An Ontology-Based Event-Driven Architecture for Integrating Information, Processes and Services Applied to International Trade

Sietse Overbeek, Marijn Janssen, and Yao-Hua Tan

Faculty of Technology, Policy and Management,
Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands
{S.J.Overbeek,M.F.W.H.A.Janssen,Y.Tan}@tudelft.nl

Abstract. In global supply chains, many public and private organizations collaborate in order to succeed in transporting goods from the seller to the buyer. Given the dynamicity of global supply chains it is hard to predict which information is needed by whom at which point in time which oftentimes causes service delivery issues. Integrating relevant information, processes and services prevents deterioration in service provisioning caused by missing information required for processes that need to be executed to supply services. In this paper, an ontology-based event-driven architecture is described for integrating information, processes and services that acts as a mechanism to coordinate service delivery. The architecture is illustrated in the context of a global supply chain of plastic toys, where it is shown how the architecture enables the availability of valuable information based on events which positively influences the delivery of a barge planning service.

Keywords: CEP, EDA, global supply chain, ontology, service delivery.

1 Introduction

In global supply chains, many clusters of public and private organizations work together to make sure goods are safely and successfully transported from the selling party to the buying party. The composition of organizations that collaborate in these chains changes over time as new organizations might enter a chain while others might withdraw dependent of, for example, the goods that are traded in a specific trade lane. Flexible mechanisms are needed to coordinate service delivery in supply chains not only because of this kind of dynamicity, but also because complex client demands are subject to change in such chains and because supply and demand of services needs to be matched [1, 2]. Well-known mechanisms to coordinate service networks are service-oriented architectures (SOAs, see e.g. [1-3]) and event-driven architectures (EDAs, see e.g. [4-6]). Each organization that offers services in a supply chain can make its services accessible as Web services. Coupling services on the interface level is technically feasible, however, specific attention should be paid to issues of integrating the required information and underlying processes for service delivery as this could lead to deterioration in service provisioning when ignored [2]. Infrastructural support for process, information and service integration is one of the grand challenges as mentioned in the service-oriented computing research roadmap [1]. The ontological models and EDA that are presented in this paper will contribute to solving this challenge by integrating the architectural information, process, and service layers. Event-driven architectures (EDAs) enable the production, detection, consumption of, and reaction to business-critical events [4]. An event is defined as 'a significant change in state', while event consumers are those entities responsible for acting on these changes [7]. Hence, there is no need to draw out in detail the activities involved in a process which may even be impossible in complex scenarios. A motivating real-life example is used to illustrate how planning issues in a global supply chain of plastic toys are dealt with by the EDA presented in this paper.

The paper is structured as follows. Section 2 provides an overview of related work. The empirical motivation for the EDA is explained in section 3. Section 4 shows the high-level design of the architecture, which is further elaborated in section 5 by means of a formal event model and ontological models of the information, process, and service layers. The illustration of the architecture in the toys trade lane is presented in section 6. Finally, section 7 presents the conclusions and plans for future research.

2 Related Work

In this section, three categories of related work are discussed, which are: (1) studies related to the core technology of the EDA as part of this paper, which is complex event processing (CEP), (2) studies that combine event-driven concepts with service-oriented concepts, and (3) studies concerning models to integrate architectural layers. These three categories of related works will be subsequently dealt with in this order.

CEP is a core technology that is suitable for dealing with complex event streams related to information, processes and services [8]. This is done by processing and analysing multiple simple events from possibly distributed sources, with the objective of extracting semantically richer events from them in a timely, online fashion [9]. CEP effectively supports the implementation of 'sense and respond' behaviors, as it enables to extract meaningful events from raw data streams [9]. CEP is also part of the architecture as designed in this paper which is elaborated in section 4. An EDA for road traffic management systems is proposed in [4], for example, where CEP is used to extract meaningful events from raw measurements provided by traffic sensors and to deliver resulting events as input to traffic control systems for decision support. Experiment CEP techniques in a radio frequency identification (RFID)-enabled framework for managing hospital data for surgical procedures is presented in [10]. The used CEP engine models basic events and event patterns in hospitals for detecting medically significant events.

The added value of an event-driven architectural style next to SOAs is discussed in [5], as it is stated that dynamic interaction of service providing organizations in networks such as global supply chains trigger a considerable amount of meaningful state changes, i.e., events. The event-driven approach supplements the service-oriented approach by facilitating real-time event processing and distributed service coordination [5]. Another solution for extending the service-oriented architecture style with EDA concepts is presented in [6], showing that combining event-driven and service-oriented concepts does not only make sense in many business applications but also reduces development efforts of the architecture itself.

Merging, supplementing and combining architectural styles is still something different than integrating the different layers of a single architecture by means of ontological models, for example, as has been the case in our approach. Works like, e.g., [3, 4] do include event ontologies to classify event instances, but they are not specifically used to integrate architectural layers as is the case with the EDA in this paper. In earlier work we have used an ontology to generate both computer-interpretable and human-readable requirements for the execution of cross-organizational processes [11, 12]. In [13], ontologies are used to represent collaboration patterns in enterprises, as by using ontologies related concepts and interrelations can be effectively modeled. The presented ontology-based EDA is then used for event detection and reaction on those events in ongoing collaborations in enterprises, while in our case ontological models are used to describe and couple the information, process and service layers which enables to understand the information requirements that follow from executed processes to deliver services. As will be seen throughout the paper and, especially in section 6, integrating the three ontological layers as part of the EDA enables the availability of valuable information based on runtime events and in the mentioned plastic toys supply chain it contributes to solving service delivery issues in the context of supply chain planning.

