Towards an Architectural Framework for Service-Oriented Enterprises

Nanjangud C. Narendra[1], Lam-Son Lê[2], Aditya K. Ghose[2], and Gandhi Sivakumar[3]

[1] IBM India Software Lab, Bangalore, India
narendra@in.ibm.com
[2] University of Wollongong, Wollongong, Australia
{lelamson,aditya.ghose}@gmail.com
[3] IBM Australia
gandhis@au1.ibm.com

Abstract. Business enterprises today are increasingly being modeled as *service-oriented enterprises (SOEs)*. That is, they are increasingly part of collaborations with other enterprises, with such collaborations being fulfilled by the exchange of business services among the participants. To that end, there is now a felt need for developing formal models of such collaborations, by leveraging past work on Enterprise Architecture (EA) models. In this paper, we present an architectural framework for modeling such collaborations as *virtual enterprises (VEs)*, since these collaborations involve interactions among multiple enterprises. Our framework is modeled by treating the VE as an enterprise itself, but with special characteristics that distinguish it from regular enterprises, viz., nature of collaborations among the participating enterprises, extent of their participation, and conflicts among the participants. The latter characteristic arises due to the autonomy of the participants and the dynamic nature of inter-organizational business interactions, and is especially crucial for VE modeling. Throughout the paper, we illustrate our architectural framework with a realistic running example. We also present and discuss some future challenges regarding modeling dynamic behavior in the VE, in particular, conflict modeling & resolution among the participating enterprises.

Keywords: service-oriented enterprise, enterprise architecture, virtual enterprises, architectural framework.

1 Introduction

Most business enterprises are now being modeled according to the principles of service-oriented computing [1] for the purposes of improving efficiency, agility and response to changing market needs. One of the key aspects of service-oriented computing is the integration of several enterpises into an entity called *virtual enterprise (VE)*. A VE possesses the following characteristics: (i) it is formed for a specific service-oriented process execution (could be short-lived or long term), and may dissolve once that process execution is done; (ii) its models are dependent on the *nature* of the interactions among the participating enterprises; (iii) it is typically formed via a joint alignment of strategies among the participating enterprises; and (iv) since the participating enterprises are autonomous, conflicts could arise among them.

A. Ghose et al. (Eds.): ICSOC 2012, LNCS 7759, pp. 215–227, 2013.

Traditionally, business enterprises have been modeled using enterprise architecture (EA) models. Several EA modeling frameworks have been developed; e.g., CIMOSA [2], TOGAF[1], Zachman [3] in industry; Archimate[2], SEAM[3] in academia; and international standards such as RM-ODP[4]. EA captures the whole vision of an enterprise in various aspects regarding both business and information technology (IT) resources. In EA, the goal is to align the business resources and IT resources in order to improve the competitiveness of the enterprise. EA is a discipline that analyzes the services offered by an enterprise and its partners to the customer, the services offered by the enterprise to its partners and the organization of the enterprise itself and of its IT.

However, most EA frameworks model only single enterprises and do not consider VEs. From our viewpoint, this makes them unsuitable for conceptual modeling of service-oriented enterprises. Hence in this paper we address this lacuna by presenting an architectural framework for VEs. However, as stated above, the behavior of a VE is critically dependent on the *interactions* between its participating enterprises (also referred to interchangeably throughout the rest of this paper as "entities"). Hence our VE model is based on the interaction types, which we refer to in this paper as *collaboration patterns*. We define four types of collaboration patterns, ranging from loosely coupled enterprises that can retain their autonomy, to subcontract-style collaboration where the subcontractor allows itself to be placed under the control of the prime contractor for the duration of the collaboration. To the best of our knowledge, this is the first attempt to develop an integrated model for a VE, with emphasis on modeling the collaboration patterns themselves as first-class objects in the model.

For our model, we adopt the 3-layer approach that consists of the *strategy* layer, the *operational* layer and the *service* layer [4, 5]. The strategy layer models the goals and business rules that define the behavior of the VE and its participants; the operational layer defines the business services [6] that are an abstraction of the actual process and service implementations that form the service layer. The unique feature of our approach is the usage of *(business) artifacts* [7] to model process implementations at the service layer. We use artifacts to model both the actual operation of the VE (collaboration artifacts) and the process executions of each participating entity (entity artifacts). We have incorporated artifact-based modeling in our approach, since this approach provides a convenient abstraction for translating business service models to lower-level IT service implementations. Moreover, since their dynamic behavior can be specified formally via communicating state machines, it is also possible to reason about them and perform formal verification before they are then translated to IT service implementations. Additionally, since IT service implementations are typically too low-level for business managers and analysts to monitor and track progress, artifacts can be used instead. The communicating state machine property of artifacts also renders them amenable to be used to model the collaboration artifact and the associated entity artifacts, and the interactions thereof, in a manner similar to that described in our earlier work [8].

