Multi-staged and Multi-viewpoint Service Choreography Modelling

Alistair Barros[1], Gero Decker[2], Marlon Dumas[3]

[1] SAP Research Centre, Brisbane, Australia
alistair.barros@sap.com
[2] Hasso-Plattner-Institute, University of Potsdam, Germany
gero.decker@hpi.uni-potsdam.de
[3] Queensland University of Technology, Australia
m.dumas@qut.edu.au

Abstract. Recent approaches to service-oriented systems engineering start by capturing the interactions between services from the perspective of a global observer, leading to so-called service choreographies. The rationale is that such choreographies allow stakeholders to agree on the overall structure and behaviour of the system prior to developing new services or adapting existing ones. However, existing languages for choreography modelling, such as WS-CDL, are implementation-focused. Also, these proposals treat choreographies as monolithic models, with no support for multiple viewpoints. This paper proposes a multi-staged and multi-viewpoint approach to choreography modelling. For the initial stages, the approach promotes the partitioning of choreography models and the design of role-based views; while for subsequent stages, milestone and scenario models are used as an entry point into detailed interaction models. The paper presents analysis techniques to manage the consistency between viewpoints. The proposal is illustrated using a sales and logistics model.

1 Introduction

As implementation-level web service interoperability standards mature, the need for analysis and design methods for service-oriented systems becomes even more pressing. In early proposals for service-oriented analysis and design methods [8, 2, 22, 16], global models of service interactions, also known as *conversation protocols* or *service choreographies*, play a key role during the initial stages of analysis and design. Service choreographies capture the interactions in which a collection of services engage, from the perspective of an ideal observer who would be able to see every interaction occurring between the involved services. By providing a "birds-eye" view over interactions, service choreographies allow business and IT stakeholders to build a common understanding of the structure and behaviour of the overall system, prior to approaching implementation issues.

Languages such as BPSS [20] and WS-CDL [14] have been proposed as a basis for modelling choreographies. However, these languages focus on detailed inter-

action flows and treat choreographies as monolithic development artifacts. WS-CDL in particular goes down to supporting the description of choreographies in a quasi-executable form, using programming constructs such as sequence, block-structured conditional and loop structures, and variable assignment. While these languages partly support an implementation-independent approach to service-oriented system design, it is questionable that they are suitable for the early phases of the development lifecycle, where the incremental acquisition of a common understanding by multiple stakeholders with different concerns and backgrounds is crucial. It is unlikely that business stakeholders and system analysts operating at a high level of abstraction will benefit from manipulating choreography models that include executable data manipulation steps. Instead, they are likely to be interested in viewing choreographies without the interaction flow details (e.g. role-based viewpoints), or to view milestones and specific scenarios. Only at later stages does the refinement of choreography models into executable code becomes a concern. Thus, languages and methods for choreography modelling should be compatible with multi-staged and multi-viewpoint design. By supporting multiple modelling viewpoints, it is possible to break down a service design into smaller more manageable parts that are handled by different stakeholders.

In this setting, the proposition of this paper is an approach to choreography modelling based on viewpoints layered on top of a choreography modelling language, namely Let's Dance [21]. Let's Dance has a formal semantics defined in terms of π-calculus [6] and has been shown to be expressive enough to capture common service interaction patterns [3] and to serve as a basis for generating local views on service interactions for subsequent refinement into executable models [21]. In this paper, we define role-based, milestone-based, and scenario-based viewpoints into service choreographies, and propose techniques for managing the consistency between these viewpoints and the interaction-based viewpoints natively supported by Let's Dance. The outcome is a set of notational elements and consistency checking techniques that provide a basis for defining service-oriented analysis and design methods that bridge the gap between business and IT concerns. The notational elements have been used to capture a refined version of a logistics collaboration model proposed by the Voluntary Inter-industry Commerce Solutions Association (VICS).[4] Meanwhile, the consistency checking algorithms have been validated through implementation and testing.

After an informal introduction to the proposal using the VICS global logistics model (Section 2), the paper defines an abstract syntax for each of the proposed modelling viewpoints (Section 3). Next, the techniques for inter-viewpoint consistency verification are presented in Section 4. Finally, related work is discussed in Section 5 while Section 6 concludes.

[4] See www.vics.org

2 Multi-view choreography design by example

This section provides the practical setting through which the proposed choreography modelling viewpoints and extensions to the Let's Dance language are motivated and developed. Insights are drawn from a case study inspired by the Sales and Logistics component of the VICS EDI Architecture Guide. The case study is related to the supply chain between retailers and manufacturers, covering processes where products are ordered through cyclic stock replenishment agreements over a time-horizon (e.g. 12 months), shipped and paid for. Along the way, shipments need to be managed and optimised through different types of carriers (land, rail, air, ocean), consolidated at intermediaries, crossed through the "red tape" of customs and quarantine, and delivered to consignment nodes where they are dispatched to retail stores. As a result of delivery, fulfilment of an order needs to be assessed with respect to quantity, timeliness and damage to ensure quick turn-around for payment and reimbursement. To close the loop, supply and consumer patterns are dynamically fed back into the next cycles of merchandising and collaborative forecasting, planning and replenishment.