3 Empirical Motivation

A toys trade lane that goes all the way from a toy factory in China via the Port of Shenzhen to the Port of Rotterdam, the Netherlands and ends in the hinterland in Venlo, the Netherlands serves as the context for the empirical motivation to design an ontology-based EDA for integrating information, processes and services. Toys that are ready after manufacturing are shipped to a warehouse nearby Shenzhen from where they are shipped to the Port of Shenzhen by road. Once loaded on a freighter, the cargo is shipped to the Port of Rotterdam where it is imported in the EU. From there, the cargo will be transported to a warehouse in Venlo. This trade lane is visualized in figure 1.

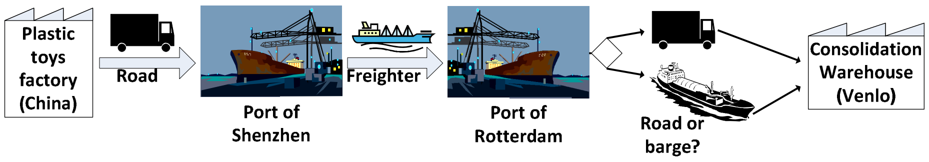

Fig. 1. A trade lane of plastic toys from China to the Netherlands

In such an international trade lane, all kinds of public and private organizations collaborate. These are, for example, providers of logistic services, regulatory authorities such as: customs, the tax administration, the food and consumer product safety authority, but also the consignor (aka the selling party) and the consignee of the toys (aka the buying party). These organizations exchange *information* with each other to make sure the toys are exchanged from the consignor to the consignee in a transparent, reliable and secure way. The Port of Rotterdam is an important decision point in the toys trade lane, because a decision has to be made whether the cargo of toys will be transported by road or by barge to Venlo. The logistic re-planning of the remainder of the cargo transportation route has to be done dynamically in Rotterdam based on the information at hand. Road transport is much more expensive than barge transport and causes considerable emissions of CO_2, but the advantage of road transport is that the destination can be reached faster when compared to barge transport. If an inland barge operator wants to make use of an online barge planning *service* at the Port of Rotterdam, it requires that the information *when* the cargo arrives at Rotterdam and on *which* freighter the cargo is stored needs to be known by the local *port community system* (PCS) two days before the scheduled barge departs from Rotterdam. A PCS is an entity that acts as a neutral hub which offers all kinds of online services to traders in the port such as the barge planning service. The PCS communicates with the terminal operator, which is the entity that loads the inland barges in Rotterdam with containers that have arrived from overseas. The mentioned information is required that early because the barge planning *process* has to be performed as efficiently as possible by only letting barges depart when they are fully loaded. This is why barges are loaded with mixed cargo, for example, a container on a barge may contain boxes with toys but also boxes with automotive parts that need to be dropped off in Venlo as well.

The problem in this outlined scenario is that it is difficult to get the information required for proper barge planning on time in Rotterdam. This is because it is hard to know the exact time a ship will arrive in Rotterdam. With an EDA, events that are triggered when a freighter with toy cargo leaves the Port of Shenzhen, changes its lane or arrives at the Port of Rotterdam can be processed and information can be derived from events that have occurred. A PCS can make use of this information to improve the delivery of their barge planning service. Currently, however, the required information for barge planning is often not available within the two days before the last inland barge departs that would arrive at the desired point in time in Venlo. This causes that often the only option left is to make use of costly and polluting road transport. The information on which freighter the cargo is stored and when cargo arrives would be available for the PCS when each state change of the cargo would trigger an event. Once the required information has been collected to supply the barge planning service, the PCS can perform the process to deliver the service to the barge operator. By integrating information for process execution with process and service descriptions through the EDA presented hereafter, the mentioned service provisioning issues can be remedied.

4 Integrating Information, Processes and Services

In the mentioned real-life example in international trade, the three layers of information, processes and services are of importance. First, information such as: where are the goods, what type of goods are transported, and an export declaration is needed by organizations involved in the toys trade lane. Second, processes are needed to understand who needs to do what in the context of the toys trade lane. For example, customs want to perform physical inspections on the toys to check for health or environmental risks and a terminal operator is responsible for transferring toy cargo from a freighter to an inland barge. Third, services are used to guide the goods from seller to buyer, such as goods ordering, customs clearing, barge planning, etc. An ontology-based event-driven architecture to integrate information, processes and services is shown in figure 2. The figure shows that the three layers are described by means of an ontology. This ontology is modeled as an Object-Role Modelling (ORM) [14] model and is explained in section 5. ORM is, like UML or ER, a conceptual modeling language that can be used for a variety of modeling purposes, such as the modeling of databases or ontologies. A specific advantage of using ORM is that objects are treated as concepts, which makes ORM immune to changes in the model that cause attributes to be remodeled as objects or relationships. The ontology describes the core concepts and relationships between those concepts on each layer and is used to determine how the three layers are interrelated. As can be concluded from figure 2, the information layer feeds the process layer, while the process layer feeds the service layer. Business processes require information for successful execution, otherwise a process cannot be fulfilled.

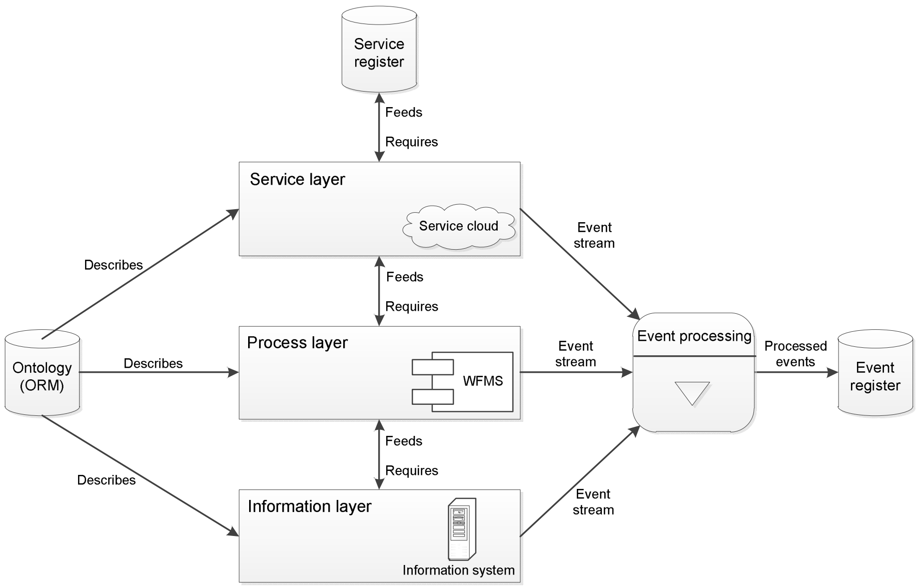

Fig. 2. An ontology-based EDA for integrating information, processes and services

