Synthesizing an Automata-based Representation of BPMN2 Choreography Diagrams*

Marco Autili, Davide Di Ruscio, Amleto Di Salle, and Paola Inverardi

Università degli Studi di L'Aquila, Italy
{marco.autili,davide.diruscio,amleto.disalle,paola.inverardi}@univaq.it

Abstract. Choreographies are an emergent Service Engineering approach to compose together and coordinate distributed services. They represent a global specification of the interactions between the participant services. BPMN2 provides a dedicated notation, called Choreography Diagrams, to define choreographies. This paper presents a model transformation to automatically transform a BPMN2 choreography specification into an automata-based representation called Choreography LTS (CLTS). The latter is a LTS suitably extended to, on one side model the complex interactions that can be specified by choreography diagrams, on the other provide modelers with a means to precisely extract the not-easy-to-grasp coordination logic "hidden" into BPMN2 Choreography Diagrams. Dedicated Eclipse plugins, within the CHOReO*Synt* tool, have been developed to support the presented transformation.

1 Introduction

Choreographies are an emergent Service Engineering approach to compose together and coordinate distributed services. They describe the interactions between the participant services by specifying the way business participants coordinate their interactions from a global perspective. The OMG BPMN2 [18] Choreography Diagrams are the standard de facto for specifying service choreographies by providing powerful constructs to specify complex interactions where message exchanges between participants go far beyond simple request-response interactions that follow a sequential flow. Choreography diagrams permit to specify inclusive and exclusive conditional branches, parallel branches to be joined at later execution points, looping tasks, and so on.

BPMN2 specifications can be very complex and the standard specification introduces constraints that a choreography designer shall obey to achieve well-formed choreography specifications. Unfortunately, the standard only provides a textual description for these constraints, hence making their correct understanding difficult. In the literature many approaches have been proposed to deal with the problems of choreography realizability, conformance, and enforcement, e.g., [21,8,4,23,9,2]). These approaches are based on different interpretations of the choreography interaction semantics in terms of both the subset of considered choreography constructs and the used formal notation. Moreover, the adopted notations, although powerful and well known in the formal community,

* This work is partially supported by the EU FP7/2007-2013 under grant agreement number 257178 (project CHOReOS - www.choreos.eu).

do not completely satisfy requirements related to usability and pragmatism that BPMN2 choreography modelers usually require.

In this paper we overview the model transformation we have adopted within the CHOReOS EU project (`www.choreos.eu`) to generate from a BPMN2 choreography diagram an automata-based specification of the coordination logic "implied" by the choreography. The transformation constitutes the core of the overall methodology we apply in CHOReOS to solve the problem of automatic choreography enforcement. The core objective of CHOReOS is to leverage model-based methodologies [5] and relevant SOA standards, while making *choreography development* a systematic process to the reuse and the assembling of services discovered within the Internet. Our approach has two main advances: (i) most of the complex constructs of BPMN2 choreography diagrams, e.g., inclusive and parallel gateways, are handled; (ii) an extension of LTSs, called *Choreography LTS* (CLTS), is provided to enable explicit descriptions of the coordination logic that must be applied to enforce the choreography, by adopting a notation that is closer to the BPMN2 choreography one; (iii) CLTS makes explicit coordination-related information that in BPMN2 is implicit. This allows to statically infer the information needed for enabling distributed coordination that, otherwise, should be calculated at run time for each choreography instance and for each execution of it. For instance, the CLTS model specifies the source and target state from which a task is initiated and terminated, the corresponding transition and enabling condition.

The paper is structured as follows. Section 2 introduces the considered BPMN2 choreography constructs. Section 3 briefly outlines the model transformation we have developed and how the introduced BPMN2 constructs are mapped to CLTS constructs. Related works are discussed in Section 4 and conclusions and future directions are given in Section 5.