[1] http://www.togaf.info/
[2] http://www3.opengroup.org/subjectareas/enterprise/archimate
[3] http://www.seam.ch/
[4] http://www.rm-odp.net/

This paper is organized as follows. Section 2 introduces our running example. Our virtual enterprise architecture models are explained in Section 3. Related work is discussed in Section 4. Finally, we present concluding remarks and suggestions for future work in Section 5.

2 Running Example

We model the VE as per the 3-layered approach as described in [4, 5]. The topmost layer is the Strategy layer, and it represents the following: businesss goals that the VE must fulfill, and policies and business rules of the VE and its participating entities that need to be taken into account while fulfilling the goals. The next lower layer is the Operational layer, which represents the Business Services [6] that each partner offers, and also describes how they are integrated to provide the overall functionality needed to fulfill the business goals specified at the Strategy layer. The actual operations executed and data exchanged are represented at the bottom layer, i.e., Service layer, which is the IT realization of the abstract concepts in the upper two layers.

For our running example we model a car manufacturer $CarMan$ with three partners. $SupSt$ and $SupTy$ are suppliers of steering wheels and tyres, respectively, while $Ship$ is a shipper who transports the completed cars. Together these entities form a (part of a) VE for manufacturing and selling cars. The relationship between $CarMan$ and its partners varies depending on the type of partner.

At the Strategy layer, the common goals of our VE would be "Manufacture Car" and "Ship Manufactured Car", which would also be the goals for $CarMan$. For $SupSt$, its goals could be "Manufacture Steering Wheel" and "Deliver Steering Wheel". For $SupTy$, its goals could be "Manufacture Tyres" and "Deliver Tyres". In order for the VE to be successful, those goals of each participating entity that pertain to the VE should be derivable from the common goals of the VE.

For $CarMan$ one of its business rules could be R1 = "If order amount for any shipment is greater than $1 million, choose lowest cost shipper; else choose $Ship$", whereas a business rule for $SupSt$ could be R2 = "Choose $Ship$ for all shipments". A business rule for $SupTy$ could be R3 = "If order amount between $100,000 and $250,000, choose Ship; else choose the lowest cost shipper". Clearly, these business rules could result in conflicts. For example, for order amounts exceeding $1 million, a conflict between $CarMan$ and $SupSt$ could arise. Whereas, for order amounts between $250,000 and $1 million, a conflict between $CarMan$ and $SupTy$ could arise. This would be handled by specifying consistency rules among the goals of the VE and the participating entities, so as to eliminate conflicts.

The above business goals are then mapped to the appropriate business services at the Operational layer. Some business services for $CarMan$ could be "Car Manufacturing" and "Car Shipment", and would be directly derivable from the common goals. For $SupSt$, some of its business services could be "Steering Wheel Manufacturing", "Steering Wheel Testing" and "Steering Wheel Shipment". While the former two would be directly derivable from the goals of $SupSt$, the third business service would need to be

directly linked to "Ship Manufactured Car" goal. This is because delivery of steering wheels is a necessary condition for shipping the finished product, i.e., the car.

It is to be noted that conflicts at the Strategic layer would directly impact the Operational layer also. For example, from the viewpoint of $SupSt$, its "Steering Wheel Shipment" business service would be affected by its policy of choosing $Ship$ for all shipments, and this may need to be redesigned in case of a conflict with $CarMan$. This would also have to be handled via the consistency rules introduced above.

At the Service layer, the (more abstract) business services are mapped onto the (more concrete) IT realizations. We subdivide this layer into two sub-layers, with the top sub-layer being modeled via business artifacts, which are an abstraction of the following: the data exchanged among the participating entities and the operations that they execute in order to fulfill the business goals of the VE. The bottom sub-layer comprises the actual IT services and business processes through which the implementation of the business services takes place. For our running example, some artifacts are `CarAssembly`, `CarShipment`, `SteeringWheelManufacture`.

Our running example raises several interesting research questions. First, we need to investigate what the overall metamodel of the VE and its participating entities should be, with emphasis on the interactions among them. Second, we need to specify the goals, business rules and consistency rules in a manner that enables easy analysis and reasoning. Third, at the Operational and Service layers, we need to investigate how business services [6] should be represented so as to facilitate easy derivation of (collaboration & entity) artifacts and IT service implementations. These research questions will form the focus of the rest of our paper.