Domains and roles. End-to-end modelling of interactions of value-chains as vast as Sales and Logistics require a careful scoping of processes being analysed. To facilitate such scoping and provide a focus on models (e.g. interaction models) developed for common business objectives, we propose the notion of *collaboration domains*. These domains group a set of logically related models. The set of collaboration domains for the Sales & Logistics case study is depicted in Figure 1. As apparent from the figure, the collaboration domains (ellipses) scope different areas of business interest. For example, a distinction is made between Collaborative Forecasting Product Replenishment (out of which an order is produced), Logistics (governing shipment of goods), Payments, Exceptions, Order Release and Product Merchandising. Given the size and complexity of domains, we propose that they should be arranged in a hierarchical structure. For example, in Figure 1, we can see that the logistics domain is decomposed into four sub-domains: Tendering, Carrier Appointment, Delivery, and Claims & Returns.

To go from collaboration domains into individual processes with detailed interactions, we propose an intermediate viewpoint: the *role-based choreography view*. A role-based choreography is defined for each leaf-level domain (since non-leaf domains are purely used for the purpose of abstraction). This viewpoint is illustrated for the Delivery domain in Figure 1. This viewpoint shows *collaborating roles* (boxes) and their *interaction dependencies*, expressed through *channels*. A channel captures the fact that there is at least one direct interaction between two or more roles and the purpose of this/these interactions. Channels are represented by small circles on lines.

Cardinality constraints on channels are used to express how many participants of one role can interact with one participant of the other role. As illustrated, a Shipper interacts with a number of Carriers for Carrier Planning, while a Carrier interacts with one Shipper. A Shipper, a Consignee and a Consolidator all interact for the purpose of agreeing on a Detailed Shipment Schedule.

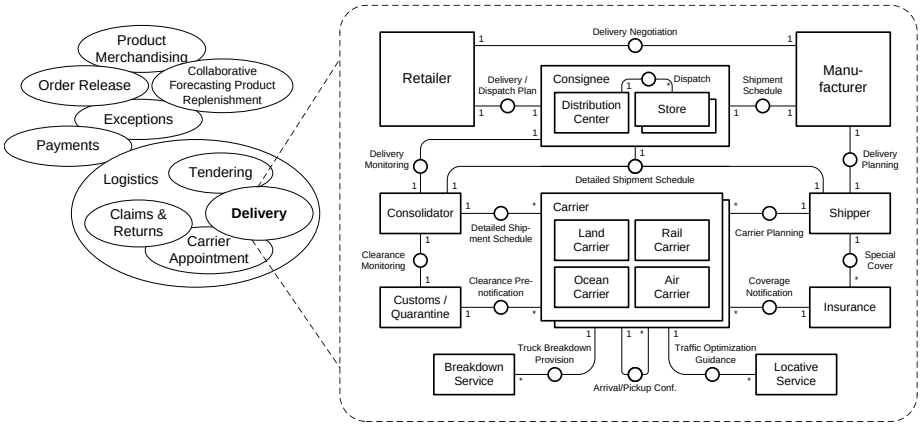

Fig. 1. High-level role-based view for Delivery domain of Logistics

Multiplicity of roles can be explicitly shown, as with Shipper and Carrier for instance. This indicates that several participants of role Carrier (overlaid boxes) are involved in one choreography instance as opposed to only one participant of role Shipper taking part (single box).

Another notational element, used in the representations for the Carrier and the Consignee roles, is that of role hierarchies (roles within roles). Role hierarchies can be used to express different relationships, and the exact relationship being represented needs to be specified by the modeller. In the case of the Consignee, the hierarchies mean that at least one of the sub-roles is involved in the interaction expressed against the super-role, i.e. a *part-of* relationship. Consignee also illustrates that further interaction dependencies can be expressed between sub-roles. Alternatively, the relationship between super- and sub-roles could reflect *specialisation*, where all sub-roles carry the same interaction dependencies as the super-role, and each may carry additional dependencies. This applies to the Carrier and its sub-roles.

With interaction dependencies between roles, through channels, in place, individual message exchanges can be captured. We propose that channels are assigned a set of message interactions in terms of message type and direction of flow. By this assignment to channels, the message interactions between collaborating roles is captured, although they remain unordered.

Milestones, scenarios and interactions. The models described so far are static: they do not describe control flow relationships between interactions. Below, we introduce viewpoints where interactions and their relationships are captured in more detail. Since the number of participants and interactions can be very large, these detailed viewpoints can be difficult to build and comprehend. Thus, a mechanism is needed to partition these models. This is achieved through the notion of milestones depicted in Figure 2.