The service layer requires processes to be executed, otherwise the services cannot be supplied to those actors in the toys trade lane that demand services. An example of such an actor is the buyer of the toys. Services are fed into a service register by the service providing organizations. The model shown in figure 2 does not have a separate software layer, as software applications are present in the three layers themselves. Computer-based support is delivered on the information layer in terms of enterprise information systems. Service providing organizations make use of dynamic workflow management systems (WFMSs) to execute, manage and monitor their business processes which is shown on the process layer. These WFMSs need to be dynamic because they need to be able to cope with runtime changes in business processes and process descriptions that are not always predefined. If services are offered in an online form then they are part of the cloud of Web services as is shown on the service layer.

The final parts of the architecture are the event processing part and the event register. Events from each layer are processed and then fed into an event register, which are next to the ontology parts of the architecture to integrate the three layers. Events are dealt with by means of complex event processing. In the next section we will zoom in on the parts of figure 2 that are related to event processing. Events are part of each layer, because the states of information, states of processes and service states change all the time when information is exchanged, business processes are executed and when services are delivered. A main principle of CEP is that events are not independent from each other, but correlated in space and time [4]. An example of correlation in space in the context of the toys trade lane can be events that are triggered at the port of departure in Shenzhen and at the port of arrival in Rotterdam. Those events can be correlated as it concerns toy cargo which could be tracked based on those correlated events. Event correlation in time means that events that are observed at one time instant are indicative of events observed at the next time instant or other future time instants. An example is event information showing that a shipment has departed from its port of departure or when an event that is triggered when a freighter changes its lane. Based on the estimated time of arrival (ETA) at the port of arrival, this event information can be indicative for the moment in time when events are observed that the shipment has indeed arrived at the port of arrival. These kind of correlations between events provide additional information that would remain hidden without time and space correlation of events. This event-driven approach is applicable to the toys trade lane, where supply chain planning in advance is considered inefficient and expensive as dynamic re-planning at the Port of Rotterdam is needed in order to reduce costs and increase efficiency. Having the ability to choose from different transport modalities at the node in Rotterdam means that costs can be lowered and efficiency can be increased which is made possible by applying the CEP mechanism instead of using top-down hierarchical planning.

Figure 3 shows the building blocks of CEP applied to the three layers that are found in figure 2. The event streams that are related to service provisioning, process execution and information exchange are processed by an event processing engine. This engine has to deal with a continuous flow of events. To process such a continuous flow, continuous queries are issued once and then run continuously over the event stream [15]. The event model is used to classify event instances based on their types and is further elaborated in section 5.1. The event processing rules define correlations between events in the form of event patterns and can be expressed by event processing languages based on event algebras or as SQL-like queries over event streams [15]. Events are stored in an event register after the events are processed by the engine, classified by means of the event model and correlated by means of the rules.

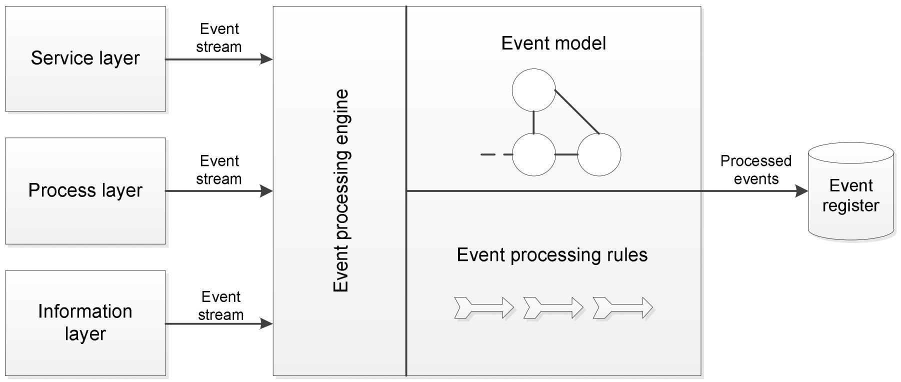

Fig. 3. Complex event processing building blocks related to the three layers, adapted from [4]

A software application based on CEP in the context of the toys trade lane is a dashboard on which it is shown that, for example, twenty containers arrive as one shipment from Shenzhen. Ten of these containers may contain toys that can be transported cheaply to the hinterland by means of barge transport and the other ten containers contain automotive parts that require a means of transport that is quicker than barge transport, as they need to arrive at a point in time at a consolidation warehouse in Venlo, the Netherlands that cannot be reached when a slower way of transport than road transport is chosen. In the next section, we will explain the event model as part of the architecture together with the ontological models.

5 Decomposed Ontological Models and an Event Model

The event model that is elaborated hereafter is comparable to the ontological models related to all three layers, but the event model serves a different purpose, which is the classification of event instances based on their types, event IDs, time- and date stamps, and their coordinates in space. Correlations can be made by the event processing engine based on this model and the event processing rules.

