Collaboration vs Choreography Conformance Checking in BPMN 2.0: from Theory to Practice

***Abstract*—The BPMN 2.0 standard is nowadays largely used to model distributed components both in academic and industrial contexts. The notation makes possible to represent systems from different perspectives. A global perspective, using *choreography* diagrams, to describe the interactions between components without exposing their internal structure, and a local perspective, using *collaboration* diagrams, where the internal behavior of a component can be highlighted.**

In this paper, we propose a formal approach for checking conformance of choreographies, representing global constraints, with related collaborations, representing possible implementations. In particular, we provide a direct formal operational semantics for both BPMN collaboration and choreography diagrams, and we formalize the conformance concept by means of two relations defined on top of the semantics. We have developed the C^4 tool to support the approach into practice. We illustrate possible benefits of its usage by means of a simple, yet realistic, running example concerning a traveling scenario.

1. Introduction

Distributed components are at the basis of service-oriented applications and have been largely adopted in the last years. Components are usually implemented as software services and they are designed to support systems interoperability. Generally, these kinds of systems are composed by participants agreeing on communication patterns without accessing to internal technical details. A way to enable interoperability is to refer to global specifications [1], [2].

A lot of effort has been devoted to study specification languages for such kind of systems, and then to conceive techniques to assess how much a real implementation corresponds to a pre-designated model. Despite this effort, the results obtained so far did not lead to a widely accepted achievement both in academic and industrial communities. This is partially due to the fact that most of the works found in the literature approach the problem only at a foundational level, highly abstracting from widely used specification languages.

To overcome such a limitation, in this work we rely on the standard BPMN 2.0 [3] (in the following just BPMN), since this graphical notation has been largely adopted both in academia and industry. In particular, we consider the *choreography* and *collaboration* diagrams provided by BPMN. A choreography is a global specification that models interactions as sequences of messages among participants, while a collaboration describes the implementation of each single participant in terms of exchanged messages and internal behavior. Notably, even if the standard correlates the two models types, it does not give them a formal semantics suitable to guarantee the definition and checking of a precise conformance relation. In this paper we try to address this aspect, so to equip software engineers both with sound theories, and a tool to check the conformance of a composition of software components with respect to a global specification.

The proposed theoretical framework has been developed into the C^4 *(Collaboration vs Choreography Conformance Checker for BPMN 2.0)* tool that allows a modeler to design or select a collaboration diagram, obtained from the composition of a set of components, and then to check it against a choreography specification. The tool uses standard formats, to permit the easy integration with external IDEs, and makes the usage of formal methods completely transparent to the modeler.

Summing up, the major contributions of this paper are:

1) the definition and the implementation in Java of a formal operational semantics for BPMN collaborations and choreographies, enabling the *formal analysis* of BPMN models;
2) the definition and implementation of two conformance relations, namely bisimulation-based and trace-based conformances;
3) the C^4 tool integrated with mCRL2 [4] for automatic conformance checking.

It is worth noticing that the distinctive aspect of this work is the specific focus on the well-established standard BPMN, taking into account its specificities and peculiarities typically overlooked by other works (see Sec. 2 for a detailed comparison with the literature).

The rest of the paper is organized as follows. Sec. 2 motivates our work detailing its differences in comparison to related works. Sec. 3 provides background notions on BPMN choreographies and collaborations, together with a running example. Sec. 4 introduces formal syntax and semantics both for choreographies and collaborations, while Sec. 5 defines conformance relations. Sec. 6 presents the C^4 tool and illustrates its usage in practice. Sec. 7 concludes the paper and discusses directions for future work.

2. Motivations and Related Works

A lot of effort has been devoted by the research community to study specification languages for choreographies and collaborations. To this aim, different notations and approaches have been defined and used, but none of them is able to combine in a unique framework the modelling and verification capabilities considering BPMN, and its expressiveness. There is a clear lack of results specifically devoted for BPMN, and this is the main motivation that drove our work.

In the following we relate the distinctive aspects of our work to the available literature, to clarify how the proposed approach actually contributes to the state of the art.