Milestones (diamonds) represent the main global states choreography instances can be in and as such are used as "sign-posts". In complex choreogra-

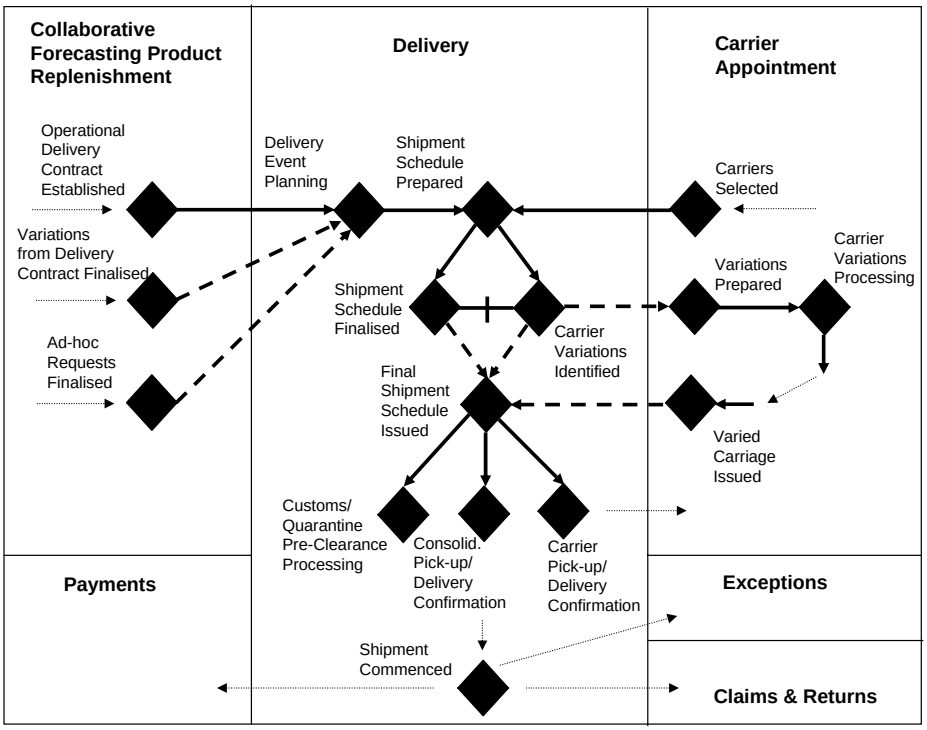

Fig. 2. Milestones related to Delivery domain

phies, milestones are useful since the details of processes which lead to milestones being reached can be omitted. To illustrate the point, Figure 2 depicts milestones primarily concerning Delivery, with some links to related milestones in processes of other domains shown. In this example, some milestones are related through *Precedes* relationships (arrowed lines). Some milestones might never be reached for a particular instance of a choreography. For example and as explained in details later, if a guard attached to an interaction evaluates to false, the interaction is skipped, and so are all other interactions that directly or transitively follow it according to the *Precedes* relation. Also, some milestones may be skipped as a result of *Inhibits* relationships. An *Inhibits* relationship (line crossed by a bar) expresses that if a milestone is reached, the target milestone can no longer be reached for the current instance of the choreography (i.e. it will be skipped). In the example, a Shipment Schedule Prepared milestone being reached will result in either a Shipment Schedule Finalised (SSF) or Carrier Variations Identified (CVI) milestone being eventually reached, but not both since these milestones "inhibit" each other. If we want to express that a milestone can still be reached even if a "preceding" milestone has been skipped, we should use a *Weak Precedes* relationship (arrowed dashed line) instead of a *Precedes* ones. Following the same example, Final Shipment Schedule (FSS) is "weak preceded" by the both the SSF and CVI milestones, and thus the FSS milestone will be reachable after one

of these two milestones has been reached and the other has been skipped. This example corresponds to a more general pattern where a *Weak Precedes* relationship is used to join two mutually exclusive paths. Another example of a *Weak Precedes* is given by the Delivery Event Processing milestone (e.g. next week's delivery) which is reached after the Operational Delivery Contract Established milestone is reached, whether or not the milestones Variations from Delivery Contract Finalised and Ad-Hoc Requests Finalised have been reached.

The last extension is to introduce *scenarios*, or specific "threads" of interactions, and to merge these scenarios to obtain detailed *interaction-based choreography models* (also called *interaction models* for short). The modelling of interactions is the central theme of choreography languages, however support for capturing scenarios (a well-established feature of analysis and design) is left open. With milestones in place, under our approach, scenarios identify interactions which serve to progress the milestones. Figure 3 illustrates how scenarios relate to milestones drawn from Figure 2, starting with a scenario yielding a milestone that is used as input for a second scenario. In the third scenario detailed in the example, the Retailer and Manufacturer negotiate required stock, and finally the Manufacturer releases order quantities. The Manufacturer then determines through the Shipper whether the allocated Carriers have capacity or not for the shipment, and accordingly two exclusive milestones result from the scenario. How large or small a scenario is, should reflect user requirements. In addition, scenarios might be split into sub-scenarios in order to allow for different variants of parts of a scenario.

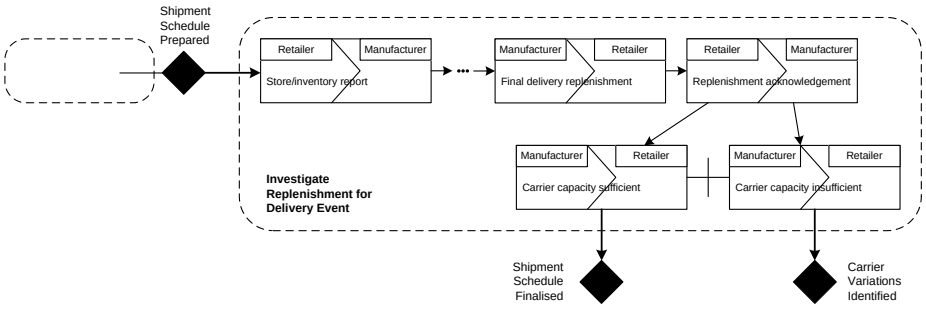

Fig. 3. Scenario for replenishment

Design method. The above considerations are summarised as a choreography design method in Figure 4. The rounded rectangles in this figure depict the activities of the choreography design method. The arrows describe which other activities are influenced by the outcome of an activity. First, domains need to be identified and decomposed into sub-domains. Next comes the identification of participants in the different domains. This identification mostly takes place early in the process but participants can also be included in later stages. With participants in place, role-based choreography models can be obtained by defining interaction dependencies between them.