2 BPMN2 choreography diagram constructs

In the following we leverage the in-depth study of the "meanders" of the BPMN2 standard specification document, and introduce the considered BPMN2 choreography diagram constructs by concisely describing their crucial characteristics. In Figures 1 and 2, besides the BPMN2 constructs on the left side, we report the corresponding CLTS translation that will be then discussed in Section 3.

The selection of the considered BPMN2 constructs has been performed by analysing the intrinsic aspects related to the choreography enforcement problem and by fulfilling the requirements of all the CHOReOS use cases.
With reference to Figure 1 **(a)..(d)**, a choreography `Task` is an atomic activity that represents an interaction by means of one or two (request and optionally response) message exchanges between two participants. Graphically, BPMN2 diagrams uses rounded-corner boxes to denote choreography tasks. Each of them is labeled with the roles of the two participants involved in the task, and the name of the service operation performed by the initiating participant and provided by the other one. A role contained in the white box denotes the initiating participant. In particular, we recall that the BPMN2 specification employs the

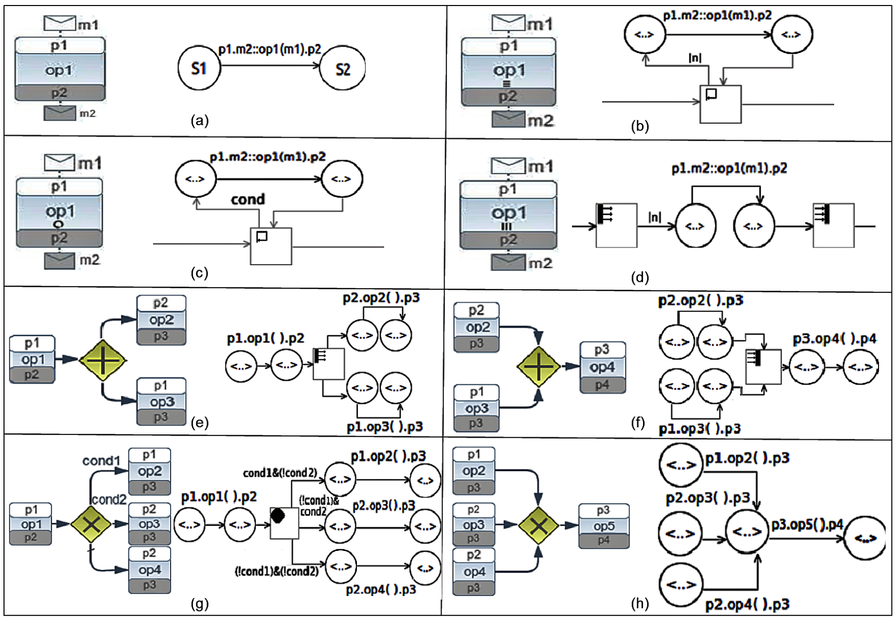

Fig. 1. From BPMN2 choreography to CLTS

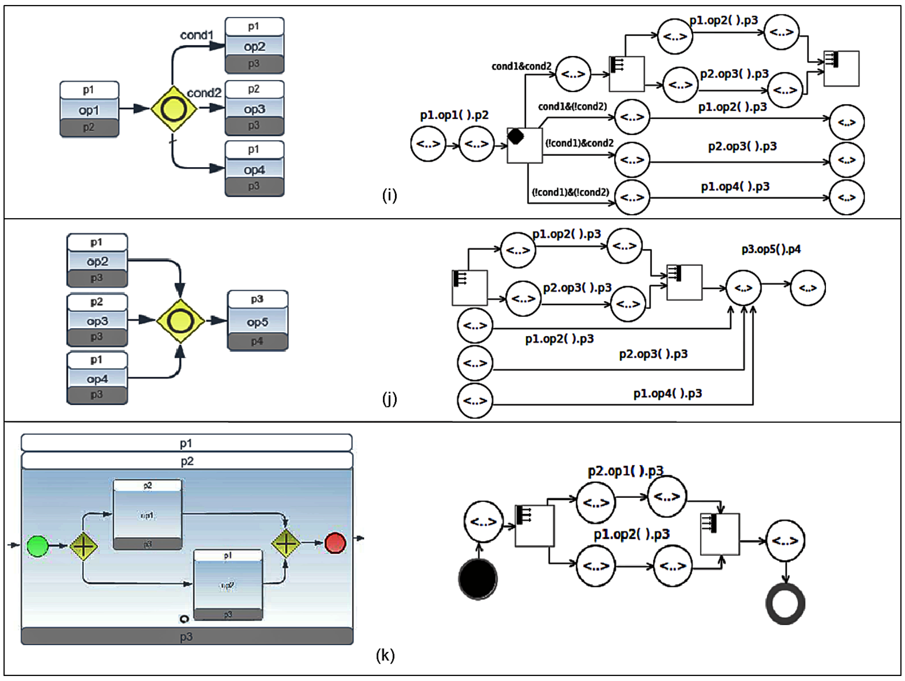

Fig. 2. From BPMN2 choreography to CLTS (Cont'd)

theoretical concept of a token that, traversing the sequence flows and passing through the elements in a process, aids to define its behavior. The start event generates the token that must eventually be consumed at an end event.

Depending on its type (i.e., `ChoreographyLoopType` in the BPMN2 specification), a task may have one of the three markers: sequential multi-instance (b), standard loop (c), and parallel multi-instance (d).

With reference to Figure 1 **(e) & (f)**, a Parallel Gateway is used to (e) create and/or (f) synchronize parallel flows without checking any condition. Each outgoing flow receives a token upon execution of this gateway. For incoming flows, this gateway will wait for all incoming flows before triggering the flow through its outgoing arrow. They create parallel paths of the choreography that all Participants are aware of. With respect to the constraints imposed by the BPMN2 official specification, the initiator participant(s) of all the tasks after the gateway must be involved in all tasks that immediately precede such gateway. The task that precedes the chain must also satisfy this constraint in the case where there is a chain of gateways with no tasks in between.