3 Models

In this Section we present the key contribution of our paper, i.e., metamodel for VEs. But first we define what we mean by business strategy and business goals. Business strategy is generally regarded as a high-level *plan* specified to achieve an objective. We follow the taxonomy proposed in the KAOS methodology [9] to categorize business goals[5] into: *achieve* (an organization seeks to achieve a condition at some point of time in the future), *cease* (seeks to undo a condition at some point of time in the future), *maintain* (strikes to maintain a condition for a period of time), *avoid* (prevents a condition from becoming true for a period of time) and *optimize* (usually articulated in the form of *maximization* or *minimization*). In our running example, for simplicity, we have restricted business goals to those conditions that are to be *achieve*d, and we will be using the running example to illustrate our metamodel throughout the rest of this section. However, our metamodel would be able to cater to the other goal types also.

Our metamodel definition for VEs will therefore be driven by the overall business strategy of the VE, along with how it impinges on those of the participating enterprises. Also, as already stated earlier in our paper, the VE metamodel will be based on the interactions among the participating enterprises as derived from their business strategies.

[5] Goal-oriented modeling in the literature [10] includes both business goals and system goals. As we do business strategy modeling, we address only business goals.

3.1 Collaboration Patterns

We model the interactions among the participating service-oriented enterprises as *collaboration patterns* [11], defined in increasing order of coupling among the enterprises as follows:

- *CP1: Informal Joint Venture*: In this collaboration, a set of enterprises get together on a relatively ad-hoc basis, for a limited period of time, driven by a common business goal. In such a collaboration, each participating entity would have its own business services and derived business processes, but without revealing any internals of their business processes. Hence the entire collaboration would be implemented via exchanges of messages based on a commonly agreed protocol. Conflict resolution in such a collaboration is accomplished primarily via negotiations.

  For example, $CarMan$ could have a short-lived adhoc arrangement with $SupTy$ for the immediate purchase & delivery of a batch of car tyres, and this would be a collaboration of type CP1. The possible conflict in business rules between $CarMan$ and $SupTy$ as described in Section 2 would have to be negotiated among the partners.
- *CP2: Association*: In this collaboration, the participating enterprises agree on a common set of business processes that each enterprise has to comply with. This may necessitate them exposing parts of their internal business processes & operations. Bound as they are together with the common business processes, the participating enterprises' negotiating positions during conflict resolution get constricted. In other words, they would experience a lower degree of freedom as opposed to participation in a CP1-type pattern.

  For example, in the collaboration between $CarMan$ and $SupTy$, if it were of type CP2, then $SupTy$ would be bound by a delivery arrangement, and in any conflict with $CarMan$, such as the one described in Section 2, the onus would be on $SupTy$ to ensure conflict resolution.
- *CP3: Formal Joint Venture*: Moving further onto increased coupling among the participants, this collaboration pattern externalizes all the business rules and constraints on each participant, and makes them part of the overall collaboration. (In the earlier two collaboration types, the rules and constraints governing the actions of each participant are not made public, since many participants may consider them confidential information.) In such a pattern, the VE as a whole dictates conflict resolution.

  For example, in the collaboration between $CarMan$ and $SupTy$, $CarMan$ would dictate how conflict resolution, if any, were to be implemented, since $SupTy$'s business rules would become visible to it.
- *CP4: Subcontract*: This collaboration pattern is the most formal and binding of all; here, one participant - the contractor - controls all the business processes to be executed, as well as the business rules and constraints affecting the participants in the collaboration. Conflicts betwen the contractor and its subcontractors are resolved in favor of the contractor; whereas, conflicts among subcontractors are resolved by the contractor as per preference rankings among the business rules.

  For example, in the collaboration between $CarMan$ and $SupTy$, the latter would become $CarMan$'s subcontractor, hence even its business rules would be

under $CarMan$'s control. Similarly, any conflict beween $SupSt$ and $SupTy$, e.g, a scheduling conflict regarding integrated supply of tyres and steering wheels, would also be resolved by $CarMan$.

3.2 Metamodel

Fig. 1 presents our metamodel for VEs. We model the VEs from two different perspectives: collaboration-centric and entity-centric (Subsections 3.3 and 3.4, respectively). In this figure, the metamodel constructs are presented at the three layers, namely the strategy layer, operational layer and service layer (in this order from the top down to the bottom of Fig. 1). A *collaboration* occurs as per a *collaboration pattern*, introduced above. Each collaboration has a *goal*, which is realized by a *plan*, which is a set of steps to achieve the goal. Each plan can be realized via a *schedule*, which is a sequence of *business service* executions in a particular order. Each plan can be realized by more than one schedule. Each business service is defined via its *inputs, outputs, preconditions and postconditions*, which are in turn derived from the attributes of the plan and schedule.

3.3 Service- and Collaboration-Centric Modeling

As depicted in Fig. 1, we model the VE itself – as stated earlier, this model will be based on the nature of the collaborations – and ensuing interactions thereof, that comprise the VE. The entity-centric conceptual model will be presented in Section 3.4.