5.1 A Formal Event Model

Figure 4 shows the event model modeled with the ORM language, which is used to classify events so that they can be stored in an event register. In an ORM model, rounded rectangles represent object types (which are the counterparts of classes), while boxes represent relationships between object types. Bold arrows express specialization relationships. The event instance object type is obviously in the centre of the event model. This object type has the most relationships with the other object types in the model. An event instance can be classified as a certain event type. This can be formalized as follows: $\text{EType}: \mathcal{EV} \rightarrow \mathcal{ET}$. The expression $\text{EType}(e) = t$ shows that an event $e \in \mathcal{EV}$ is of the type $t \in \mathcal{ET}$, where $\mathcal{EV}$ is the set of events and $\mathcal{ET}$ is the set of event types. Figure 4 also shows three arrows that are drawn from the object type 'Event Type' to the object types 'Information Event', 'Process Event', and 'Service Event', which implies that each 'Information Event', 'Process Event', and 'Service Event' is also an 'Event Type'. Instances of information, process and service events are streamed from three different layers. The stream equation that captures this is modeled as follows: $\text{Stream}: \mathcal{EV} \rightarrow \mathcal{AL}$. The expression $\text{Stream}(e) = a$ shows that an event $e \in \mathcal{EV}$ is streamed from layer $a \in \mathcal{AL}$, where $\mathcal{AL}$ is the set of layers. However, there is a value constraint on the object type 'Architectural Layer' showing that only the values '`IL`', '`PL`', and '`SL`' are permitted. This means that it can only be modeled that events can be streamed from the information layer (`IL`), the process layer (`PL`), and the service layer (`SL`). These event subtypes form the relationship between the event model and the ontological models of the three architectural layers that are elaborated in the following sections. Because events can be correlated in space by the event processing engine, the ORM event model of figure 4 shows a visualization of the coordination equation. It is possible to reason about the spatial relationships between events if the coordinates are known where events take place. The three-dimensional coordinates of an event can be found by means of the coordinates equation: $\text{Coord}: \mathcal{EV} \rightarrow \mathbb{R} \times \mathbb{R} \times \mathbb{R}$. The coordinates are plotted on a three-dimensional Cartesian coordinate system, with an origin and axes X, Y, and Z.

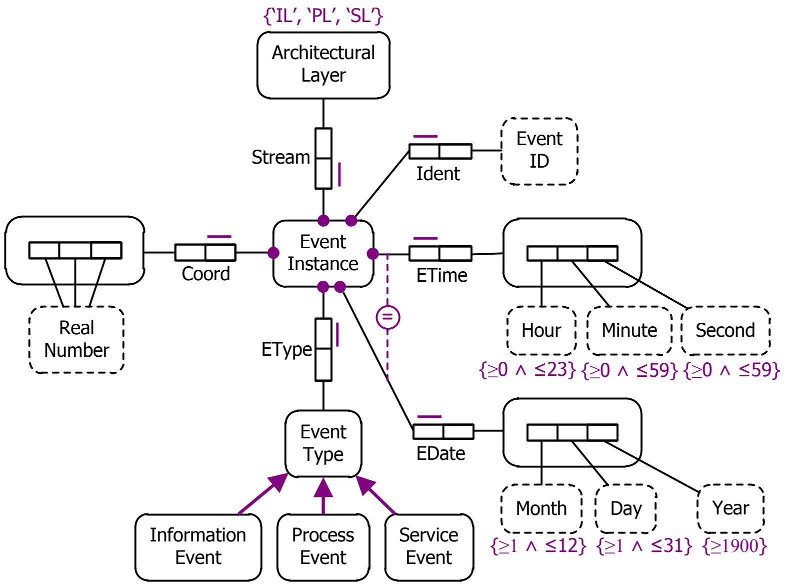

Fig. 4. An ORM-based event model

Events can also be correlated in time next to the correlation of events in space. This is why the event time and event date functions are shown in figure 4. The signatures of these equations are: $\mathrm{ETime}, \mathrm{EDate} : \mathcal{EV} \rightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{N}$. For example, $\mathrm{ETime(e)} = (22, 15, 42)$ expresses that an event instance took place at 22 hours, 15 minutes and 42 seconds. Figure 4 shows that a *set equality* constraint is added to the roles of the event time and date equations that are coupled to the 'Event Instance' object type. This constraint forces that time and date stamps should always be given together when some event instance occurs. Finally, each event instance has a unique ID. Therefore, the event identification equation is introduced to complete the event model: $\mathrm{Ident} : \mathcal{EV} \rightarrow \mathbb{N}$. Now that a formal event model has been explained as part of the architecture for integrating information, processes and services the ontology to describe the three layers will be elaborated.

5.2 Ontological Model of the Information Layer

The ontological model for describing the information layer is shown in figure 5. The model shows that the 'Agent State' object type plays a pivotal role in the ontological model and has the most relationships with other object types. Agents working for service providing organizations require information and, therefore, exchange information during the execution of business processes to deliver services. The basis for the ontological model of the information layer is the theory for demand and supply of information as presented in [16].