With reference to Figure 1 **(g) & (h)**, a Diverging (Decision) Exclusive Gateway (g) is used to create alternative paths within a choreography. If none of the conditional expressions (see `cond1` and `cond2`) evaluate to true, a default path can optionally be specified (see task `op4`). A Converging Exclusive Gateway (h) is used to merge alternative paths. Each incoming flow token is routed to the outgoing flow without synchronization. Being in a fully decentralized setting, there is no central mechanism to store the data that will be used in the condition expressions of the outgoing flows. The gateway's conditions may have natural language descriptions but, as clarified by the BPMN2 official specification, such choreographies would be underspecified and would not be enforceable. To create an enforceable choreography, the gateway conditions must be formal expressions that can be precisely (and automatically for tool supported approaches) checked. Still according to the BPMN2 official specification, the initiating participants of the choreography tasks that follow the gateway must have sent or received the message that provided the data upon which the conditional decision is made. In addition, the message that provides the data for the gateway conditional decision may be in any choreography task prior to the gateway (i.e., it does not have to immediately precede the gateway). Thus, for the gateway to be automatically enforced, we assume to have the specification of what messages provide the data upon which the conditional decision can be actually made.

With reference to Figure 2 **(i) & (j)**, a Diverging Inclusive Gateway (i) can be used to create alternative but also parallel paths. Unlike the Exclusive Gateway, all condition expressions are evaluated. All flows that evaluate to true will be traversed by a token. Since each path is considered to be independent, any combination of the paths may be taken, from zero to all. However, it should be designed so that at least one path is taken. If none of the conditional expressions (see `cond1` and `cond2`) evaluate to true, a default path can optionally be specified. A converging Inclusive Gateway (j) is used to merge a combination of alternative and parallel paths. A control flow token entering an Inclusive Gateway may be synchronized with some other token that arrives later.

With reference to Figure 2 **(k)**, a Sub-Choreography is a compound activity task that defines a flow of other tasks. Each sub-choreography involves two or more participants.

