Sequential Composition of Multi-Party Choreographies

Andreas Schönberger and Guido Wirtz
Distributed and Mobile Systems Group
University of Bamberg
Bamberg, Germany
$\{$*andreas.schoenberger*$|$*guido.wirtz*$\}$@*uni-bamberg.de*

***Abstract*—For Business-To-Business (B2Bi) scenarios, the application of choreography and orchestration technology has become a core technique for resolving discrepancies between the interaction logic of individual partners and the intended overall message flow. While orchestrations govern the message exchanges of each single partner, choreographies define constraints and requirements for the message flow between all partners. Using choreographies, B2Bi scenarios can be analyzed from a global perspective before the business services of the integration partners for implementing orchestrations are developed.**
So far, B2Bi choreographies mostly have been binary, i.e., performed by exactly two partners. This paper shows how multi-party B2Bi choreographies can be composed from binary choreographies, how the multi-party perspective lends itself to attacking the so-called *partial termination* problem and how projections for the individual partners can be derived.

***Keywords*-multi-party choreographies; business services; business process management; B2Bi; ebXML BPSS**

I. Introduction

The choreography-orchestration tool-chain is a natural candidate to be applied to Business-to-Business integration (B2Bi). Choreographies may be used for modeling the interactions between integration partners from a global point of view and thus for defining and agreeing upon the business documents to be exchanged and the sequence of these exchanges. In a second step, each integration partner can leverage orchestration technology for implementing its obligation defined by the choreography. Thus, defining and analyzing choreographies has become an important method for defining the requirements and setting the context of the implementation of integration services. In the B2Bi domain, many approaches ([1], [2], [3], [4]) focus on strictly binary choreographies, i.e., on interactions between exactly two integration partners. While binary choreographies cover the majority of current B2Bi scenarios, multi-party scenarios actually are an implication of the concepts of supply chains/supply networks. Consequently, Huemer and Hofreiter argue [5] that interactions with more than one business partner at least have to be defined locally. Moreover, there are some real world examples that are not binary. For example, RosettaNet, a leading supply chain community of the ICT industry, defines the so-called "Order-To-Cash With Logistics Service Provider Scenario"[1] depicted in figure 1. In this scenario, a Customer, a Supplier and a Logistics Service Provider (LSP) role (represented by BPMN pools) are using RosettaNet PIPs (small cuboids labeled 3A4, 3A8 and so on) for exchanging business documents. Moreover, the local actions of each role for processing the business documents exchanged via PIPs are given. However, figure 1 only describes the intended flow of interactions and leaves out what happens if communication errors occur or if, for example, the Supplier and LSP role are not able to agree upon the provision of transportation services. Note that such technical/business errors only affect two of the three roles immediately (send and receive actions are defined for one partner only). This raises the question whether or not erroneous behavior may have an effect on the remaining role and how to detect problematic execution paths.

In order to provide a widely applicable solution to this problem, this paper defines how multi-party choreographies can be composed from existing binary choreographies. Further, the negative effect of technical/business errors between two partners on the remaining partners is captured as the so-called *partial termination* problem and an algorithm for identifying problematic execution paths is sketched. Additionally, an algorithm for deriving role projections from multi-party choreographies is given for fostering straightforward systems development.

Section II pins down the notion of B2Bi choreography used here. Section III introduces sequential multi-party (SeqMP) choreographies as new class of multi-party B2Bi choreographies. Also, the *partial termination* and *role projection* problems are introduced and their solutions are described. Finally, section IV discusses related work and section V concludes and points out directions for future work.

II. B2Bi choreographies

ebXML BPSS (ebBP, [6]) is the leading B2Bi choreography interchange format. The availability of domain specific concepts such as support for community-defined business document libraries or B2Bi Quality-of-Service (QoS) parameters like security and reliability make ebBP particularly useful for capturing the specification of B2Bi scenarios. ebBP concentrates on the interactions between integration partners, i.e., the sequence and types of business document exchanges (e.g., PIP cuboids in figure 1) and not on the local activities of integration partners for sending, receiving and processing these (e.g., activities within the BPMN pools in

[1]cf. http://www.rosettanet.org/Support/ImplementingRosettaNetStandards/eBusinessProcessScenarioLibrary/OrdertoCash/tabid/3320/Default.aspx