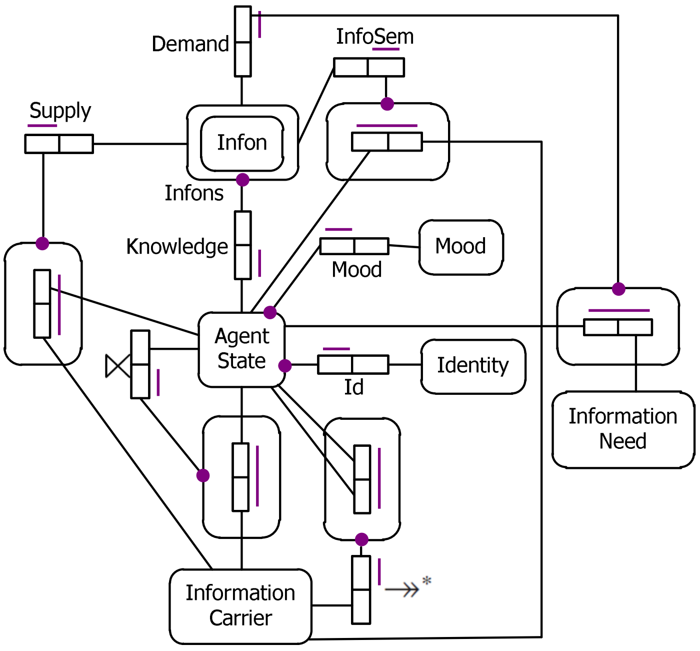

Fig. 5. An ontological model for describing the information layer

First of all, information needed by agents is provided on information *carriers*, such as Web pages, PDF documents, and e-mails. After the information retrieval from a carrier an agent is in a different state, i.e. that agent has absorbed more information than before. Each state belongs to a unique agent, determined by the identity function [16]: $\mathrm{Id} : \mathcal{AS} \rightarrow \mathcal{ID}$, where $\mathcal{AS}$ is the set of agent states and $\mathcal{ID}$ is the set of agent identities. This function shows that the identity of an agent $i \in \mathcal{ID}$ is determined by the agent state. When an agent in state a experiences an information carrier $c \in \mathcal{IC}$, where $\mathcal{IC}$ is the set of information carriers, then this agent will end up in a new state denoted as $a \ltimes c$. This is an expression of the following state change function: $\ltimes : \mathcal{AS} \times \mathcal{IC} \rightarrow \mathcal{AS}$. Note that for readability purposes the notation $a \ltimes c$ is a different notation than $\ltimes (a, c)$ but the semantics of both are the same. The knowledge that an agent has accrued is administrated by the knowledge function [16]: $\mathrm{Knowledge} : \mathcal{AS} \rightarrow \wp(\mathcal{IN})$. The expression $\mathrm{Knowledge}(a) = I$ shows that a set of information particles $\mathcal{IN}$, or *infons* [17] have been accrued by an agent in state a. For example, when a customs officer interprets an export declaration of toys he has acquired information from that declaration, causing him to change to another state.

The mood of an agent is also taken into consideration when designing the ontological model for the information layer, as this may influence how much knowledge is acquired from an information carrier. The moods of agents thus also influence the results of information exchange. The mood function is modeled as follows [16]: $\mathrm{Mood} : \mathcal{AS} \rightarrow \mathcal{MO}$. The mood $m \in \mathcal{MO}$ of an agent in state a can be expressed as $\mathrm{Mood}(a) = m$. Assume that a customs officer in state a interprets information from an export declaration form a couple of times. This form is an information carrier $c \in \mathcal{IC}$. Further assume the following: $\{\mathtt{interested}, \mathtt{informed}, \mathtt{bored}\} \subseteq \mathcal{MO}$. Before interpreting the export declaration the customs officer is interested: $\mathrm{Mood}(a) = \mathtt{interested}$. After reading the export declaration form the customs officer is informed: $\mathrm{Mood}(a) = \mathtt{informed}$. After reading the form another time the officer is bored: $\mathrm{Mood}(a) = \mathtt{bored}$. The customs officer will most probably hardly consume any new information after reading it for a third time, making him bored.

Agent states can also be ordered by means of the experience operator: $\twoheadrightarrow^* : \mathcal{AS} \times \mathcal{AS} \rightarrow \mathcal{IC}$. This means that $a_1 \twoheadrightarrow^* a_2$ can be interpreted as an agent in state $a_1 \in \mathcal{AS}$ ends up in state $a_2 \in \mathcal{AS}$ after experiencing an information carrier [16]. Next to the agents that have a certain role in the context of information exchange, the information itself is transported from an information supplier to an information requester. The supply of information is modeled as follows: $\mathrm{Supply} : \mathcal{AS} \times \mathcal{IC} \rightarrow \wp(\mathcal{IN})$. This function implies that an agent in state a supplies information $I \subseteq \mathcal{IN}$, which is carried by an information carrier c. The potential information that a carrier may provide to an agent can be expressed by using the information semantics function [16]: $\mathrm{InfoSem} : \mathcal{IC} \times \mathcal{AS} \rightarrow \wp(\mathcal{IN})$. For example, the overall information content $J \subseteq \mathcal{IN}$ of an information carrier c for a given agent in a state a can be expressed as $\mathrm{InfoSem}(c, a) = J$. In this case, a customs officer that is interested in receiving information about toy cargo that he wants to inspect probably acquires more infons from an information carrier than a customs officer that is uninterested and does not want to receive information about a shipment. The information need that an agent has corresponds to a need for infons, which is modeled as follows: $\mathrm{Demand} : \mathcal{AS} \times \mathcal{IM} \rightarrow \wp(\mathcal{IN})$, where the set $\mathcal{IM}$ is the set of all information needs [16].

5.3 Ontological Model of the Process Layer

The ontological model of the process layer describes those concepts and relationships between the concepts that are of importance for using the information that has been exchanged in the underlying information layer for specifying the workflow that needs to be followed to realize service delivery. Figure 6 shows the ORM model for describing the process layer which visualizes the formalisms shown in this section. The process ontology in [18] forms a basis for the ontological model. First of all, process input is processed and transformed into output. Process input concerns the *infons* that are produced after process execution, which is done by one or more agents performing the process. The transformation of infons from service input to infons as service output is modeled as follows: Transform $\subseteq \mathcal{PR} \times \wp(\mathcal{IN}) \times \wp(\mathcal{IN})$. An expression like $(p, I_1, I_2) \in$ Transform shows that some process $p \in \mathcal{PR}$ transforms input $I_1 \subseteq \mathcal{IN}$ to output $I_2 \subseteq \mathcal{IN}$. The equation to express which agents perform which processes is modeled as follows: Perform $\subseteq \mathcal{AG} \times \mathcal{PR}$. For example, $(a, p) \in$ Perform shows that some agent $a \in \mathcal{AG}$ performs some process $p \in \mathcal{PR}$. The OWL-S specification [18] shows that there are three subtypes of the super type 'Process'. These are 'Atomic Process', 'Simple Process' and 'Composite Process'. An atomic process is one that has no internal structure. It corresponds to a single interchange of inputs and outputs. A composite process consists of a set of component processes linked together by control flow structures. The control flow is described using typical programming language or workflow constructs such as sequences, conditional branches, parallel branches and loops.