3 BPMN2-to-CLTS

In this section we discuss how it is possible to derive CLTS models out of BPMN2 choreography specifications. Such a translation can be done in different manners including the adoption of general purpose programming languages. However, as previously said, the presented work has been done in the context of the CHOReOS EU project where model-driven principles and techniques [5] have been employed to support the development of choreography-based systems.

Before describing the model transformation, we introduce the *CLTS* metamodel defined for extending LTSs, and constitutes the foundation of the coordination logic extracted by the synthesis process. The definition of these metamodels is the result of a survey we have conducted within CHOReOS. Specifically, in the literature a number of valuable approaches have been proposed to transform different kinds of choreography notations into more formal specifications. For instance, by leveraging the concept of token, alternative models, such as free-choice Petri Nets, might have been adopted (see Section 4). However, the deep study we have initially conducted within CHOReOS to precisely define, at the project level, the integration architecture for the CHOReOS Integrated Development and Run-time Environment[1] (IDRE), led the whole consortium to agree on the definition of the CLTS model, which best met the (both formal and technical) requirements of all the software tools now integrated by the IDRE. Indeed, to the purposes of defining an integrated suite of tools to support the whole choreography life cycle, the CLTS model brings together many features of already existing formalisms and notations in the literature, and filters out those ones not strictly needed. Last but not least, the main requirement was to have a notation as close as possible to the BPMN2 choreography diagrams, while enabling formal reasoning and automatic treatment by all the IDRE components.

A fragment of the CLTS metamodel is shown in Figure 3. The metamodel extends the basic notion of LTS state by introducing new elements to model complex states, i.e., `initial` and `final` states, `fork` and `join` states, as well as, `activity loop` and `alternative` states. The basic notion of labeled transition has been extended to have the possibility of specifying `participants roles`, `service operations request/response/fault` messages and related `types`, as well as, `conditions`.

As discussed later in the section, the BPMN2-to-CLTS transformation has been implemented in a model-driven setting by means of the ATLAS Transformation Language (ATL) [13]. A model transformation takes as input one or more models conforming to the source metamodels and generates one or more models conforming to the target metamodels. Thus, to develop the BPMN2-to-CLTS

[1] `http://www.choreos.eu/bin/view/Documentation/WebHome`

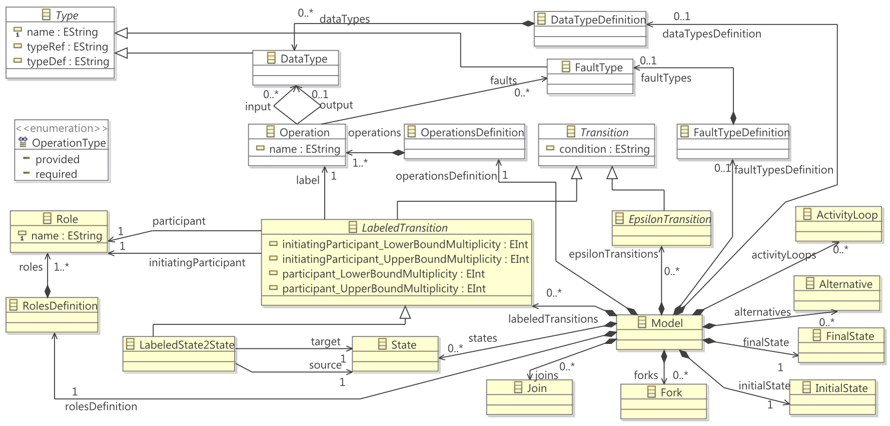

Fig. 3. CLTS metamodel

transformation, both the BPMN2 and CLTS metamodels are required. The former is available in the Eclipse ecosystem; the latter consists of the following elements (see Figure 3):

▷ the metaclass `Model` and its composition relations represent a choreography;

▷ plain states are represented by means of the metaclass `State`;

▷ initial and final states are represented by means of the metaclasses `InitialState` and `FinalState`, respectively;

▷ loop and alternative elements are represented by means of the metaclasses `ActivityLoop` and `Alternative`, respectively;

▷ fork and join elements are represented by means of the metaclasses `Fork` and `Join`, respectively;

▷ the set or roles played by the different participants in the choreography are represented by means of the metaclass `RoleDefinition`, which consists of a number of `Role` elements.