At the service layer, the business service is finally realized by a *collaboration artifact* [8], which models the business service executions. This artifact represents the execution of the business services involved in the collaboration, and consists of a lifecycle composed of *states*. Each state has *incoming and outgoing transitions*, each of which are implemented by *IT services*. Each IT service is modeled via its own *inputs, outputs, preconditions and postconditions*, which are derived from the conditions governing the incoming and outgoing transitions.

Formally, we represent a collaboration $Coll = \langle \{E_i\}, \{P_j\}, G, \{R_k\} \rangle$, where E_i is a participating enterprise, P_j is a collaboration pattern, G is the goal of the collaboration expressed as a boolean condition to be achieved (as described above), and R_k is a business rule that constrains the dynamic behavior of the collaboration. (Our definition of business rules also includes business compliance requirements [12] that typically apply to all participating entities.) The collaboration pattern P_j is defined as $P_j = \langle \{E_j\}, CP_k \} \rangle$, where E_j are the participating enterprises in the collaboration pattern and CP_k is the collaboration pattern type as introduced earlier.

We define the goal G of the collaboration as a boolean condition to be achieved. Without loss of generality, we represent G in CNF as $G = G_1 \wedge G_2 \wedge ... \wedge G_n$, where the G_i are sub-goals, typically assigned to the participating enterprises. In other words, the overall goal of the collaboration should be consistent with the goals of the participating enterprises. We express a business rule R_k also as a boolean condition that should be satisfied in addition to the goal. An example of the goal for the VE would be $Car\_Delivered = Car\_Components\_Received \wedge Car\_Manufactured \wedge Car\_Shipped$.

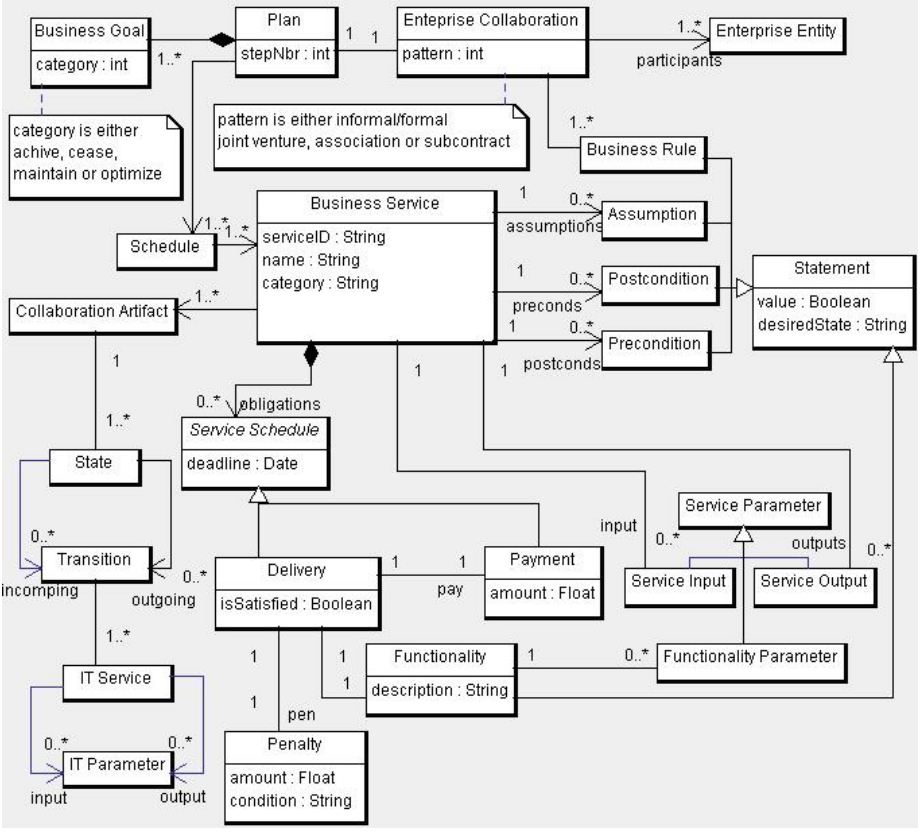

Fig. 1. Virtual Enterprise Architecture Model

Since the Plan is derived from the Goal, we define it as $Plan = \langle G, \{R_k\}, Sch\rangle$, where Sch is the Schedule, defined as $Sch = (\{Pref_j\langle B_i, B_j, rel\rangle\})$. Here, B_i stands for a business service, and rel denotes either "immediately" (IMM) or "eventually" (EVE). That means the schedule defines a set of ordering preferences $Pref_j$ on the execution of the business services to fulfill the Plan. The R_k refer to the business rules that constrain the ordering and execution of the business services while fulfilling the goal.