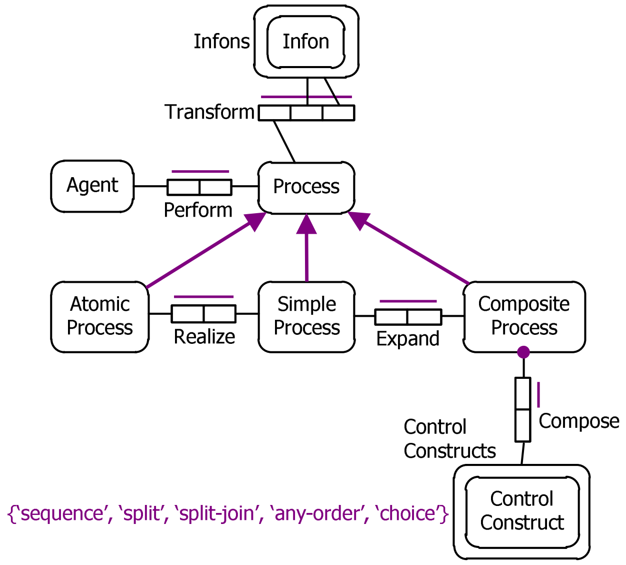

Fig. 6. An ORM-based ontological model for describing the process layer

A third type of process, the simple process, can be used to provide abstracted, non-invocable views of atomic or composite processes. A simple process is realized by an atomic process, while a simple process can be expanded to a composite process. A composite process on its turn is composed of the control constructs sequence, split, split-join, any-order and choice. The following equations and one function are introduced to formalize the relationships between the process subtypes: Realize $\subseteq \mathcal{AP} \times \mathcal{SR}$, Expand $\subseteq \mathcal{SR} \times \mathcal{CP}$ and Compose $: \mathcal{CP} \to \wp(\mathcal{CC})$. The set $\mathcal{AP}$ is the set of atomic processes, the set $\mathcal{SR}$ is the set of simple processes, the set $\mathcal{CP}$ is the set of composite processes, and the set $\mathcal{CC}$ is the set of control constructs. The expression $(p_1, p_2) \in$Realize shows that some atomic process $p_1 \in \mathcal{AP}$ realizes some simple process $p_2 \in \mathcal{SR}$, while a simple process $p_2 \in \mathcal{SR}$ that is expanded to a composite process $p_3 \in \mathcal{CP}$ is expressed as $(p_2, p_3) \in$ Expand. The expression Compose$(p_3) = C$ shows that a composite process is composed of control constructs $C \subseteq \mathcal{CC}$.

5.4 Ontological Model of the Service Layer

The ontological model of the service layer describes those concepts and relationships between the concepts that are of importance for the actual service delivery to clients by service providing organizations and it is shown in figure 7. The reference service model (RSM) [19] is suitable to serve as a basis because it aims to facilitate the semantic interlinking between services annotated using different semantic models and it accommodates bottom-up social annotation of services. The transformation of infons from service input to infons as service output is modeled as follows: Transform $\subseteq \mathcal{SE} \times \wp(\mathcal{IN}) \times \wp(\mathcal{IN})$, where $\mathcal{SE}$ is the set of services. This equation is almost identical to the transform equation used in the process model. A service is executed in a service *context*, which may differ at every new execution of a service. The context of a service is adapted by the circumstances of the service client [19].

