Choreography-based design of business collaborations

Choreography-Based Design of Business Collaborations

Remco Dijkman

Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
R.M.Dijkman@tue.nl

Abstract. We define choreography-based design as a design approach in which a business process for a business collaboration (or *choreography*) is established by the stakeholders in the collaboration. Subsequently, the stakeholders adapt their internal business processes to the established choreography, to allow successful collaboration. To assist in such a design approach, this paper presents techniques to: (i) construct the choreography in a gradual manner, based on step-wise refinement, allowing a designer to focus on the most important problems of the choreography first and then gradually introduce more detail; and (ii) check if at one level of detail the aspects from the previous level of detail are observed. We define these techniques at a sufficient level of detail, such that they can be implemented and supported by tools.

1 Introduction

This paper presents design techniques for choreography-based design. We define *choreography-based design* as a design approach in which a business process for a business collaboration is established by the stakeholders in the value chain. We call the established business process a *choreography*. A choreography specifies the messages that stakeholders in a value chain exchange and a contract that applies to these messages. This contract can, for example, specify the possible orderings of the messages, time constraints and encoding of the messages that allows automated exchange of the messages. Stakeholders that want to participate in the value chain must then conform to the choreography.

To assist in choreography-based design, this paper presents:

1. deliverables that can be produced during a choreography-design trajectory based on the ideas of step-wise refinement. These ideas allow a choreography designer to first focus on the important aspects of a business collaboration and gradually introduce more detail, until a level of detail is reached at which the collaboration can be implemented. The deliverables that we use are taken from the Let's Dance approach to choreography-based design [3];
2. formal relations between the deliverables, which allow us to check if, at one level of detail, the aspects from the previous level of detail are observed.

We focus our techniques on the control flow aspect of choreography-based design.

We define our formal relations, based on a formalization of the deliverables in terms of Petri nets. Subsequently, we analyse the relations between the deliverables

and present them formally as relations on the Petri nets. This leads to a specification of the techniques at a sufficient level of detail for them to be implemented in a tool.

The work presented in this paper is an application of the multi-viewpoint theory from [4] to the Let's Dance approach to choreography-based design [3]. We apply the theory by considering each of the Let's Dance deliverables a viewpoint in terms of the theory from [4].

Another approach to service-oriented design that defines relations between deliverables is given in [10]. This approach uses a different formalism to define the relations between the deliverables. Also, although it considers the design of a choreography at increasing levels of detail, it does not consider some of the deliverables that in our opinion help to structure the design process further. The P2P approach [1] presents formal relations, between deliverables in inter-organisational workflows, based on Petri nets. However, this approach does not consider some of the deliverables that in our opinion help to structure the design process further. Martens [7] also introduces a means, based on Petri nets, to specify services from different viewpoints. However, he performs different analysis.

The structure of this paper is as follows. Section 2 presents the deliverables that we propose in a choreography-based design approach. Section 3 explains how these deliverables can be represented formally, by means of Petri nets. Section 4 investigates the relations between the deliverables and defines them formally. Section 5 presents the conclusions and future work.

2 Deliverables in Choreography-Based Design

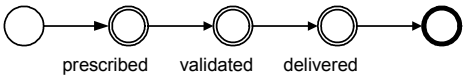
This section identifies the deliverables in a choreography-based design, based on the deliverables proposed by Let's Dance [3]. We distinguish: the choreography milestone model; the choreography scenario model; the choreography; and the provider behaviour model.

The choreography milestone model represents the behaviour of a business collaboration at a high level of abstraction. It does this by representing the milestones that can be reached in a collaboration and the relationships between these milestones. In this paper we represent these relationships by means of control flow relations.