An example of a business rule for the VE could be the rule R1 described in Section 2; this constrains the VE to select a particular type of shipper to deliver the manufactured cars. This in turn constrains the plan and schedule to be designed; for example, if a Shipping business service were to be invoked, then rule R1 may force the VE to ensure that shipment of all cars occurs at the end of the schedule, instead of shipping the cars as and when they are manufactured.

We define a business service [6] as $B_i = \langle I, O, P, E, \{A_i\}\rangle$, where I is the set of inputs, O is the set of outputs, P is the set of preconditions (expressed as a conjunction of boolean conditions) and E is the set of effects (also expressed as a conjunction of

boolean conditions). Here, A_i refers to the collaboration artifact(s) that realize the implementation of the business service. Hence we define A_i as
$A_i = \langle S, s_0, S_f, L, T, L', M \rangle$ [8], where S is the set of states of the artifact, s_0 is its initial state, S_f is its final state, L is the set of transition labels, T is the set of transitions, L' is the set of message labels, and M is the set of messages. In this definition, we assume that the modeled collaboration artifacts communicate with each other via message passing, which will be one of the triggers for transitioning artifacts from one state to the next. Hence we define a state transition in an artifact from state s_i to state s_j as $t = \langle s_i, l, s_j \rangle$. Whereas, we define a message from one artifact to another as $m = \langle s_i, l', s_j \rangle$, where s_i and s_j are states in the sender and recipient artifact, respectively.

As per our metamodel, state transitions in an artifact are realized via the implementation of a set of IT services in a predefined sequence; hence the transition is implemented by $ITS = \langle \{ITS_i\}, \langle ITS_i, ITS_j, rel \rangle$, where rel signifies either "eventually" or "immediately". That is, once the transition t is triggered, the set of IT services ITS execute, thereby transitioning the artifact to the next state as defined by the transition.

In our running example, let us assume the existence of a Manufacturing business service in the VE. Some of its inputs would be $\{\{Comp_j\}, \{Matl_k\}\}$, where $Comp_j$ and $Matl_k$ are the car components and other raw materials needed for manufacture, respectively. Its primary output would be $\{Car_i\}$, i.e., the manufactured cars. Some preconditions for the business service could be
$\{Components\_received, Raw\_Materials\_received\}$ and the effect would be
$\{Cars\_Manu\_Completed\}$. One collaboration artifact that would help model the manufacturing process could be `CarAssembly`, with the following states in its lifecycle:
$Not\_Activated$, $Components\_Obtained$, $Raw\_Materials\_Obtained$,
$Engine\_Created$, $Chassis\_Created$, $Engine\_Assembled$,
$Other\_Components\_Assembled$, Car_Tested, $Car\_Manufacture\_Completed$. An example of a transition between states could be $\langle Other\_Components\_Assembled, l_t,$
$Car\_Tested \rangle$, which signifies that `CarAssembly` has transitioned (with the label l_t) from the state where all components have been assembled to the state where the car has been fully tested before it is declared ready for shipment. An illustration of message passing between artifacts is provided in Section 3.4, once the entity metamodel is introduced.

3.4 Entity-Centric Modeling

Each participating enterprise or entity could be part of one or more collaboration patterns with different other entities, even within the same collaboration. This entity also has its own *goal*, realized by a *plan*. Ideally, it is expected that the entity goal should not conflict with that of the collaboration depicted in Fig. 1. This plan is realized by an *entity schedule*. The entity schedule is then realized at the operational layer by the *business services* of the entity. Similar to Fig. 1, each business service is realized by one or more *entity artifacts*, with their respective lifecycles, states, transitions and IT service realizations.

Formally, we define a business enterprise as $E_i = \langle G_i, \{P_j\}, \{R'_k\} \rangle$, where G_i is the goal of E_i, P_j denotes the collaboration patterns of which E_i is a part, and R'_k defines the business rules constraining E_i's behavior. The goal G_i and business rules R'_k are

defined in a manner similar to those of the collaboration as already introduced earlier in Section 3.3. The plan for E_i is $Plan_i = \langle G_i, \{R'_k\}, Sch_i \rangle$, where the schedule Sch_i is defined as $Sch_i = (\{< B'_i, B'_j, rel >\})$, where rel is as defined in Section 3.3, and B'_i & B'_j are the business services of the enterprise.

From our running example, for $SupSt$, its goal could be $Deliver\_Steering\_Wheel = Manufacture\_Steering \wedge Deliver\_Steering$. Its business service could be "Steering Wheel Manufacturing", while one of its business rules could be the rule R2 from Section 2. If $SupSt$ were in a CP1-type collaboration with $CarMan$, then the two enterprises would need to negotiate in case business rules R1 and R2 created a conflict. For CP2- and CP3-type collaborations, $SupSt$ would have to modify or remove R2 for conflict resolution. For a CP4-type collaboration, $SupSt$'s business rules may be invalidated by the VE, thereby preventing conflicts from arising in the first place. Again, these business services are as defined in Section 3.3, and their associated enterprise artifacts are also modeled as defined in Section 3.3.

