Computers in Industry 83 (2016) 92–107

Contents lists available at ScienceDirect

Computers in Industry

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

A service-oriented framework for agent-based simulations of collaborative supply chains

Mariana Dorigatti[a,*], Armando Guarnaschelli[b], Omar Chiotti[a], Hector E. Salomone[a]

[a] INGAR-CONICET, Avellaneda 3657, S3002GJC Santa Fe, Argentina
[b] Pontificia Universidad Católica de Valparaíso, ITRA, 2147 Brasil Avenue, 2362804 Valparaíso, Chile

ARTICLE INFO

Article history:
Received 10 November 2015
Received in revised form 28 July 2016
Accepted 28 September 2016
Available online xxx

Keywords:
Agent-based simulation
Collaborative supply chains
Service oriented framework

ABSTRACT

Current collaborative practices of supply chain management are limited to some known configurations where a dominant member sets the pace for the collaboration extent. Extending collaborative models to supply chains without a dominant member requires defining a fair assessment of costs and benefits and how they are distributed among members. To understand collaborative models and their mechanisms, simulation-based approaches are recommended as they can afford the complexity of real scenarios. However, building ad-hoc simulation models for studying complex supply chain interactions can be prohibitive in terms of both cost and time. Therefore, the availability of simulation frameworks, to be used easily by business managers and facilitating the development of those models, has a strong incentive in the quest of current business efforts to increase their supply chain performance. The objective of this work is to present a systematic and reusable serviceoriented framework for agent based simulation to support the analysis of collaborative interactions in supply chains. Results of a requirement analysis performed to this aim are described, and the fulfillment of identified requirements by the proposed framework, and capabilities thereof, are discussed.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Supply chain collaboration implies that two or more independent members work jointly in planning and executing supply chain operations for enhancing their performance. E-business environment enabled a series of collaboration mechanisms for information sharing, operation coordination, and joint decision making that convey the promise of improving competitive advantages for all members engaged in a common supply chain [1].

In spite of the strong theoretical arguments about the improved overall performance obtained through collaboration, current practice remains rather limited to some known configurations where a dominant member of the supply chain sets the pace for the collaboration extent. One of the main hurdles found in extending collaborative models to supply chains without a dominant member is the difficulty in performing a fair assessment of their benefits and required efforts, and how both are distributed among members. While analytical models have provided very valuable qualitative and quantitative insights, for a better understanding of collaborative models and their mechanisms, simulation-based approaches are recommended as they can afford the complexity of real scenarios [2,3].

Building ad-hoc simulation models for studying complex supply chain interactions can be prohibitive in terms of both cost and time. Therefore, the availability of simulation frameworks, easy to use by business managers and facilitating the development of those models, has a strong incentive in the quest of current business efforts to increase their supply chain performance.

The objective of this work is to present a systematic and reusable service-oriented framework for agent-based simulation to support the analysis of collaborative interactions in supply chains. Artifacts comprised in the framework were designed following a systems analysis and design methodology including the steps of requirement analysis, conceptual design, prototype design and implementation and case study validation of the enunciated requirements. The requirement analysis was performed by reviewing and extending the functionality highlighted in current literature about supply chain frameworks. Traditional steps of information elicitation among customers and users were

* Corresponding author.
E-mail addresses: marianadorigatti@gmail.com, marianadorigatti@santafe-conicet.gov.ar (M. Dorigatti), armando.guarnaschelli@pucv.cl (A. Guarnaschelli), chiotti@santafe-conicet.gov.ar (O. Chiotti), salomone@santafe-conicet.gov.ar (H.E. Salomone).

http://dx.doi.org/10.1016/j.compind.2016.09.005
0166-3615/© 2016 Elsevier B.V. All rights reserved.

replaced by the thoughtful interpretation of a well-known reference model widely adopted by the supply chain community [4]. Required business processes were documented and specified by using a standard notation [5] and the analysis of interactions of different lanes in the business processes was used to identify the required service interfaces as suggested in traditional service oriented design methodologies [6].

The framework proposed in this paper, builds on top of the existing knowledge for simulating supply chains and addresses the specific issues derived from collaborative interactions. The main contribution is a novel combination of service oriented architecture, agent-based simulations and business processes perspective that resulted in a framework organization capable to handle complex supply chain collaborative scenarios. The capability of the framework in addressing the requirements was validated with a case study.

In this work, we have identified several features a simulation framework should have in order to support the modeling of collaborative supply chain arrangements. We analyzed some of the relevant proposals in the literature under their ability to provide with those features. The application of the framework for modeling an actual supply chain demonstrated the ability of the proposed generalized components to represent different planning coordination strategies and collect simulated results of each scenario performance under varying conditions of the external demand. The comparison of different cases confirmed the expected qualitative behavior and validated its usage as an assessment tool for obtaining quantitative measurements of different collaboration strategies.

The remainder of this work is structured as follows. Section 2 summarizes the results of a requirement analysis. Related works are discussed in Section 3. Section 4 presents a framework overview and a description of its structural and functional components. Section 5 describes a prototype that implements the framework. Section 6 discusses the fulfillment of identified requirements and evaluates the framework capabilities. Finally, conclusions and future work are outlined in Section 7.

2. Requirement analysis

This section presents a summary of the identified capabilities that a simulation framework should offer in order to adequately support the analysis of collaborative interactions in supply chains. They were gathered by reviewing the requirements and features described in the literature about supply chain simulation frameworks. A critical analysis of those requirements was performed to identify the contributions that enable or enhance desired functionality for simulating non-centralized, dynamic and multi-actor interactions as the one typically arising during the execution of collaborative supply chain business processes. The following requirements are not intended to be a complete specification, they are focused on the functional aspects relevant to the representation of collaborative interactions and are provided as the driving forces that guided the design of the framework we are proposing in this paper.

I *Provide reusable components that can be easily assembled to set a wide variety of supply chain scenarios among independent partners*. Building ad hoc simulation models for every new simulation analysis is expensive; and skilled modeling abilities are required. To make the analytical tool usable for business analysis, the framework should provide a comprehensive set of generic components that can be easily specialized through parametric options and can be available within a general modeling environment. This requirement has been implicitly considered in most agent based SC simulation proposals and it has been pointed out as a fundamental requirement in the work of [7]. In their work, the concept of reusability is discussed and explained as the ability to reuse simulation model components by making small changes, and sometimes without any change at all. Clearly, reusability shortens the simulation modeling cycles and reduces the complexity of creating new models. It is a common practice in the literature to adopt some form of componentization, mostly associated to the structural aspects (used to describe the physical network) and organizational aspects (used to describe the interacting actors). We remark here the additional need to extend this practice to the business processes executed.

II *Allow for establishing dynamic links among partners without requiring a predefined network structure*. In order to support collaborative interactions, the framework should allow for relationships among members to be dynamically established, without imposing an a priori definition of rigid network structures. Representing real world SC collaboration is more difficult using rigid network structures. The fact is that even when there are strong partnerships among logistics nodes the risk of conflict arising from different goals and opposite interests persist, so flexibility in the network topology emerges as relevant requirement. Moreover, the agent based SC simulation methodological framework proposed by [8] relies on this fact to support a changing and evolving environment by the addition or removal of agents representing SC nodes and/or partners.

III *Adopt a business process-oriented perspective to reflect the activities in the supply chain, following some established reference model known to most business analysts*. One of the aspect less developed in most of the current frameworks is the organization and componentization of the supply chain activities and how those activities are orchestrated in business processes. To facilitate the understanding of the model and the interpretation of the simulation behavior, it is desirable to simulate and organize supply chain activities by using a business process orientation and resorting to widely accepted supply chain reference models such as SCOR [4]. The work of [9] highlight the need of a standardization, using the activities and processes as described and understood by the SC community. The SCOR model is a widely accepted standard and it is based on the description of business processes. A framework both based on the definition of business processes and in the semantics of their elements as stated by SCOR will have more impact and success on the SC community.

IV *Provide for explicit representations of control and decision making activities separated from execution activities*. In order to provide proper support to simulate the impact of different interaction policies and supply chain management actions, control and decision making activities should be clearly separated from execution activities and explicitly defined avoiding built-in replenishment strategies or inventory control mechanisms embedded in the execution simulation. This requirement has been previously highlighted in the literature in the works of [10,11]. The first work emphasizes the need of an explicit and specific modeling approach for control structures and discusses how these structures when embedded into the physical flow of a simulation severely harm the quality of the simulation approach. This is due to the fact that key decision variables are hidden within physical transactions building blocks and dispersed throughout the simulation model. The second work makes similar observations in the context of manufacturing simulation stating that decision makers, control rules and their interactions are mostly hidden and this diminishes modeling flexibility and modularity. This latter observation is also pointed out by [12].

V *Support the interaction among members by using message-based and document-oriented protocols that resemble actual business interactions*. Collaborative interactions among supply chain members are better described by explicitly defining standard interaction protocols that are supported by business document exchanging. In this way, it is easier to deploy different implementations of internal processes of each member, still allowing interoperability and providing a natural control on the information visibility that is shared among members. Collaborations between independent partners deal with the problematic of sharing private and sensitive information [13]. As a consequence of this, modeling collaborative interactions requires the specification of communication protocols as well as messages and business documents to be shared. Agent based simulation frameworks such as the ones proposed in [7,8,14] adopt message based interaction but are not intended to specify and outline the information to be shared.

3. Related works

Discrete Event simulations and System Dynamics are modeling approaches widely used to analyze supply chain behavior. An extensive literature review [5] records the usage of both approaches to deal with different supply chain problems. Despite the large list of contributions, the study of dynamic supply relationships among independent members appears as barely explored.

Umeda and Lee [6] propose a generic hybrid-modeling framework that combines discrete-event simulations with system-dynamics simulations. Discrete event models represent operational processes within the supply chain; and system dynamic models represent reactions in supply chain management circumstances. Traditional discrete-event approaches usually adopt a network perspective and are focused on representing the supply chain's topology and infrastructure, while generally assuming implicit representations of control and decision processes.

Agent-based simulation offers a promising framework to capture the dynamic aspects of logistics coordination among supply chain partners. Agent-based approaches focus on individual participants' behavior and decision processes, often at the expense of event-oriented aspects of supply chains, as well as more global activities and policies.

A review of agent-based formalisms for supply chain simulation can be found in [7]. In this area, the work by Swaminathan et al. [8] outstands as one of the most comprehensive attempts to build a generalized framework. They were pioneers in having a vision of agents for the purpose of a flexible and reusable modeling and simulation framework that allows for the development of models to address issues related to configuration, coordination, and contracts. Models are made of reusable components representing different entities in the supply chain. Interaction protocols are introduced to support agents' interactions by regulating the flow of materials, information, and cash through message passing. Although in their library they classify components as either structural or control elements, the proposed controlling structures fall short in providing explicit representations of the relationship between decision making activities and their corresponding execution actions.