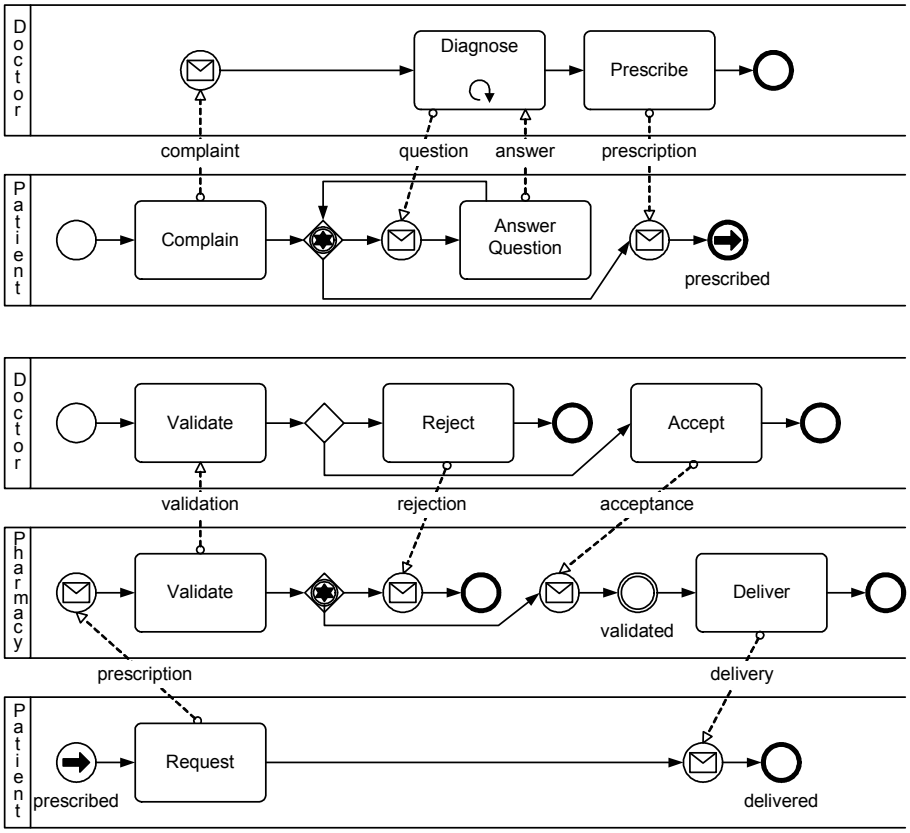
The choreography scenario model represents possible scenarios, in terms of conversations between partners, that lead from one milestone to another. We represent the conversations between partners by means of message exchange interactions and control flow relations between those interactions.

The choreography represents all interactions between stakeholders in a business collaboration, in terms of message exchange. It constitutes all interactions decided upon in the choreography scenario model and it should observe the relations between those interactions in the choreography scenario model.

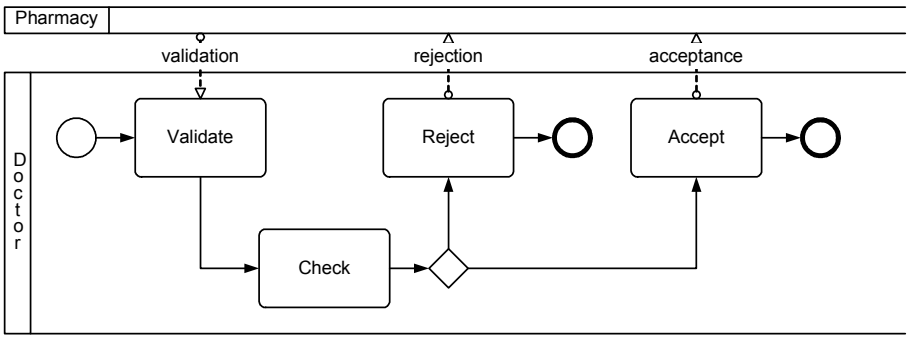
A provider behaviour model represents all interactions in which a single stakeholder engages in the context of a business collaboration. It only represents the externally observable behaviour, leaving the internal realization thereof to the stakeholder. The provider behaviour model must observe the interactions and their relations of the part of the choreography that the stakeholder fulfils.



i. Example of a milestone model



ii. Examples of scenario models



iii. Example of a provider behaviour model

Figure 1. Examples of deliverables in choreography-based design

Figure 1 shows examples of a milestone model, two choreography scenario models and a provider behaviour model. The models are represented in the Business Process Modelling Notation (BPMN) [9]. Figure 1.i shows the milestones that are reached when ordering a prescription drug. First the doctor has to prescribe a recipe, this recipe has to be validated by the pharmacy, then the patient receives the drug. We developed the choreography in more detail, by showing two scenarios that lead from one milestone to another. Figure 1.ii shows these scenarios. Figure 1.iii shows the provider behaviour of a doctor that participates in the choreography.