Referring to the illustration in Section 3.3, one possible artifact for $SupSt$ could be `SteeringWheelManufacture`, with the following states: $Components\_Obtained$, $Raw\_Materials\_Obtained$, $SteeringWheel\_Manufactured$, $SteeringWheel\_Tested$, $SteeringWheel\_Shipped$. An example transition could be $\langle SteeringWheel\_Manufactured, l_v, SteeringWheel\_Tested \rangle$, which signifies that `SteeringWheelManufacture` has transitioned (with the label l_v) from the state where the steering wheel has been assembled to the state where it has been fully tested, bringing it one step closer to shipment to $CarMan$. The interaction between `SteeringWheelManufacture` and `CarAssembly` artifacts can be modeled via message sending from the former to the latter. Once the `SteeringWheelManufacture` artifact has reached $SteeringWheel\_Shipped$ state, the message can be sent to `CarAssembly` artifact, which enables the `CarAssembly` artifact to be instantiated.

Once all components (from other suppliers such as $SupSt$) are obtained, the `CarAssembly` artifact attains the $Components\_Obtained$ state. One of the triggers for this transition would be a message from `SteeringWheelManufacture` artifact while in state $SteeringWheel\_Shipped$, to `CarAssembly` artifact while it is in state $Not\_Activated$.

3.5 Consistency Rules

In order for the VE to work successfully, the dynamic behavior of the collaboration and its participating entities should be consistent. To that end, we define the following consistency rules. At the Strategy layer, the goals and business rules of the collaboration and its participating entities need to be consistent. This can be formally denoted by the following:

- $G \Rightarrow G_i, \forall i$, i.e., the goal of the collaboration should serve as the overall goal for the participating entities
- $R \wedge R'_i \not\models \perp, \forall i$, i.e., the business rules of the collaboration should not conflict with those of any of the participating entities

– $For\ P_j \ni \{E_i, E_j\}, R'_{il} \wedge R'_{kl} \not\models \perp, \forall i, j, k, l$, i.e., no two business rules from each of the participating entities, especially those that are part of a collaboration pattern, should conflict with each other

For example, rules R1 and R2 from Section 2 would be consistent with each other, but this would not be the case for R1 and R3. At the Operational layer, the following consistency rules can be defined:

– The schedules of the collaboration and that of any of the participating enterprises should not conflict. For the collaboration, let the schedule be $Sch_{Coll} = (\{Pref_j = \langle B_i, B_j, rel\rangle\})$ and the schedule of a participating enterprise be $Sch_E = (\{Pref_k = \langle B'_i, B'_j, rel\rangle\})$. Let the combined schedule be the set $Sch' = Sch_{Coll} \cup Sch_E$. Then for any two preference rules $Pref_m$ and $Pref_n$, where $Pref_m \in Sch'$ and $Pref_n \in Sch'$, the following should hold: $Sch' \wedge Sch_i \not\models \perp$.
– Let there be N business services B_i in the collaboration, with the effect of each being Eff_i. Then $Eff_1 \wedge Eff_2 \ldots \wedge Eff_N \models G$, i.e., the business services in the collaboration will entail the goal G of the collaboration.
– For any participating enterprise with goal G_i, let there be M business services B'_i with the effect of each being Eff'_i. Then, in a manner similar to that of the collaboration, $Eff'_1 \wedge Eff'_2 \ldots \wedge Eff'_M \models G_i$, i.e., the business services of the participating enterprise will entail its goal G_i.

For example, one preference rule for our running example from Section 2 could be $\langle Steering\ Wheel\ Manufacturing, Steering\ Wheel\ Shipment, IMM\rangle$ and $\langle Steering\ Wheel\ Shipment, Car\ Manufacturing, EVE\rangle$. The former rule states that steering wheeels should be shipped as soon as their manufacture is completed; the latter rule states that steering wheels are needed in order to complete the manufacture of the cars.

Similarly, at the Service layer, the following consistency rules can be defined:

– The collaboration artifact & enterprise/entity artifacts should not conflict with each other. Since each such artifact is modeled as a state machine, techniques from communicating state machine verification (such as [13]) can be used to verify that the artifacts do not conflict with each other.
– The modeled IT services for the collaboration and the participating enterprises should also be consistent with each other. The verification of this can be accomplished via techniques such as those described in [14].