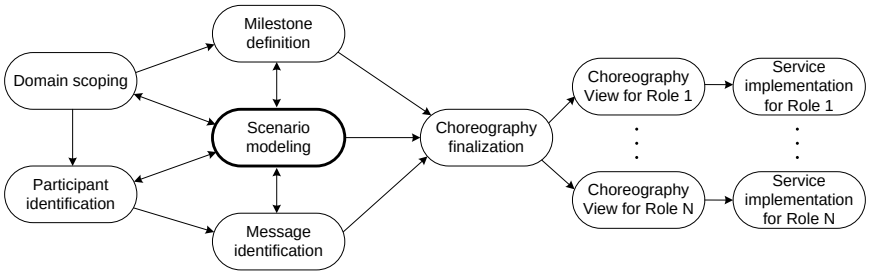

Fig. 4. Method for choreography design

Milestone models provide a high-level view of the behavioural aspect of choreography models, describing the main global states choreography instances can be in. Scenarios describe which interactions are needed to get from one milestone to another and thus only focus on one part of a choreography. During scenario modelling the designers might realise that they have to introduce more participants than they have considered so far. Furthermore, re-discussing the scope of a domain might be needed when scenario modelling goes down to the level of message exchanges. It might not be obvious to what domain a certain interaction belongs to, and this may affect the grouping of scenario models and interaction models into domains. For example, does a scenario triggered by an interaction "Pickup Appointment Change Request" belong to the domain "Carrier Appointment" or to "Delivery"? Message exchanges between the different participants are identified in this activity. First, only high-level descriptions are given. Later, message structures and contents are specified in detail.

The domain-relevant parts of the scenarios are aggregated into an integrated interaction model for the domain. Therefore, all participants, milestones, interactions and relationships between them are captured in this integrated choreography. Subsequently, the individual participants' views on the choreography model are generated and distributed to the participants who now proceed to design and implement their parts of the choreography. In our previous work [21], we have proposed algorithms for generating such local participants' views in the context of the Let's Dance language. Finally, existing implementations might be checked to determine if they already comply with the choreography model or if changes have to be made.

3 Choreography modelling viewpoints

The top-level viewpoint of the proposed choreography design method is the domain model. A domain model is composed of a set of domains arranged in a hierarchical relation. Leaf-level domains are mapped to different models corresponding to the proposed viewpoints on a choreography. Specifically, each leaf domain maps to: (i) a role-based model, (ii) a milestone model, (iii) a set of interaction models corresponding to scenarios, and (iv) an integrated interaction

model. In this section, we present an abstract syntax for each of these types of models.

3.1 Role-based choreography models

The previous section has already motivated role-based models and has introduced the intended meaning of the different elements of such models. Figure 5 summarises the corresponding graphical representations: Rectangles represent roles. Concrete participants are bound at design-time or at run-time. Small

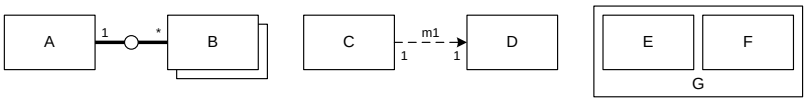

Fig. 5. Diagram elements for role-based models

circles represent channels while message links are depicted as dashed arrows. Message links are directed channels with a particular message type assigned. Cardinality of channels is represented by either "1" or "*" attached to the endpoint of a channel. Multiplicity of roles is represented by a double rectangle like for role B. Hierarchy is represented by containment in the diagram.

We define a role-based choreography model C_R to be a tuple (R, RM, C, $Senders$, $Receivers$, $Card$, Msg, CM, $Parent$) where:

- R is the set of all roles,
- $RM : R \rightarrow \{one, many\}$ is a function assigning a multiplicity to a role,
- C is the set of all channels,
- $Senders, Receivers : C \rightarrow \wp(R)$ assign the set of roles to a channel who can send / receive messages over the channel, if the sets of senders and receivers are disjunct, the channel is a message link ($ML := \{ml \in C \mid Senders(c) \cap Receivers(c) = \emptyset\}$),
- $Card : \{(c, r) \in C \times R \mid r \in Senders(c) \cup Receivers(c)\} \rightarrow \{one, many\}$ is a function assigning a cardinality to role r for channel c,
- Msg is the set of all message types,
- $CM : ML \rightarrow Msg$ is a partial function linking message links to message types,
- $Parent \subseteq R \times R$ specifies the hierarchical relationships between a role and its sub-roles ($Parent$ must be acyclic) and
- the cardinality "many" is only used in the presence of the multiplicity "many" of the corresponding role: $\forall (c, r) \in Card\ [c = many \Rightarrow RM(r) = many]$.