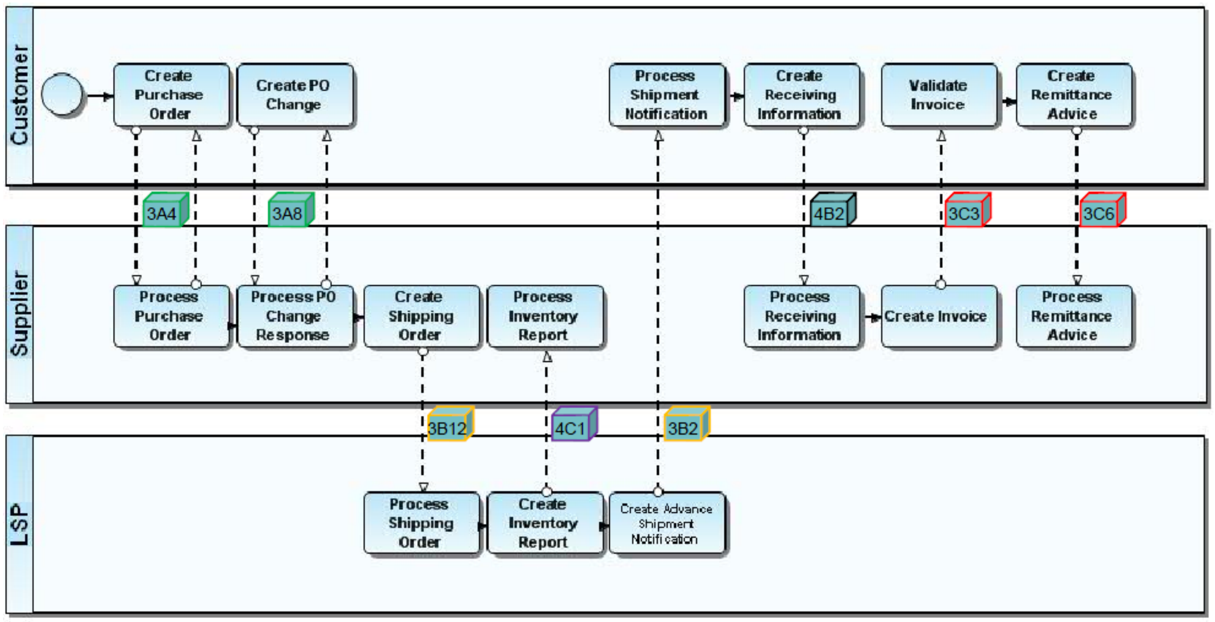

Figure 1. RosettaNet Order to Cash with Logistic Service Provider Scenario (from RosettaNet[1])

figure 1). In the terminology coined in [7], this corresponds to the specification of *interaction* choreographies instead of *interconnection* choreographies. For the purpose of the work at hand, it is sufficient to know that the concept of ebBP *BusinessCollaborations* (BC) can be used to capture the choreography of business document exchanges (the PIPs of figure 1) between two or more integration partner roles. ebBP *BusinessCollaborationActivities* (BCA) can be used to specify the execution of an existing BC within another BC. This work builds upon the concept of binary BCAs, i.e., BCAs with exactly two roles, that are performed in sequence according to some guards that distinguish between the results of a BCA. As long as these concepts are available, this work is also applicable to other interaction-style choreography languages. Note that ebBP is an XML-based B2Bi choreography interchange format and therefore needs a visual language to be useful for B2Bi modeling. The UN/CEFACT Modeling Methodology (UMM, [8]), the Business Choreography Language (BCL, [9]) as well as the upcoming BPMN 2.0 choreographies ([10], section 11) are visual B2Bi choreography languages.

Figure 2 shows the use case of figure 1 in BPMN choreography notation. The use case is remodeled as a series of binary BCAs that are composed of 1 to 3 PIPs. The binary choreographies (BCAs) are modeled as so-called BPMN *Call Choreographies (Collapsed)* and visualized as rounded rectangles with a '+' at the bottom. The two bands at the top and at the bottom contain the integration partner roles participating in the call choreographies. The text in the middle contains an id (c1...c4), a name and the PIP types contained in the call choreographies (3A4, 3A8 and so on). Using the PIP types, it is easy to identify which call choreography corresponds to which part of the original RosettaNet choreography definition. For defining the detailed structure of each binary call choreography, existing approaches are ready for use (cf. [4], [11], [12]).

III. SEQ-MP CHOREOGRAPHIES

This section first motivates the class of *sequential multi-party* (SeqMP) choreographies and gives its formal definition in subsection III-A. Subsection III-B then identifies two important problems in SeqMP choreographies and subsection III-C provides algorithms for solving these.

A. Definition

The class of SeqMP choreographies is tailored to the needs of B2Bi. By analyzing 100 scenarios of the publicly available *RosettaNet implementation guidelines* (for implementing B2Bi processes), we have discovered that the majority of interactions is binary (84 scenarios), i.e., are performed between exactly two integration partners. This is in line with academic research (cf. section I). The remaining multi-party interactions of our analysis can be split up into binary interactions. We have identified two factors that foster decomposability into binary interactions. First, the atomic building blocks of many B2Bi processes are binary transaction-like concepts for the exchange of request business documents and optional response business documents. In the case of RosettaNet, these atomic building blocks are called Partner Interface Processes (PIP) and despite the simple structure of PIPs the economic value exchanged using PIPs is worth billions of dollars (RosettaNet Standards Assessment 2008[2]). Similar concepts

[2]http://www.rosettanet.org.my/Download/2009 ImplementationStatistics 05.26.09.pdf