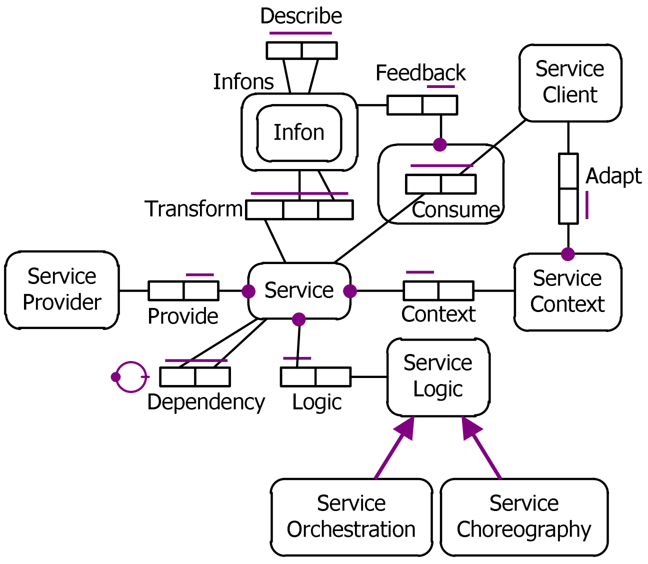

Fig. 7. An ORM-based ontological model for describing the service layer

For these reasons, the following two functions are introduced: $\text{Context} : \mathcal{SE} \rightarrow \mathcal{SC}$ and $\text{Adapt} : \mathcal{SC} \rightarrow \mathcal{CL}$. The context function is used to show that each service has a context, while the context adaptation function is used to show from which client the circumstances are adapted to form the context of a service. As the service *logic* concerns the business logic that is implemented by a service it determines both the input required by a service as well as the output produced by it. That each service implements a logic can be expressed by the logic function: $\text{Logic} : \mathcal{SE} \rightarrow \mathcal{SL}$. The interface of a service is described by defining its choreography and orchestration. Therefore, 'Service Orchestration' and 'Service Choreography' are introduced as two subtypes of the 'Service Logic' object type. The necessary input and output information for each service's capability are *described* in the choreography of the service. The orchestration reflects the *dependency* of a service with another service [19]. The symbol next to the dependency fact type in the figure depicts the ORM 'ring constraint' for *irreflexivity*. This means that in case of service orchestration some service cannot be dependent of itself. Each service is uniquely provided by some service provider and this is shown by the service provision function: $\text{Provide} : \mathcal{SE} \rightarrow \mathcal{SP}$. The consume equation is written as: $\text{Consume} \subseteq \mathcal{CL} \times \mathcal{SE}$ and is used to show that some client $c \in \mathcal{CL}$ consuming a service s is expressed as $(c, s) \in \text{Consume}$. Finally, a service provider can improve a service based on the feedback delivered by a client who has used the service. This is modeled as follows: $\text{Feedback} : \mathcal{CL} \times \mathcal{SE} \rightarrow \wp(\mathcal{IN})$. The situation in which a client c has consumed service s and provides feedback $F \subseteq \mathcal{IN}$ is expressed as $\text{Feedback}(c, s) = F$. With the description of the ontological model of the service layer complete, the international toy trade lane is used to illustrate the models.

6 Illustrating the Architecture in the Plastic Toys Trade Lane

The ontology-based event-driven architecture is illustrated in the context of the plastic toys trade lane to show how the utilization of the introduced formalisms can solve the barge planning issues raised in the trade lane. First, example events and expressions of equations in the formal event model are described. The expression $\text{EType}(e_1) = i$ shows that an event instance e_1 is an information event type i, which has been triggered when a freighter with toy cargo leaves the Port of Shenzhen. The event instance e_1 is streamed from the information layer, as it is an informational event, which is expressed as: $\text{Stream}(e_1) = \texttt{IL}$. The coordinates in space where the event takes place are expressed by $\text{Coord}(e_1) = (51,12,11)$. These could be the GPS coordinates of the Port of Shenzhen.

The following two expressions $\text{ETime}(e_1) = (10,05,23)$ and $\text{EDate}(e_1) = (01,05,2012)$ show that a freighter with toy cargo leaves the Port of Shenzhen at 10:05:23 on 5 January 2012. Second, related expressions of the information layer can be introduced. Assume that a barge planning agent interprets a dashboard containing event-based information about freighters arriving at the Port of Rotterdam. When interpreting this information he ends up in a new state $a \ltimes c$. The dashboard is an information carrier c. After interpreting the information on the dashboard, the expression $\text{Knowledge}(a \ltimes c) = I_1$ shows that the barge planning agent has accrued information I_1. Assume that the set I_1 contains information about: (1) what the coordinates are of specific toy cargo that might be shipped by barge, (2) on which freighter this cargo is stored, and (3) the expected time of arrival of the freighter in Rotterdam.

When involving the process layer model it can be determined how to view the toys trade lane in terms of processes. The first path of the trajectory from Shenzhen to Rotterdam can be viewed as a simple process p_1. In that part of the trajectory no complex dynamic re-planning has yet to be done, which causes that this first part can be depicted as a *black box* process to avoid overspecification. Once the cargo arrives in Rotterdam, the simple process can be expanded to a composite process which is expressed as $(p_1, p_2) \in \text{Expand}$. It is possible to decompose process p_2 into one of two sub-processes $p_3$and $p_4$at the point where the toy cargo arrives in Rotterdam, because at that point it has to be decided if the trajectory from Rotterdam to Venlo is traversed by road or barge. By using control constructs such as `If-Then-Else` a choice can be made which sub-process has to be performed to continue the transport from Rotterdam. An `If-Then-Else` control construct occurs at the point in the process where it is identified that *if* the current ETA of the vessel arriving in Rotterdam is 2 days or less before the point in time when the last inland barge that would arrive on time in Venlo departs from Rotterdam, *then* road planning should be performed, *else* barge planning should be performed. As we know from the expressions as part of the information layer, choosing between road or barge is dependent whether required information from the event dashboard has been accrued.

Finally, meaningful expressions from the service layer model are shown. A barge planning service s tranforms input I_1 to output I_2, that is expressed as $(s, I_1, I_2) \in \text{Transform}$. Recall that the information contained in I_1 is the barge planning information that has been acquired by the barge planning agent as mentioned above. The set I_2 contains information about (1) on which inland barge the toy cargo will be stored, (2) when it departs to Venlo, and (3) when it is expected to arrive there. Service s is a Web service and the output I_2 is supplied online to the terminal operator and the inland barge operator. The expression $\text{Logic}(s) = o$ shows that logic o is the service orchestration of the barge planning service. The orchestration shows that this service is dependent of the 'freight announcement service' u, as it is required for each freighter that arrives in Rotterdam to announce itself. This is expressed as: $\mathtt{Dependency}(s, u)$. The service choreography of the barge planning service describes that the input of the service is I_1 and the output is I_2, which is expressed as: $\mathtt{Describe}(I_1, I_2)$. The illustration in this section shows that triggered events are catalysts for generating a source of information, which, once made visible on a dashboard for in this case the PCS as the service provider, is used to improve the level of their service provisioning. Based on events, the service provider can get information about what the coordinates are of specific toy cargo that might be shipped by barge, on which freighter this cargo is stored, and the expected time of arrival of the freighter in Rotterdam. The EDA as elaborated in this paper shows that this architecture enables the availability of valuable information based on events. We have shown some example events, but it is conceivable that all kinds of other events are triggered when the states of the tracked toy cargo change when the cargo goes from China to the Netherlands. These events then provide additional valuable information for exchange between agents, which enables the execution of business processes to deliver logistic services, such as the barge planning service.