Requirements for a simulation modeling framework suitable for supporting the analysis of collaborative supply chains have been thoughtfully described by van der Zee & van der Vorst [9]. In this work, authors recognize the need for an explicit definition of control policies and coordination mechanisms. They also highlight the need for the/an explicit definition of timing and execution of decision activities. Upon these requirements, they propose a modeling framework based on agents and jobs concepts. Planning and control concepts are explicitly represented through decision making agents carrying out control jobs. In their implementation, they provide examples of aggregated agents as *Producer*, *DC* (for distribution center), *Outlet*, and *Costumer*. *DC* agent is composed of three agents: *DC_Distr*, taking care of order receipt and picking, *DC_DistrPlanning* deciding on the timing and quantity of sent orders, and *DC_Purchasing* for ordering replenishments. Although relationships between agents are governed by a generalized concept of flow (including materials, information, and jobs), these interactions are less structured and do not present well defined interaction protocols as it is desirable to have in a generalized and systematic framework for agent-based simulation models. The internal structure of agents is only defined at a very high level (jobs and resources) and the definition of actual supply chain entities are left to the specialization of this structure.

Pundoor and Herrmann [10] present a framework for building hierarchical simulation models that integrate discrete event simulation and spreadsheets. The first level is the simulation model; the second level has submodels that correspond to supply chain participants (consumers, producers, and traders) and the third level has submodels that correspond to process elements performed by each participant. These process elements are standardized modules that can be reused. Examples of these modules are: *Schedule Product Deliveries, Receive Product, Schedule Production, Produce and Test, Receive Orders*, etc. Each participant in the supply chain has its own set of modules and they are connected by using standard interfaces that represent the material, information, and cost flow. Planning activities are carried out by using Excel VBA. Execution is carried out in Arena. Activities are conceived as business processes by adopting SCOR reference model; and their implementation distinguishes between control and decision making activities.

Rosetti and Chan [11] describe an object-oriented framework providing a list of reusable classes for modeling and simulating a supply chain. Their classes are organized into several conceptual submodels. Structural aspects are captured by a *RelationshipNetwork*, composed of *Nodes* and *Relationships*. A Node in this *RelationshipNetwork* can represent a *Facility*, *Region*, or *OrderGenerator*. A *Relationship* represents a conceptual connection between two *Nodes* and indicates the possible flow of information or material between *Nodes*. The role of a *Facility* in the network is to manufacture products, distribute products, consolidate shipments, and deliver shipments. *Facility* class is further specialized into five different concepts in the framework design. They are: *ManufacturingCenter, DistributionCenter, TransportationCenter, Shipper,* and *Contractor*. Control and decision making activities are not clearly decoupled from execution activities except for a class *InventoryPolicy* which encapsulates rules to control the associated inventory. Although the framework provides a class to represent the concept of an Order, transactions do not follow an explicit interaction protocol.

Labarthe et al. [12] present a framework for multi-agent simulations integrated by cognitive and reactive agents. Cognitive agents are responsible for decision making or deliberative activities and they are based on decision models or procedures that require information interaction with other agents and communicate through messages. In turn, reactive agents perform execution activities by introducing a clear separation between the two levels. The framework components are specific to the proposed case study and their lack of generality reduces their potential reusability for other supply chain scenarios. The interaction among different supply chain entities is managed by communication protocols that coordinate agents' behavior. In this way there is no need for previous definition of network structures;

and links are dynamically established through the use of typical multi-agent registration and naming services. Communication among agents is supported by interaction protocols following ACL-FIPA standard.

Chatfield et al. [7] review different representation formalisms to outline a conclusion on the advantages of combining agent-based simulation for representing participants and process-driven simulation to capture the order-driven nature of supply processes. They divide the supply chain representation into five types of constructs: *node*, *arc*, *component*, *action*, and *policy*. Physical aspects of the supply chain are contained in *node*, *arc*, and *component* objects, whereas activities and logical (managerial) aspects of the supply chain are described by using *action* and *policy* objects. *Nodes* and *Arcs* represent locations and their connections. Orders, messages, and material are represented by *Components*. The framework provides: (1) a basic set of managers, *DemandGenerator, InventoryManager, OrderPlacementManager, OrderProcessingManager, ShippingManager, ReceivingManager, and TransportManager*, responsible for decision making; and (2) a basic set of actors, *OrderPlacementActor, ShippingActor, ReceivingActor* and *TransportActor*, responsible for the execution of related activities. The proposal was implemented as a set of Java classes called SISCO library, which can be instantiated through a graphical editor and then automatically mapped into entities of a general purpose, process-oriented simulator named Silk [13].

Umeda and Zhang [14] also propose a hybrid model combining discrete events and system dynamics. Model components are defined in different levels. The higher level corresponds to Feature-Elements Model that defines the members in the supply chain (*Supplier*, *Source*, *Storage*, *Consumer*, *Deliverer*, and *Manager*). The next level is Function-Element Model that defines business process functions (e.g. purchasing, procurement, delivery, etc.). The next

Table 1
Comparison of related works.

	I: Reusable components	II: Dynamic Links	III: Business Process Orientation	IV: Explicit Control and Decision Making	V: Message-Based Interaction Protocols
Rossetti & Chan [15]	Provides a set of Java classes that can be used with an open-source object-oriented library for executing discrete-event simulation models in Java programming Language (JSL)	Not supported. Members interactions should be predefined and captured in a *Relationship-Network*	Not supported	Neither explicit nor clearly decoupled from execution activities except for a class *InventoryPolicy* which encapsulates rules to control the associated inventory	Not supported. Provides a class for representing the concept of an Order but transactions do not follow an explicit interaction protocol. Orders are exchanged by invoking class methods.
Pundoor & Herrmann [9]	Generalized process elements implemented as Rockwell's Arena submodules extended with Excel VBA logic	Not supported. Predefined as connections among Arena modules	Activities and processes organized following SCOR reference model	Planning activities are defined in Excel VBA extensions; execution activities performed in Arena sub-modules	Not supported. Information flow is controlled by activities that trigger events. No message interactions among members
van der Zee & van der Vorst [10]	Main concepts of the framework: Agent, FlowItem, and Job are implemented extending the class library of eM-Plant. They can be combined to build aggregated reusable structures.	Not supported. Predefined as Connections in eM-Plant	Not supported.	Decision making separated from execution by using the concept of Job and Controller Agent responsible for creating job definitions	Not fully supported. Interaction is managed by passing Job definitions and FlowItems through interface ports of defined connections.
Labarthe et al. [8]	Case-Specific agents. Generalized reusable components are not provided.	Fully Supported. Information flow is managed by messages and physical flow by signal passing among agents. Links are dynamically established through the use of typical multi-agent registration and naming services	Not supported.	Cognitive agents responsible for decision making and reactive agents perform execution activities.	Interaction is message-based complaint with FIPA Agent Communication Language standard.
Chatfield et al. [16–18]	SISCO library provides a set of reusable Java classes: *Order, Node, Arc*, and multiple *Manager* and *Actor* classes for specialized supply chain functions.	Not supported. Predefined – Using arcs to connect nodes that represent locations	Business processes are not explicit. Some of them represented by managers or actors.	Managers are responsible for decision making activities, Actors for execution.	Interaction is order-centric but not through message passing among entities.
Umeda et. al [19]	Model components for generalized members: Supplier, Source, Storage, Deliverer, Consumer, and Planner	Not supported. Assumed predefined connections are needed in order to fit into System Dynamics/Discrete Events framework	Basic business processes defined at the level of Function-Element model	Not clearly separated. Some model components are parametric with two types of operational modes: Schedule-driven and Stock-driven to differentiate policy	Not supported.
Long et al. [20]	A library with five meta-agents: machine agent (MA), buffer/Inventory agent (BA/IA), transport agent (TA), Machining task agent (MTA) and Transport task agent (TTA)	Fully Supported. Links are dynamically established through the use of typical multi-agent registration and naming services	Not Supported.	Task agents responsible for providing operational policies to controlled agents	Interaction is a message-based complaint with FIPA Agent Communication Language standard
this proposal	A library of generalized structural components and services to configure agents representing supply chain members	Fully Supported. Links are dynamically established through the use of typical multi-agent registration and naming services	Inter-organizational business processes are supported by service oriented interactions. Internal private processes organized following SCOR reference	Separated business processes for decision making activities generating orders for execution activities.	Interaction is message-based following service-oriented interaction protocols exchanging business documents

level is Implementation Model providing description of activity elements expressed in terms of a neutral discrete event language; and the last level is Execution Modules that transfers the neutral discrete events language into a specific simulation source code. *Supplier*, *Source*, and *Storage* components defined in the feature model are parametric with two types of operational modes: Schedule-driven and Stock-driven. Schedule-driven mode involves the definition of orders controlled by the Manager component. On the other hand, Stock-driven mode implies some sort of automatic replenishment based on the observation of stock levels. In this case, the role of *Manager* is to control replenishment set points [6].

Long [15,16], presents a distributed framework for agent-based simulations where three types of agents are defined to represent the network structure: *Machine Agent*, *Buffer Agent*, and *Transport Agent*. These agents are basic components that can be instantiated with different attributes and methods to obtain specialized agents to model the material flow. In order to model the information flow and coordination, two types of agents are defined to act as managers or coordinators: *Machining Task Agent* and *Transport Task Agent*. These managers are responsible for handling orders, extracting task information from them, and providing operational policies to controlled agents. The supply chain structure is defined by instantiating nodes in the matrix-based model by using agent classes which appear to be rather complex and not very flexible to accommodate dynamic relationships.

In order to position our proposal in comparison with related works, we provide information in Table 1, which summarizes the capabilities of the most recent proposals in the context of the already mentioned requirements.

4. Framework overview

As a proposal to address the requirements outlined in the previous section, we introduce a service-oriented framework for performing agent-based simulations to support the analysis of collaborative relationships in supply chain interactions.

We explicitly adopt a business process perspective to capture all relevant activities in the supply chain, distinguishing between intra-organizational processes of each participating member and inter-organizational (collaborative) processes.

To represent the public interactions involved in collaborative processes we adopt a service-oriented approach. It defines the exposed operations, the interaction protocol, and the contractual requirements in terms of business information that need to be used in the process of invoking and responding to them.

To organize the different concepts and entities that compose the framework, we distinguish among three aspects that need to be accounted for: *organizational*, *structural*, and *functional*.

The *organizational* aspect refers to the different organizational units that participate as members in the supply chain. The term member is used to indicate an independent organizational unit with autonomous decision making ability and its own business goals. All members participating in the supply chain should be represented by a unique type of agent responsible for managing communications with other members by means of message-based interaction protocols defined by different services each member can offer and request. In this way, the supply chain will be modeled as a homogeneous agency of agents *SCMember*, each one representing a supply chain member. Agent *SCMember* has the generic ability to freely define its internal structure and business functions implementations but is obliged to be complaint with the standardized services interfaces.