3.2 Milestone, scenario and interaction models

In previous work [21] we have presented Let's Dance as a language for modelling service interactions and their dependencies. Below, we enrich the language with

milestones. That way models like they were motivated in section 2 are specified. Figure 6 contains the graphical elements for representing milestone and interaction models. Milestones are represented by diamond shapes and interactions

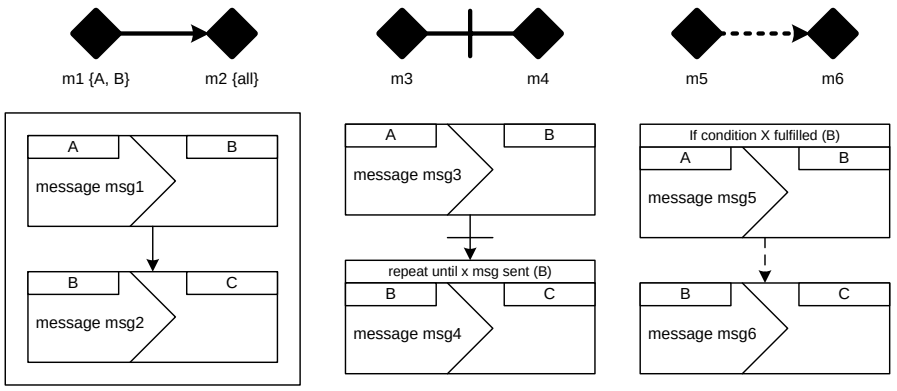

Fig. 6. Diagram elements for milestone and interaction diagrams

using rectangles. These roles indicated in brackets describe which participant has to be notified as soon as the milestone is reached. If all participating roles are to be synchronised "all" appears in brackets or the brackets are omitted.

The *Precedes* relationship between two elements (milestones or interactions) e1 and e2 indicates that e1 must have been executed / reached before e2 can be executed / reached. A *WeakPrecedes* relationship between e1 and e2 indicates that e2 can only be executed / reached after e1 has been either executed / reached or after it has been skipped. An *Inhibits* relationships between e1 and e2 indicates that e2 cannot be executed / reached after e1 has been executed / reached (i.e. e2 is then skipped). Two-directional *Inhibits* relationships are represented like in the case of m3 and m4.

Interactions can either be elementary or composite. In the case of elementary interactions a participant of a certain role sends a message of a given type to another participant. In Figure 6 a participant of role A sends a message of type m1 to a participant of role B. Composite interactions allow for grouping of one or more interactions. Interactions can be guarded, i.e. they may only occur if a guard condition evaluates to true, or they can be repeated. It is specified which participant evaluates the guard condition and the repetition expression.

Below, we introduce three types of models corresponding to different viewpoints into a choreography: milestone, scenario and interaction models. Milestone models are composed only of milestones and relationships between them. Meanwhile, scenario models are composed of milestones, interactions and relationships between them. The purpose of a scenario model is to show how to go from a set of milestones to another. Scenario models are not limited to one domain and may be used to show dependencies between milestones and interactions from different domains. Scenario models can be nested, i.e. different sub-scenario models

may refine a given scenario model, showing specific variants. A similar notion can be found in Message Sequence Charts [18] which capture specific paths of interactions between a number of parties. But in contrast to traditional MSCs, we allow scenario or sub-scenario model to show alternative paths (i.e. we allow conditional branching in scenario models)[5]. Finally, interaction models show all milestones and interactions from one domain as well as their relationships.

We introduce a unified abstract syntax for milestone, scenario and interaction models by defining interaction models as the most general viewpoint, and the other two as special cases. An interaction model C_I is a tuple (I, M, RI, RT, GI, R, RM, c_0, $Precedes$, $WeakPrecedes$, $Inhibits$, $Parent$, $Sends$, $Receives$, MR, Msg, IM) where

- I is the set of milestones and interactions,
- $M \subseteq I$ is the set of milestones,
- $RI \subseteq I \setminus M$ is the set of repeated interactions,
- $RT : RI \rightarrow \{w, r, fs, fc\}$ links a repeated interaction to a repetition type (either while, repeat, for each (sequential) or for each (concurrent)),
- $GI \subseteq I \setminus M$ is the set of guarded interactions,
- R is the set of roles,
- the function $RM : R \rightarrow \{many, one\}$ specifies whether many or just one participant of a role is involved in the choreography,
- $c_0 \in I \setminus M$ is the top-level interaction of the choreography,
- $Precedes, WeakPrecedes, Inhibits \subseteq I \times I$ are binary relations over I,
- $Parent \subseteq (I \setminus M) \times I$ is the relation between interactions and their direct sub-interactions and milestones, defining the set of elementary interactions $EI := \{i \in (I \setminus M) \mid i \notin range(Parent)\}$
- the partial functions $Sends, Receives : EI \rightarrow R$ link elementary interactions to sending and receiving roles,
- $MR : M \rightarrow \wp(R)$ links milestones to sets of roles that are to be synchronised,
- Msg is the set of all message types and
- $IM : EI \rightarrow Msg$ links elementary interactions to message types.

