ServiceInterface>> collaborative repair.* This service offers the following operations:
Request for Collaboration requestForCollaborationMsg: CollaborationRepairRequestMsg): Used by the Feasibility Manager participant for requesting the Controller to engage in a collaboration to repair a damaged schedule. A request for collaboration is specified using the message described in Figure 19. It contains the scheduling entities which are needed to enable a collaborative repair process, and the detail of the repair collaboration in course.
Cancel Collaboration (collaborationCancellationMsg: RepairCollaborationCancellationMsg): Used by the Feasibility Manager to inform a repair collaboration cancellation.
Receive Collaboration Solution Report collaborationSolutionReport: FeasibilityReportMsg): Used by the Feasibility Manager to send the result of successful repair collaboration. A collaboration solution report is also a *FeasibilityReportMsg* (Figure 19), showing the result of a repair collaboration and the changes introduced.

4.1.3. *<<ServiceInterface>> collaborative monitoring.* Offers the operations exposed by the Controller in order to enact the functionalities described in the Monitoring Schedule sub-process diagram (Figure 3). The operations are:
Notify Monitoring Completion (monitoringCompletion: MonitoringCompletionMsg): The message associated to this operation has the identification of the schedule which completed its monitoring, in order to communicate the completion of the monitoring process of the schedule.
Notify Disruptive Event (disruptiveEvent: DisruptiveEventMsg): This message (Figure 19) contains a new specification of the resource or order affected by a disruptive event. It is shown in the diagram class through of the relations *newResourceSpecification* and *newSPOSpecification*. This message may be sent to the Controller if after updating the observed data in the control structure a disruptive event is detected.
Request Observed Data Update (requestDataUpdate: Variable UpdateMsg): This message contains information about an attribute variable or an environment variable. Each variable has a timeStamp attribute which represents when the data is required.
Notify Monitoring Object Status (notifyObjectStatus: MonitoringObjectStatusMsg): Used to notify the Controller about the monitoring status of an object (start or completion).

4.2. **Capabilities and services provided by the monitor.** In order to enact the functionalities described in the in the Monitoring Schedule sub-process diagram in Figure 3, the Monitor provides three operations exposed through the service interface depicted in Figure 5.

4.2.1. *<<ServiceInterface>> monitoring.* This service offers the following operations:
Request Monitor Schedule (schedule: ScheduleMsg): This operation is used by the Controller participant to request the monitoring of a schedule. The message associated to this operation contains a schedule which is sent to the Monitor, in order to be monitored. Figure 19 shows the *ScheduleMsg* message structure.
Cancel Monitoring (monitoringCancellation: MonitoringCancellationMsg): This operation is used by the Controller participant to communicate the monitoring process of a schedule has ended. This message contains the identification of the schedule that must cancel its monitoring process. It is shown in the CMDESC collaborative business process diagram (Figure 1).
Update Observed Data (updateObservedDataMsg: VariableUpdateMsg): This operation is used by the Controller participant to send an update on the values of some data.

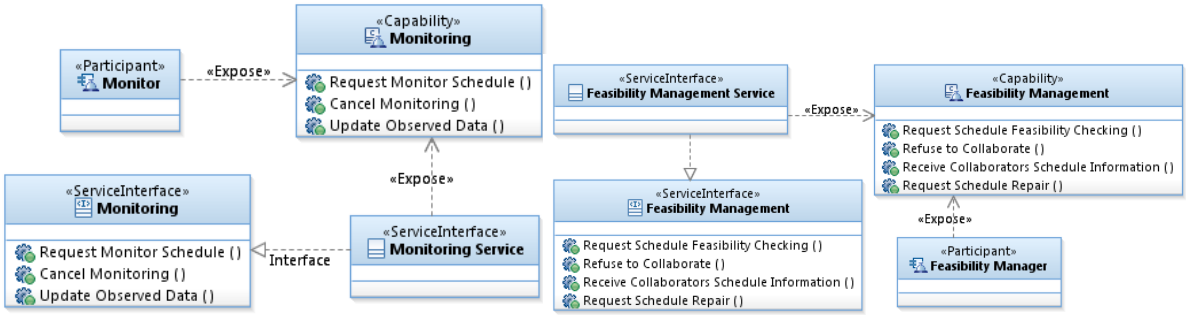

FIGURE 5. Capabilities exposed by the monitor

FIGURE 6. Capabilities exposed by the feasibility manager

4.3. **Capabilities and services provided by the feasibility manager.** In order to enact the functionalities described in the Repair Schedule Sub-process diagram in Figure 2, the Feasibility Manager provides four operations exposed through the service interface:

4.3.1. *<<ServiceInterface>> feasibility management.* This service offers the following operations depicted in Figure 6.
Request Schedule Feasibility Checking (eventScope: EventScopeMsg): This operation is exposed so that a Controller can request the assessment of feasibility in a given schedule, or only on the scope of an event. The message associated to this operation contains the information resulting from the Define Event Scope task in the CMDESC collaborative business process diagram (Figure 1). The *EventScopeMsg* is defined in Figure 19.
Refuse to Collaborate (collaborationRefuse: CollaborationRefuseMsg): This operation allows a possible collaborator in a collaborative repair process (a Controller), to refuse to collaborate. The message associated to this operation captures the refuse to collaborate from a Controller as shown in the Repair Schedule Sub-process diagram (Figure 2). The refuse details in the message have the code of the collaboration request which is refused.
Receive Collaborators Schedule Information (scheduleInfo: ScheduleMsg): This operation allows a collaborator to send its schedule information to a Feasibility Manager in order to participate in a Collaborative Schedule Repair. The message in this operation is a schedule or a portion of a schedule and is shown in Figure 19.
Request Schedule Repair (eventScope: EventScopeMsg): This operation allows any controller calling it, to request the repair of a Schedule. This message is shown in Figure 19.

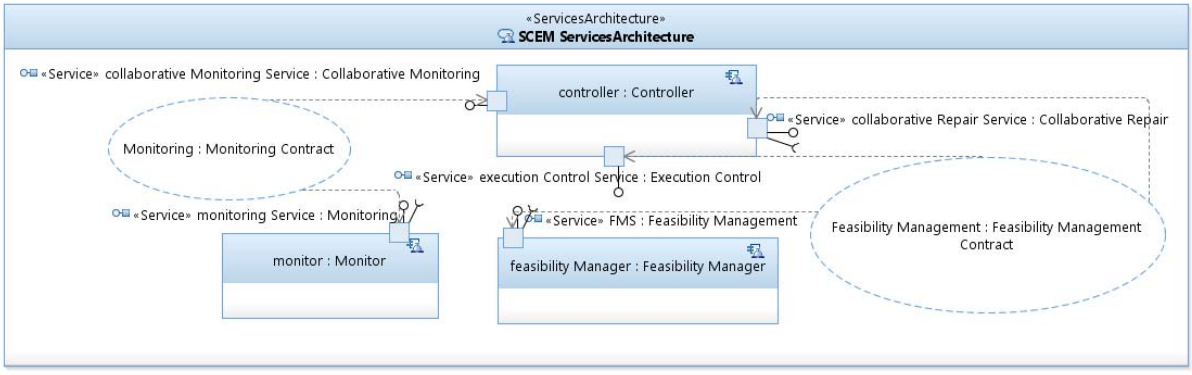

FIGURE 7. CMDESC service architecture

The final architecture resulting from organizing the participants and services through specific service contracts is graphically represented in Figure 7.

Two service contracts are defined: Feasibility Management and Monitoring. These contracts contains all the metadata regarding the description about the CMDESC SOA based solution services that is: the purpose and function of their operations, defined in service identification (Section 4.1); the messages that need to be exchanged in order to engage the operations, defined in the service interface specification (Section 4.2) and the data models used to define the structure of the messages, defined in the service data model (Appendix A).