In order to describe the *structural* aspect, the framework provides entities *Location* and *InventoryItem* in order to represent the physical logistic structure of each member. Entity *InventoryItem*, designed to represent the widely used concept of SKU (stock-keeping unit), in combination with entity *Location* allow each member to define its own physical logistic structure for the goods being managed in the supply chain.

The *functional* aspect is represented in the framework by a list of generalized services that can be added to an agent *SCMember* to define intra-organizational business processes a member performs, including public interactions with other members. The addition of a service to an agent *SCMember* will augment the list of service ports a member can answer and include additional *Resources* needed to execute related business processes.

In this framework, services were implemented by the following components: *Source_Service*, *Make_Service*, *Deliver_Service*, *Transport_Service*, and *Plan_Service*. In order to support the operation of these services, components representing different types of resources are also provided in the framework. These are:

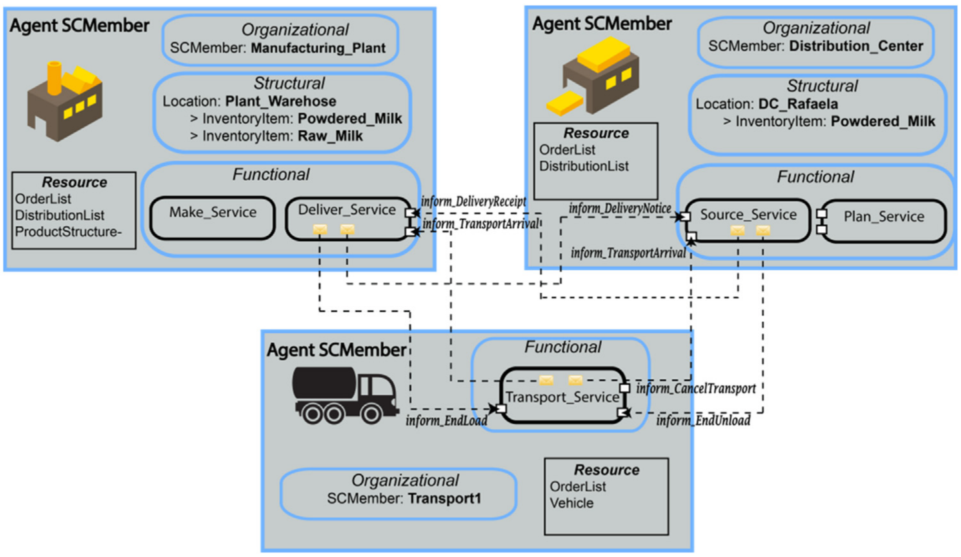


Fig. 1. Example of framework application to a simple Supplier-Transporter-Customer relationship.

i) *OrderList*, designed to represent a consolidated repository of inbound, outbound, internal transfers, and transformation orders managed by each agent SCMember. This repository can be thought as the order's database in the ERP (Enterprise Resorce Planning) system of the supply chain member.
ii) *DistributionList*, designed to represent all replenishment options of each entity *InventoryItem*.
iii) A *ProductStructureList,* designed to represent a bill-of-material for describing the production of items in transformation operations.
iv) *Vehicle*, designed to represent any resource that can move among entities *Location* carrying limited amount of items and account for transit times and in-transit inventory.

Before going to a more detailed description of the framework and its components, let us illustrate how the main concepts of the framework can be articulated to model a simplified example of a particular supply chain case. Fig. 1 depicts the usage of elements provided by the framework to represent a case involving three members engaged in a supply chain relationship with roles Supplier, Transporter, and Customer.

There are three agents *SCMember* representing the members involved in the supply chain. SCMember: *Manufacturing_Plant* controls a single *Location*: *Plant_Warehouse* managing two items: *InventoryItem*: *Powdered_Milk* and *InventoryItem*: *Raw_Milk*. It includes two services: *Make_Service* and *Deliver_Service* and their required resources. *SCMember*: *Transport1* has no Locations and just deploys a *Transport_Service* and its required resources. In turn, *SCMember*: *Distribution_Center* controls a single *Location*: *DC_Rafaela* managing a single item: *InventoryItem: Powdered_Milk* and deploying two services: *Plan_Service* and *Source_Service* with their required resources.

As an example of the inter-organizational business processes enabled by this configuration, the following sequence of interactions (Fig. 1) can be observed:

i) *Transport_Service* invokes *inform_TransportArrival* service port in *Deliver_Service* (this will trigger the internal process for loading the shipment)
ii) *Deliver_Service* invokes *inform_EndLoad* service port in *Transport_Service* (this will trigger the vehicle start the transit to destination)
iii) *Deliver_Service* invokes *inform_DeliveryNotice* service port in *Source_Service*
iv) *Transport_Service* invokes *inform_TransportArrival* service port in *Source_ Service* (this will trigger the internal process for unloading the vehicle)
v) *Source_Service* invokes *inform_EndUnload* service port in *Transport_Service* (this will free the Vehicle to leave for the next destination)
vi) *Source_Service* invokes *inform_DeliveryReceipt* service port in *Deliver_Service*

4.1. Structural components

The structural aspect of the supply chain is modeled by adding components of type *Location* to a given *SCMember*. Each *Location* represents a storage point and will contain a list of *InventoryItems* to represent all goods being physically managed in this storage point (Fig. 2). Each agent SCMember is responsible for the internal movements and interacts with other agents SCMember just through their external shipping and reception points.

InventoryItem provides the basic business processes to manage the storage point; i.e. put or draw units in storage, keep track of the current inventory and record performance indicators related to the stock position such as Average Inventory, Perfect Order Fulfillment, and Percentage of Orders Delivered In Full. These variables and indicators are aggregated into the Location containing a set of inventory items and further aggregated at *SCMember* level. Note that structural components do not provide any functionality other than basic recording. All supply chain activities are provided by functional components that will be responsible for deploying business processes, which control structural components.

4.2. Functional components

All activities in the supply chain are modeled as business processes organized in components representing services that can be offered by *SCMember*. In the current version of the framework, we have defined five generalized services that can be used as components to model the functional aspects of the member. *Deliver_Service* encapsulates all the functionality for *SCMember* to behave as a supplier; *Source_Service* enables the functionality to be a client receiving materials from a supplier; *Plan_Service* includes functions to generate internal transfer and production orders and external delivery orders based on demand requirements, distribution lists, and product structure lists; *Make_Service* have functions for executing production orders transforming raw materials into products; and *Transport_Service* provides functions for transporting materials among different locations.

Fig. 3 depicts five functional components defined as services indicating the proposed interaction interface with exposed operations.

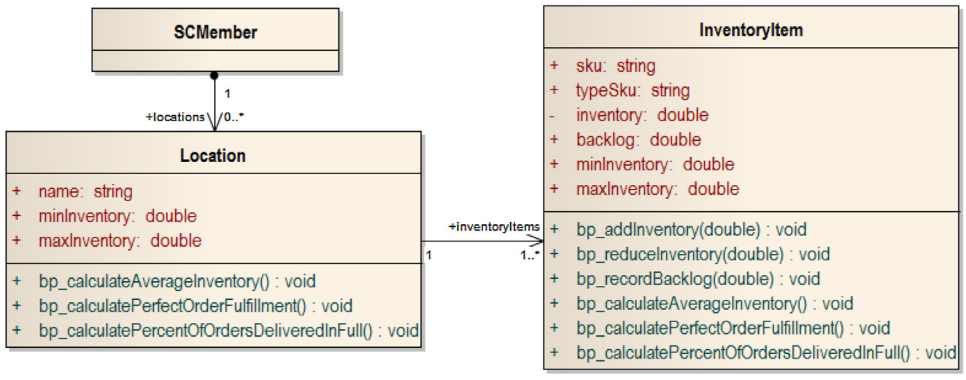

SCMember
1
+locations
0..*
Location
+ name: string
+ minInventory: double
+ maxInventory: double
+ bp_calculateAverageInventory() : void
+ bp_calculatePerfectOrderFulfillment() : void
+ bp_calculatePercentOfOrdersDeliveredInFull() : void
+inventoryItems
1
1..*
InventoryItem
+ sku: string
+ typeSku: string
- inventory: double
+ backlog: double
+ minInventory: double
+ maxInventory: double
+ bp_addInventory(double) : void
+ bp_reduceInventory(double) : void
+ bp_recordBacklog(double) : void
+ bp_calculateAverageInventory() : void
+ bp_calculatePerfectOrderFulfillment() : void
+ bp_calculatePercentOfOrdersDeliveredInFull() : void

Fig. 2. Structural components.

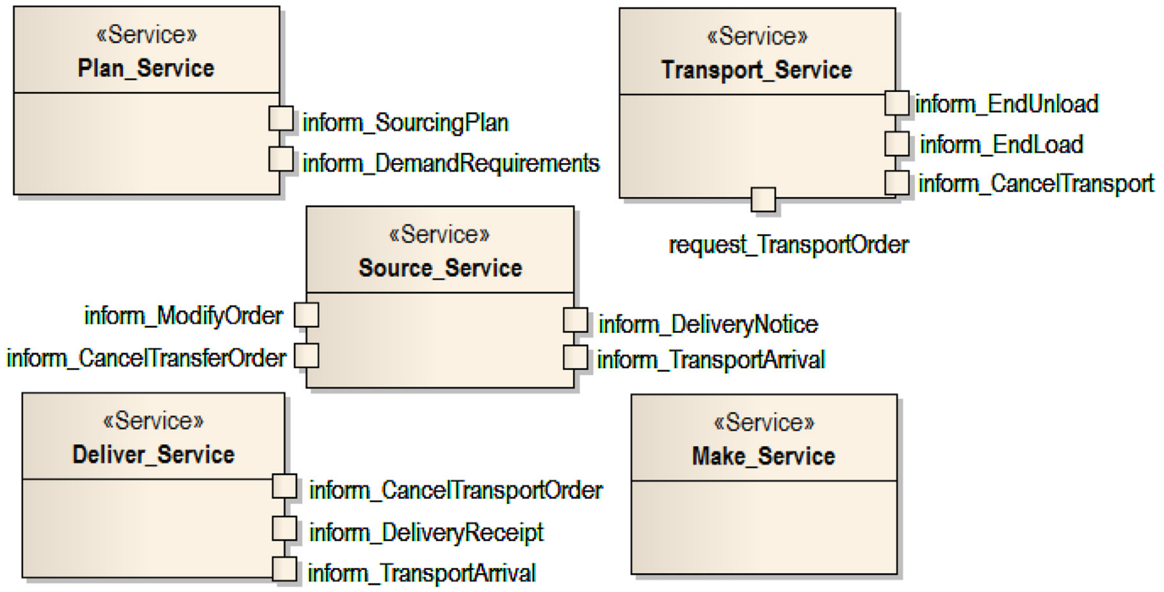


Fig. 3. Functional components as services.

When any of these components are added to agent *SCMember*, all the corresponding ports for receiving requests for exposed operations and all internal business processes for supporting required activities are added to the agent. The agent should be given as well as with all resources required by added business processes.