▷ the set of transition labels is represented by means of the metaclass `OperationDefinition`, which contains `Operation` elements.

▷ transition relations are represented by means of the metaclass `LabeledTransition`. The references **`initiatingParticipant`**, and **`participant`** are defined to represent the participants involved in the considered interactions;

In the remaining of the section we discuss how BPMN2 choreography diagram constructs can be mapped to CLTS model element. The discussion is based on the representative cases shown in Figure 1 and Figure 2.

(a)..(d) – As previously said, depending on its type (i.e., `ChoreographyLoopType` in the BPMN2 specification), a task may have one of the three markers: sequential multi-instance (b), standard loop (c), and parallel multi-instance (d). Accordingly, a task is transformed into a basic state-to-state transition if no marker is specified (a), a CLTS `ActivityLoop` transition with a fixed number |n| of possible iterations if a BPMN2 sequential multi-instance marker is specified (b), a conditional CLTS `ActivityLoop` transition if a BPMN2 standard loop marker is specified (c), and a CLTS state-to-state transition that can be forked (and then joined) a fixed number |n| of times if a BPMN2 parallel multi-instance marker is specified (d). In the corresponding CLTS fragments, the transition label $p1.m2::op1(m1).p2$ specifies that the participant $p1$ initiates the task $op1$ by sending the message $m1$ to the receiving participant $p2$ which, in turn, returns the message $m2$.

Note that the BPMN2 graphical elements do not show, neither the condition expressions nor the specified fixed number, which can only be internally specified. In/out messages (i.e., request/response messages) are reported in the CLTS transitions on the right-hand and left-hand sides of the operation name, respectively. To bound the number of times a loop is repeated, either the condition expressions must be evaluated (based on the data contained in the exchanged messages) in the case of a standard loops, or counters must be employed (updated upon the observed message exchanges) in the case of sequential and parallel multi-instance loops. In any case, the task must be performed at least once, before checking the condition or the counter. In this respect, it is worth to mention that one of the reported critical issues (issue number 16554 - published in the official web site) is about "underspecification of `ChoreographyLoopType`". The reported issue basically says that it is mandatory to specify either the number of loop repetitions or the expression that must be evaluated, on what messages and (for reasons of enforceability) by which participant(s). In our model transformation we have anticipated the resolution of this issue.

(e) & (f) – When used to create parallel flows, the parallel diverging gateway is transformed using a CLTS `Fork` state that splits into all the outgoing flows. Note that, in order to enforce the coordination logic implied by a parallel gateway, the `Fork` state is used in a CLTS to model real parallelism (and not abstract parallelism by means of interleaving). Complementarily, when a parallel converging gateway is used to join parallel flows, a CLTS `Join` state is used.

The sequences for the remaining cases **(g)..(k)** can be easily obtained by following the same method as for the previous cases.

(g) & (h) – When used to create alternative paths, a diverging exclusive gateway is transformed using a CLTS `Alternative` state. Note that the conditions `cond1` and `cond2` are suitably combined to achieve exclusivity. When used to merge alternative paths, a converging exclusive gateway is transformed using state-to-state transitions that, by modeling the flows immediately preceding the gateway, collapse into a further state-to-state transition that models the flow immediately following the gateway. As a further clarification, the very same converging exclusive gateway behavior can be equivalently specified in BPMN2 without using the gateway construct. That is, with reference to the figure, it is sufficient to have three arrows that directly connect the tasks on the left to the task on the right.