3 Formal Description of Deliverables

Our formal model is based on labelled marked Petri nets. We first present our formal model and then we explain how it can be used to represent the deliverables.

3.1 Formal Model

A labelled Petri net is a tuple (P, T, F, l), such that:

- P is the set of places, which is partitioned into P^I that represents the set of places that are internal to some business partner and P^C that represents the (connector) places on which (tokens representing) messages exchanged between partners are stored.
- T is the set of transitions, which is partitioned into T^I that represents the set of transition that are internal to some business partner and T^S and T^R that represent the set of transitions that correspond to sending and receiving messages, respectively. T^M is a subset of T^I that represents the set of milestones.
- $F \subseteq (T^S \times P^C) \cup (P^C \times T^R) \cup (T \times P^I) \cup (P^I \times T)$ is the flow relation that connects internal places to transitions and connector places to send and receive transitions. Every connector place must be connected to exactly one send and one receive transition, because connector places are only used to connect send transitions to receive transitions.
- $l : (T^S \cup T^R \rightarrow Msg) \cup (T^I \rightarrow L)$ is the labelling function that labels send and receive transitions with the message sent or received and an internal transition with the action or milestone that it represents. A special 'silent' label τ represents that no observable action or send or receive action occurs and that no milestone is reached. Send and receive transitions that are connected to the same connector place by the flow relation must have the same label. The label represents the message exchanged. In this paper we write a '!' or a '?' behind the label of a send or receive transition, respectively. We write the label of a milestone between square brackets: '[', ']'. These are notational conventions to distinguish the different kinds of transitions; the '!', '?', '[' and ']' are *not* part of the label itself.

$M : P \rightarrow \mathbb{N}$ represents the marking of a Petri net.

In this paper we use a particular form of Petri nets called *workflow nets* [2]. Workflow nets satisfy the criteria that:

1. an 'initial' place $i \in P$ and a 'final' place $f \in P$ exist, such that i does not have incoming flows and f does not have outgoing flows;
2. all places and transitions are on a path from i to f; i.e.: if we add a transition t that connects f to i by the flow relation (i.e. $\{(f, t), (t, i)\} \subseteq F$), then, for every two places or transitions x and y, there is an (indirect) flow from x to y and vice versa.

In a workflow net $M^i = \{(i,1)\}$, the *initial marking* that has only one token on i, marks the beginning of a process and $M^f = \{(f,1)\}$, the *final marking* that has only one token on f, marks the completion of a process.

We denote possible firing of transition t in a Petri net N with marking M as: (N, M) $[t>$. We denote firing of transition t in a Petri net N with marking M, causing it to change into a marking M' as: (N, M) $[t>$ (N, M'). We denote a sequence of transitions as t^*; if all transitions labelled τ, we write τ^*. We denote the successive firing of a sequence of transitions t^*, causing a Petri net N with marking M to change into a marking M' as: (N, M) $[t^*>$ (N, M'). We say that a marking M' is *reachable* from a marking M in a net N, if there exists a sequence of transitions t^*, for which (N, M) $[t^*>$ (N, M'). The set of reachable markings from a marking M in a net N is the set that contains all markings M' for which there exists a t^* such that (N, M) $[t^*>$ (N, M'). We denote the set of reachable markings as: $(N, M)[>$. For a precise definition of firing and reachability, we refer to the Petri net theory [8].

In this paper we refer to the elements of a net by subscripting the element with the name of the net. For example, we refer to the places P of a net N as P_N.

3.2 Formal Model Applied to Represent Deliverables

We can represent the deliverables in a choreography-based design as follows.