Note that by using this service-oriented architecture, the framework allows that different implementations of services can interoperate as long as they comply with service interface. In this way, the framework organizes functionality by encapsulating all internal processes as much as possible and reducing public interactions to a minimum set of message-based requests to

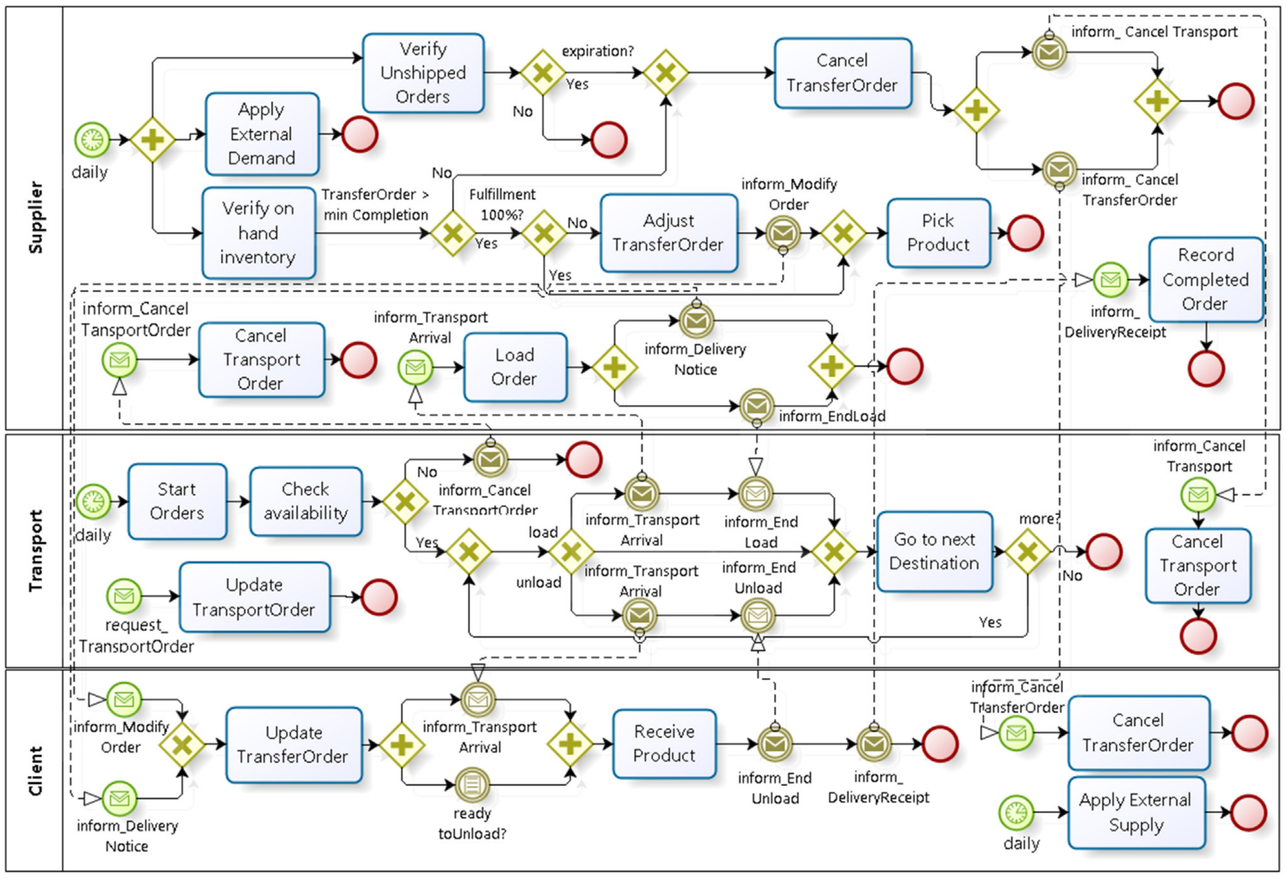


Fig. 4. Collaborative Business Process for executing a delivery order.

exposed operations. For instance, as *Make_Service* only define internal business processes, it has no exposed operations.

The proposed interface is the result of the definition of a standardized business process for the included supply chain activities. By using these business processes definitions, we identify both interactions among roles and the internal task needed to support interactions. The identified interactions are used to define service operations and internal activities guided the design of the internal functionality of the service. The business process for execution activities involving a *Supplier*, a *Client*, and a *Transport* is depicted in Fig. 4.

The collaborative business process for executing a delivery order has three lanes, one for each participating role. From this business process, functional components *Delivery_Service*, *Source_Service*, and *Transport_Service* were indentified and defined following each role in the collaboration. Interactions among roles (i.e. message arrows crossing the role lane) were used to identify public operations exposed by each service. Details of their involved internal business processes and resources are shown in Fig. 5.

All interactions among roles are modeled as a service operation invoked through a standard message carrying the needed business information. In the framework, these messages have been modeled by a Class *SCMessage* adopting a FIPA (Foundation for Intelligent Physical Agent) Agent Communication Language standard. The Class *SCMessage* includes the following attributes: *sender* and *receiver*, denoting the identification of the corresponding *SCMember*; an attribute *performative*, indicating the type of the communicative act; and a *content* including business objects needed to understand the message.

Interactions in execution activities are supported by two main business documents: a *TransferOrder* and a *TransportOrder*. *TransferOrder* indicates the origin and destination of a delivery and it is composed of a list of order items, each one indicating SKU and quantity. *TransportOrder* is composed of a list of locations to visit in a journey and –for each location- a list of references to transfer orders to be processed for loading or unloading on the site.

By exchanging these two documents, participants are able to conduct all internal activities involved in the full execution of the business process in Fig. 4.

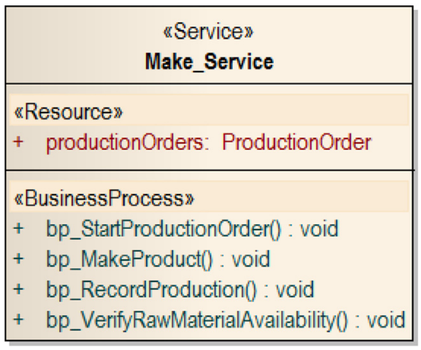

«Service»
Make_Service
«Resource»
+ productionOrders: ProductionOrder
«BusinessProcess»
+ bp_StartProductionOrder() : void
+ bp_MakeProduct() : void
+ bp_RecordProduction() : void
+ bp_VerifyRawMaterialAvailability() : void

Fig. 6. Make Service detail.

There is another execution process for transforming raw materials or intermediates into products with all its internal activities and not requiring interactions with other members. These activities have been encapsulated in a functional component *Make_Service* which is responsible for executing planned production orders. Details of this service are shown in Fig. 6. This service is driven by a list of ProductionOrders indicating what to do and when. A production order indicates start and end dates, origin and destination locations for raw materials and products, a reference SKU with the corresponding production quantity, and a product structure list composed of consumed and produced SKUs with the corresponding quantities, relative to the reference SKU quantity. This structure is suitable to represent bill-of-material and sub-products relationships.

The definition of the list of transfer, production, and transport orders to be executed is the result of the decision making activities belonging to another standardized business process for planning the supply chain that is shown in Fig. 7.

This process implements the activities carried on by a *SCMember* interacting with other agents *SCMember* having the same process and requires two interactions: one to receive the demand requirements from a client and another one to inform about the resulting sourcing plan. In this version of the framework, the processes corresponding to SCOR make-to-stock mode were

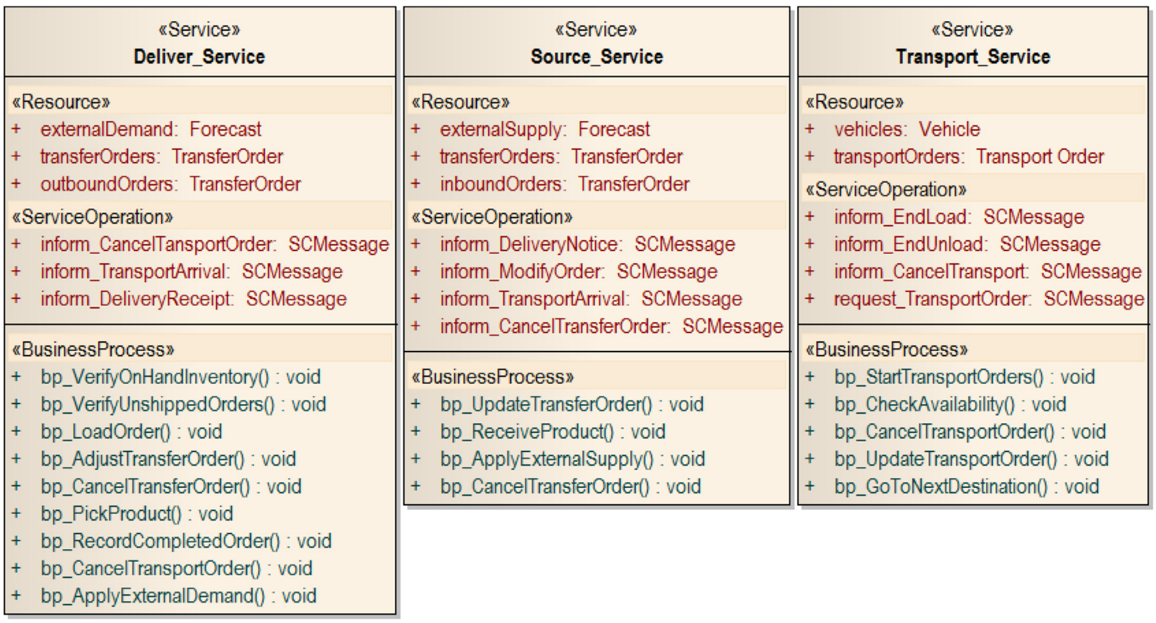

«Service»
Deliver_Service
«Resource»
+ externalDemand: Forecast
+ transferOrders: TransferOrder
+ outboundOrders: TransferOrder
«ServiceOperation»
+ inform_CancelTansportOrder: SCMessage
+ inform_TransportArrival: SCMessage
+ inform_DeliveryReceipt: SCMessage
«BusinessProcess»
+ bp_VerifyOnHandInventory() : void
+ bp_VerifyUnshippedOrders() : void
+ bp_LoadOrder() : void
+ bp_AdjustTransferOrder() : void
+ bp_CancelTransferOrder() : void
+ bp_PickProduct() : void
+ bp_RecordCompletedOrder() : void
+ bp_CancelTransportOrder() : void
+ bp_ApplyExternalDemand() : void
«Service»
Source_Service
«Resource»
+ externalSupply: Forecast
+ transferOrders: TransferOrder
+ inboundOrders: TransferOrder
«ServiceOperation»
+ inform_DeliveryNotice: SCMessage
+ inform_ModifyOrder: SCMessage
+ inform_TransportArrival: SCMessage
+ inform_CancelTransferOrder: SCMessage
«BusinessProcess»
+ bp_UpdateTransferOrder() : void
+ bp_ReceiveProduct() : void
+ bp_ApplyExternalSupply() : void
+ bp_CancelTransferOrder() : void
«Service»
Transport_Service
«Resource»
+ vehicles: Vehicle
+ transportOrders: Transport Order
«ServiceOperation»
+ inform_EndLoad: SCMessage
+ inform_EndUnload: SCMessage
+ inform_CancelTransport: SCMessage
+ request_TransportOrder: SCMessage
«BusinessProcess»
+ bp_StartTransportOrders() : void
+ bp_CheckAvailability() : void
+ bp_CancelTransportOrder() : void
+ bp_UpdateTransportOrder() : void
+ bp_GoToNextDestination() : void

Fig. 5. Execution Services detail.