Modeling Notation. In the literature, significant contributions on conformance rely on the WS-CDL standard [5]–[9]. However, WS-CDL has never obtained wide approval in practice, also due to its strong relationship with the Web Services technology. In order to have a concrete impact on practice we based our study on the BPMN standard [10]. This notation received a lot of interest both from the industry, thanks to the consolidation of BPMN management systems, and from the academia, as also testified by several EU projects that adopted it as the modeling notation to be used to describe choreographies (e.g. CHOReOS [11], CHOReVOLUTION [12]). Certainly there are other proposals considering BPMN as the modeling notation, but none of them provide a direct semantics.

Direct Semantics. The usage of a direct semantics for BPMN reduces errors potentially introduced during the translation phase, and do not ask to translate syntactical terms of the source language (i.e. BPMN) into other syntactical terms of a target language, equipped with a formal semantics. Differences in expressiveness of the source and target languages could even hinder the definition of the translation for the whole language. In the literature many proposals can be found following a translation-based approach and distinctions exist in relation to the used target language, such as process algebras [7], [13], [14], more complex languages (e.g. LOTOS) [15]–[20], transition-based models (e.g. Petri Nets) [6], [9], [21] or session types [1]. The direct semantics proposed here is inspired by [22], but finalized to a different goal. In particular here we provide a global semantics able to catch all the system states from a global point of view.

It is worth to remark some of the advantages of a direct semantics approach with respect to mappings from BPMN to Petri Nets [23], also in relation to their possible usage for conformance checking [21]. In particular Petri Nets are not enough expressive to describe situations where the system is waiting for a message that will never occur. This restriction leads to assume that events always occur along a path, somehow merging the semantics of message based events with that of message tasks. This also affects the semantics of event-based gateways treated as exclusive gateways, since the event-based is not anymore triggered by events. Finally, the terminate event has a non local behavior that is difficult to reproduce with Petri Nets. The direct semantics we provide instead does not suffer from any of the listed issues.

Arbitrary topology. The direct semantics we propose supports a wider class of models, i.e. that of models with an *arbitrary topology*, without imposing any constraint on the structure of models. Indeed, even if structuredness has been considered as a good modeling practice [24], designers often do not follow such a guideline [25], because in this way the modeling activity results to be less complex [26] and more expressive [27], [28]. Instead, most of the approaches to formal semantics of BPMN found in the literature impose constraints on the structure of the model. We believe that considering models with an arbitrary topology can contribute to an extensive adoption of our approach, thus having a real impact on the development of process-aware systems.

Conformance vs Communication models. In order to catch dissimilarity defined in the BPMN standard, between choreographies and collaborations, we defined two different semantics. The behavioral aspects of the two models are successively related using two conformance relations that are able to deal with inhomogeneous communications, as choreographies are synchronous and collaborations asynchronous. In this regard we differ from the literature where, to deal with asynchronous communications, additional constructs are used, such as buffers [17] or other language structures [13], [19], and others [5], [6], [14], [16] where asynchronous models are reduced in synchronous ones. Moreover, the peculiarity of these relations is on the observed actions. In particular, messages in a choreography are related to received messages in the collaboration, differently from most works in the literature [5], [7] that focus the observation on send actions.

Conformance vs Non-determinism. Notably, one of the main novelties of our contribution is the definition of conformance relationships that deal with different forms of non-determinism generated by message exchange. In BPMN there are two different elements for describing choices: the event-based gateway, that produces non-dominated non-determinism (roughly no one in the model as complete knowledge on the decision that will be taken), and the exclusive gateway, that produces dominated non-determinism (roughly the decision is taken by one party and followed by the others). For such scope we developed two relations, Trace-Based and Bisimulation-Based Conformance, and considered in the semantics the dominated and non-dominated non-determinism as prescribed by the BPMN standard. This is somehow similar to [7], [8] that rely on the concept of internal and external choice defined in the CSP process algebra.