3.6 Discussion - Conflicts and Conflict Resolution

One of the key advantages of developing a metamodel for VEs, is the ability to use it as a basis for reasoning about dynamic behavior. For our purposes, from the viewpoint of VE modeling, beyond the usual applications of orchestration & choreography for modeling inter-organizational inteactions [15, 16], the crucial question is of modeling conflicts and their resolution. Conflicts can be classified along three orthogonal dimensions - type, origin and impact [17]. The type of conflict would be determined by the highest layer at which they manifest; for example, a conflict in goals or business rules

would manifest at the Strategy layer, whereas a conflict among interacting IT services from different participants would manifest at the Service layer. The origin of a conflict would define the participant, or one of its constructs (e.g., business service, IT service, etc.) where the conflict originated; multiple origins could exist for a single conflict. The impact of a conflict would model the side-effects of the conflict on the rest of the participants and the VE itself in general. For example, a minor scheduling conflict among two interacting entities, which can be resolved by the entities themselves, would have local impact. Whereas, a product supply conflict, that could affect the collaboration artifact of a VE, would have global impact.

As part of future work, we will be integrating our earlier work on conflict modeling in B2B applications [17] and formalizing it by incorporating it into our VE metamodel.

4 Related Work

EA Modeling Frameworks: The importance of EA modeling has given rise to several frameworks. One of the earliest was the Zachman framework [3], which provides a formal and highly structured way of viewing and defining an enterprise. It consists of a two dimensional classification matrix based on the intersection of six communication questions (What, Where, When, Why, Who and How) with six rows representing six reification criteria (scope, business, system, technology, component, operations). CIMOSA [2] is a similar EA framework, organized along three dimensions, viz., view (organization, resource, information, function), lifecycle (requirements, design, implementation) & generic (generic, partial, particular). Unlike the Zachman framework, CIMOSA provides a methodology and supporting technology. The other popular framework, TOGAF is modeled at four layers, viz., business, application, data and technology, and provides an integrated approach for designing and implementing an EA. It is to be noted, however, that none of these frameworks attempt to explicitly model VEs.

The VE model described in our paper is inspired in part by our earlier ODP-based work on enterprise modeling [18], and also on the 3-layer service architecture model described in [4, 5]. The latter model also describes how inter-organizational service interactions can be modeled at the strategic, operational and service layers. We have borrowed the layers from [4, 5], but we have redefined them with VEs in mind, i.e., with emphasis on collaboration patterns and consistency rules for conflict handling.

Virtual Enterprise Modeling: The earliest notable work in this realm was about capturing the requirements of a VE using the ODP enterprise language [19]. There exists some recent work focusing on modeling and detailing collaboration pattern types [20, 21, 11]. In those works, the authors have delineated the various ways in which an enterprise and its partners can interact, with emphasis on the various ways in which work can be outsourced from one business to another. Indeed, our idea of collaboration patterns is inspired by those works; however, those works do not consider a layered approach, nor do they model consistency rules for conflict handling. The e^3 Value project[6] has also focused on inter-organizational aspects, but primarily from the viewpoint of how B2B interactions can be modeled as value exchanges. Hence we

[6] `http://e3value.few.vu.nl/`

consider e^3 Value to be complementary to our work, and our future work will investigate how it can be integrated into our VE metamodel.

Our work in this paper on conflict handling is based on our earlier work on modeling B2B conflicts in B2B interactions [17], as we have already described in Section 3.6.

Goal & Strategy-Based Business Process Modeling: In the area of goal-based business process modeling [22–26], the citation [22] showed how to model and evaluate business processes at a level of abstraction higher than the business process level. Based on that idea, our earlier work has focused on how to derive business process models from goals [27], which also leveraged the KAOS stepwise refinement approach [9]. We have leveraged this same approach in this paper, by modeling business processes as derivable from goals, via plans and schedules. Our paper also incorporates some of our ongoing work on business service modeling [6], which aims to model business services as an abstraction of IT service implementations. We have further enhanced this abstraction by incorporating our earlier work [8] on artifacts and Web services.

5 Conclusions and Future Work

In this paper, we have addressed the crucial but little-researched area of modeling service-oriented enterprises as virtual enterprises. Since the key constructs in such an enterprise are the types of collaborations among the participating enterprises, we have presented and detailed a metamodel for VEs where these types are modeled upfront as first-class objects. We have also positioned this within a 3-layer framework comprising Strategy, Operational and Service layers, with interactions among the participating enterprises modeled at all these three layers. We have also formally defined consistency rules that are needed in order to ensure that the VE can function without conflicts among the (relatively autonomous) participating enterprises. To the best of our knowledge, this is the first attempt at developing a formal metamodel for VEs.

Future work will include incorporating conflict resolution in our metamodel via techniques such as those presented in [28, 17], and developing a prototype implementation to evaluate and refine our metamodel.