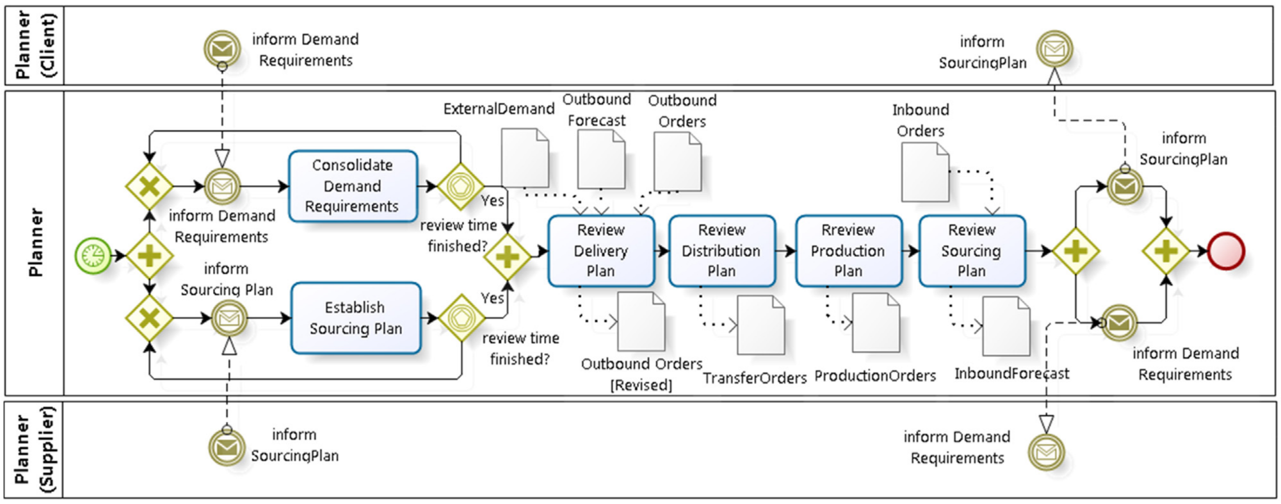


Fig. 7. Collaborative Business Process for Planning Delivery Orders.

implemented. The planning business process is periodically triggered and initiates with a review period where demand requirements are provided by clients and consolidated to generate an outbound forecast. And the current sourcing plan is established by collecting inbound orders by all suppliers. Once the review period for input information is finished, a plan review process is triggered updating outbound orders, internal transfer orders, production orders, and an inbound forecast. Updated outbound orders are communicated to all clients by invoking their respective *inform_SourcingPlan* service operation; and the updated inbound forecast is communicated to the corresponding suppliers by invoking their corresponding *inform_DemandRequirements* service operation.

Note that this process is a generalized one-pass review cycle by each member in the supply chain and that more than one review cycle may be necessary to synchronize the whole supply chain. Different synchronization strategies can be implemented in the specific logic that generates updated outbound orders given the outbound forecast and the updated inbound forecast given the current proposed sourcing plan.

In accordance with this collaborative business process for planning, a functional component *Plan_Service* was defined to encapsulate internal activities and expose public operations needed to support interactions. Details of this functional component are shown in Fig. 8.

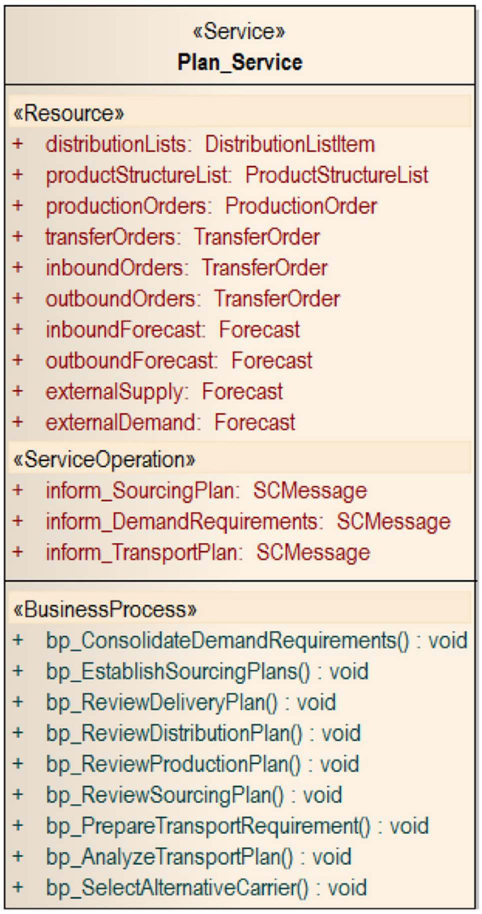
With the introduction of this planning functional component whose objective is the definition of orders to be executed by execution services, a complete separation of execution, control, and decision making activities is achieved.

As the planning business process is also simulated, there are periodical events that trigger the planning review processes. At this moment, a process (*bp_ConsolidateDemandRequirements* in the *Plan_Service*) is run to estimate the future demand. The result of this process is non-deterministic since we modeled a varying error in the forecast. For instance, when forecasting the demand of month 6 in the first month, the error will be higher than forecasting the same period in the review of month five.

The planned transfer orders will be reviewed accordingly, generating a new plan every time the review process is triggered and every time using new forecast estimations.

The actual realization of the demand is simulated by the implementation of the process *bp_ApplyExternalDemand* in the *Deliver_Service*. This method is expected to generate a value for the demand of every item (for instance, by random drawing form a normal distribution around the mean demand for the item in the period).

In this way, every simulation run produces different planning results (different set of planned orders) and different demand realization results therefore resulting in different inventory projections and performance metrics.


«Service»
Plan_Service

«Resource»
+ distributionLists: DistributionListItem
+ productStructureList: ProductStructureList
+ productionOrders: ProductionOrder
+ transferOrders: TransferOrder
+ inboundOrders: TransferOrder
+ outboundOrders: TransferOrder
+ inboundForecast: Forecast
+ outboundForecast: Forecast
+ externalSupply: Forecast
+ externalDemand: Forecast

«ServiceOperation»
+ inform_SourcingPlan: SCMessage
+ inform_DemandRequirements: SCMessage
+ inform_TransportPlan: SCMessage

«BusinessProcess»
+ bp_ConsolidateDemandRequirements() : void
+ bp_EstablishSourcingPlans() : void
+ bp_ReviewDeliveryPlan() : void
+ bp_ReviewDistributionPlan() : void
+ bp_ReviewProductionPlan() : void
+ bp_ReviewSourcingPlan() : void
+ bp_PrepareTransportRequirement() : void
+ bp_AnalyzeTransportPlan() : void
+ bp_SelectAlternativeCarrier() : void

Fig. 8. Planning Service detail.

5. Implementation

A prototype of the presented framework was implemented by using the hybrid discrete event and agent-based simulation platform Anylogic 6.8 [21]. This commercial platform provides object-oriented java-based programming with some already available high level constructors for representing communication-enabled agents, graphical visualization of simulation results, and simulation experiment management capabilities. This platform was selected for it allowed a quick implementation of the artifacts in the framework and was used for concepts proof and validation. Nevertheless, we would like to remark that the proposed framework could be implemented in any other simulation platform providing basic agent oriented simulation support such as message-based communication coupled with message-triggered behavior.

The implementation of framework' components in terms of Anylogic entities is summarized in Table 2.

There are plain java classes for supporting the communication data structure, active objects for those entities having graphical visualization and active objects of type agent for entities requiring agent messaging capabilities.

SCMember is the main agent*;* and all functional components were implemented as active objects of type agent, exploiting the ability of active objects to be composed of other embedded active objects in a hierarchical way. By using this composition, we allow for members with different supply chain capabilities to be configured by adding functional components (as active objects) to the main *SCMember* agent. For implementing the services, we used the concept of "port" that is embedded in the platform a message queue for receiving invocations from other agents. A message queue is a standard feature of any agent-based platform supporting messaging. We followed a naming convention for the port that matches the operation name of the service. In this way, the service invocation is done by sending a message to the corresponding port of the agent containing an object SCMessage that includes the business documents required by the service operation. An example of a fully embodied SCMember with all functional components included as services is depicted in Fig. 9. In the left side panel, all implemented components are available to be reused in configuring a new SCMember with specific capabilities.

Every SCMember has a collection of *Locations* with each *Location* having a collection of *InventoryItems.* These collections are instantiated at run time by reading the desired configuration from a database file. A similar approach was adopted for initializing the distribution list and product structure. This provides the framework with a very flexible and efficient way to represent different supply chain configurations with essentially the same underlying model.

The framework includes the computation of standard indicators for measuring the supply chain performance at different levels in a hierarchical way. Every *InventoryItem* component records its delivered and stock-out quantities (both for internal transfers and external customer demand) and the average inventory. Every *Location* summarizes these indicators for all its controlled inventory items and every *SCMember* summarizes further for all its controlled locations. Additionally, every *SCMember* records the order's execution performance by summarizing orders' execution as perfect, delayed, adjusted, and canceled (stating both number of orders and quantities). Finally, the global model summarizes the indicators for all *SCMembers* to provide global supply chain metrics.

6. Application and evaluation of the modeling framework

In the previous sections, we have described a framework designed to support the analysis of collaborative relationships in supply chains by means of agent-based simulations.

In this section, we will discuss its application to a case example with the aim of verifying the fulfillment of the identified requirements and evaluating its capabilities.

The case example represents an actual supply chain of dairy products covering the central region of Argentina. It involves a manufacturing site for processing raw milk to produce a variety of dairy products. For the purpose of this example, four main families of products are considered: Powder Milk (PM), Long-life Milk (UHT), Yogurt, and Fresh Cheese. The first two families, PM and UHT, do not require refrigeration during storage and transportation, and use different warehousing and transportation resources from those required by fresh products.

There are 9 regional distribution centers capable of storing and delivering these four products to satisfy local demand. Each distribution center has two separate storage facilities (one for dry products, one for fresh). Products are shipped by trucks from the plant; and the transit time ranges from hours within the same day

Table 2
Implementation of the framework components.