A milestone model C_M is an interaction model where $I = M \cup \{c_0\}$. Meanwhile, a scenario model is an interaction model where no milestone is both the source and the target of $Precedes$ and/or $WeakPrecedes$ relationships, that is: $\forall m \in M\ [\neg\exists i, j \in I\ ((i\ Precedes\ m \vee i\ WeakPrecedes\ m) \wedge (m\ Precedes\ j \vee m\ WeakPrecedes\ j))]$

4 Consistency analysis

Consistency checking is an essential aspect in multi-viewpoint approaches [8]. In this section we introduce consistency rules between role-based choreography models and interaction models as well as between milestone models and interaction models based on the abstract syntaxes given in the previous sections.

[5] While this feature is not supported in traditional MSCs, it is supported in various extensions to MSCs such as Live Sequence Charts (LSCs) [10].

4.1 Role-based choreography models vs. interaction models

Consistency between a role-based choreography model $C_R = (R_R,\ RM_R,\ C_R,\ Senders_R,\ Receivers_R,\ Card_R,\ Msg_R,\ CM_R,\ Parent_R)$ and an interaction model $C_I = (I_I,\ M_I,\ RI_I,\ RT_I,\ GI_I,\ R_I,\ RM_I,\ c_0,\ Precedes,\ WeakPrecedes,\ Inhibits,\ Parent_I,\ Sends_I,\ Receives_I,\ MR_I,\ Msg_I,\ IM_I)$ is given if:

- all roles of C_I are present in C_R and have the same multiplicity: $R_I \subseteq R_R$ and $RM_I \subseteq RM_R$,
- for all elementary interactions there is a corresponding channel (one channel can correspond to many interactions): $\forall i \in EI_I\ [\exists c \in C_R\ (Sends_I(i) \in Senders_R(c) \wedge Receives_I(i) \in Receivers_R(c) \wedge (CM_R(c) = IM_I(i) \vee c \notin dom(CM_R)))]$ and
- each role has to be involved in at least one corresponding elementary interaction for every channel it is connected to: $\forall c \in C_R \forall r \in Senders_R(c) \cup Receivers_R(c)\ [\exists i \in EI\ ((r \in \{Sends_I(i)\} \cap Senders_R(c) \vee r \in \{Receives_I(i)\} \cap Receivers_R(c)) \wedge (CM_R(c) = IM_I(i) \vee c \notin dom(CM_R)))]$.

4.2 Milestone models vs. interaction models

Consistency between a milestone model $C_M = (I_M,\ M_M,\ RI_M,\ RT_M,\ GI_M,\ R_M, RM_M, c_{0M}, Precedes_M, WeakPrecedes_M, Inhibits_M, Parent_M, Sends_M,\ Receives_M,\ MR_M,\ Msg_M,\ IM_M)$ and an interaction model $C_I = (I_I,\ M_I,\ RI_I, RT_I, GI_I, R_I, RM_I, c_{0I}, Precedes_I, WeakPrecedes_I, Inhibits_I, Parent_I,\ Sends_I,\ Receives_I,\ MR_I,\ Msg_I,\ IM_I)$ is given if all constraints defined in the milestone model are ensured in the interaction model. Constraints are given by the $Precedes_M$, $WeakPrecedes_M$ and $Inhibits_M$ relationships.

1: $I_{(1,1)} := \{i \in I_I \mid \neg\exists j \in RI_I\ (j\ Parent_I^*\ i \wedge RT_I(j) \neq r) \wedge$
2: $\quad \neg\exists j \in I_I (j\ Precedes_I^*\ i \wedge (j \in GI_I \vee (\exists k \in I_I (k\ Inhibits^+\ j))))\}$
3: $\forall (m_1, m_2) \in Precedes_M\ [m_1\ Precedes_I^+\ m_2 \vee$
4: $\quad (m_1 \in I_{(1,1)} \wedge m_1 (Precedes_I \cup WeakPrecedes_I)^+ m_2)]$
5: $\forall (m_1, m_2) \in WeakPrecedes_M\ [m_1 (Precedes_I \cup WeakPrecedes_I)^+ m_2]$
6: $\forall (m_1, m_2) \in Inhibits_M [\exists i, j \in I_I (i\ Inhibits_I\ j \wedge j\ Precedes_I^*\ m_2 \wedge$
7: $\quad (i\ Precedes_I^*\ m_1 \vee (i \in I_{(1,1)} \wedge i (Precedes_I \cup WeakPrecedes_I)^* m_1)))]$

Fig. 7. Consistency checking between milestone models and interaction models

Figure 7 presents how consistency between a milestone model C_M and an interaction model C_I can be checked. We assume that all composite interactions are repeated and that all interactions and milestones are reachable. Furthermore, all *Inhibits* relationships must have an effect. In previous work [21] we have introduced algorithms for expanding choreography models and for identifying unreachable interactions and obsolete *Inhibits* relationships. A $Precedes_M$ relationship is ensured in C_I if there is a path of $Precedes_I$ relationships from one milestone to the other or if the first milestone is always eventually reached ($m_1 \in I_{(1,1)}$) and there is a path of $Precedes_I$ and $WeakPrecedes_I$ relationships (lines 3-4). Lines 1-2 present how $I_{(1,1)}$ can be identified: There must be

no preceding guarded interaction or an $Inhibits_I$ relationship targeting a preceding interaction. A $WeakPrecedes_M$ relationship is ensured if there is a path of $Precedes_I$ and $WeakPrecedes_I$ relationships. Finally, an $Inhibits_M$ relationship is ensured if a preceding interaction of m_1 is the source of an $Inhibits_I$ relationship targeting a preceding interaction of m_2.