5. **Reference Models.**

5.1. **The disruptive event management reference model.** A full description of this reference model can be found in [41]. It is relevant to emphasize that an instance of this Reference Model is a self-contained description of the feasibility of an execution schedule that can be automatically transformed into a Constraint Satisfaction Problem (CSP). This CSP, checks the feasibility of the execution schedule, and also gives the possibility of identifying slacks in order to device a repair mechanism with minimum modifications, as the one presented here. Following the two main modeling views of this Reference Model are described.

5.1.1. *Modeling supply processes.* In this reference model any instance of a supply process scheduled to be executed is described by an order coming from a Scheduling or Planning System (Materials Requirement Planning, Enterprise Resource Planning, Distribution Resource Planning, etc.). These orders can be of different types: transfers, production, shipments, etc. In this model all of them are described as generic *SupplyProcessOrder* objects.

A relevant aspect of the model is the ability to capture the feasibility of execution. Therefore, the concept of Resource is introduced to represent every resource that is required by a *SupplyProcessOrder* in order to be executed. For these resources it is desired to efficiently describe their availability in order to trace its feasibility to meet the requirements of an order.

Resources used in general supply processes can be of very different nature: transportation equipment, production units, storage facilities, and also materials including work in progress, etc. In order to generalize the ability for a resource to satisfy a supply process requirement, the concept of *FeasibilityDimension* is introduced.

A *FeasibilityDimension* is a characteristic of a Resource describing its capability to fulfill a requirement from a *SupplyProcessOrder*. The availability of the resource is therefore conditioned by them and every requirement for the resource should be expressible in terms of its feasibility dimensions.

Any ongoing execution *Schedule* is described as a net of *Resources* linked by *SupplyProcessOrder*. This structure captures the relationships for describing the propagation of a disruptive event, allows monitoring and controlling disruptive events and possible exceptions at their origin, communicating disruptions and exceptions to the proper receptor. In this work, the receptor will be one of the Controller participants. Using this representation the disruption propagation and impact can be assessed as different SCs and different business partners in a single SC are represented as supply process orders and resources having different Controllers.

Figure 8 presents an UML class diagram representation of a general supply process. Linked supply processes conform a net of resources and supply process orders. In the class diagram, a supply process is defined, through a *SupplyProcessOrder*, which is composed by

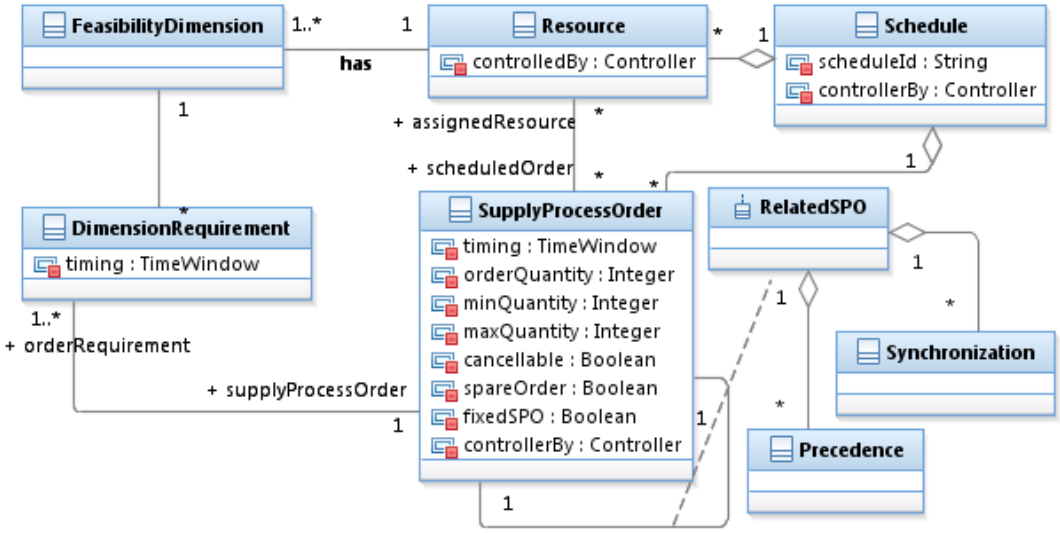

FIGURE 8. UML class diagram of a general supply process

a set of *DimensionRequirements* imposed to every *FeasibilityDimension* of the resources assigned for the execution of the supply process order. When two supply process orders belong to different business partners, or even to different SCs their relation is captured by the association class *RelatedSPO*, which implies relationships between the two supply process orders, such as the same *orderQuantity* and the same *timing*.

5.1.2. *Describing resources by feasibility dimensions.* In order to assess the availability of a resource, the feasibility of its schedule and to evaluate the effects of disruptive events, it is necessary to describe the resources. A possible attempt is to classify resources by types as in [42], but this has the drawback that resources in an SC can be quite diverse, a more generic and extensible characterization is needed.

As the purpose is to model the availability of resources and the feasibility of its scheduled supply process orders, the characterization of resources by introducing the concept of feasibility dimensions is proposed (Figure 9).

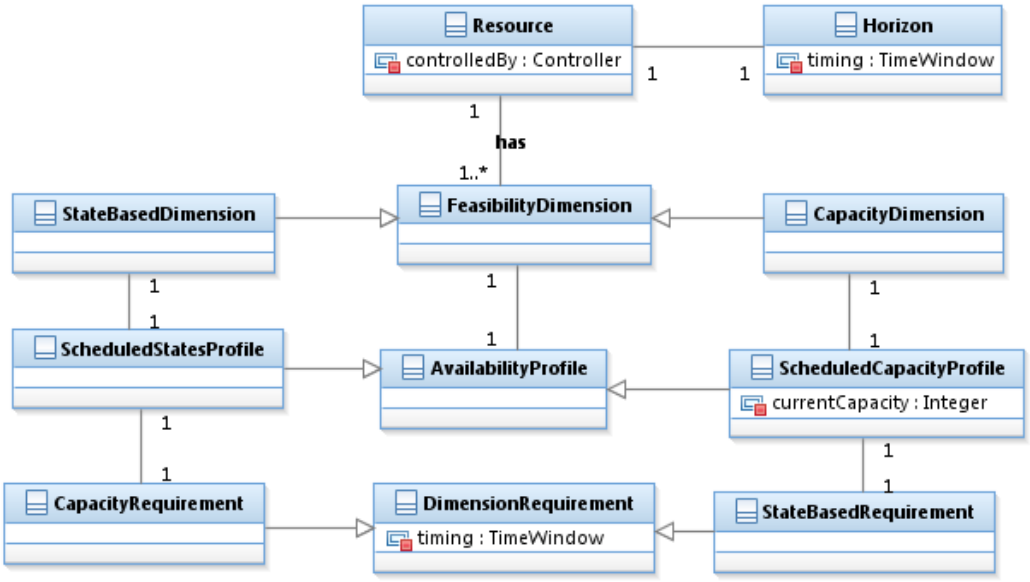

FIGURE 9. Feasibility dimension of a resource

Two types of feasibility dimensions are defined: Capacitated Dimension and State Based Dimension. Each one of them has an *AvailabilityProfile* that describes its intrinsic and planned availability in a given *Horizon*.

Every *SupplyProcessOrder* has a set of requirements over the resources assigned for its fulfillment. These requirements should be expressed according to the availability of each resource, which is expressed in feasibility dimensions. The concept of *Dimension-Requirement* to define all the possible uses of resources is introduced, in correspondence with each type of feasibility dimension. Therefore, there are two types of requirements *StateBasedRequirement* and *CapacityRequirement*.