Milestone model. We formally represent a milestone model by a Petri net that only contains internal places, milestone transitions and transitions marked 'silent', because a milestone model consists of milestones and their relations. Figure 2.i shows an example of a Petri net that represents a milestone model. The figure is a possible formalization of Figure 1.i. We are in the process of defining a mapping from BPMN to Petri nets, but this mapping is not complete yet. Therefore, the formalization is only for illustration purposes.

Scenario and Choreography models. We formally represent a scenario or a choreography model by a Petri net that can contain all the constructs explained in the previous section. Figure 2.ii shows an example of a Petri net that represents a scenario model. The figure is a possible formalization of Figure 1.ii. We subscripted the interactions in which 'prescriptions' are exchanged, because these interactions are different and (since below we rely on different interactions having different labels) therefore they should have different labels.

Provider Behaviour Model. We formally represent a provider behaviour model by a Petri net that must not contain connector places, because a provider behaviour represents the behaviour of a single provider and not the interaction between providers. A provider behaviour can represent a provider's participation in an

interaction, so send and receive transitions are allowed in the formalization. Figure 2.iii shows an example of a Petri net that represents a scenario model. The figure is a possible formalization of Figure 1.iii.

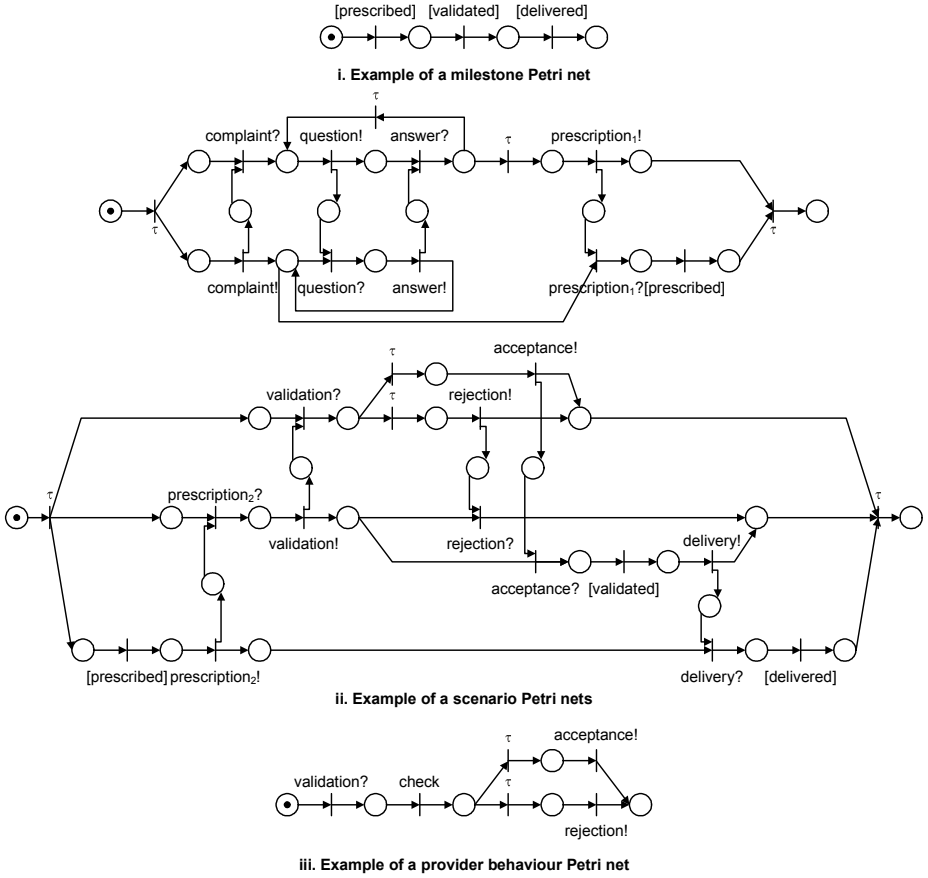


Figure 2. Examples of formalized deliverables

4 Formal Relations between Deliverables

Figure 3 illustrates the relations between the deliverables in choreography-based design. Below we explain these relations and how we can check formally, if they are observed by the deliverables.