Additional constraints can be added in the interaction model. For example, if two milestones m_1 and m_2 are not ordered in the milestone model, we can introduce a $Precedes_I$ relationship between m_1 and m_2 in the interaction model without violating the consistency rules.

5 Related work

Service choreography description has been the subject of intensive research and standardisation. An early attempt was BPSS [20] where global models are captured as flows of interactions using flowchart-like constructs. WSCI [1] represents another approach wherein global service interaction models are defined as collections of inter-connected local models (as opposed to a single global model). Control dependencies are described within each individual local model. More recently, the WS-CDL initiative [14] led to a language that follows the line of BPSS insofar as global service behaviour is described as flows of interactions. WS-CDL goes further than BPSS in the level of details at which interaction flows are described. In fact, WS-CDL can be seen as a programming-in-the-large language for Web services since it relies on imperative programming constructs. The work presented in this paper is complementary to these initiatives, as it defines viewpoints and notational elements that operate at a higher level of abstraction.

In [4], the authors consider the use of state machines for describing local models of service interactions. While state machines lead to simple models for highly sequential scenarios, they may lead to spaghetti-like models when used to capture scenarios with parallelism and cancellation (e.g. scenarios where a given interaction may occur at any time during the execution of another set of interactions). Nonetheless, state machines have been shown to be a suitable formal foundation for reasoning about service models, e.g. determining the boundedness of service queues in service conversations [11]. This latter reference surveys a number of approaches for describing service interaction models based on communicating state machines.

The concept of multi-viewpoint modelling of distributed systems has been advocated in the RM-ODP reference model [12], which defines various viewpoints such as enterprise viewpoint (high-level purpose and policies), computational viewpoint (functional decomposition and interface definition), information viewpoint, etc. Dijkman [7] defines a framework for capturing multiple viewpoints over distributed systems and applies the framework to formalise RM-ODP's enterprise and computational viewpoints. Dijkman's framework is defined as an extension to an Architecture Description Language (ADL), namely ISDL, that includes notational elements similar to those found in the role-based and interaction viewpoints considered in this paper, although ISDL does not directly sup-

port our role decomposition construct. A discussion on the application of ISDL for service choreography modelling is presented in [17]. However, the suitability of ISDL for capturing complex service interactions (e.g. involving multicast) is unproven. Also, ISDL does not have a counter-part for the milestone-based viewpoint which is useful when dealing with large service choreographies.

Colombo et al [5] propose a methodology for service composition that starts with the definition of so-called *social models* that capture business entities and their dependencies. These models are similar to our role-based models, with the difference that our role-based models capture more detail than social models, e.g. role-based models capture the multiplicity of interaction dependencies between roles. In the second phase of the methodology of Colombo et al., a process model capturing the behaviour of a service composition is constructed. This process model is derived from a set of ECA rules and it is encoded as a finite state machine. This approach is suitable for capturing sequential interactions, but arguably not for capturing concurrent interactions. In contrast, we take as starting point a language in which concurrent interactions can be naturally captured. Indeed, if two interactions in a Let's Dance model are not related through a "Precedes" dependency, either directly or transitively, these interactions may occur in any order or concurrently.

Foster et al. [9] propose a method for Web service composition in which scenario models expressed as MSCs are compared with orchestration models expressed in BPEL. In this context, orchestration models are choreography models projected over a single role (i.e. a local view on a choreography model). To check consistency between scenario models and orchestration models, these models are compiled into Labelled Transition Systems (LTSs). The resulting LTSs are compared in terms of their traces to check that the behaviour of the scenario model is contained in the behaviour of the orchestration model. Our proposal is complementary insofar as we focus on capturing the relationships between scenario models and higher-level models (i.e. milestone models and role-based models).

Seel et al. [19] present a requirements framework for inter-organizational business process models. A distinction is made between interaction points for collaborating employees and departments and interaction points for information systems. Corresponding extensions to Event-driven Process Chains (EPC [15]) are introduced.

The role-based view presented in this paper can be seen as a formalized representation of the "service decomposition" diagrams proposed in [13]. Our role-based views contain more information than those of [13] and they can be directly linked with milestone, scenario and detailed interaction models.

6 Conclusion and outlook

We motivated the need for choreography languages to unhinge from present focus on implementation considerations concerning message interactions, in service to analysis and design of wide-spanning B2B domains, and the collaborations of interacting participants in particular. Our proposal was illustrated using the

VICS global supply chain standard, offering insights into the large and intricate landscape that needs to be penetrated to get down to detailed interaction-based choreography models. We developed domain scoping (essentially equivalent to process hierarchies) and role-based choreography models as horizontal partitions, together with milestones as vertical partitions. For lower levels, we refined interaction-based choreography modelling to support scenarios through which milestones are progressed. Consistency of models was formally analysed, with one question being left open: how to integrate a set of scenario models for a given domain into a single choreography model? This integration, which is left as future work, should be guided by the milestone model of the domain, given that each scenario model covers a different set of milestones.