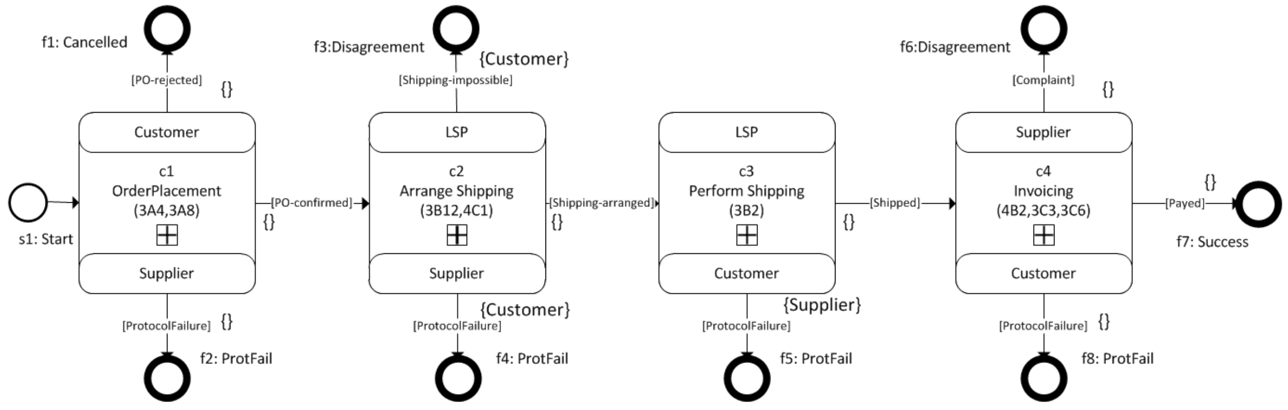

Figure 2. SeqMP model of the RosettaNet use case

can also be found in ebBP, UMM or BCL. Second, the control flow defined typically is fairly simple, i.e., does not apply concepts like parallel structures or hierarchical decomposition. This, in turn, is in line with a multi-case study of Reijers et al. [13] who report the results of an investigation of 16 business processes from six Dutch organizations: *"Business processes turned out to be completely sequential structures. Their routing complexity was only determined by choice constructs and iterations."* This finding is also backed by the B2Bi models created for the eBIZ-TCF project (http://www.moda-ml.net/moda-ml/repository/ebbp/v2008-1/en/) that do not use concurrent behavior.

Now, as control flow of B2Bi interactions tends to be simple and the atomic building blocks are binary, multi-party choreographies can be viewed as sequences of binary choreographies, i.e., as binary BCAs.

The advantage of using binary BCAs as building blocks for multi-party choreographies is that integration partners can be assumed to have agreed upon the result of the binary BCA. Also, both partners start and terminate the BCA more or less in lock-step. Consequently, the result of the binary BCAs can be used for routing the control flow of the multi-party choreography. In figure 2, this is indicated by guards (expressions placed in brackets) that are attached to the transitions. The guard `[PO-confirmed]` after BCA `c1` captures confirmation of the purchase order exchanged whereas `[PO-rejected]` captures rejection. The corresponding transitions of these guards link to BCA `c2` or end state `f1` accordingly. The annotations in curly braces are explained later. This concept of defining multi-party choreographies as sequentially performed binary BCAs with branching structures for defining control flow is reflected in the following definition.

Definition 3.1 (SeqMP Choreography):
A SeqMP choreography is a directed graph SeqMP (s_0, F,SBCA,T,R,RA) with the following elements:

- s_0 the (unique) start state.
- F a non-empty set of final states.
- SBCA a non-empty set of binary BCAs.
- T the union of the following transition sets
  - $T^{start} = \{(s_0,\text{true},bca)\}$, bca $\in$ SBCA
  - $T^{end} \subseteq$ SBCA $\times$ G $\times$ F
  - $T^{flow} \subseteq$ SBCA $\times$ G $\times$ SBCA

  where G is a set of boolean guards consisting of the constants {true, else} and any disjunction of terms that consist of the names of the possible results of the BCA just performed. A term is evaluated upon termination of a BCA and becomes true when the BCA produces the corresponding result. 'else' becomes true if all guards of all other transitions with the same source become false.
- R the set of roles of the SeqMP process.
- RA: SBCA $\rightarrow$ R^2, a role assignment function that assigns exactly two roles to each BCA. □

Further, the following auxiliary functions are defined.

Definition 3.2 (SeqMP Auxiliary Functions):

- .-notation/#-notation is used for accessing the components of a tuple by name/index.
- *namesB* the function that computes the names of the results of a BCA.
- *namesG* the function that computes the names contained in a guard.
- A path *path*(a,b) between two nodes a,b $\in \{s_0\} \cup F \cup$ SBCA is a sequence of nodes a,$n_{1..x}$,b such that for all i=1...x-1, $(n_i,g_i,n_{i+1}) \wedge (a,g_a,n_1) \wedge (n_x,g_x,b) \in T$. The length of a path(a,b) *length*(path(a,b)) is the number of nodes in the sequence. Let Path(a,b) be the set of all paths between a and b. □

Based on this definition, it is possible to characterize the validity of SeqMP processes using the following three conditions.