References

1. Huhns, M.N., Singh, M.P.: Service-oriented computing: Key concepts and principles. IEEE Internet Computing 9(1), 75–81 (2005)
2. Cuenca, L., Ortiz, A., Vernadat, F.: From uml or dfd models to cimosa partial models and enterprise components. Int. J. Computer Integrated Manufacturing 19(3), 248–263 (2006)
3. Zachman, J.A.: The information systems management system: A framework for planning. DATA BASE 9(3), 8–13 (1978)
4. Orriens, B., Yang, J., Papazoglou, M.P.: A Rule Driven Approach for Developing Adaptive Service Oriented Business Collaboration. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 61–72. Springer, Heidelberg (2005)
5. Orriens, B., Yang, J.: A rule driven approach for developing adaptive service oriented business collaboration. In: IEEE SCC, pp. 182–189 (2006)
6. Lê, L.-S., Ghose, A., Morrison, E.: Definition of a Description Language for Business Service Decomposition. In: Morin, J.-H., Ralyté, J., Snene, M. (eds.) IESS 2010. LNBIP, vol. 53, pp. 96–110. Springer, Heidelberg (2010)
7. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Syst. J. 42, 428–445 (2003)

8. Narendra, N.C., Badr, Y., Thiran, P., Maamar, Z.: Towards a unified approach for business process modeling using context-based artifacts and web services. In: IEEE SCC, pp. 332–339 (2009)
9. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Journal of Science of Computer Programming 20(1-2), 3–50 (1993)
10. Mylopoulos, J., Chung, L., Yu, E.S.K.: From object-oriented to goal-oriented requirements analysis. Commun. ACM 42(1), 31–37 (1999)
11. Norta, A., Grefen, P.W.P.J.: Discovering patterns for inter-organizational business process collaboration. Int. J. Cooperative Inf. Syst. 16(3/4), 507–544 (2007)
12. Governatori, G., Milosevic, Z., Sadiq, S.W.: Compliance checking between business processes and business contracts. In: EDOC, pp. 221–232 (2006)
13. Peng, W., Purushothaman, S.: Data flow analysis of communicating finite state machines. ACM Trans. Program. Lang. Syst. 13(3), 399–442 (1991)
14. Benatallah, B., Casati, F., Toumani, F.: Representing, analysing and managing web service protocols. Data Knowl. Eng. 58(3), 327–357 (2006)
15. Barros, A., Decker, G., Dumas, M., Weber, F.: Correlation Patterns in Service-Oriented Architectures. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 245–259. Springer, Heidelberg (2007)
16. McIlvenna, S., Dumas, M., Wynn, M.T.: Synthesis of orchestrators from service choreographies. In: APCCM, pp. 129–138 (2009)
17. Maamar, Z., Thiran, P., Narendra, N.C., Subramanian, S.: A framework for modeling b2b applications. In: AINA, pp. 12–19 (2008)
18. Lê, L.-S., Wegmann, A.: Hierarchy-oriented modeling of enterprise architecture using reference-model of open distributed processing. Special Issue on RM-ODP, Computer Standards & Interfaces Journal (February 2012)
19. Oldevik, J., Aagedal, J.: ODP-Modelling of Virtual Enterprises with Supporting Engineering Architecture. In: Proceedings of 3rd EDOC, pp. 172–182. IEEE Computer Society (September 1999)
20. Grefen, P.W.P.J.: Towards dynamic interorganizational business process management. In: WETICE, pp. 13–20 (2006)
21. Norta, A., Grefen, P.W.P.J.: A framework for specifying sourcing collaborations. In: ECIS, pp. 626–638 (2006)
22. Kueng, P., Kawalek, P.: Goal-based business process models: Creation and evaluation. Business Process Management Journal 3, 17–38 (1997)
23. Cardoso, E., Almeida, J., Guizzardi, R.: On the support for the goal domain in enterprise modelling approaches. In: 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), pp. 335–344 (October 2010)
24. Xu, T., Ma, W., Liu, L., Karagiannis, D.: Synthesizing enterprise strategic model and business processes in active-i*. In: 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), pp. 345–354 (October 2010)
25. Neiger, D., Churilov, L.: Goal-Oriented Business Process Modeling with EPCs and Value-Focused Thinking. In: Desel, J., Pernici, B., Weske, M. (eds.) BPM 2004. LNCS, vol. 3080, pp. 98–115. Springer, Heidelberg (2004)
26. De la Vara Gonzalez, J.L., Diaz, J.S.: Business process-driven requirements engineering: a goal-based approach. In: Business Process Management Workshops, `http://lams.epfl.ch/conference/bpmds07/program/Gonzalez_23.pdf`
27. Ghose, A.K., Narendra, N.C., Ponnalagu, K., Panda, A., Gohad, A.: Goal-Driven Business Process Derivation. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 467–476. Springer, Heidelberg (2011)
28. Bentahar, J., Moulin, B., Bélanger, M.: A taxonomy of argumentation models used for knowledge representation. Artif. Intell. Rev. 33(3), 211–259 (2010)