Tool support. The final distinctive result, which incorporates all the previous advantages, is the availability of a tool, where concepts like formal semantics and conformance

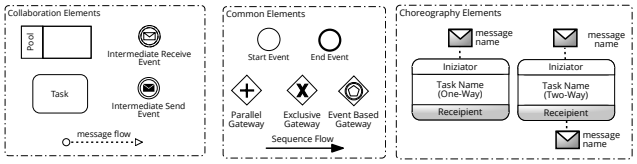


Figure 1. BPMN 2.0 Elements.

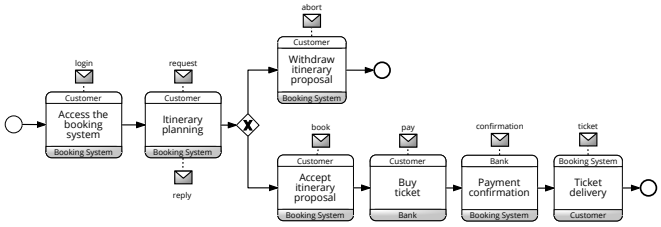


Figure 2. Booking Choreography.

relations are included and made easily accessible to non-expert users by means of a GUI. A similar endeavor has lead to the VerChor tool [19]. However, VerChor purpose is different: it uses conformance to check the realizability of a set of peers obtained from a projection of a given choreography.

3. BPMN 2.0 Overview

The focus of this section is not a complete presentation of BPMN, but a discussion of relevant aspects for choreography and collaboration diagrams we use in the paper. We also introduce a scenario to be used as a running example.

BPMN Standard. Fig. 1(center) depicts the main used modeling elements that are included in both diagrams. **Events** are used to represent something that can happen. An event can be a *Start Event*, representing the point in which the choreography/collaboration starts, while an *End Event* is raised when the choreography/collaboration terminates. **Gateways** are used to manage the flow of a choreography/collaboration both for parallel activities and choices. Gateways act as either join nodes (merging incoming sequence edges) or split nodes (forking into outgoing sequence edges). Different types of gateways are available. A *parallel gateway (AND)* in join mode has to wait to be reached by all its incoming edges to start, and respectively all the outgoing edges are started simultaneously in the split case. An *exclusive gateway (XOR)* describes choices; it is activated each time the gateway is reached in join mode and, in split mode, it activates exactly one outgoing edge. An *Event Based gateway* is similar to the XOR-split gateway, but its outgoing branches activation depends on the occurrence of a catching event in the collaboration and on the reception of a messages in the choreography; these events/messages are in a race condition, where the first one that is triggered wins and disables the other ones. **Sequence Flows** are used to connect collaboration/choreography elements, they are used to specify the flow of the collaboration/choreography.

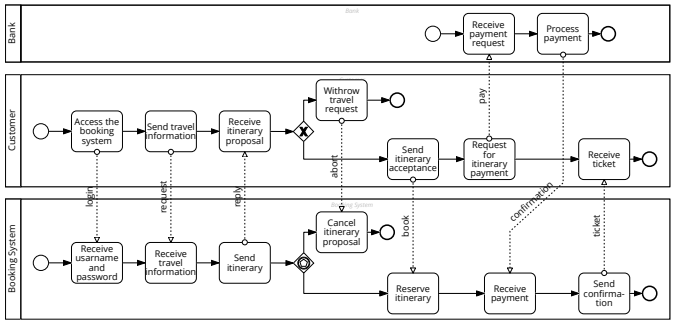


Figure 3. Booking Collaboration.

In the collaboration diagram, the following elements are also included (Fig. 1, left-hand side). **Pools** are used to represent participants involved in the collaboration. **Tasks** are used to represent specific works to perform within a collaboration by a participant. **Intermediate Events** represent something that happens, such as sending or receiving of a message. **Message Edges** are used to visualize communication flows between different participants, by connecting communication elements within different pools.