Definition 3.3 (Valid SeqMP Choreography):
A SeqMP choreography smp is valid iff the following three conditions hold:

1) **Subsequent role participation**:
$\forall$ (s_1, g, s_2) $\in$ smp.T^{flow}: $RA(s_1) \cap RA(s_2) \neq \emptyset$, i.e., for two subsequent BCAs at least one of the assigned

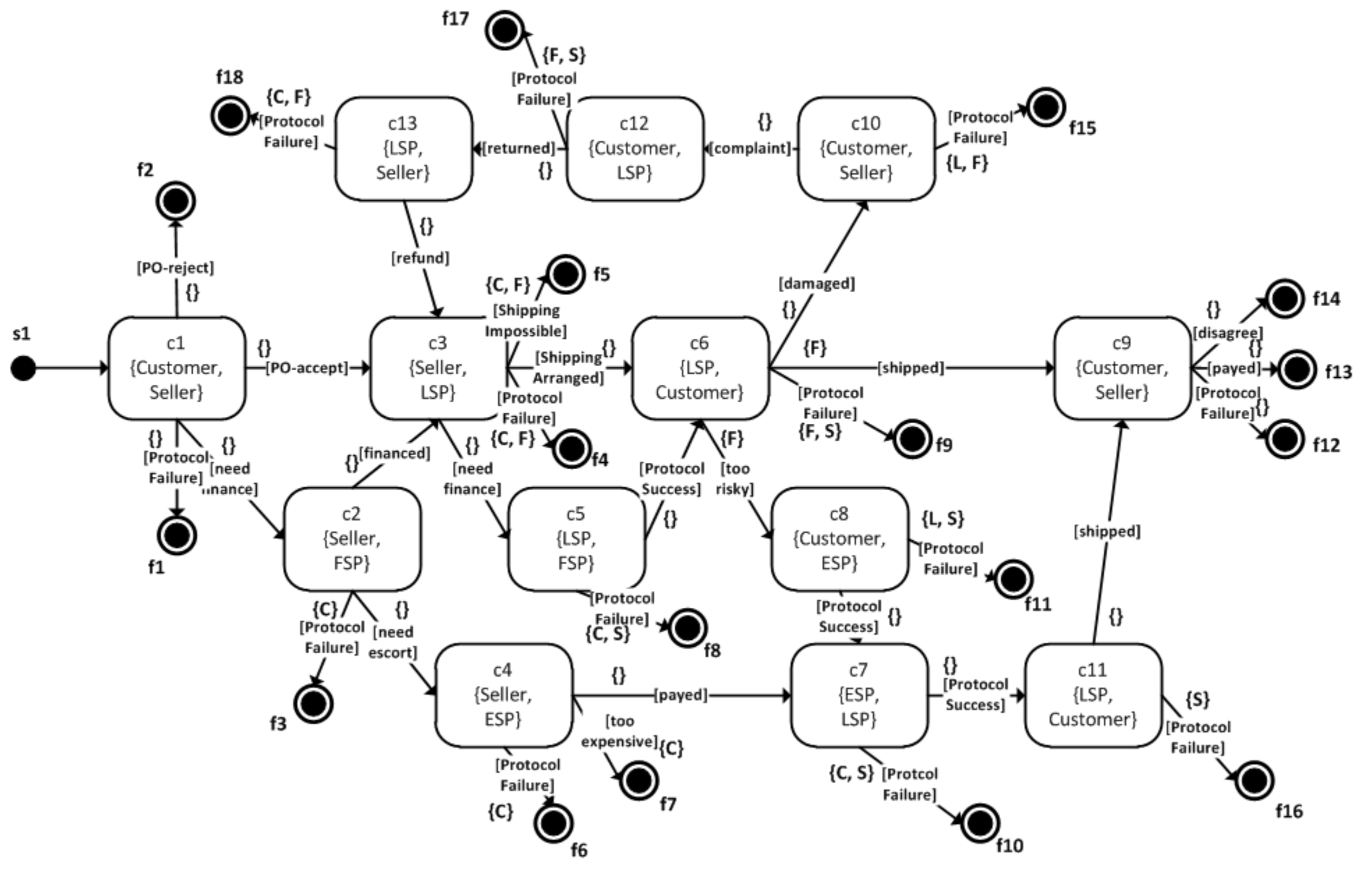

Figure 3. Seq-MP model of a complex use case (conflated visualization)

roles must be the same (thence enabling synchronization between terminating one BCA and starting the next).

2) **Guard validity**:
Let $\text{SUCC}_{bca} \subseteq$ smp.T be the set of outgoing transitions from some bca $\in$ smp.BCA with: $\forall$ t (t#1, t#2, t#3) $\in \text{SUCC}_{bca}$: t#1 = bca. Then, the guards of bca are valid iff:
$(|\text{SUCC}_{bca}| = 1 \wedge \text{for } \{t\} = \text{SUCC}_{bca}\text{: t\#2 = true}) \vee (\forall \text{ t} \in \text{SUCC}_{bca}\text{: t\#2} \neq \text{true} \wedge ((\bigcup_{t \in \text{SUCC}_{bca}} \text{namesG(t\#2)} = \text{namesB(bca)}) \vee (\bigcup_{t \in \text{SUCC}_{bca}} \text{namesG(t\#2)} \subset \text{namesB(bca)} \wedge \exists \text{ t1} \in \text{SUCC}_{bca}\text{: t1\#2 = else} \wedge \nexists \text{ t2} \in \text{SUCC}_{bca}\text{, t1} \neq \text{t2: t2\#2 = else})) \wedge \forall \text{ t3, t4} \in \text{SUCC}_{bca}\text{, t3} \neq \text{t4: namesG(t3\#2)} \cap \text{namesG(t4\#2)} = \emptyset)$

3) **Connectedness**:
$(\forall \text{ f} \in \text{smp.F: Path(smp.s}_0\text{, f)} \neq \emptyset) \wedge \forall \text{ bca} \in \text{smp.SBCA: Path(smp.s}_0\text{, bca)} \neq \emptyset \wedge \exists \text{ f} \in \text{smp.F: Path(bca, f)} \neq \emptyset$. □

Actually, it would not be hard to extend this definition to using multi-party BCAs as building blocks (and even the algorithms in section III-C would work) as long as an agreed-upon result among all participants of the BCAs would be guaranteed. This, however, does not seem to hold true for many real-world scenarios.