Further research will spawn in two directions which are relevant for impact in web services composition environments. The first is to validate the modelling views against further use cases and to refine the modelling proposals accordingly. The second is to determine how well such extended considerations of choreography modelling can be mapped into intermediate, more implementation focused languages such as WS-CDL and WS-BPEL. Along the way, the Let's Dance tool will be extended to support the extensions proposed.

Acknowledgement. The authors wish to thank Remco Dijkman for valuable feedback on a draft of this paper. The third author is funded by a fellowship from Queensland Government and SAP.

References

1. Assaf Arkin et al. Web Service Choreography Interface (WSCI) 1.0. Technical report, Aug 2002. http://www.w3.org/TR/2002/NOTE-wsci-20020808.
2. K. Baïna, B. Benatallah, F. Casati, and F. Toumani. Model-driven web services development. In *Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAISE'04)*, Riga, Latvia, 7-11 June 2004. Springer.
3. A. Barros, M. Dumas, and A. ter Hofstede. Service interaction patterns. In *Proceedings of the 3rd International Conference on Business Process Management*, Nancy, France, September 2005. Springer.
4. B. Benatallah, F. Casati, F. Toumani, and R. Hamadi. Conceptual modeling of web service conversations. In *15th International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 2681 of *LNCS*, pages 449–467, Klagenfurth, Austria, June 2003.
5. E. Colombo, J. Mylopoulos, and P. Spoletini. Modeling and analyzing context-aware composition of services. In *Proceedings of the 3rd International Conference on Service-Oriented Computing (ICSOC)*, pages 198–213, Amsterdam, The Netherlands, December 2005. Springer.
6. G. Decker, J. M. Zaha, and M. Dumas. Execution semantics for service choreographies. In *Proceedings 3rd International Workshop on Web Services and Formal Methods (WS-FM 2006)*, Vienna, Austria, Sept 2006. Springer.
7. R. Dijkman. *Consistency in Multi-Viewpoint Architectural Design.* PhD thesis, University of Twente, The Netherlands, 2006.
8. R. Dijkman and M. Dumas. Service-oriented design: A multi-viewpoint approach. *International Journal of Cooperative Information Systems*, 13(4):337–368, December 2004.

9. H. Foster, S. Uchitel, J. Magee, and J. Kramer. LTSA-WS: a tool for model-based verification of web service compositions and choreography. In *Proceeding of the 28th international conference on Software Engineering (ICSE) – Research Demonstration*, pages 771–774, Shanghai, China, May 2006. ACM Press.
10. D. Harel and R. Marelly. *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, 2003.
11. R. Hull and J. Su. Tools for composite web services: a short overview. *SIGMOD Rec.*, 34(2):86–95, 2005.
12. ITU-T/ISO. Open distributed processing reference model. Technical Report ITU-T X.901..904 – ISO/IEC 10746-1.4, ITU-T/ISO, 1994–1997.
13. S. Jones. *Enterprise SOA Adoption Strategies*. InfoQ Enterprise Software Development Series, 2006. Available at: http://www.infoq.com/minibooks/enterprise-soa.
14. N. Kavantzas, D. Burdett, G. Ritzinger, and Y. Lafon. Web services choreography description language version 1.0, w3c candidate recommendation. Technical report, November 2005. http://www.w3.org/TR/ws-cdl-10.
15. G. Keller, M. Nttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Verffentlichungen des Instituts fr Wirtschaftsinformatik (IWi) 89, Universitt des Saarlandes, January 1992.
16. M. Papazoglou and W. van den Heuvel. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology (IJWET)*, 2006.
17. D. Quartel, R. Dijkman, and M. van Sinderen. Methodological support for service-oriented design with ISDL. In *Proceedings of the 2nd Intenational Conference on Service-Oriented Computing (ICSOC)*, pages 1–10, New York NY, USA, November 2004. Springer.
18. E. Rudolph, J. Grabowski, and P. Graubmann. Tutorial on Message Sequence Charts. *Computer Networks and ISDN Systems*, 28(12):1629–1641, 1996.
19. C. Seel and D. Vanderhaeghen. Meta-Model based Extensions of the EPC for Inter-Organisational Process Modelling. In *Proceedings 4th Workshop on Geschft-sprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK 2005)*, volume 167 of *CEUR*, pages 117–136, Hamburg, Germany, December 2005.
20. UN/CEFACT and OASIS. ebXML Business Process Specification Schema (Version 1.01). http://www.ebxml.org/specs/ebBPSS.pdf, 2001.
21. J. M. Zaha, M. Dumas, A. ter Hofstede, A. Barros, and G. Decker. Service interaction modeling: Bridging global and local views. In *Proceedings 10th IEEE International EDOC Conference (EDOC 2006)*, Hong Kong, China, October 2006.
22. O. Zimmermann, P. Krogdahl, and C. Gee. Elements of service-oriented analysis and design. Available at: www.ibm.com/developerworks/library/ws-soad1, 2004.