5.2. **The monitoring reference model.** A full description of this reference model can be found in [43]. It is relevant to emphasize that an instance of this Reference Model is a self-contained description of the monitoring and monitoring structure of an order or a resource, which can be automatically transformed into a particular cause-effect model. The UML class diagram in Figure 10 presents the monitoring model which has a monitoring structure based on a *cause_effect* relationship among *Variables*. These variables represent *AttributeVariable* (resource or order specifications) or *EnvironmentVariable* affecting a resource or a supply process order.

The monitoring structure has a set of *Milestones*. Each milestone defines a point where a set of variables will be observed. Each *Variable* of the monitoring structure has one State that can be *ObservedState* or *EstimatedState*. When the state is *ObservedState*, the *Variable* is observed and its value is given. A *Variable* has an observation policy. An *ObservationPolicy* defines the mode, the recurrence and the updating time of the observed variable. When the state is *EstimatedState*, the Variable value is estimated from the value of other variables using the *cause_effect* relationship net. To perform this task, the *MonitoringStructure is_analysed_by* a *MonitoringStructureAnalyzer*.

The *TargetVariable* has a *planned Value* that is an order specification (for example, amount and end time) or a resource parameter (for example, geographical positions planned and production rate planned of the machine). Also, the target variable must have an *estimated Value* which is defined by a *MonitoringStructureAnalyzer* or an observed Value which is a given value of a variable in the monitoring structure.

The *TargetVariable* is used to evaluate if a disruptive event can occur. This is done by evaluating conditions between the estimated/observed value and the planned one. When the disruption condition is verified, a *DisruptiveEvent* is reported.

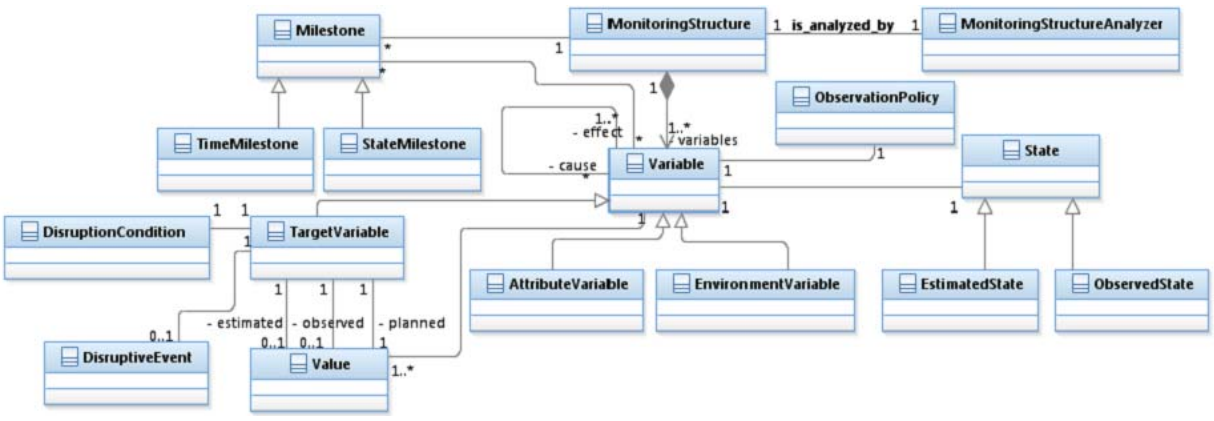

FIGURE 10. UML class diagram of a general monitoring model

6. Service Implementation.

6.1. **Feasibility manager service implementation.** In Figure 11, an UML class diagram shows the implementation design. As described in the business processes of Section 3, a Feasibility Manager is proposed to evaluate the feasibility of a schedule and to collaboratively repair a disrupted schedule. Any schedule described using the Reference Model for Disruptive Event Management (Section 5.1) can be automatically transformed into a Constrain Satisfaction Problem (CSP). This is possible because the feasibility of every order is captured as generic requirements on two types of resource dimensions: capacity based and state based. We have derived appropriate constraint templates for each type of dimension. The resulting Schedule Feasibility Problem is composed by variables and constraints on these variables that determine whether a set of *DimensionRequirements* of the supply process orders can be fulfilled by the *assignedResources*. By allowing modifications to resource's planned usage and the attribute values of a supply process order, different mechanisms for solution searching can be derived.

The Repair Schedule sub-process (Section 3.3) and the CMDESC collaborative business process (Section 3) define three private operations (Figure 11) for the Feasibility Manager participant: Define Collaboration Scope, Check Schedule Feasibility and Generate Feasibility Report.

6.1.1. *Operation: check schedule feasibility.* This is implemented by transforming a portion of a schedule (a disruptive event scope), or a complete schedule into a Schedule Feasibility Problem using a Model Driven Development approach [44, 45]. This approach is materialized by a transformation engine which holds the precise mapping between an instance of the Reference Model for Disruptive Event Management (Section 5.1) (used to define schedules and schedule related artifacts in this work) and the necessary constraints and variables to determine the Schedule Feasibility Problem.

This operation is also used to repair a disrupted schedule by the definition of suitable domains for the variables in the scheduling feasibility problem. These domains imply modifications using only planned buffers.

The communication implied by the association between the Feasibility Manager and the CSPSolver in Figure 11 is implemented by two documents, UML artifacts, an XML

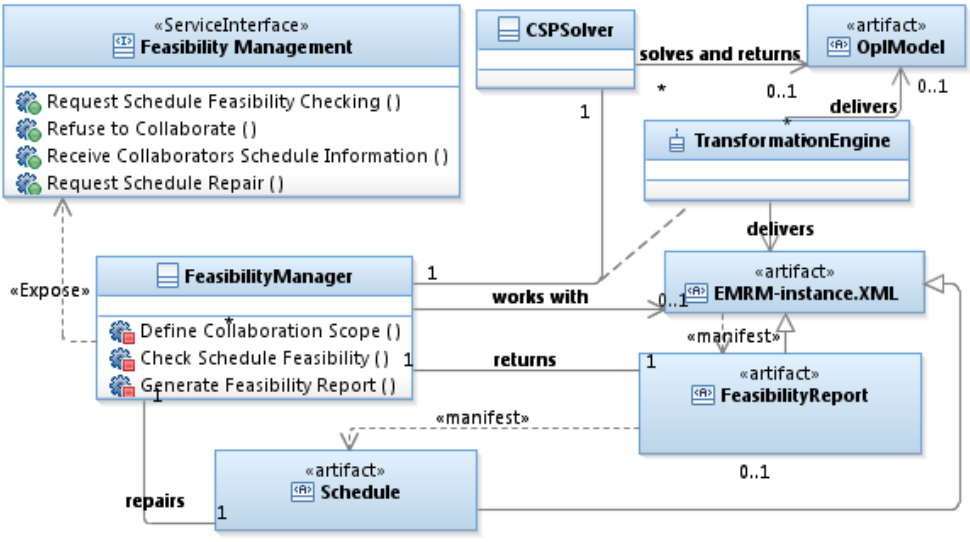

FIGURE 11. UML class diagram of the implementation design

instance of the Disruptive Event Management Reference Model *EMRM.XML*, compliant with the XMLSchema (EMRM.XSD) of the reference model; the other artifact is a *OplModel* instance, the CSP representation in the IBM ILOG OPL Development Studio API [46], used as the engine to solve the Schedule Feasibility Problem.

Whenever a successful feasibility search is done, the solution is communicated using the transformation engine, converting the solution of a CSP into a new specification of resource usage and planned requirements of a schedule expressed using the EMRM.

6.1.2. *Operation: define collaboration scope.* This operation determines what additional information is required to augment the search space of the Schedule Feasibility Problem. The additional information in terms of a schedule is: Resources and Supply Process Orders some of them suitable to be modified. An order can be modified, only if every of its assigned resources belong to the collaboration scope. A Resource usage can be modified only if all its scheduled orders belong to the collaboration scope. A radial expansion mechanism was chosen in the implementation of this operation, if feasibility is not found with a given collaboration scope, it is augmented in a radial fashion, allowing modification to previously fixed orders. Resources and orders are controlled by different participants, so in this operation it is also determined who is going to be asked to collaborate. By modifying the orders in this set, a radial expansion is done including the orders associated to the resources in the current collaboration scope.