B. Problems in Multi-Party Choreographies

Remodeling the RosettaNet Order-To-Cash use case as depicted in figure 2 immediately reveals two important problems, *partial termination* and creation of *role projections*. *Partial termination* becomes obvious when looking at the transitions that lead into final states `f3`, `f4` and `f5` of figure 2. As the source of these transitions are binary BCAs, only those roles participating in the respective BCA will be aware of the termination of the overall SeqMP choreography. However, the *Customer* role (in case of BCA `c2` *Arrange Shipping*) or the *Supplier* role (in case of BCA `c3` *Perform Shipping*) may still wait for some interaction to happen. This may not be a problem in case the individual BCAs of a SeqMP choreography are independent of each other, but the business semantics of RosettaNet's Order-To-Cash scenario contradicts independence of, for example, `c1` and `c4`. One possibility to attack this problem is adding additional BCAs implementing exception handling routines. However, modeling such multi-party choreographies may be hard because such exception handling BCAs may fail as well and suitable business documents for communicating exception handling semantics are not always available in business document libraries. Further,

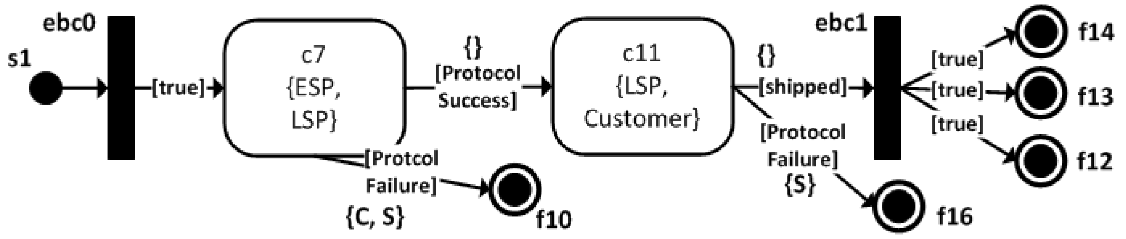

Figure 4. 1 out of 2 Possible Projections for the LSP Role (conflated visualization)

a business level problem like disagreeing on the conditions of shipping between *Supplier* and *LSP* may require business escalation routines between *Supplier* and *Customer* that are not intended to be implemented using business-document based choreographies. Note, that the *partial termination* problem implicitly is contained in the original RosettaNet definition of the use case as well. It just is not as obvious because only the *intended* flow of interactions is modeled.

Creation of *role projections* is an obvious problem when considering that some roles may not participate in every BCA of a SeqMP choreography. Hence, the possible sequences of BCA executions with participation of a particular role *r* must be derived from the overall choreography and the BCAs with participation of *r* must be suitably abstracted.

Both problems are not hard to solve if the use case is as simple as in figure 2. However, more complex SeqMP choreographies such as the artificial use case depicted in figure 3 may be more challenging. The use case depicts a multi-party order-to-cash choreography between a *Customer*, a *Seller*, a *Logistics Service Provider* (LSP), a *Financial Service Provider* (FSP) and an *Escort Service Provider* (ESP). Due to space limitations, the BPMN choreography notation has been conflated in an ad-hoc manner by only showing the participating roles and an id for each call choreography. The business semantics of the use case may be derived to some extent from the guards on the transitions, but, for the purpose of this paper, the control flow of the use case is decisive. One striking observation is that *partial termination* is not a problem associated to final states only, but actually of transitions that partition a SeqMP choreography in parts with or without possible participation of a particular role. For example, by firing the transition between *c6* and *c8* as depicted in figure 3, there is no possibility that the *FSP* role will become active in the particular SeqMP instance anymore whereas participation still would be possible in *c6*.

C. Seq-MP Algorithms

This section presents algorithms for dealing with the *partial termination* and the *role projection* problem identified in the last section. Note that the algorithms are defined for SeqMP that is defined for binary BCAs as building blocks. However, the algorithms themselves also would work for multi-party BCAs as building blocks. For validation, the algorithms have been implemented for an abstract model of SeqMP in Java.

Algorithm 1: Projection Computation

```
input     : A valid SeqMP smp to be analyzed
output    : A mapping of roles to their projections:
            smp.R → Set<State>
variables : Set<State> computed;
\\initially maps each role of smp to an empty set;
Map<Role,Set<State>> projs;
algorithm:
1  foreach State s of each path without loop in a depth first traversal of smp do
2      Set<Role> known = $\bigcup_{r \in \text{states on the current path}} \mathtt{RA}(r)$;
3      Set<Role> curr = RA(s);
4      curr = curr \ known;
5      if $curr \neq \emptyset \land s \notin computed$ then
6          foreach Role r in curr do
               // Add role projection for r starting at s
7              projs.get(r).add(doProjection(r, s, processBegin));
8          end
9          computed.add(s);
10     end
11     if processBegin then processBegin = false;
12 end
13 return projs;
```

The prototype implementation together with the test graphs for figure 2 and 3 are available[3].