Component	Implementation	Component	Implementation
Structural		Functional	
SCMember	Active Object-Agent	Deliver_Service	Active Object-Agent
Location	Active Object	Make_Service	Active Object-Agent
InventoryItem	Active Object	Plan_Service	Active Object-Agent
Resources		Source_Service	Active Object-Agent
distributionLists	Collection of DistributionListItem	Transport_Service	Active Object-Agent
externalDemand	Collection of Forecast	Business Objects	
externalSupply	Collection of Forecast	Destination	Java Class
inboundForecast	Collection of Forecast	DistributionListItem	Java Class
inboundOrders	Collection of TransferOrder	Forecast	Java Class
outboundForecast	Collection of Forecast	OrderItem	Java Class
outboundOrders	Collection of TransferOrder	ProductionOrder	Java Class
productionOrders	Collection of ProductionOrder	ProductStructureList	Java Class
productStructureList	Collection of ProductStructureList	SCMessage	Java Class
transferOrders	Collection of TransferOrder	SourcingPlan	Java Class
vehicles	Collection of Vehicle	Timing	Java Class
		TransferOrder	Java Class
		TransportOrder	Java Class
		Vehicle	Java Class

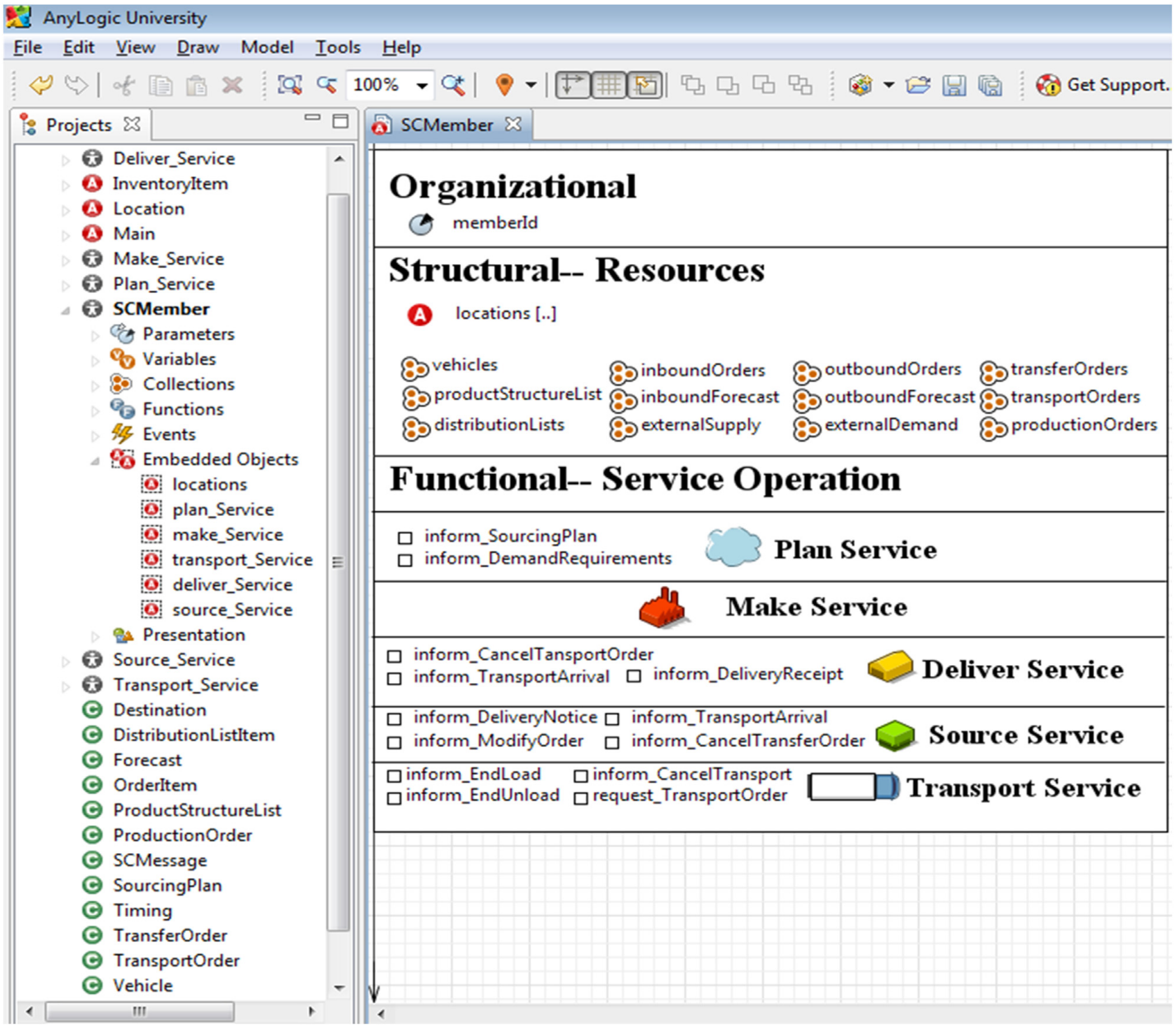


Fig. 9. A fully embodied SCMember with all functional services.

to two days for the most distant destinations. The example supply chain is depicted in Fig. 10.

The manufacturing plant has on-site storage capabilities for raw milk and the four products. The production process has been modeled at the level of individual production lines, converting raw milk into the final product ready to be shipped. The final demand of each distribution center was modeled by an average daily demand for the whole horizon subject to stochastic variability provided by a normal distribution. This demand is estimated during the planning process and captured by a daily forecast with some stochastic error which increases with the horizon, i.e., the later in future the prediction, the higher the forecast error.

This supply chain was modeled with the proposed framework for illustrating its capabilities to represent and evaluate the performance of two different planning coordination strategies between the manufacturing plant and the various distribution centers. In the first strategy, distribution centers conduct an independent planning by forecasting their external demand and placing orders to the manufacturing site. In the second one, coordination is centralized, and the replenishment of all distribution centers (DCs) is decided by a single decision maker who does a global distribution planning based on the observed total external demand. This second strategy represents a vendor-managed inventory collaboration. The impact of planning horizon and review frequency was also investigated, leading to the formulation of four cases:

- Case 1: DCs independent ordering with a weekly review and placing orders for the following week.
- Case 2: DCs independent ordering planning for the next 180 days.
- Case 3: Centralized replenishment planning for 180 days without revision.
- Case 4: Centralized replenishment planning for 180 days with revisions every 30 days.

To represent the strategy where distribution centers are independent, each of them was modeled as a separate SCMember having a two storage locations, one for dry products and one for fresh products. The manufacturing site was represented with another SCMember with three storage locations, raw material, plant dry products, and plant fresh products. In this way, each distribution center behaves as a client of the manufacturing site, performing its internal planning process based on their own estimated demand, informing the required replenishments as

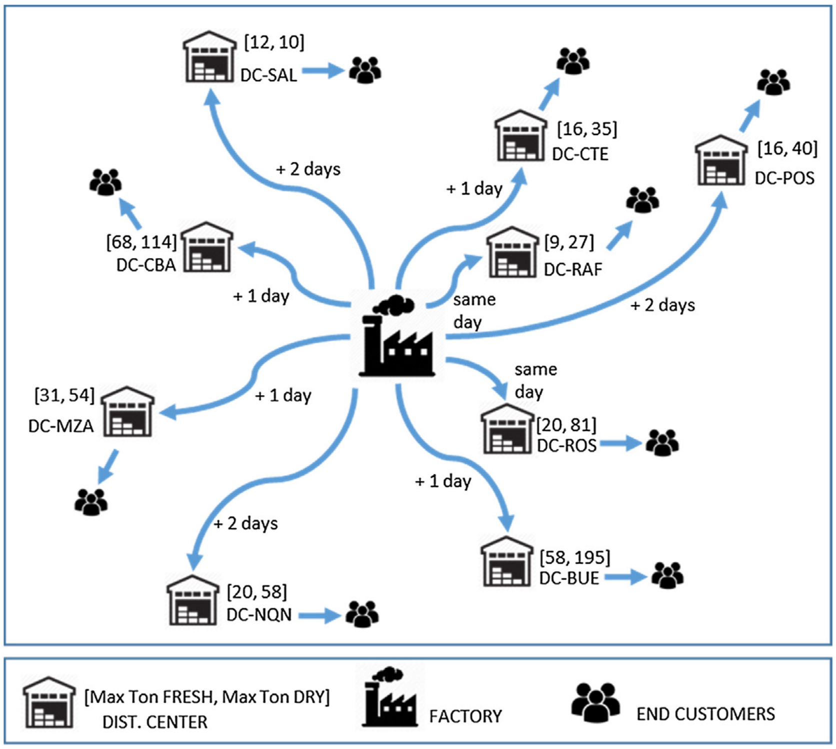


Fig. 10. Dairy products supply chain example.

demand requirements to the manufacturing site, and receiving a sourcing plan after the manufacturing site conducts its own planning process. For the manufacturing site, each distribution center behaves as a separate customer.

In the case of centralized planning, there is just one SCMember having two locations for each distribution center plus three locations for the plant. The planning process estimates the external demand for all distribution centers and generates a global distribution plan for all internal locations as a set of transfer orders. In all cases, transportation service was provided by the manufacturing site. Simulations were conducted for a total time horizon of 180 days. Every simulation is presented with random values for external demand and forecast error. The simulation platform was set up on a computer with Intel Core 2 Quad CPU at 2.4 GHz running Windows XP.

Simulation results for one particular inventory item in the supply chain are illustrated in Fig. 11. On the bottom of the figure, the behavior of demand can be observed. The green line indicates the forecasted value that was used in the planning process; and the orange line stands for the actual realized demand. In order to reproduce realistic conditions, the forecast process was modeled in such a way that error increases with the predicted horizon, and therefore the deviation of the forecasted value from the actual average demand is randomly higher for the periods ahead. Simulation results also include the tracking of inventory evolution as a result of executing all inbound and outbound orders, and a summary of performance indicators for a particular inventory item.

Performance indicators are hierarchically aggregated for every location and for all locations of every *SCMember*, and they can be inspected at any level. For instance, Fig. 12 shows the aggregated indicators at the root of one member.

For each case, a simulation experiment including 30 repetitions with different values of random parameters was conducted for collecting the expected values of performance indicators. This number of repetitions was found enough to provide a stable value of the averaged indicator. A summary of performance indicators for all cases including CPU time is presented in Table 3.