(i) & (j) – Similarly to a diverging exclusive gateway, diverging inclusive gateway is transformed using a CLTS `Alternative` state. However, to model that all combinations of the paths may be taken, combined forking and joining paths are used. To conform with this characteristic, the conditions `cond1` and `cond2` are

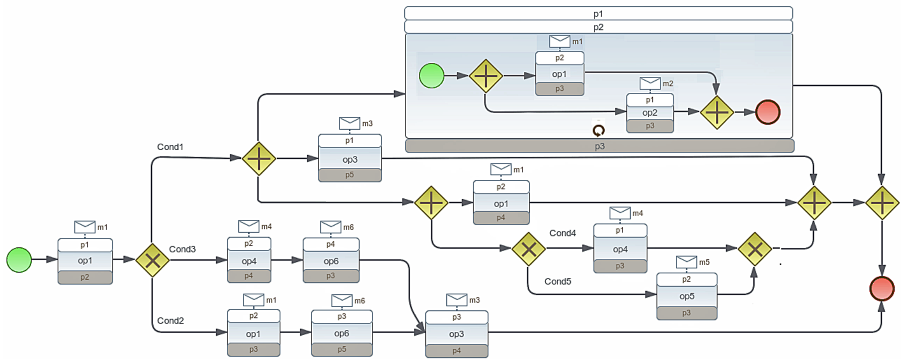
Fig. 4. BPMN2 choreography diagram example

suitably combined to achieve exclusivity between not only single paths, but also between the combined forking and joining paths and single paths. Considering the previous explanations for the converging exclusive gateway and the diverging inclusive gateway, the transformation for a converging exclusive gateway, when merging combinations of alternative and parallel paths, is rather intuitive.

(k) Compound activities tasks are transformed by recursively applying the previous rules. In Figure 2, only a very simple case is shown.

All the previously discussed mappings have been automatized by means an ATL [13] model transformation consisting of about 4.000 lines of code. By applying the transformation rules described above, the BPMN2 choreography diagram of Figure 4 is transformed to the corresponding CLTS diagram in Figure 5 (the CLTS diagram has been drawn by means of the GMF-based editor we have developed in CHOReOS). It is worth to clarify that the choreography in the figure has been aptly created to highlight, in one choreography, most of the crucial subtitles the transformation needs to handle. Therefore, it looks artificial from a use case point of view. For a set of realistic use cases, provided by the CHOReOS industrial partners, and for downloading the Eclipse plug-

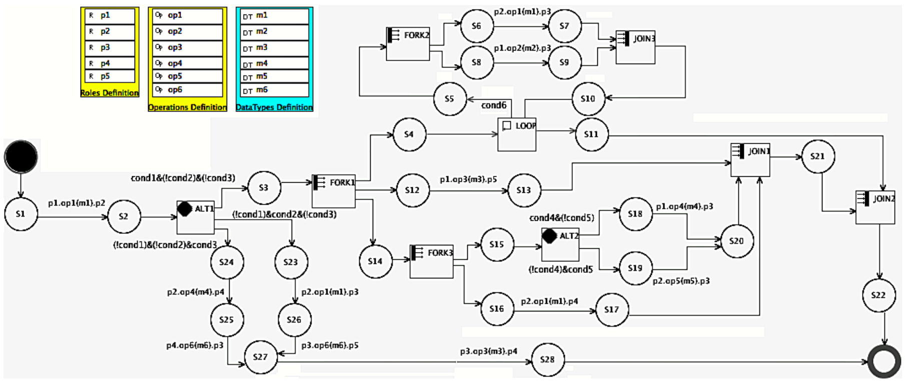
Fig. 5. CLTS derived from the BPMN2 choreography diagram in Figure 4

ins implementing the model transformation the interested reader can refer to `http://choreos.disim.univaq.it/`.

4 Related Work

The approach presented in this paper is related to a number of other valuable approaches in the literature that, to different purposes, transform different kinds of choreography notations into more formal specifications.

The definition of our approach required the consideration of valuable approaches in the literature (most notably [21,8,4,23,9,2]) and state-of-the-art languages, systems, and techniques that have emerged in different contexts including SOA, model-transformations, and seminal work on distributed coordination [15]. Their consideration within the same research and development space [1] is so far being representing the opportunity for us to harness our knowledge towards the systematic development of choreography-based systems.