1) Computing Escalation Sets: As really safe multi-party choreographies that specify exception handling logic in full (cf. above) are hard to design, we propose the computation of so-called *escalation sets* for tackling the *partial termination* problem. Definition 3.4 defines the concept of escalation sets formally and essentially says that an escalation set is the set of roles that, without proper notification, potentially is disabled from participating in a SeqMP instance anymore by firing a particular transition.

Definition 3.4 (Escalation Set):
An escalation set *es* is a subset of the roles of a valid SeqMP choreography *smp* that is attached to a transition t=(t#1,

[3]http://www.uni-bamberg.de/pi/seqMP-Algorithms

Algorithm 2: Role Projection at State s: Part 1

```
input     : State s and Role r of valid SeqMP smp;
boolean first telling whether s is the first BCA in smp
output    : The first state of the role projection $s_{out}$
variables : Map<String, State> proj;
Map<State,State> visitMap;

procedure: walk(Map<State,State> vMap, State curr) :
State mSelf = vMap.get(curr);
foreach Transition t=(t#1,t#2,t#3) in SUCC_curr do
    State mTarg = vMap.get(t#3);
    if mTarg $\neq$ null then
        if Not(mSelf is an EBCState $\wedge$ mSelf = mTarg) then
            if mSelf is an EBCState then tr(mSelf, mTarg,
            "true");
            else tr(mSelf, mTarg, t#2);
        end
    else
        if proj.containsKey(t#3.id()) then tr(mSelf,
        proj.get(t#3.id()), t#2);
        else
            Map<State,State> cpVMap =
            vMap.scopy();
            if t#3 is a FinalState $\vee$ r $\in$ RA(t#3) then
                State mNext = t#3.scopy();
                cpVMap.put(t#3, mNext);
                proj.put(t#3.id(), mNext);
                if mSelf is an EBCState then tr(mSelf,
                mNext, "true");
                else tr(mSelf, mNext, t#2);
            else
                if mSelf is an EBCState then
                    cpVMap.put(t#3, mSelf);
                else
                    EBCState ebc = new EBCState();
                    cpVMap.put(t#3, mSelf);
                    proj.put(ebc.id(), ebc);
                    tr(mSelf, ebc, t#2);
                end
            end
            walk(cpVMap, t#3);
        end
    end
end
end procedure // continue...
```

Algorithm 3: Role Projection at State s: Part 2

```
algorithm:
// ...continue
$s_{out}$ = new StartState("s1");
State currCopy = s.scopy() // copy s without transitions
proj.put("s1", $s_{out}$);
proj.put(s.id(), currCopy);
if first then
    proj.put("ebc0", new EBCState("ebc0"));
    // define transition between #1 and #2 with guard #3
    tr ($s_{out}$, proj.get("ebc0"), "true");
    tr (proj.get("ebc0"), proj.get(s.id()), "true");
else
    tr (proj.get("s1"), proj.get(s.id()), "true");
end
visitMap.put(s, currCopy);
walk(visitMap, s);
return $s_{out}$;
```

t#2,#3) $\in$ smp.T such that: $\forall$ r $\in$ es: ($\exists$ path(s_1, t#1), s_1 $\in$ smp.SBCA, length(path) > 1: r $\in$ RA(s_1)) $\wedge$ ($\exists$ path(t#1, s_2), s_2 $\in$ smp.SBCA, length(path) > 1: r $\in$ RA(s_2)) $\wedge$ ($\forall$ path(t#3, s_3), s_3 $\in$ smp.SBCA, length(path) > 1: r $\notin$ RA(s_3)) □

In the use cases of figures 2 and 3 the escalation sets are attached to each transition by enumerating the according roles in curly braces. For figure 3, only the initial letters of each role are given. For example, the escalation set {F} of the transition between `c6` and `c9` says for the Financial Service Provider (FSP) role that it may have participated in an instance of the depicted SeqMP graph (if `c2` or `c5` were on the path to `c6`), that the FSP role still could be triggered (via the path (`c6`,`c10`,`c12`,`c13`,`c3`,`c5`)) and that it cannot be triggered any more (because from `c9` no BCA with participation of FSP can be reached). This may or may not be an issue depending on whether or not the FSP considers each BCA `c5` as a completely independent business case. The LSP role, in turn, is not included in the escalation set between `c6` and `c9` because it knows from the `[shipped]` outcome and the global SeqMP model that there is no way of being involved in the current SeqMP instance any more.

We have a fully working algorithm for computing escalation sets as of definition 3.4 in the prototype implementation. Due to space limitations, we only sketch the algorithm:

1) For a particular state s of a SeqMP choreography, calculate the roles that may have participated in any BCA on any path leading to s and store it as *es*.
2) Remove the roles participating in s from *es* because they are informed about the outcome of s.
3) Remove all roles that do not participate in any BCA reachable in forward direction as they have been informed about termination earlier or have been considered in an escalation set earlier. If *es* is empty then store it for all outgoing transitions of s. Otherwise proceed.
4) For each outgoing transition t of s
    a) Calculate the roles that participate in any BCA reachable in forward direction via t and store them as *tfollow*.
    b) Store *es* $\setminus$ *tfollow* as escalation set of t.