6.1.3. *Operation: generate feasibility report.* This operation analyzes which resources and orders have been modified after a feasibility search. It is used to generate the further message called *FeasibilityReportMsg* detailed in Appendix A.

6.2. **Monitor service implementation.** In Figure 12, an UML class diagram shows the implementation design. As described in the Monitoring Schedule sub-process (Section 3.4) the Monitor is responsible of monitoring the execution of a schedule. The Monitoring Reference Model (Section 5.2) allows the definition of all entities required for an automatic derivation of a monitoring structure that will be used by the Monitor for deciding which variables are being observed and compared at any time. This structure may include estimated variables whose values can be inferred from causal relationship with other observed variables. For these cases, the sub-structure corresponding to the cause-effect network is transformed to a model representation suitable to be processed by the Bayesian Network

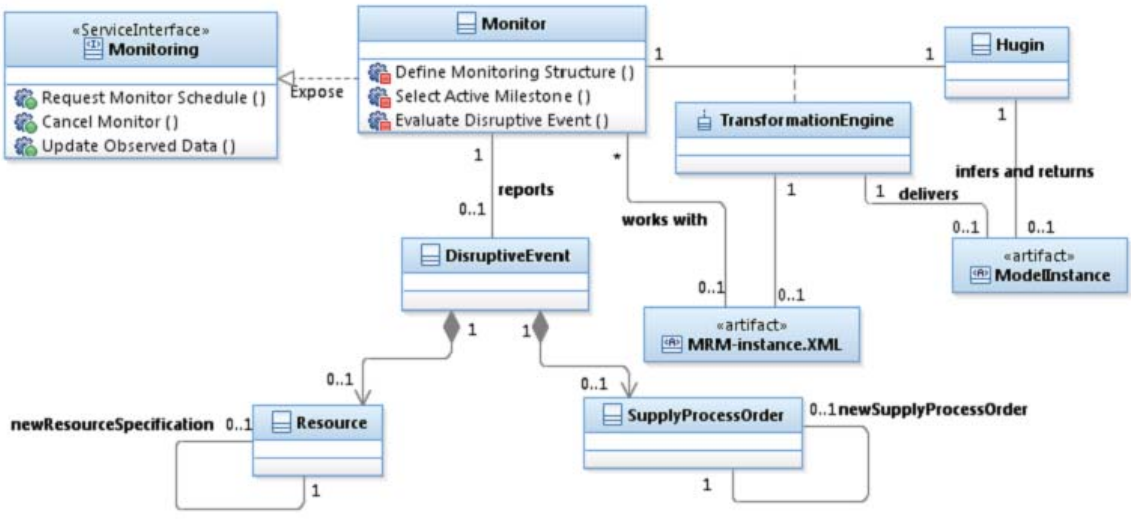

FIGURE 12. UML class diagram of the implementation design

inference engine. The Monitoring Schedule sub-process (Section 3.4) defines three private operations (Figure 12) for the Monitor participant: Define Monitoring Structure, Select Active Milestone and Evaluate Disruptive Event. The implementation of these operations for each kind of monitoring activity identified in the Use Case 2.3.1 is described next.

6.2.1. *Operation: monitoring order specification changes.* The monitoring structure associated with this activity defines three target variables for each SPO. Each variable has a planned value and an observed value. The planned value corresponds to an attribute of the SPO (start time, end time or quantity). By default, the Monitor has two milestones to evaluate if a disruptive event may occur in an order. These are schedule start, order start. Once the schedule start milestone is activated, the monitor will capture any change in the planned values for the order target variables (i.e., changes in the order specification) and when the change is assessed as significant, according to a threshold, the disruptive event is concluded. When the order start milestone is activated, the actual start time will be observed and compared with the planned value to evaluate a possible disruptive event. After this milestone, this activity finishes.

6.2.2. *Operation: monitoring current status of resource feasibility.* The monitoring structure associated with this activity defines a target variable for each *FeasibilityDimension* associated with a resource. Each variable has a planned value which represents an expected value of the corresponding feasibility dimension of the resource. It also has an observed value (current value). The Monitor has a milestone for each *DimensionRequirement* where the current value is observed and compared against the planned value to evaluate the occurrence of a disruptive event in that dimension.

6.2.3. *Operation: monitoring order progress.* The monitoring structure associated with this activity in general will depend on the type of process since complex cause-effect relationship among variables may be introduced to improve the predictive capabilities of a disruptive event. As the order execution may follow different stages, different milestones can be defined to identify those stages. Each milestone will introduce the relevant variables to be considered in the stage. The Monitor is responsible for observing the variables at each milestone, updating their values in the monitoring structure and evaluating the impact of these variable values on the estimated value of the target variables using the inference engine of the Bayesian Network, to evaluate if a disruptive event can occur.

Unless an specific structure is designed for the process, a default structure is generated having two milestones: order start and order end, two target variables: corresponding to the order end time planned and order amount planned, and one observed variable indicating a measure of the order progress. This measure is used to infer estimated values for the target variables and anticipate a disruptive event.

6.2.4. *Operation: monitoring changes on resourceś future expected availability.* The monitoring structure is associated with the *AvailabilityProfiles* of the resource. For each capacity profile item there are two target variables in correspondence with the planned maximum and minimum capacity bounds. For each item in the state profile the planned value is a given state. The monitoring structure will have one milestone for the schedule start and one milestone for each item in the availability profile indicating its start time.

Once the schedule start milestone is activated, the monitor will capture any change in the planned availability values and when the change is assessed as significant, according to a threshold, the disruptive event is concluded. When every availability item milestone is activated, the actual availability will be observed and compared with the planned value to evaluate a possible disruptive event.

The Monitor uses two documents, defined as UML artifacts in Figure 12. One is an XML instance of the Monitoring Reference Model MRM.XML, compliant with the XMLSchema (MRM.XSD) of the reference model. The other artifact, named *ModelInstance* is used whenever a cause-relationship network is being used to predict values for the estimated variables and is a representation of the Bayesian Network suitable to be processes by the inference engine [44].

7. **Case Studies for Validation.** Empirical validations of the CMDESC participants have been carried out. To validate the Monitor participant, three case studies have been carried out, and the supply processes modeled are a cheese production process of a dairy industry [41], a castings production process of a molding industry, and a marine freight transport process.

Based on the Monitoring Reference Model (Section 5.2) the cause-effect relationship models of the Monitor participant have been generated. Particularly these are probabilistic models based on Bayesian network for prediction functions. To process these models the Monitor participant uses the inference engine of [47]. In each case the ability of the Monitor to anticipate disruptive events has been evaluated.

To validate the Feasibility Manager participant, a commodity chemical supply chain has been used as case of study. Based on the Monitoring the Schedule Feasibility Problems are built using a Model Driven Development [44, 45] transformation described in Service Implementation section. Following, a case study is described.

7.1. **Case study for a commodity chemical supply chain.** The control actions are generated using the abilities of the Feasibility Manager participant. Schedules are described using the same reference model, and exception detection and repair actions are fulfilled using the Repair Mechanism.

In this supply chain a fertilizer (Urea) is produced, warehoused and distributed to three geographically distant distribution centers (DCs). The factory warehouse is located in Bahia Blanca (FWBahiaBlanca), Argentina, and DCs are: "Urea-DCSanLorenzo" at San Lorenzo, Argentina; "Urea-DCUruguay", at Montevideo, Uruguay; "Urea-DCBrasil", at Rio Grande, Brasil. The distribution centers are sourced by means of dedicated ships through fluvial and maritime routes. Table 1 presents the average trip times in hours.