The common idea underlying the approaches in [6,7,16,17,20] is to assume a high-level specification of the requirements that the choreography has to fulfill and a behavioral specification of the services participating in the choreography. From these two assumptions, by applying data and control-flow analysis, a BPEL, WSCI or WS-CDL description of a centralized choreographer specification is automatically derived. This description is derived in order to satisfy the specified choreography requirements.

The works described in [22,24,3,11,4] address the problem of checking whether a choreography can be realized by a set of interacting services, each of them synthesized by projecting the choreography specification on the role to be played. This problem is known as choreography realizability check. The focus is on verifying whether the set of services, required to realize a given choreography, can be easily implemented by simply considering the role-based local views of the specified choreography. That is, this verification does not aim at extracting the global coordination logic that, as we do in CHOReOS, is needed to check whether the collaboration among the discovered services leads to global interactions that violate the choreography behavior.

In [9] the authors presents a framework for verifying choreographies using model and equivalence checking techniques. Leveraging a translation of the choreography into LOTUS NT algebra, the framework enables the verification of some analysis tasks, i.e., repairability, realizability, conformance, synchronizability, and control for enforcing the choreography. In order to check in sequence the system synchronizability and realizability using equivalence checking, distributed controllers are generated through an iterative process presented in [10].

In [19], the authors show how to automatically generate a partial DAML-S process model out of a WSDL description. The generated process model can be then possibly completed manually by the developer. Although the aim of this work is completely different from ours, it shows that DAML-S can be considered as another possible notation BPMN2 choreographies could be mapped to.

The work in [14] presents an approach that generates a set of related orchestrations from a choreography specification. Specifically, a transformation to derive a set of BPEL specifications out of a CDL specification of the choreography is formalized. Similarly to us, the transformation mappings is implemented as transformation rules in ATL.

In [12] the authors propose a model-driven method to develop collaborative systems. The methods uses a graph transformation to derive a flow-local choreography from a flow-global choreography. UML Activity Diagrams are used to specify both the source model and the target model.

Most of the previous approaches consider as input choreography specified by using different notations and formalisms. Only few of them uses BPMN2 Choreography Diagrams, notably, [9] and [21]. Depending on the specific purposes, the different approaches transform the choreography into different (formal) representations such as Petri Net, LOTUS NT, various state machines, etc. Moreover, all these approaches are based on different interpretations of the choreography interaction semantics and consider different subsets of choreography constructs. A weakness here resides on the fact that all the adopted formal notations are distant from BPMN2.

5 Conclusions and Future Work

This paper presents a model transformation to extract from a BPMN2 choreography specification the global coordination logic and codify it into an extended LTS, called *Choreography LTS* (CLTS). The expressiveness of the CLTS metamodel allows us to fully automate the approach and to transform very complex choreography specifications into rigorous descriptions. The presented approach is implemented as a REST service and it part of a model-based tool chain (named CHOReO*Synt*[2]) released to support the development of choreography-based systems in the CHOReOS EU project.

The approach has been applied to real-world use cases, provided by the CHOReOS industrial partners, in the *Airport*, *Marketing and Sale*, and *Taxy Transportation* domains. Interested readers can refer to the CHOReOS project and CHOReO*Synt* web sites for documentation. The application of our approach to these use cases has shown that the method is viable and practical. However, although our preliminary validation has been carried out in a context in which the presence of relevant to the approach stackholders are present, a real quantitative assessment of the method needs further investigation. This is part of our ongoing work together with an industrial partner of the CHOReOS project. Another direction of future work will address the extension of the implemented transformations to transform also BPMN2 choreography specifications containing events, as well as event-based and complex gateways.

References

1. M. Aksit, I. Kurtev, and J. Bézivin. Technological Spaces: an Initial Appraisal. International Federated Conf. (DOA, ODBASE, CoopIS), Industrial Track, Los Angeles, 2002.
2. M. Autili, D. Ruscio, A. Salle, P. Inverardi, and M. Tivoli. A model-based synthesis process for choreography realizability enforcement. In *FASE*, volume 7793 of *LNCS*, pages 37–52. 2013.