7 Conclusions and Future Research

Organizations that collaborate with each other in global supply chains are dependent of the information that is available to supply their services to demanding parties. Yet, the dynamic nature of global supply chains complicates the exchange of information as each time different organizations might be involved. In this paper, an ontology-based event-driven architecture is introduced that integrates information, processes, and services. This architecture can be used as a mechanism to coordinate service delivery in the context of, for example, a certain global supply chain where public and private organizations collaborate. It is illustrated by means of a real-life example in the context of a plastic toys trade lane that events are triggered by changes in the state of toy cargo. These events provide information to service providing organizations, for example, a port community system (PCS) uses information to supply a barge planning service for terminal operators and barge operators. A PCS is an entity that acts as a neutral hub which offers all kinds of online services to traders in a port.

For proper barge planning in the toys trade lane, the PCS needs to know when the toy cargo arrives at the Port of Rotterdam, the Netherlands, on which freighter the cargo is stored and when the cargo needs to be in some consolidation warehouse in Venlo. The PCS needs to have this information two days before the last inland barge departs that would arrive at the desired point in time in Venlo. If this information is not available on time, which happens frequently in the contemporary situation, then the only option to transport the toy cargo on the remainder of the trade lane from Rotterdam to Venlo is by road. This is much more expensive and polluting than barge transport. The illustration shows that the availability of information based on events positively influences the delivery of services. The architecture couples the information required to conduct a barge planning process with the delivery of the results of that process as a service to demanding parties.

Future research is concentrated on improving the architecture design, especially by further analyzing the exact ways how the three layers are coupled, which will probably lead to more insights in the relationships between which information is required for which processes to further enhance service delivery. Finally, the development of a prototype dashboard application that shows how information is derived from triggered events in the context of the plastic toys trade lane will also be researched. With this exercise, it is aimed to contribute to solving current planning issues in the toys trade lane by means of exploiting the architecture.

Acknowledgements. This paper results from the CASSANDRA project, which is funded by the 7th Framework Programme of the European Commission (FP7; SEC-2010.3.2-1) under grant agreement no. 261795. The CASSANDRA project addresses the visibility needs of business and government in the international flow of containerized cargo. It does so by developing a data-sharing concept that allows an extended assessment of risks by both business and government, thereby enabling enhanced supply chain visibility and cost-efficient and effective security enhancement. Ideas and opinions expressed by the authors do not necessarily represent those of all partners.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: A research roadmap. International Journal of Cooperative Information Systems 17, 223–255 (2008)
2. Chung, J.-Y., Chao, K.-M.: A view on service-oriented architecture. Service Oriented Computing and Applications 1, 93–95 (2007)
3. Yuan, S.-T., Lu, M.-R.: An value-centric event driven model and architecture: A case study of adaptive complement of SOA for distributed care service delivery. Expert Systems with Applications 36, 3671–3694 (2009)
4. Dunkel, J., Fernández, A., Ortiz, R., Ossowski, S.: Event-driven architecture for decision support in traffic management systems. Expert Systems with Applications 38, 6530–6539 (2011)
5. Kong, J., Jung, J.-Y., Park, J.: Event-driven service coordination for business process integration in ubiquitous enterprises. Computers & Industrial Engineering 57, 14–26 (2009)
6. Juric, M.: WSDL and BPEL extensions for Event Driven Architecture. Information and Software Technology 52, 1023–1043 (2010)
7. Scheer, A.-W.: ARIS - Business Process Modeling. Springer, Berlin (2000)
8. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Boston (2002)
9. Zappia, I., Paganelli, F., Parlanti, D.: A lightweight and extensible Complex Event Processing system for sense and respond applications. Expert Systems with Applications (in press)
10. Yao, W., Chu, C.H., Li, Z.: Leveraging complex event processing for smart hospitals using RFID. Journal of Network and Computer Applications 34, 799–810 (2011)
11. Overbeek, S.J., Janssen, M.F.W.H.A., van Bommel, P.: A standard language for service delivery: enabling understanding among stakeholders. Computer Standards & Interfaces 34, 355–366 (2012)
12. Overbeek, S.J., Janssen, M.F.W.H.A., van Bommel, P.: Designing, formalizing and evaluating a flexible architecture for integrated service delivery: combining event-driven and service-oriented architectures. Service Oriented Computing and Applications (in press)
13. Papageorgiou, N., Verginadis, Y., Apostolou, D., Mentzas, G.: Event-driven adaptive collaboration using semantically-enriched patterns. Expert Systems with Applications 38, 15409–15424 (2011)
14. Halpin, T.: Information Modeling and Relational Databases: from Conceptual Analysis to Logical Design. Morgan Kaufmann, San Mateo (2001)
15. Arasu, A., Babu, S., Widom, J.: The CQL Continuous Query Language: Semantic Foundations and Query Execution. VLDB Journal 15, 121–142 (2006)
16. van Bommel, P., Proper, E., van der Weide, T.: Information coverage in advisory brokers. International Journal of Intelligent Systems 22, 1155–1188 (2007)
17. Devlin, K.: Infons and types in an information-based logic. In: Cooper, R., Mukai, K., Perry, J. (eds.) Situation Theory and its Applications, vol. 1, pp. 79–96. Center for the Study of Language and Information, Stanford (1990)
18. Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D., Sirin, E., Srinivasan, N.: Bringing Semantics to Web Services with OWL-S. World Wide Web 10, 243–277 (2007)
19. Loutas, N., Peristeras, V., Tarabanis, K.: Towards a reference service model for the Web of Services. Data & Knowledge Engineering 70, 753–774 (2011)