TABLE 1. Average trip times in hours

	Urea-DCSanLorenzo	*Urea-DCUruguay*	*Urea-DCBrasil*
FWBahiaBlanca	96	144	168
Urea-DCUruguay	–	–	60

A Distribution Resource Planning system is used to generate a distribution schedule for a scheduling horizon of 33 days. In this schedule, product availability in *FWBahiaBlanca* is considered to be unlimited, this means that stock, demand and supply is managed for each distribution center attending constraints regarding to: Ships routes and availability, loading dock availability at factory warehouse, and inventory size and safety stocks constraints at each DC.

7.1.1. *Resources and their availability.* In Figure 13 the projected available inventory of urea is shown for distribution centers Brasil, based on the outcome of the DRP. As seen on the figure there are two replenishments for DC-Brasil. These replenishments imply coordination and timing for the involved resources. Using the reference model replenishments are modeled as transfers from Bahia Blanca to each DC, using the corresponding ship,

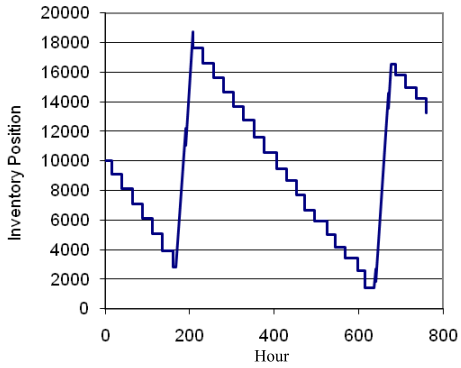

FIGURE 13. Urea-DCBrasil projected available inventory

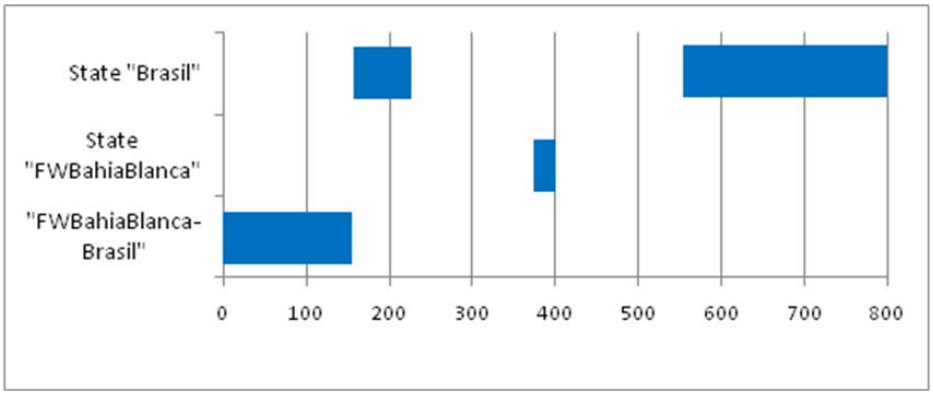

FIGURE 14. Ship-DCBrasil scheduled states profile

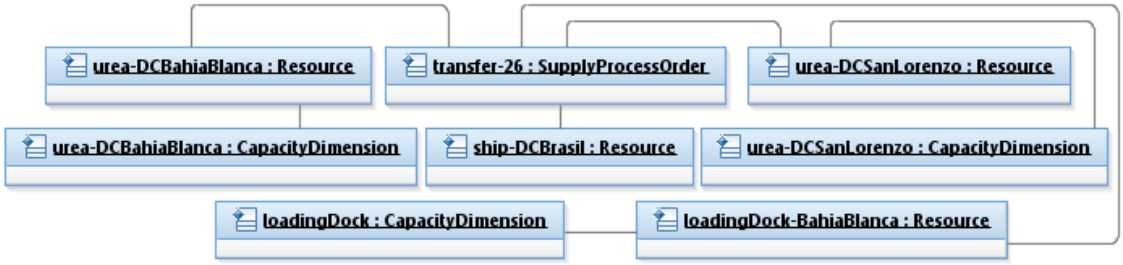

FIGURE 15. Transfer-26 supply process order

loading dock and inventory resource at DC, like the supply process *transfer-26* depicted in Figure 15. This figure also shows how the implied resources are modeled.

Ships are coordinated with regards to its capacity and geographical position, Figure 14 shows the geographical position (a *stateBasedDimension* of every ship Resource) in form of a Gantt chart (representing the *ScheduledStatesProfile*) for ship Ship-DCBrasil along scheduling horizon.

Inventory resources in this case of study, modeled with a single *CapacityDimension*, only have a maximum capacity constraint captured in the *scheduledCapacityProfile*. There are three of them, one for each DC, with a maximum capacity of 30000, 20000, 20000 tons in San Lorenzo, Uruguay and Brasil respectively. The loading dock in this case of study acts as a renewable resource, and the requirements it attends are of type Renewable and its projected availability can be seen in Figure 16.

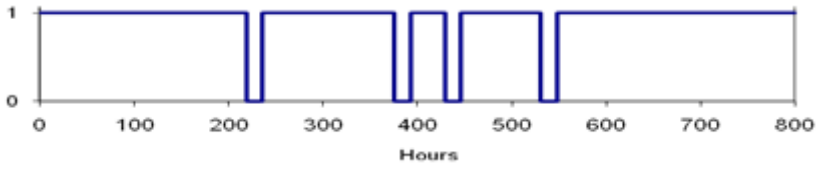

FIGURE 16. loadingDock-BB ProjectedAvailableCapacity

7.1.2. *SupplyProcesOrders and their specification.* Each DC has a list of Supply Process Orders (*SPO*) (customers' orders) to serve, which for this example are set together as a daily shipment for each of the 33 days. Each one of these *SPO* imposes a *CapacityRequirement* on the corresponding inventory resource.

In this case, the shipments from each DC have the following scheduling/rescheduling policy: Each order has a time window for its dispatch, if it was originally scheduled for day d, in case of a repair procedure it cannot be dispatched outside the week that contains day d. Some orders allow quantity modifications buy they cannot exceed 10 percent of the original value.

The replenishments to the DCs, that is transfers SPOs, are scheduled with a specific timing in the scheduling horizon, but, in case a of a repair procedure, their timing specification can be changed as long as resources capacities (inventories and ship capacity) and states (geographical position of ships) allow these changes. Transfer quantities are only limited by ships.

At the factory warehouse, the ships are loaded at a constant rate of 1000 tons/hour. And in DCs ships are downloaded at a constant rate of 250 tons/hour in San Lorenzo and Uruguay, and at 425 tons/hour in Brasil.

In exceptional situations the supply from Bahia Blanca to Brasil can be done by a 3PL (third party logistics provider) that by contract provides a transportation capacity of 17000 tons with a delivery time of 7 days. This optional process is modeled as a *spareOrder* (also *cancellable*) that requires for its execution resources loadingDock-BB (to load Urea) and Urea-DCBrasil (to download urea at Brasil).

7.1.3. *Predict and detect disruptive events.* The monitoring function performs four main activities during the execution of a schedule (Use Case 2.3.1). These are: monitoring order specification changes, monitoring current status of resource feasibility, monitoring order progress and monitoring changes on resource's future expected availability. The monitoring structure is generated using the Monitoring Reference Model (Section 5.2).

In this supply process, the navigation conditions of the ship can be unfavorable due to the weather conditions (storms, winds, etc.). These unfavorable weather conditions are more frequent in the winter season and can produce a delay in the ship arrives to the port. The delay can be increased if in the arrival port or in the intermediate ports there are unfavorable weather conditions or the port is congested. This prevent to carry out unload operations. The Ship-DCBrasil can carry orders to ports on Uruguay and Brasil. The MonitoringStructure for monitoring the progress of this maritime transport order is graphically represented in Figure 17.