Focusing on the choreography diagram, we underline its ability to specify the message exchange between two or more participants. This is done by means of **Choreography Tasks** (Fig. 1, right-hand side). They are drawn as rectangles divided in three bands: the central one refers to the name of the task, while the others refer to the involved participants (the white one is the initiator, while the gray one is the recipient). Messages can be sent either by one participant (One-Way tasks) or by both participants (Two-Way tasks).

Running Example. We introduce here a choreography and a possible implementation in a collaboration model.

Choreography Example. The choreography in Fig. 2 combines a booking system, a customer and a bank that have to interact in order to book a travel. After accessing to the booking system the customer requests an itinerary and receives a tentative planning. Then, the choreography can proceed following two different paths according to the customer decision. The upper path is followed when the customer decides to withdraw the travel proposal; while the lower path is used for the the proposal acceptance. In particular, when the proposal is accepted, the customer interacts with the bank for the payment of the ticket, and then the bank sends the confirmation to the booking system. This completes then the procedure by sending the ticket to the customer.

Collaboration Example. The collaboration in Fig. 3 combines the work-activities of the same participants. After the customer logins to the booking system, he requests some travel information and receives a proposal from the booking system. The customer then decides whether to withdraw or accept the proposal; this is represented by means of a XOR gateway. According to this decision, either the upper path, for the proposal withdraw, or the lower path, for the confirmation, is activated. The booking system waits for the decision of the customer and behaves accordingly. This is represented by means of an Event-based gateway. In case of

withdraw, the two participants terminate with an end event. In case of confirmation, the customer sends the itinerary acceptance to the booking system and asks for payment to the bank. As soon as the bank processes the payment and confirms it to the booking system, the customer receives the ticket.

4. C^4 Formal Framework: Semantics

This section presents our formalization of the BPMN semantics at the bases of the proposed framework. In the formalization we follow a pragmatic approach, as we focus on those elements regularly used to design process models [29]. Specifically, we first summarize its distinctive aspects in relation to the BPMN modeling principles, and then we illustrate its formal definition.

4.1. Linguistic Aspects and Design Choices

Considering the choreography diagrams, we made some specific design choices. In relation to the *Two-Way choreography task*, the OMG standard states that it is "an atomic activity in a choreography process" execution [3, p. 323]. However, this does not mean that the task blocks the whole execution of the choreography. In fact, participants are usually distributed, and we assume that other choreography tasks involved in different parallel paths of the choreography can be executed. Thus, here we intend atomicity to mean that both messages exchanged in a Two-Way task have to be received before triggering the execution along the sequence flow outgoing from the task. Therefore, even if we allow Two-Way tasks in the choreography models, we safely manage them as pairs of One-Way tasks preserving the same meaning.

A further distinctive aspect of our formal semantics concerns the *communication model* that, to be compliant with the BPMN standard, is different for choreographies and collaborations. In the former case, the communication is expressed using synchronous messages. Indeed, according to the standard [3, p. 315], a choreography task completes when the receiver participant reads the message, triggering in this way the execution of the element connected to the task by means of its outgoing sequence edge. Synchronous communication requires choreography tasks to be blocking activities, which resume the execution only when an exchanged message is actually received. The communication model of collaborations, instead, is asynchronous, like that of distributed systems in reality. This means that a message sent by one participant is enqueued by the receiving one, which can then consume and process it subsequently, while the sender is free to proceed with its execution.The use of two different communication models also impacts on the definition of the conformance relation as illustrated in Sec. 5.

4.2. Semantics of BPMN Choreographies

To enable a formal treatment of a BPMN choreography we defined a BNF syntax of its model structure (Fig. 4). In

$$
\begin{aligned}
Ch ::= {} & \mathsf{start}(\mathsf{e}_o) \mid \mathsf{end}(\mathsf{e}_i) \mid \mathsf{andSplit}(\mathsf{e}_i, E_o) \mid \mathsf{andJoin}(E_i, \mathsf{e}_o) \\
& \mid \mathsf{xorSplit}(\mathsf{e}_i, E_o) \mid \mathsf{xorJoin}(E_i, \mathsf{e}_o) \mid \mathsf{task}(\mathsf{e}_i, \mathsf{e}_o, \mathsf{o}_1, \mathsf{o}_2, \mathsf{m}, \mathsf{t}) \\
& \mid \mathsf{eventBased}(\mathsf{e}_i, T_1, T_2) \mid Ch_1 | Ch_2 \\
T ::= {} & (\mathsf{e}_o, \mathsf{o}_1, \mathsf{o}_2, \mathsf{m}, \mathsf{t}) \mid T_1, T_2
\end{aligned}
$$