Using escalation sets, the participants of a SeqMP choreography can find out at design time which transitions potentially suffer from the *partial termination* problem at runtime and may agree on appropriate actions for dealing with this issue, e.g., informing their integration partners via mail or phone. Clearly, when firing a problematic transition at runtime the defined escalation set has to be compared to the roles that really have participated in that particular instance.

2) Computing Role Projections: Our approach for computing projections of a SeqMP *smp* for a role *r* is based on abstracting interactions without participation of *r* using so-called event-based choices as depicted in figure 4. Figure 4 shows the smaller one out of two possible projections for the LSP role of the use case depicted in figure 3. It starts out by transitioning from the start state to an event-based choice *ebc0* which means that the LSP has to wait until one of the outgoing transitions of *ebc0* is fired (which is why it is denoted as event-based choice). As there only is one single outgoing transition that leads into BCA c7, the LSP role has to wait until the ESP role triggers c7. As no BCA without participation of the LSP is directly reachable from c7, all directly following states are simply copied. However, in c11, c9 is directly following which is a BCA without LSP participation. To resolve this, another event-based choice *ebc1* is introduced. All final states and BCAs with LSP-participation that are reachable via the transition to c9 are then connected to *ebc1*. If there was a path(c9,c7) then there would be a transition from ebc1 to c7 (you can try that using the prototype implementation).

We choose deliberately to allow for multiple projections of a particular SeqMP *smp* for a particular role *r*. The reason for that is that different branches of a multi-party choreography potentially may comprise completely unrelated BCAs of r. For example, if the use case of figure 3 began with BCAs c1, c2 and c4 then only the BCAs of figure 4 would be possible for the LSP role. Assume further there was no transition between c8 and c7. Then, depending on whether transition c2-c4 or c2-c3 is taken, completely disjoint sets of BCAs with LSP-participation are possible. In that situation, we believe that integration participants should not be forced to view unrelated BCAs as part of a single collaboration. We use algorithm 1 for identifying potentially disjoint projections for a particular role r. Once the first BCA with participation of r is found that could be the starting point of such a projection, the computation of a role-specific projection at a particular state is started (algorithm objects 2 and 3). For the case of the LSP role of the use case of figure 3, this approach leads to two individual projections that overlap. In this case, the individual projections still may be considered to be two different use cases or may be merged which depends on the system setting of the integration participant.

Also, in figure 4, the various final states that are following *ebc1* could be conflated and as there are only final states following, the transition to *ebc1* could even be replaced by one single final state. Note that it is essentially irrelevant for the LSP whether the overall collaboration terminates in f12, f13 or f14 because there are no more BCAs within the LSP's projection after completion of c11. This calculation can easily be done after having performed the algorithms as described here.

IV. Related Work

The problems identified in section III-B are quite different from several problems identified in different choreography research. In publications such as [14], [15], [16], [17], problems like *enforceability* or *realizability* are researched. The according approaches have in common that the atomic building blocks of choreographies are single message exchanges and it is then researched whether or not the message sequences in the choreographies can be *enforced* by the local role-specific projections of interaction partners, and whether or not the sequence of message exchanges is the same for synchronous or asynchronous communication. In the work at hand, these problems are not relevant. By using BCAs instead of single message exchanges, we can be sure that the state of integration partners is aligned at the end of each BCA (cf. [4], [12]). BCAs are performed using according protocol machines that ensure alignment and therefore are not comparable to single messages in some communication buffer that may cause diverging states or deadlocks. Moreover, due to the *Subsequent role participation* condition, valid SeqMPs do not suffer from a local enforceability problem. To the best of our knowledge, *partial termination* as defined above has not been identified as multi-party choreography problem yet. However, deriving role projections of multi-party interactions has been a research topic for a long time. For example, van der Aalst and Weske [18] describe an approach for dissecting Petri Nets that contain role specific behavior. However, communication between participants is modeled as *send*- and *receive*-transitions that already are associated with roles. This corresponds to dissecting interconnection style choreographies and is representative for more recent research on interconnection style choreographies. However, we are considering the problem of deriving interaction style choreographies. At first sight, the proposal for deriving role projections of *Let's Dance* models as described in [14] is comparable to that problem. However, Let's Dance applies a block-structured approach for modeling loops which is different from the class of SeqMP models with arbitrary loops. In [19], so-called Interaction Petri nets that enable multi-party interactions by modeling binary message exchanges as transitions of Petri nets, are analyzed. The model of [19] therefore can be considered to be comparable for the role projection problem of the work at hand and an algorithm for calculating role projections is presented as well. However, the problem there is defined for Petri Nets and therefore the algorithm is not directly amenable to SeqMP models. Further, the algorithm in [19] introduces duplicate transitions for flattening parallelism. However, transitions correspond to message exchanges in [19] and to BCAs for SeqMP.

The business semantics of duplicate message exchanges (or duplicate BCAs for SeqMP) is not clear, though.

Apart from that, interesting choreography work has been presented in [5] where so-called local choreographies are used to model the sequence of interactions of one integration partner in multiple global choreographies. The perspective is different from the work at hand by focusing on a partner that participates in more than one choreography where the participants of the respective choreographies may only know the focal integration partner. A typical scenario for that type of integration is a manufacturer that employs a sub-contractor producing parts of a product without the customer knowing the sub-contractor. While there may be valid reasons for not revealing the interactions with business partners to different business partners, the type of integration in [5] leaves out the opportunity to perform analyses like escalation set computation that rely on a global view on choreographies.