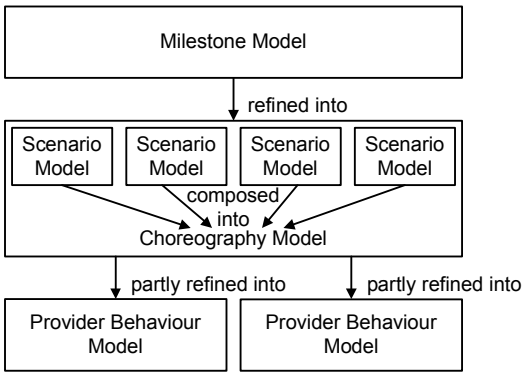


Figure 3. Relations between deliverables

4.1 Relation between Scenario Models and Choreography Model

A choreography model is a composition of the scenario model [3]. While each scenario model only represents the actions that have to be performed to get from one milestone to another, the choreography model represents all actions that a group of service providers performs jointly in the context of their collaboration. Moreover, the choreography model is not designed separately, but it is completely defined by (the composition of) the scenario models. Therefore, we can derive a choreography model from scenario models by taking the union of the scenario models, keeping only a single copy of a milestone, action or interaction that appears in multiple scenario models. This copy has all relations from the milestones, actions or interactions from which it was derived. To ensure that the resulting net is also a workflow net, we have to add a new initial place and a new final place, such that the initial place has a target transition that puts tokens on the original initial places and that the final place has an source transition that only fires after there is a token on all original final places.

Figure 4 shows an example in which we composed the scenarios from Figure 2.ii. For brevity, the example figure only shows the relevant part of the composed Petri net. These scenarios share the milestone ‘prescribed’. Therefore, the composition only contains one milestone transition labelled ‘prescribed’. This milestone transition has the incoming and outgoing flows of both scenarios from which it is derived. The newly created initial and final places with their transitions and flows are shows in grey.

This operation assumes that each milestone from a scenario model has a different label than the other milestones from that scenario model. It assumes the same for internal actions (other than the ones labelled ‘silent’), send actions and receive actions.

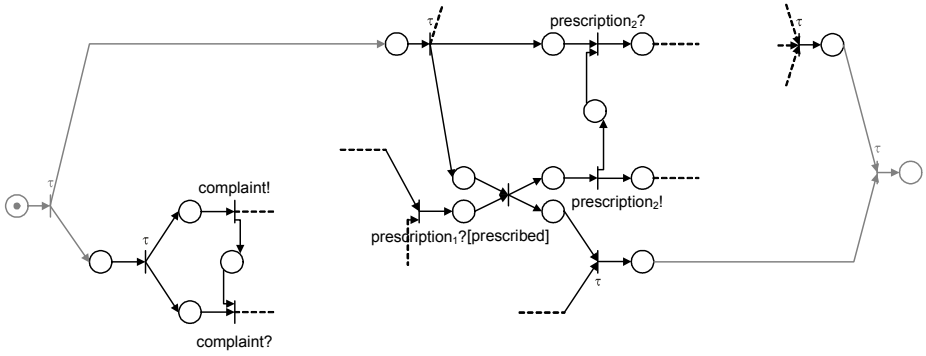


Figure 4. Composition of scenarios from Figure 2.ii

We can compose two scenario Workflow nets, N and M, into a Workflow net, (P, T, F, l, i, f), using the following algorithm:

$T_{New} := \emptyset$
$Map_N := \emptyset$
$Map_M := \emptyset$
For each pair $t_N \in T_N$, $t_M \in T_M$, for which
  $l_N(t_N) = l_M(t_M)$ and $l_N(t_N) \neq \tau$ and
  $(t_N \in T_N^S \wedge t_M \in T_M^S) \vee (t_N \in T_N^R \wedge t_M \in T_M^R) \vee$
  $(t_N \in T_N^M \wedge t_M \in T_M^M) \vee (t_N \in (T_N^I - T_N^M) \wedge t_M \in (T_M^I - T_M^M))$
    $T_{New} := T_{New} \cup (\text{new } t)$
    $Map_N := Map_N \cup \{(t_N, t)\}$
    $Map_M := Map_M \cup \{(t_M, t)\}$
$P := P_N \cup P_M$
$T := (T_N - dom(Map_N)) \cup (T_M - dom(Map_M)) \cup T_{New}$
$F := \{(e_1, e_2) \mid (e_1, e_2) \in (F_N \cup F_M) \wedge e_1, e_2 \notin dom(Map_N) \wedge e_1, e_2 \notin dom(Map_M)\} \cup$
  $\{(e_1', e_2) \mid (e_1, e_2) \in F_N \wedge e_1 \in dom(Map_N), e_1' = Map_N(e_1)\} \cup$
  $\{(e_1', e_2) \mid (e_1, e_2) \in F_M \wedge e_1 \in dom(Map_M), e_1' = Map_M(e_1)\} \cup$
  $\{(e_1, e_2') \mid (e_1, e_2) \in F_N \wedge e_2 \in dom(Map_N), e_2' = Map_N(e_2)\} \cup$
  $\{(e_1, e_2') \mid (e_1, e_2) \in F_M \wedge e_2 \in dom(Map_M), e_2' = Map_M(e_2)\}$
$l := \{(t, lab) \mid (t, lab) \in (l_N \cup l_M) \wedge t \notin dom(Map_N) \wedge t \notin dom(Map_M)\} \cup$
  $\{(t, lab) \mid t \in T_{New}, lab = l_N(Map_N^{-1}(t))\}$
$i := \text{new } pi$
$f := \text{new } pf$
$P := P \cup \{pi, pf\}$
$T := T \cup \{\text{new } ti, \text{new } tf\}$
$F := F \cup \{(pi, ti), (ti, i_M), (ti, i_N), (tf, pf), (f_M, tf), (f_N, tf)\}$
$l := l \cup \{(ti, \tau), (tf, \tau)\}$
return (P, T, F, l, i, f)

4.2 Relation between Milestone Model and Choreography Model

A choreography model refines a milestone model. It does this by specifying how one gets from one milestone to another by performing certain actions or interactions, while the milestone model only specifies the milestones that can be reached and the relations between the milestones. The choreography model cannot introduce new milestones, but only actions and interactions. Also, it should also observe the original relations between the milestones, as they are specified by the milestone model. It cannot introduce, nor remove, means to get from one milestone to another.

Hence, the relation between a milestone model and a choreography model is that: after abstracting from the actions and interactions that specify how to get from one milestone to another, the choreography model must be equivalent to the original milestone model.

To allow us to check consistency with respect to this relation, we 'abstract' from actions and interactions in a choreography model, by labelling them 'silent'. Subsequently, we check equivalence between the Petri nets, using the notion of branching bi-simulation [5]. Informally, two Petri nets, N and M, are branching bisimilar (denoted: $N \sim_{bbisim} M$) if at any time (i.e.: in any reachable marking, starting from the initial markings) N can take the same transition as M and vice versa, possibly preceded and/or succeeded by silent transitions. A (low) polynomial time algorithm for checking branching bi-simulation is developed by Groote and Vaandrager [6]. This technique is well-known for checking refinement between Petri nets in which one Petri net refines the other by only inserting transitions [1,4]. Figure 5 shows a Petri net that is an abstraction from the Petri net from Figure 4, by labelling all transitions that are not milestone transitions 'silent'. The figure shows only a part of the full Petri net. To complete the consistency check, we must check bi-similarity between the full Petri net and the milestone Petri net from Figure 2.i.