7.1.4. *Unexpected events in the case study.* To illustrate the capabilities of the Feasibility Manager in the case study two different scenarios generated due to two disruptive events notified by the Monitoring System are considered. They are the resource Ship-DCBrasil becomes unavailable, and unexpected increase in demand at DC-Uruguay has occurred.
Scenario 7.1.4.1. Resource Ship-DCBrasil becomes unavailable. Ship-DCBra- sil becomes unavailable since day 17 (400 hours in the timeline) and onwards. From visual

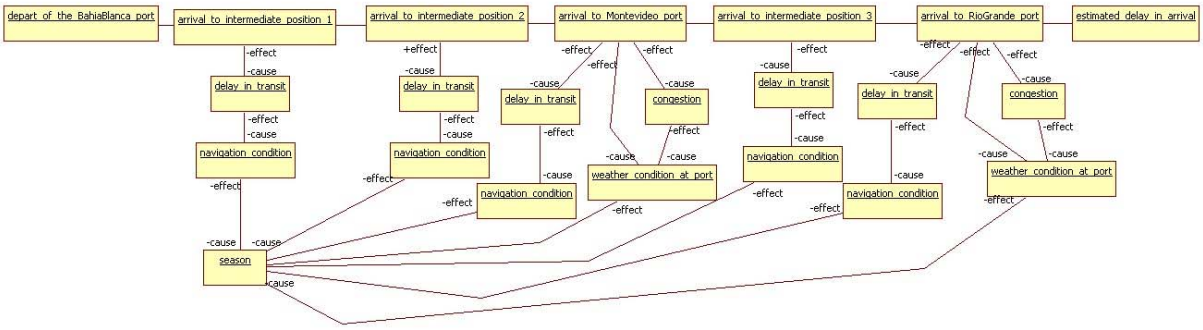

FIGURE 17. MonitoringStructure of a marine transport order

inspection on Figure 14, the second transfer from Bahia Blanca to Brasil will be infeasible, and this also will make impossible to fulfill DC-Brasil customer orders.

The Feasibility Manager receives an *EventScope* and generates the corresponding Schedule Feasibility Problem for every step of the feasibility search procedure detailed in the Service Implementation.

The *EventScope* is composed by resource Ship-DCBrasil in the set of Resources, together with its scheduled orders. Table 2 shows how the *CollaborationScope* is expanded starting from the *EventScope* by requiring successive collaborations; in the second request for collaboration feasibility is found.

TABLE 2. Schedule expansion through successive collaboration, disruptive event 1

Expansion	*Resources*	*FixedSPO*	*VariableSPO*
EventScope	Ship-DCBrasil	Ship-DCBrasil.scheduledOrders	Ø
Collaboration Scope 1	Ship-DCBrasil Urea-DCBrasil loadingDock-BB	Urea-DCBrasil.scheduledOrders loadingDock-BB.scheduledOrders	transfer-Urea-BahiaBlanca-DCBrasil-7 transfer-Urea-BahiaBlanca-DCBrasil-26
Collaboration Scope 2	Ship-DCBrasil Urea-DCBrasil loadingDock-BB spareShip	Urea-DCBrasil.scheduledOrders loadingDock-BB.scheduledOrders	transfer-Urea-BahiaBlanca-DCBrasil-7 transfer-Urea-BahiaBlanca-DCBrasil-26 spare-Urea-BahiaBlanca-DCBrasil-1

Summary of changes introduced in the schedule to restore feasibility:

SPO transfer-Urea-BahiaBlanca-DCBrasil-26 has been cancelled.

SPO spare-Urea-BahiaBlanca-DCBrasil-1 was activated during the repair process with the following specification: *order quantity: 17000; order startTime: 383; order endTime: 608* and the following requirements specification:

CapacityRequirement: load urea at Bahia Blanca, involving resource loadingDock-BB starts at 383h and ends at 400h, with a duration of 17h.

CapacityRequirement: load urea at Bahia Blanca, involving resource spareShip starts at 383h and ends at 400h, with a duration of 17h.

StateBasedRequirement: arrive and stay in state Rio Grande, to download Urea at DC in Rio Grande Brasil, involving resource spareShip starts at 568h ends at 608h, with duration of 40h.

CapacityRequirement: download urea to DC in Rio Grande Brasil, involving resource spareShip starts at 568 and ends at 608, with duration of 40h.

CapacityRequirement: download urea to DC in Rio Grande Brasil, involving resource Urea-DCBrasil starts at 568 and ends at 608, with duration of 40h.

Scenario 7.1.4.2. Unexpected increase in demand at DC-Uruguay. The supply chain of this example shares the Urea Market with another provider and its supply chain. At Uruguay a competitorś Distribution center temporarily runs out of stock, forcing its clients to supply from Urea-DCUruguay. As a result, the daily shipments scheduled from day 10 to 19 have an increase in their order quantity.

Simulating the effects of this event on Urea-DCUruguay inventory (*Capacity Dimension: Urea-DCUruguay-capDim*), an important infeasibility clearly appears as seen in Figure 18, as the curve labeled "Exception".

The Feasibility Manager receives this time the whole schedule as the *EventScope* and returns the following results: A set of 9 orders are modified (within planned implicit and explicit slacks) in order to restore feasibility. In Figure 18 is visible how the solution of the mechanism is closely related to the original schedule. The second Urea transfer is put forward and the other modifications consisted in slightly reducing some orders quantities in order to restore feasibility preserving most of the original schedule.

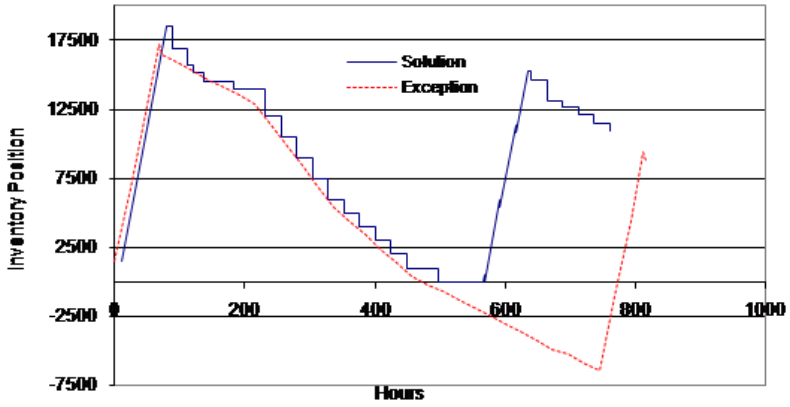

FIGURE 18. Urea-DCUruguay projected available inventory (exception and solution)

8. **Discussion and Future Work.** In this work, we have presented a comprehensive proposal to systematically address the problem of *Collaborative Management of Disruptive Events in Supply Chains*. Both academic and industrial researchers have identified this problem as not being adequately covered by state of the art solutions in the SCEM systems. Moreover, the ability to automatically detect disruptions and repair them locally without affecting coordinated and coexistent schedules within a supply chain, is recognized to be a major competitive advantage in next generation of SCM systems.

Our proposal includes the definition of a business process conceived to be executed collaboratively by independent supply chain partners with the intention of providing system support for companies willing to engage in collaboration agreements for controlling the execution of their supply processes.

Using this novel business process, we developed a complete service-oriented solution applying standard SOMA techniques in order to derive the architecture, define the services interfaces, the service data models and the choreographies representing the collaborations. The application of this technique allows the generation of the SOA specifications in full compliance with the business process and its requirements.

Following the principles of MDD (model driven development), the SOA specification proposed is independent of the implementation and therefore suitable for expressing the details of the collaboration contracts among different companies running their processes on different technological platforms.

The service operations identified in this solution can be implemented following different strategies, but the interfaces specified provide a consistent system support for the companies to interoperate.