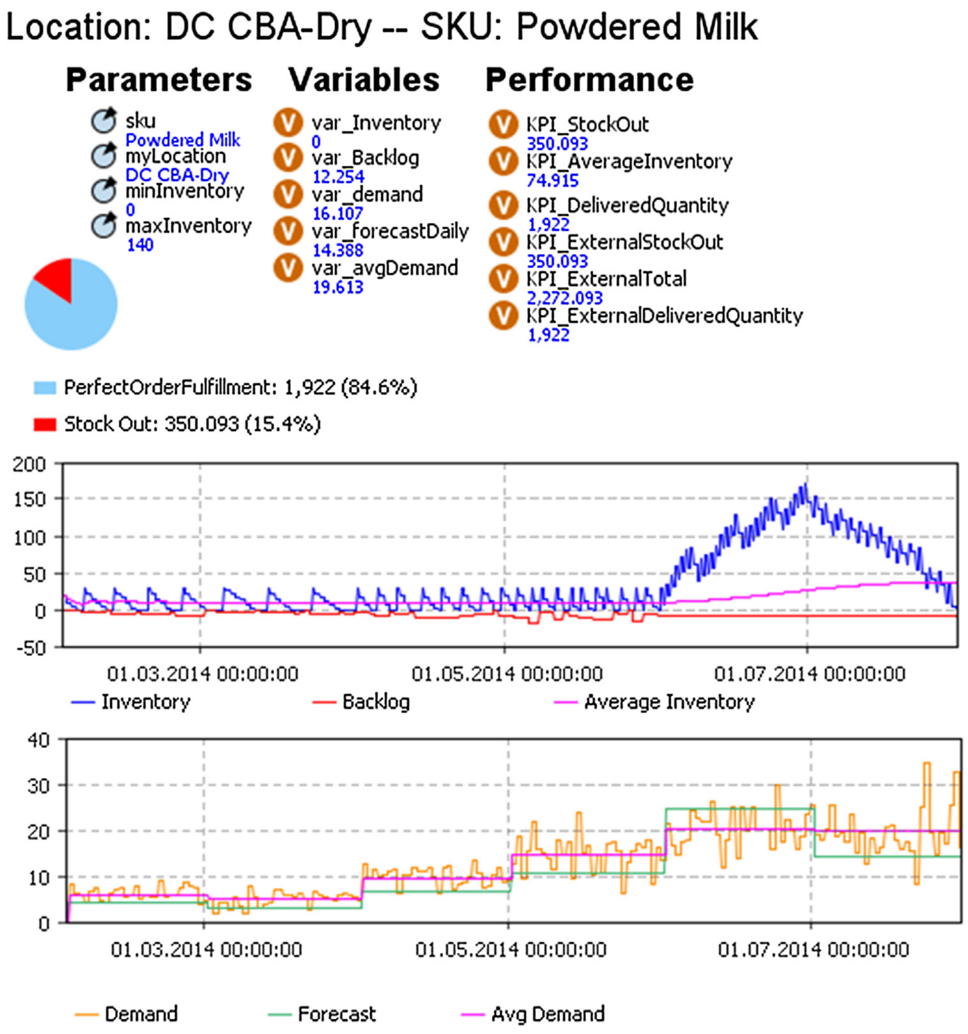
In this comparison, it can be observed that although the four cases were facing the same external demand over the 180-day horizon, its fulfillment, and therefore the overall service level at the DCs is quite different from the independent weekly ordering review to the centralized, with a 30-day revision, showing the combined benefits of centralized decision making and increased planning horizon with reviews. Note that a similar performance is achieved with both independent and centralized strategies when orders are decided for the whole horizon without revisions.

A similar behavior is observed for the performance of the total supply chain average service level (measured as the total fulfilled tons fulfilled vs. total planned tons planned, including production and transfers) and the average inventory.

Results also show that when doing independent weekly reviews, the delivery performance of transfer orders is very high, almost as good as in centralized planning (manufacturing site is able to deliver as requested), therefore the reduced performance to fulfill external demand is a reflection of a too short planning horizon (very low forecast error but reduced future demand visibility).

It should be highlighted that for generating the four cases, the same framework components were utilized an instantiated from a single configuration file indicating different members, their structure and functional components, and their product structure and distribution list configuration. Components were reutilized without changes validating the framework flexibility to be adjusted to very different supply chain configurations.

For instance, when distribution centers were configured as independent SCMembers, they were automatically given with the ability to place demand requirements to the manufacturing site and receive a sourcing plan after the manufacturing site has consolidated its demand and performed its planning process. The resulting orders were treated by the manufacturing site as purchase orders from a client.


Location: DC CBA-Dry -- SKU: Powdered Milk

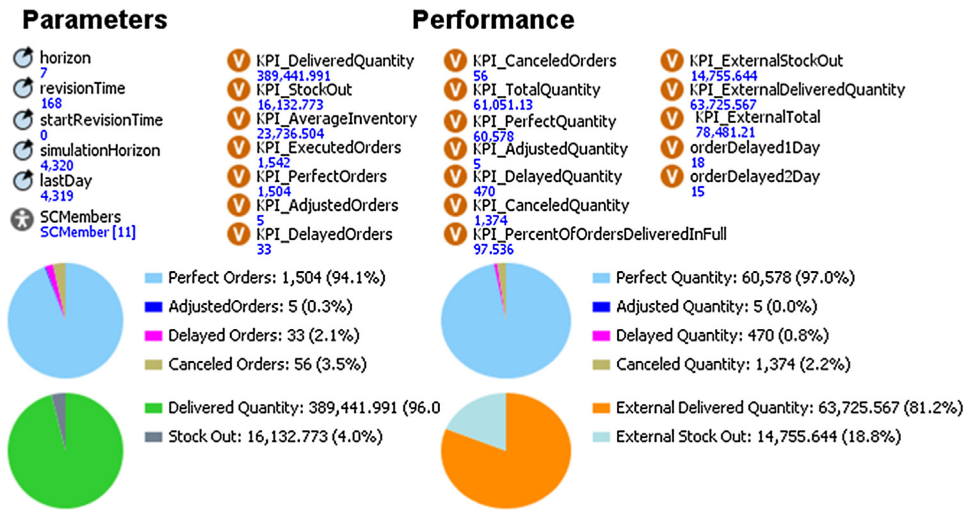
Parameters	Variables	Performance
sku: Powdered Milk	var_Inventory: 0	KPI_StockOut: 350.093
myLocation: DC CBA-Dry	var_Backlog: 12.254	KPI_AverageInventory: 74.915
minInventory: 0	var_demand: 16.107	KPI_DeliveredQuantity: 1,922
maxInventory: 140	var_forecastDaily: 14.388	KPI_ExternalStockOut: 350.093
	var_avgDemand: 19.613	KPI_ExternalTotal: 2,272.093
		KPI_ExternalDeliveredQuantity: 1,922

PerfectOrderFulfillment: 1,922 (84.6%)
Stock Out: 350.093 (15.4%)

Fig. 11. Simulation results for one particular inventory item in the supply chain.

When distribution center warehouses were configured as locations of the same centralized SCMember, they were automatically included in the internal distribution replenishment planning process; and the resulting orders were treated as transfer orders between internal locations. The frequency and horizon of the planning process are parameters of the corresponding service and can be easily changed to reproduce different planning strategies.

It should be also remarked that all interactions among SCMember instances (representing an autonomous supply chain entity) are strictly conducted through message exchanges by


Parameters

horizon	7
revisionTime	168
startRevisionTime	0
simulationHorizon	4,320
lastDay	4,319
SCMembers	SCMember [11]

Performance

KPI_DeliveredQuantity	389,441.991
KPI_StockOut	16,132.773
KPI_AverageInventory	23,736.504
KPI_ExecutedOrders	1,542
KPI_PerfectOrders	1,504
KPI_AdjustedOrders	5
KPI_DelayedOrders	33
KPI_CanceledOrders	56
KPI_TotalQuantity	61,051.13
KPI_PerfectQuantity	60,578
KPI_AdjustedQuantity	5
KPI_DelayedQuantity	470
KPI_CanceledQuantity	1,374
KPI_PercentOfOrdersDeliveredInFull	97.536
KPI_ExternalStockOut	14,755.644
KPI_ExternalDeliveredQuantity	63,725.567
KPI_ExternalTotal	78,481.21
orderDelayed1Day	18
orderDelayed2Day	15

Perfect Orders: 1,504 (94.1%)
AdjustedOrders: 5 (0.3%)
Delayed Orders: 33 (2.1%)
Canceled Orders: 56 (3.5%)
Perfect Quantity: 60,578 (97.0%)
Adjusted Quantity: 5 (0.0%)
Delayed Quantity: 470 (0.8%)
Canceled Quantity: 1,374 (2.2%)
Delivered Quantity: 389,441.991 (96.0
Stock Out: 16,132.773 (4.0%)
External Delivered Quantity: 63,725.567 (81.2%)
External Stock Out: 14,755.644 (18.8%)

Fig. 12. Aggregated performance indicators for a member.

Table 3
Summary of performance indicators.

		Independent Ordering		Centralized replenishment	
		orders placed every week for next week	orders placed for next 180 days	180 days without revision	180 days with every 30 days revision
External Demand (tons)	Total	78561	78711	78628	78479
	Fulfilled	64128	72307	72560	74697
	Missed	14433	6404	6068	3782
DCs Service Level	%	81.6	91.9	92.3	95.2
Inventory (tons)	Average	22958	15012	14889	12567
Supply Chain Planned (tons)	Total	405162	419497	419152	413688
	Fulfilled	390128	410801	411130	409848
	Missed	15034	8696	8022	3840
Supply Chain Service Level	%	96.3	97.9	98.1	99.1
# of Transfer Orders	Perfect	1470	2256	2248	2398
	Delayed 1 day	10	132	138	24
	Delayed 2 days	14	253	248	11
	Canceled	29	186	180	6
Orders Delivered In Full	%	98.2	84.9	84.8	98.5
Transfer Orders (tons)	Perfect	60621	68768	68759	69783
	Delayed	375	4762	4962	525
	Canceled	598	2280	1940	54
CPU Time (in seconds)	1 run	4,7 s	5,8 s	8,4 s	23,1 s
	Experiment (30 repetitions)	129,6 s	162,1 s	196,2 s	547,7 s

invoking their exposed services interface. All internal variables and processes are private for each member. This is an essential design feature of our framework that enables interoperation of different implementations of internal business processes.

For instance, in cases where DCs are independent SCMembers, Fig. 13 shows interactions among every DC, the Factory, and the Carrier. The first part of the sequence diagram displays interactions held at every planning review. The sequence starts with the independent DC informing the factory about its demand requirements; then the factory consolidates requirements for all its customers, performs its planning review and, after that, it will request a transport order to carrier and inform DC of the resulting sourcing plan.

The second part of the sequence diagram in Fig. 13 shows interactions held later on at the moment of delivery order execution. The sequence starts with the Carrier informing the transport arrival in the factory. The factory loads the order and informs the Carrier about the end of load operations and informs DC the delivery notice. After the transit period, the Carrier informs the transport arrival in the DC, the DC unloads the order and informs the Carrier about the end of unload operations and the factory about the delivery receipt.

7. Conclusions

Benefits of information sharing, increased visibility, and collaborative coordination in the supply chain performance have been largely discussed in the literature both at a theoretical level and at analytical results of simulation experiments for specific scenarios.

Several schemes for collaboratively managing a supply chain have been proposed. Examples of them are: Vendor Managed Inventory (VMI), Continuous Replenishment (CR), Collaborative Planning, Forecasting and Replenishment (CPFR), Quick Response (QR), and Efficient Customer Response (ECR).