V. Conclusion and Future Work

This paper introduces an approach for creating multi-party B2Bi choreographies from binary B2Bi choreographies. The identification of the *partial termination* and the *role projection* problem show the potential of SeqMP for optimizing multi-party B2Bi scenarios. Solutions to these problems which have been validated using a prototype implementation have been provided as well.

Future work comprises defining an integration architecture for supporting the communication of *partial termination* events among the participants of a SeqMP choreography and configuration options for defining when to consider the BCAs of a SeqMP choreography as independent. Moreover, the derivation and monitoring of legally binding obligations to communicate *partial termination* events is still an open issue.

References

[1] J.-H. Kim and C. Huemer, "From an ebXML BPSS choreography to a BPEL-based implementation," *SIGecom Exch.*, vol. 5, no. 2, pp. 1–11, 2004.

[2] A. Schönberger and G. Wirtz, "Using Webservice Choreography and Orchestration Perspectives to Model and Evaluate B2B Interactions," in *2006 Int. Conf. on Software Engineering Research and Practice (SERP), Las Vegas, USA*, pp. 329–335.

[3] S. Wieczorek, A. Roth, A. Stefanescu, V. Kozyura, A. Charfi, F. M. Kraft, and I. Schieferdecker, "Viewpoints for modeling choreographies in service-oriented architectures," in *WICSA/ECSA 2009, Cambridge. Los Alamitos/Calif.*

[4] A. Schönberger and G. Wirtz, "Towards executing ebBP-Reg B2Bi choreographies," in *Proceedings of the 12th IEEE Conference on Commerce and Enterprise Computing (CEC'10), Shanghai, China*. IEEE, November 10-12 2010.

[5] B. Hofreiter and C. Huemer, "A model-driven top-down approach to inter-organizational systems: From global choreography models to executable BPEL," in *Joint Conference on E-Commerce Technology (CEC'08) and Enterprise Computing, E-Commerce, and E-Services (EEE'08)*. Crystal City, Washington D.C., USA: IEEE, 7 2008.

[6] OASIS, *ebXML Business Process Specification Schema Technical Specification*, 2nd ed., OASIS, December 2006.

[7] G. Decker, O. Kopp, and A. Barros, "An introduction to service choreographies," *Information Technology*, vol. 50, no. 2, pp. 122–127, 2008.

[8] UN/CEFACT, *UN/CEFACT's Modeling Methodology (UMM): UMM Meta Model - Foundation Module Version 1.0*, 1st ed., UN/CEFACT, 10 2006.

[9] M. Zapletal, T. Motal, and H. Werthner, "The business choreography language (BCL) - a domain-specific language for global choreographies," in *The 5th 2009 World Congress on Services (SERVICES 2009 PART II), Bangalore, India*. IEEE, September 2009.

[10] OMG, *Business Process Model and Notation, v2.0 Beta 2, DRAFT*, OMG, June 2010. [Online]. Available: http://www.omg.org/cgi-bin/doc?dtc/10-06-04

[11] A. Schönberger, C. Pflügler, and G. Wirtz, "Translating shared state based ebXML BPSS models to WS-BPEL," *(to appear in) International Journal of Business Intelligence and Data Mining - Special Issue: iiWAS 2009*, vol. 5, no. 4, 2010.

[12] C. Pflügler, A. Schönberger, and G. Wirtz, "Introducing partner shared states into ebBP to WS-BPEL translations," in *Proc. iiWAS 2009, 11th Int. Conf. on Information Integration and Web-based Applications & Services, 14.-16. December 2009, Kuala Lumpur, Malaysia*. ACM, December 2009.

[13] H. A. Reijers and W. M. van der Aalst, "The effectiveness of workflow management systems: Predictions and lessons learned," *International Journal of Information Management*, vol. 25, no. 5, pp. 458 – 472, 2005.

[14] J. Zaha, M. Dumas, A. ter Hofstede, A. Barros, and G. Decker, "Bridging global and local models of service-oriented systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Trans.*, vol. 38, no. 3, may 2008.

[15] G. Decker, A. Barros, F. M. Kraft, and N. Lohmann, "Non-desynchronizable service choreographies," in *ICSOC 2008: Proc. of the 6th Int. Conf. on Service-Oriented Computing*. Springer, pp. 331–346.

[16] T. Bultan, J. Su, and X. Fu, "Analyzing conversations of web services," *IEEE Internet Comp.*, vol. 10, no. 1, pp. 18–25, 2006.

[17] V. Kozyura, A. Roth, and W. Wei, "Local enforceability and inconsumable messages in choreography models," in *Proceedings of 4th South-East European Workshop on Formal Methods (SEEFM'09),Thessaloniki, Greece*, 2009.

[18] W. M. P. van der Aalst and M. Weske, "The p2p approach to interorganizational workflows," in *Proc. of the 13th Int. Conf. on Advanced Information Systems Engineering 2001*, London, UK, pp. 140–156.

[19] G. Decker and M. Weske, "Local enforceability in interaction petri nets," in *Proceedings of the 5th International Conference on Business Process Management (BPM 2007), Brisbane, Australia, September 24-28*, 2007, pp. 305–319.