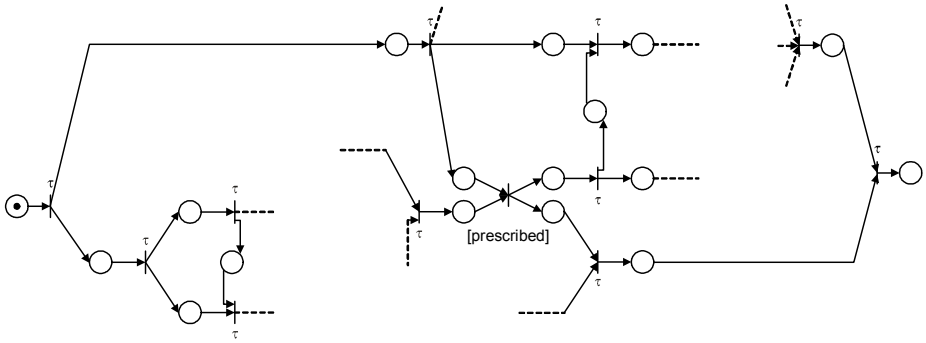


Figure 5. Abstraction from non-Milestones in Figure 4

We can check consistency between a milestone Workflow net M and a choreography Workflow net C, using the following algorithm:

$C' := (P_C, T_C, F_C, l_C', i_C, f_C)$, where

$l_C' := \{(t, l') \mid (t, l) \in l_C \text{ and } l' := l \text{ if } t \in T_C^M, l' := \tau \text{ otherwise}\}$

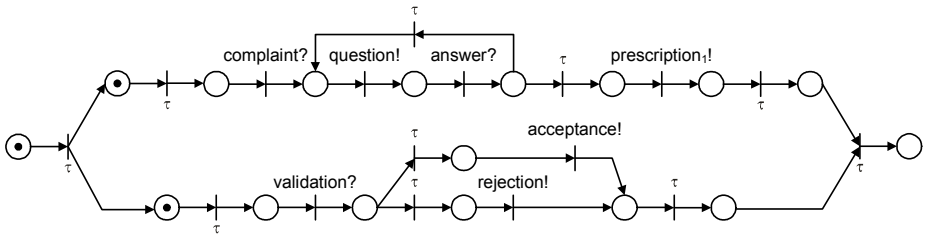
return $C' \sim_{bbisim} M$

4.3 Relation between Provider Behaviour Model and Choreography Model

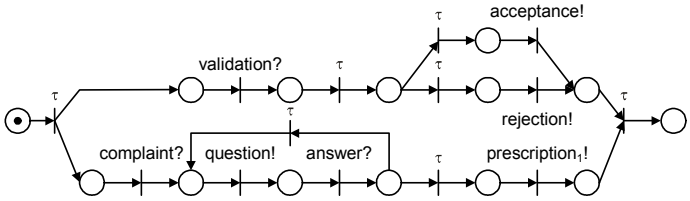
A provider behaviour model refines a part of a choreography model. It does this by specifying not only interactions between parties in the collaboration, but also internal actions of one of the parties in the collaboration, focussing on the part of the choreography that that party will perform.

Therefore, to check consistency between a provider behaviour model and a choreography model, we: (i) extract the part of the choreography that the provider performs and abstract from milestones in that part by labelling them 'silent'; (ii) abstract from internal actions of the provider behaviour by labelling them 'silent'; and (iii) check equivalence (branching by bi-similarity) between the results of (i) and (ii).

As an example Figure 6.i is an extraction from the choreography Petri net from Figure 4. It extracts the part of the choreography that is performed by the doctor and abstracts from milestones by labelling them 'silent'. Figure 6.ii shows an abstraction from the provider behaviour from Figure 2.iii. It abstracts from all internal actions of the 'doctor' (the only internal action is 'check'). To complete the consistency check, we must check bi-similarity between Figure 6.i and Figure 6.ii.



i. Choreography that only contains actions and interactions performed by the doctor



ii. Provider behaviour with internal actions labelled 'silent'

Figure 6. Consistency between provider behaviour and choreography

We can check consistency between a choreography Workflow net C and a provider behaviour Workflow net P, considering that, in the choreography, the provider performs the actions and interactions with labels in ls (This information can be obtained from the role assignment in the model from which the Petri nets are derived. For example Figure 2.ii shows exactly which actions and interactions are performed by the 'doctor'.), using the following algorithm:

$C' := (P_C', T_C', F_C', l_C', i_C, f_C)$, where

$controlflows := \{(s, t) \mid (s, t) \in F_C \text{ and } s, t \notin P_C{}^C\}$

$transclos := controlflows$

repeat
temp := *transclos*
transclos := *transclos* $\cup$ $\{(s_1, t_2) \mid (s_1, t_1), (s_2, t_2) \in transclos$ and $t_1 = s_2\}$
until *transclos* = *temp*
P_C' := $\{p \mid p \in P_C$ and $\exists ts \in Ts$ for which $(i, p) \in transclos \wedge (p, ts) \in transclos$
or $\exists ts \in Ts$ for which $(ts, p) \in transclos \wedge (p, f) \in transclos\}$
T_C' := $\{t \mid t \in T_C$ and $\exists ts \in Ts$ for which $(i, t) \in transclos \wedge (t, ts) \in transclos$
or $\exists ts \in Ts$ for which $(ts, t) \in transclos \wedge (t, f) \in transclos\}$
F_C' := $F_C \cap ((P_C' \times T_C') \cup (T_C' \times P_C'))$
l_C' := $\{(t, l') \mid (t, l) \in l_C$ and $t \in T_C'$ and l' := τ if $t \in T_C^M$, l' := l otherwise$\}$
P' := $(P_P, T_P, F_P, l_P', i_P, f_P)$, where
l_P' := $\{(t, l') \mid (t, l) \in l_P$ and l' := τ if $t \in T_P^I$, l' := l otherwise$\}$
return $P' \sim_{bbisim} C'$

5 Conclusions and Future Work

This paper presents an approach to choreography-based design that introduces deliverables in a choreography-based design trajectory, shows how these deliverables can be formalised with Petri nets and, based on this formalization, how the consistency of the deliverables can be checked. The paper presents algorithms to perform these checks.

As a next step we aim to implement the algorithms in a tool and apply them to choreography-based designs. This serves to evaluate the algorithms and to make them more practically applicable. We also aim to further integrate the algorithms with the Let's Dance approach to choreography-based design, by developing a formalization of the language used in Let's Dance in terms of Petri nets. This further serves to make the algorithms practically applicable.

References

1. van der Aalst, W., Weske, M.: The P2P approach to Interorganizational Workflows. In: Proc. of the Int. Conf. on Advanced Information Systems Engineering. Lecture Notes in Computer Science, Vol. 2068. Springer-Verlag, 140-156, 2001.
2. van der Aalst, W.: Verification of Workflow Nets. In: Application and Theory of Petri Nets. Lecture Notes in Computer Science, Vol. 1248. Springer-Verlag, 407-426, 1997.
3. Barros, A., Decker, G., Dumas, M. Multi-staged and Multi-viewpoint Service Choreography Modelling. Technical Report 4668, Queensland University of Technology, Brisbane, Australia, 2006.
4. Dijkman, R., Dumas, M.: Service-Oriented Design: A Multi-Viewpoint Approach. Int. J. of Cooperative Information Systems 13, 337-368, 2004.
5. Glabbeek, R.J. van, & Weijland, W.P. Branching Time and Abstraction in Bisimulation Semantics. Journal of the ACM, 43(3), 555-600, 1996.
6. Groote, J.F., Vaandrager, F. An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence. In Paterson, M.S. (ed.): Automata, Languages and Programming. Lecture Notes in COmputer Science 443, pp. 626-638, 1990.

7. Martens, A. Analyzing Web Service Based Business Processes. In: Proc. of the Intl. Conf. on Fundamental Approaches to Software Engineering. Lecture Notes in Computer Schience, Vol. 3442. Springer-Verlag, 2005.
8. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proc. of the IEEE 77, 541-580, 1989.
9. OMG. Business Process Modeling Notation (BPMN) 1.0, OMG Final Adopted Specification dtc/06-02-01, 2006.
10. Quartel, D.A.C., Dijkman, R.M., and van Sinderen, M.J. Methodological Support for Service-oriented Design with ISDL. In: Proceedings of the 2nd ACM International Conference on Service Oriented Computing (ICSOC), pp. 1-10, 2004.