Despite the evidence that indicates the overall improvement achieved by such collaborative engagements, it is not easy in practice to assess how those benefits can be fairly distributed among independent partners. The establishment of a sustainable collaborative agreement depends on the mutual understanding of benefits and costs of operating under the proposed collaboration.

Only simulations can provide accurate estimations of those benefits and costs for specific scenarios. The construction of a simulation model for a realistic collaborative supply chain, including the behavior of execution and planning operations, may become an overwhelming hurdle for business analysts in producing an efficient evaluation.

In this work, we have identified several features a simulation framework should have in order to support the modeling of collaborative supply chain arrangements. We analyzed some of the relevant proposals in the literature under their ability to provide with those features.

Using those features as design requirements, we have developed a new simulation framework that was specifically conceived for supporting the modeling of collaborative relationships in supply chain for performance assessment using simulation.

Our proposed framework fulfills all these requirements:

I *Provide reusable components that can be easily assembled to set a wide variety of supply chain scenarios among independent partners.* We have shown that a basic set of generalized components were successfully reutilized to represent very different collaboration relationships as the ones illustrated in the case study. These components were organized under three different categories, functional, organizational and structural. Functional components represent supply chain processes or

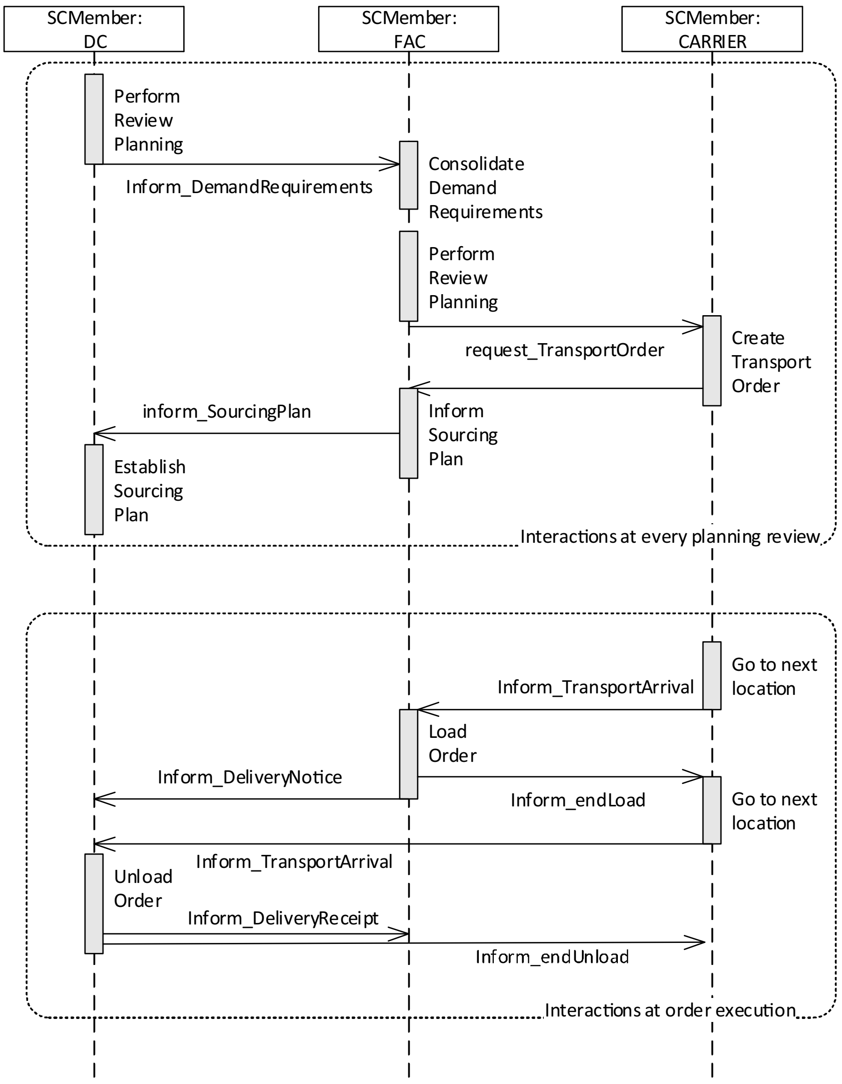


Fig. 13. Interaction messages for the case of independent DCs.

basic activities within a collaboration setting and were modeled as services. Organizational components were modeled as agents (they represent business partners within a collaboration). Structural components representing the physical structures or facilities in a supply chain were captured by entities such as location and inventory items. All these components are generalized and can be re-used to configure different scenarios without changing the implementation as shown in the case study.

II *Allow for establishing dynamic links among partners without requiring a predefined network structure*. The presented framework does not require an *a priori* definition of interaction links among partners neither any predefined supply chain structure. This essential requirement was addresses by adopting an agent-based approach and stablishing that every organizational entity is modeled as an independent agent that interacts with the others only through message exchange, invoking a series of exposed services interfaces. It is possible for agents to establish new interactions during simulation run time by using typical multi-agent registration and naming services.

III *Adopt a business process-oriented perspective to reflect activities in the supply chain, following some established reference model known to most business analysts*. Our framework makes a clear separation of aspects for describing structural representation and functional behavior. For the functional behavior, a business process-oriented perspective was utilized to encapsulate different supply chain activities. In fact, the business process becomes an important artifact in our framework. For organizing these activities, we followed a well-known reference model of SCOR [4]. The framework uses SCOR as the source of supply chain activities, their semantics and their organization for collaborative purposes into business process models. These business process models are defined using a standard process language (BPMN) [2].

IV *Provide for explicit representations of control and decision making activities separated from execution activities*. In the definition of functional components, activities have been carefully separated, distinguishing between those of the execution level and those belonging to control or decision making. The two layers interact by using the orders concept. Orders are defined in planning level activities and passed to execution level activities as instructions. This separation allows that the same execution activities can be used by completely different implementations of planning logics. Therefore, the only connection between decision making processes (planning activities) and execution processes are the set of orders.

V *Support the interaction among members by using message-based and document-oriented protocols that resemble actual business* *interactions*. All interactions among members have been standardized through interaction protocols, implemented as a series of services exposed by members in order to be invoked. These services define standard messages including business documents to be exchanged. In this way, it is possible to have different implementations of internal processes of each member, granting interoperability and providing a natural control on the information visibility that is shared among members. An analysis of the interactions of the different lanes in the business processes was used to identify the required service interfaces as suggested in traditional service oriented design methodologies.

A prototype of the proposed simulation framework was implemented by using a general purpose agent-based simulation platform, Anylogic 6.8 [21], which allowed for a quick validation on the case study.

The application of the framework for modeling an actual supply chain demonstrated the ability of the proposed generalized components to represent different planning coordination strategies and collect simulated results of each scenario performance under varying conditions of the external demand.

Functional components of the framework also included modeling and simulation of forecasting and planning activities. In this implementation, the forecasting activity was modeled in such a way that the forecast error increased with the outlook period and the planning activity implemented standard DRP (Distribution Requirements Planning) and MRP (Material Requirements Planning) procedures for deciding replenishment orders during the simulation at every review period.

The comparison of different cases confirmed the expected qualitative behavior and validated its usage as an assessment tool for obtaining quantitative measurements of different collaboration strategies.

Future work will be focuses on designing new generalized components for a more efficient support for modeling transport service planning.

References

[1] J.M. Swaminathan, S.R. Tayur, Models for supply chains in e-business, Manage. Sci. 49 (10) (2003) 1387–1406.

[2] H. Chan, F. Chan, A review of coordination studies in the context of supply chain dynamics, Int. J. Prod. Res. 48 (10) (2010) 2793–2819.

[3] U. Ramanathan, Performance of supply chain collaboration—a simulation study, Expert Syst. Appl. 41 (1) (2014) 210–220.

[4] Supply Chain Council, Supply-Chain Operations Reference-model SCOR Version 6.1 Overview (2004).

[5] OMG, Object Management Group, Business Process Model and Notation (BPMN) 2.0 (2011).

[6] Service Oriented Architecture Modelling Language (SoaML) Specification (Version 1.0.1) (2012).

[7] Q. Long, W. Zhang, An integrated framework for agent based inventory–production–transportation modeling and distributed simulation of supply chains, Inf. Sci. (N.Y.). 277 (September) (2014) 567–581.

[8] O. Labarthe, B. Espinasse, A. Ferrarini, B. Montreuil, Toward a methodological framework for agent-based modelling and simulation of supply chains in a mass customization context, Simul. Modell. Pract. Theory 15 (February (2)) (2007) 113–136.

[9] G. Pundoor, J. Herrmann, A hierarchical approach to supply chain simulation modelling using the Supply Chain Operations Reference model, Int. J. Simul. Process Modell. 2 (3–4) (2006) 124–132.

[10] D.J. van der Zee, J.G.a.J. van der Vorst, A modeling framework for supply chain simulation: opportunities for improved decision making*, Decis. Sci. 36 (1) (2005) 65–95.

[11] J.H. Pratt, D.B. Kamath, M. Mize, The separation of physical, information, and control elements for facilitating reusability in simulation modeling, Int. J. Comput. Simul. 4 (3) (1994).

[12] S.C. Karacal, J.H. Mize, A formal structure for discrete event simulation. Part I: modeling multiple level systems, IIE Trans. 28 (9) (1996).

[13] M. Ganesh, S. Raghunathan, C. Rajendran, The value of information sharing in a multi-product, multi-level supply chain: impact of product substitution, demand correlation, and partial information sharing, Decis. Support Syst. 58 (2014) 79–94.

[14] Q. Long, An agent-based distributed computational experiment framework for virtual supply chain network development, Expert Syst. Appl. 41 (9) (2014) 4094–4112.

[15] M.D. Rossetti, H. Chan, A prototype object-Oriented supply chain simulation framework, Proceedings of the 2003 Winter Simulation Conference (2003) 1612–1620.

[16] D.C. Chatfield, T.P. Harrison, J.C. Hayya, SISCO: An object-oriented supply chain simulation system, Decis. Support Syst. 42 (1) (2006) 422–434.

[17] D.C. Chatfield, J.C. Hayya, T.P. Harrison, A multi-formalism architecture for agent-based, order-centric supply chain simulation, Simul. Model. Pract. Theory 15 (February (2)) (2007) 153–174.

[18] D.C. Chatfield, T.P. Harrison, J.C. Hayya, SCML: an information framework to support supply chain modeling, Eur. J. Oper. Res. 196 (2) (2009) 651–660.

[19] S. Umeda, F. Zhang, A simulation modeling framework for supply chain system analysis, Proc. 2010 Winter Simul. Conf. (2010) 2011–2022.

[20] Q. Long, W. Zhang, An integrated framework for agent based inventory – production – transportation modeling and distributed simulation of supply chains, Inf. Sci. (N.Y.) 277 (2014) 567–581.