[2] `http://choreos.disim.univaq.it`

3. S. Basu and T. Bultan. Choreography conformance via synchronizability. In *Proc. of WWW '11*, pages 795–804, 2011.
4. S. Basu, T. Bultan, and M. Ouederni. Deciding choreography realizability. In *Procs. of POPL 2012*, pages 191–202. ACM, 2012.
5. J. Bézivin. On the Unification Power of Models. *Jour. on Software and Systems Modeling (SoSyM)*, 4(2):171–188, 2005.
6. A. Brogi and R. Popescu. Automated Generation of BPEL Adapters. In *Proc. of ICSOC'06, volume 4294 of LNCS*, 2006.
7. D. Calvanese, G. D. Giacomo, M. Lenzerini, M. Mecella, and F. Patrizi. Automatic service composition and synthesis: the roman model. *IEEE Data Eng. Bull.*, 31(3):18–22, 2008.
8. G. Gössler and G. Salaün. Realizability of choreographies for services interacting asynchronously. In *FACS*, volume 7253 of *LNCS*, pages 151–167. 2012.
9. M. Güdemann, P. Poizat, G. Salaün, and A. Dumont. Verchor: A framework for verifying choreographies. In *FASE*, volume 7793 of *LNCS*, pages 226–230. 2013.
10. M. Güdemann, G. Salaün, and M. Ouederni. Counterexample guided synthesis of monitors for realizability enforcement. In *ATVA*, LNCS, pages 238–253, 2012.
11. S. Hallé and T. Bultan. Realizability analysis for message-based interactions using shared-state projections. In *FSE*, pages 27–36, 2010.
12. F. Han, S. Kathayat, H. Le, R. Brek, and P. Herrmann. Towards choreography model transformation via graph transformation. In *Procs. of ICSESS 2011*, pages 508–515, 2011.
13. F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. ATL: A model transformation tool. *Science of Computer Programming*, 72(1-2):31–39, 2008.
14. R. Khadka. *Model-Driven Development of Service Compositions: Transformation from Service Choreography to Service Orchestrations.* Thesis for a degree in master of science in computer science, University of Twente, Enschede, The Netherlands.
15. L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM*, 21:558–565, July 1978.
16. A. Marconi, M. Pistore, and P. Traverso. Automated Composition of Web Services: the ASTRO Approach. *IEEE Data Eng. Bull.*, 31(3):23–26, 2008.
17. T. Melliti, P. Poizat, and S. B. Mokhtar. Distributed behavioural adaptation for the automatic composition of semantic services. In *In Procs. of FASE'08/ETAPS'08*, pages 146–162, 2008.
18. OMG. Business Process Model And Notation (BPMN) Version 2.0. `http://www.omg.org/spec/BPMN/2.0/`.
19. M. Paolucci, N. Srinivasan, K. Sycara, and T. Nishimura. Towards a semantic choreography of web services: from wsdl to daml-s. In *In Procs. of ICWS'03*, pages 22–26, 2003.
20. J. Pathak, R. Lutz, and V. Honavar. Moscoe: An approach for composing web services through iterative reformulation of functional specifications. *International Journal on Artificial Intelligence Tools*, 17:109–138, 2008.
21. P. Poizat and G. Salaün. Checking the Realizability of BPMN 2.0 Choreographies. In *Proc. of SAC 2012*, pages 1927–1934, 2012.
22. G. Salaün. Generation of service wrapper protocols from choreography specifications. In *Proc. of SEFM*, 2008.
23. A. Stefanescu, S. Wieczorek, and M. Schur. Message choreography modeling. *Software & Systems Modeling*, pages 1–25, 2012.
24. J. Su, T. Bultan, X. Fu, and X. Zhao. Towards a theory of web service choreographies. In *WS-FM*, pages 1–16, 2007.