The consistency of interoperation is also granted in the semantic level by complementing the SOA specification with reference models that provide the basis for the definition of the business documents being exchanged among the participants.

The proposed reference models accomplish the description of the problem information in a very high level of abstraction and therefore are applicable to a wide range of supply chain processes, from procurement, manufacturing, distribution, and retailing domains. In particular, the reference models proposed also have the characteristic of providing self-contained descriptions of the information required for the decision making activities involved in the business process. This feature enables the possibility of automating the generation of decision models expressed in standard representations for decision making tools (as mathematical programming solvers or inference engines).

In order to validate our proposal we have developed specific implementation for the two specialized services in the solution: the Feasibility Management and the Monitoring services. In both implementations, we take advantage of the generality of the reference models to deploy highly automated decision making procedures.

The Feasibility Management implementation we propose is able to provide generic feasibility checking and repair mechanisms for local adjustments of the coordinated execution schedule within the space of slacks already provided in the planned operations. This level of intervention is the most suitable for being delegated into automated procedures avoiding the need for triggering complex re-planning iterations.

The Monitoring implementation we propose also exploit the generality of the reference models by framing the monitoring task into four well-identified activities that will capture the disruptive events that are rooted either on the orders dynamic or the resource availability. For those processes where it is possible to identify measurable variables with predictive ability to anticipate the disruption through causal relationships, the implementation increase the pro-activity of the monitoring task by supporting these prediction with inference engines.

The implementation strategy has been tested and illustrated with a case study from a commodity chemical supply chain. In the case study typical disruptions were captured, assessed and repaired by using the services described in this paper. There are some issues that need to be addressed further for the proposal in this work to be effectively deployed in real world scenarios. First, the generality of the reference models impose the burden of creating very rich and dense documents that collect information normally disperse in different business applications and databases. We understand that this is not a simple task in nowadays enterprise software. However, as the service oriented approach gains momentum in the industry, and the concepts of cross-organizational information buses are becoming more and more popular, the gap for the requirements in this proposal will narrow in the short future.

The ability for the monitoring function increase their proactive capabilities by using predictive models in the current status of this proposal is still limited to the design of ad-hoc causal relationship networks. Although the construction of specialized libraries for commonly used supply process types can be conceived as a practical solution, there are opportunities for extending the autonomy in the generation of these networks.

Acknowledgment. This work is partially supported by Consejo Nacional de Investigaciones Cientficas y Tcnicas (CONICET) and Agencia Nacional de Promocin Cientfica y

Tecnolgica (ANPCyT). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] C. Soosay, P. Hyland and M. Ferrer, Supply chain collaboration: Capabilities for continuous innovation, *Supply Chain Management*, vol.13, no.2, pp.160-169, 2008.

[2] R. Derrouiche, G. Neubert and A. Bouras, Supply chain management: A framework to characterize the collaborative strategies, *International Journal of Computer Integrated Manufacturing*, vol.21, no.4, pp.426-439, 2008.

[3] P. R. Kleindorfer and G. H. Saad, Managing disruption risks in supply chains, *Production and Operations Management*, vol.14, no.1, pp.53-68, 2005.

[4] H. L. Lee, V. Padmanabhan and S. Whang, The bullwhip effect in supply chains, *Sloan Management Review*, vol.38, no.3, pp.93-102, 1997.

[5] N. Radjou, L. M. Orlov and T. Nakashima, Adapting to supply network change, *The TechStrategy Report*, 2002.

[6] L. Zhao, L. Qu and M. Liu, Disruption coordination of closed-loop supply chain network (i) – Models and theorems, *International Journal of Innovative Computing, Information and Control*, vol.4, no.11, pp.2955-2964, 2008.

[7] L. Zhao, M. Liu and L. Qu, Disruption coordination of closed-loop supply chain network (ii) – Analysis and simulations, *International Journal of Innovative Computing, Information and Control*, vol.5, no.2, pp.511-520, 2009.

[8] V. Landeghem and H. Vanmaele, Robust planning: A new paradigm for demand chain planning, *Journal of Operations Management*, vol.20, pp.769-783, 2002.

[9] H. L. Lee, The triple – A supply chain, *Harvard Business Review*, vol.82, no.10, pp.103-112, 2004.

[10] G. E. Vieira, J. W. Herrmann and E. Lin, Rescheduling manufacturing systems: A framework of strategies, policies, and methods, *Journal of Scheduling*, vol.6, no.1, 2003.

[11] H. Aytug et al., Executing production schedules in the face of uncertainties: A review and some future directions, *European Journal of Operational Research*, vol.161, no.1, pp.86-110, 2005.

[12] A. Adhitya, R. Srinivasan and I. A. Karimi, A model-based rescheduling framework for managing abnormal supply chain events, *Computers and Chemical Engineering*, 2006.

[13] A. Pfeiffera, B. Kádára and L. Monostoria, Stability-oriented evaluation of rescheduling strategies, by using simulation, *Computers in Industry*, vol.58, no.7, pp.630-643, 2007.

[14] X. Wang, A. Liang, C. Xu and D. Yang, Analysis and design of decision support system of disruption management in logistics scheduling, *International Journal of Innovative Computing, Information and Control*, vol.5, no.6, pp.1559-1568, 2009.

[15] N. Masing, *SC Event Management as Strategic Perspectiva – Market Study: CMDESC Software Performance in the European Market*, Master Thesis, Universitié du Québec en Outaouasis, 2003.

[16] R. Zimmermann, Agent-based supply network event management, in *Whitestein Series in Software Agent Techonologies*, M. Walliser, S. Brantschen, M. Calisti and T. Hempfling (eds.), 2006.

[17] K. Knickle and J. Kemmeler, Supply chain event management in the field success with visibility, *AMR Research*, Boston, 2002.

[18] N. Montgomery and R. Waheed, Event management enables companies to take control of extended supply chains, *AMR Research*, 2001.

[19] Q. Guo and M. Zhang, A novel approach for multi-agent-based intelligent manufacturing system, *Information Sciences*, vol.179, pp.3079-3090, 2009.

[20] O. Hoffmann, D. Deschner, S. Reinheimer and F. Bodendorf, Agent-supported information retrieval in the logistic chain, *Proc. of the 32nd Hawaii International Conference on System Sciences*, Maui, 1999.

[21] M. Kärkkäinen, K. Främling and T. Ala-Risku, Integrating material and information flows using a distributed peer-to-peer information system, in *Collaborative System Production Management*, H. S. Jaddev, J. C. Wortmann and H. J. Pets (eds.), Boston, Kuwer Academic Publishers, 2003.

[22] R. Schoenthaler and C. Alvarenga, A new take on supply chain event management, *Supply Chain Management Review*, 2003.

[23] J. K. Speyerer and A. J. Zeller, Managing supply networks, symptom recognition and diagnostic analysis with web services, *Proc. of the 37th Hawaii International Conference on System Sciencies*, 2004.

[24] N. B. Szirbik, J. C. Wortmann, D. K. Hammer, J. B. M. Goosenaerts and A. T. M. Aerts, Mediating negotations in a virtual enterprise via mobile agents. *Proc. of the Academia/Industry Working Conference on Research Challenges, IEEE Computer Society*, Buffalo, NY, USA, pp.237-242, 2000.
[25] F. Teuteberg and D. Schreber, Mobile computing and auto-ID technologies in supply chain event management – An agent-based approach, *Proc. of the 13th European Conference on Information Systems*, Regensburg, Germany, 2005.
[26] T. Bui, C. Hempsch, H. Sebastian and T. Bosse, A multi-agent simulation framework for automated negotiation in order promising, *Proc. of the 15th Americas Conference on Information Systems*, San Francisco, CA, USA, 2009.
[27] S. Bussmann, N. R. Jennings and M. Wooldridge, Multiagent systems for manufacturing control: A design methodology, in *Springer Series on Agent Technology*, T. Ishida, N. R. Jennings and K. Sycara (eds.), Berlin Heidelberg, Springer-Verlag, 2004.
[28] A. C. A. Cauvin, A. F. A. Ferrarini and E. T. E. Tranvouez, Disruption management in distributed enterprises: A multi-agent modelling and simulation of cooperative recovery behaviours, *International Journal Production Economics*, vol.122, pp.429-439, 2009.
[29] P. Leitao, *An Agile and Adaptive Holonic Architecture for Manufacturing Control*, Ph.D. Thesis, Department of Electrotechnical Engineering Polytechnic Institute of Braganca, University of Porto, http://www.ipb.pt/pleitao, Portugal, 2004.
[30] L. Monostori, E. Szelke and B. Kadar, Management of changes and disturbances in manufacturing systems, *Annual Reviews in Control*, vol.22, pp.85-97, 1998.
[31] L. Ribeiro, J. Barata and A. Colombo, Supporting agile supply chains using a service-oriented shop floor, *Engineering Applications of Artificial Intelligence*, vol.22, pp.950-960, 2009.
[32] W. Shen and D. H. Norrie, An agent-based approach for manufacturing enterprise integration and supply chain management, *Proc. of the 3rd International Conference on the Practical Applications of Agents and Multi-Agents Systems, PAAM*, 1998.
[33] J. Swaminathan, S. Smith and N. Sadeh, Modeling supply chain dynamics: A multiagent approach, *Decision Sciences*, vol.29, no.3, 1998.
[34] K. Van Dam, A. Adhitya, R. Srinivasan and Z. Lukszo, Critical evaluation of paradigms for modelling integrated supply chains, *Computers & Chemical Engineering*, vol.33, no.10, pp.1711-1726, 2009.
[35] L.-C. Wang and S.-K. Lin, A multi-agent based agile manufacturing planning and control system, *Computers & Industrial Engineering*, vol.57, pp.620-640, 2009.
[36] M. P. Papazoglou and W. Van Den Heuvel, Service oriented architectures: Approaches, technologies and research issues, *VLDB Journal*, vol.16, no.3, pp.389-415, 2007.
[37] Object Management Group, *Unified Modeling Language (UML), Version 2.3*, http://www.omg.org/spec/UML/2.3/, 2010.
[38] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy and K. Holley, SOMA: A method for developing service-oriented solutions, *IBM Systems Journal*, vol.47, no.3, pp.377-396, 2008.
[39] Object Management Group, *Business Process Model and Notation (BPMN), Version 2.0*, http://www.omg.org/spec/BPMN/2.0/, 2011.
[40] Object Management Group, *Service Oriented Architecture Modeling Language (SoaML), Version 1.0 – Beta 2*, http://www.omg.org/spec/SoaML/, 2009.
[41] A. Guarnaschelli, O. Chiotti and E. Salomone, Service oriented approach for autonomous exception management in supply chains, *I3E 2010*, pp.292-303, 2010.
[42] T. BedraxWeiss, C. McGann and S. Ramakrishnan, Formalizing resources for planning, *Proc. of the 13th International Conference on Automated Planning and Scheduling*, Italy, Trento, 2003.
[43] E. Fernndez, E. Salomone and O. Chiotti, Model based on Bayesian networks for monitoring events in the supply chain, *IFIP Advances in Information and Communication Technology*, vol.338, pp.358-365, 2010.
[44] S. Mellor, K. Scott, A. Uhl and D. Weise, *MDA Distilled, Principles of Model Driven Architecture*, Addison-Wesley Professional, Addison-Wesley, 2004.
[45] J. Warmer and A. Kleppe, *The Object Constraint Language: Getting Your Models Ready for MDA*, 2nd Edition, Addison-Wesley, 2003.
[46] *IBM ILOG OPL Development Studio API.*
[47] *Hugin Expert A/S, Hugin Researcher*, www.hugin.com, 2010.

Appendix A. Messages of the CMDESC Business Process. In this appendix, we define the specific messages exchanged among the participants of the proposed business

process. A first group of messages is related to conveying information about the schedule and disruptive events. The structure of these messages is shown in Figure 19.

ScheduleMsg. This message is sent by an execution system or any other client, user of the Controller in this business process. As has been defined previously, a schedule may contain a vast variety of synchronized resources and supply processes. In order to support the diversity and complexity of coexisting schedules, in this business process a schedule is defined as a message containing schedule information compliant with the Reference Model for Event Management [38]. This reference model provides a common language to describe and understand supply processes and the interaction between them and resources throughout the supply chain. The description obtained by using this reference model is enough to assess feasibility of execution. The structure of this message is shown in Figure 19.

DisrupriveEventMsg. This message contains the set of resources and supply process orders affected by the disruptive event. The structure of this message is shown in Figure 19.

EventScopeMsg. This message contains the scope of the disruptive event defined with the task Define Event Scope. The structure of this message is shown in Figure 19.

CollaborationRepairRequestMsg. This message contains the identification of resources and supply process orders required to keep searching for feasibility in the Schedule represented by the Collaboration Scope. The structure of this message is shown in Figure 19.

FeasibilityReportMsg. This message is defined whenever a Check Feasibility or Repair Schedule Sub-process ends. It contains information regarding a successful search for feasibility and the changes introduced in the schedule to obtain it. And if collaboration from other SC members was necessary a detail of the collaborations involved. When this message is sent after a Check Feasibility Request sub-process ends it only contains information on whether the schedule remains feasible or not. The structure of this message is shown in Figure 19.

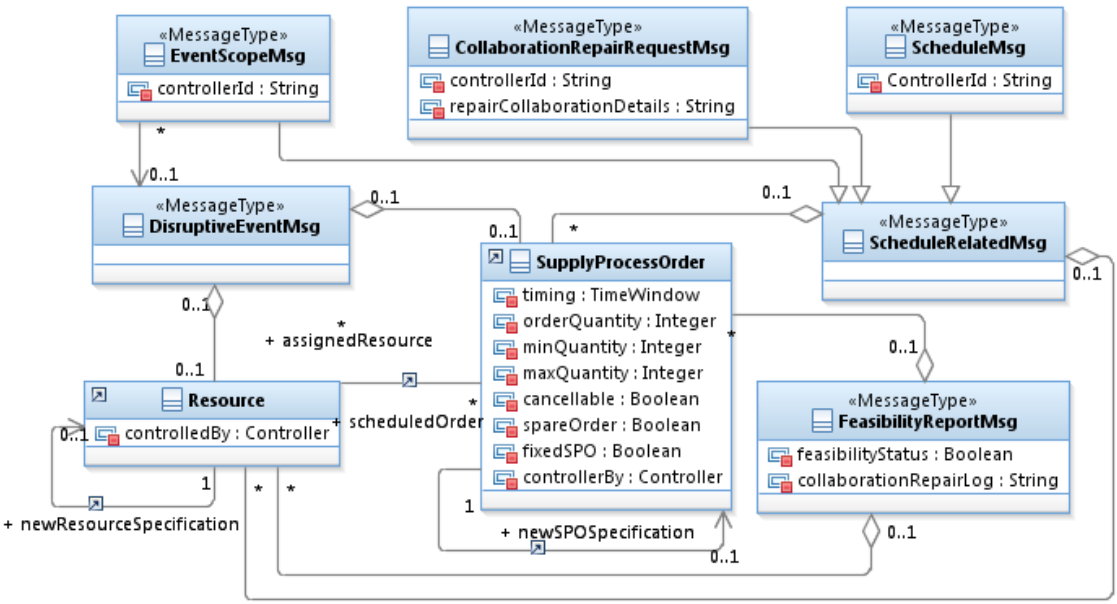

FIGURE 19. Schedule related messages and DisruptiveEventMsg