Figure 4. Syntax of BPMN Choreography Structures.

the proposed grammar, the non-terminal symbol Ch represents *Choreography Structures*, while the terminal symbols, denoted by the sans serif font, are the considered elements of a BPMN model, i.e. events, tasks and gateways. Notably, we are not proposing a new modeling formalism, but we are only using a textual notation for the BPMN elements. Indeed, with respect to the graphical notation, the textual one is more coincise for writing the operational rules and more manageable for the implementation of the semantics.

In the following $\mathsf{e} \in \mathbb{E}$ denotes a sequence edge, while $E \in 2^{\mathbb{E}}$ a set of edges; we require $|E| > 1$ when E is used in joining and splitting gateways. For the convenience of the reader we refer with e_i the edge incoming in an element and with e_o the edge outgoing from an element. o, m, and t denote names uniquely identifying an organization, a message and a task, respectively. The correspondence between the syntax used here and the graphical notation of BPMN illustrated in Sec. 3 is as follows.

- $\mathsf{start}(\mathsf{e}_o)$ represents a start event with outgoing edge e_o.
- $\mathsf{end}(\mathsf{e}_i)$ represents an end event with incoming edge e_i.
- $\mathsf{andSplit}(\mathsf{e}_i, E_o)$ (resp. $\mathsf{xorSplit}(\mathsf{e}_i, E_o)$) represents an AND (resp. XOR) split gateway with incoming edge e_i and outgoing edges E_o.
- $\mathsf{andJoin}(E_i, \mathsf{e}_o)$ (resp. $\mathsf{xorJoin}(E_i, \mathsf{e}_o)$) represents an AND (resp. XOR) join gateway with incoming edges E_i and outgoing edge e_o.
- $\mathsf{task}(\mathsf{e}_i, \mathsf{e}_o, \mathsf{o}_1, \mathsf{o}_2, \mathsf{m}, \mathsf{t})$ represents a one-way task t with incoming edge e_i and outgoing edge e_o sending a message m from o_1 to o_2. As explained in Sec. 4.1, two-way tasks are rendered in our formal framework as pairs of one-way tasks, hence they are not explicitly included in the syntax.
- $\mathsf{eventBased}(\mathsf{e}_i, T_1, T_2)$ represents an event-based gateway with incoming edge e_i, and a list of (at least two) tasks T_1, T_2 to be processed. It is worth noticing that the definition of the task list T is composed by elements of the same structure of the one-way task except for the incoming edge, which is subsumed in the structure of the event-based gateway. When convenient, we shall regard a task list simply as a set.
- $Ch_1 | Ch_2$ represents a composition of elements in order to render a process structure in terms of a collection of elements.

To achieve a compositional definition, each sequence edge of the BPMN model is split in two parts: the part outgoing from the source element and the part incoming into the target element. The two parts are correlated by means of unique sequence edge names in the BPMN model. To avoid malformed structure models, we only consider structures in which for each edge labeled by e outgoing from an element, there exists only one corresponding edge labeled

by e incoming into another element, and vice versa.

The operational semantics we propose is given in terms of configurations of the form $\langle Ch, \sigma \rangle$, where Ch is a choreography structure, and σ is the execution state storing for each edge the current number of tokens marking it. Specifically, a state $\sigma : \mathbb{E} \to \mathbb{N}$ is a function mapping edges to numbers of tokens. The state obtained by updating in the state σ the number of tokens of the edge e to n, written as $\sigma \cdot \{\mathsf{e} \mapsto \mathsf{n}\}$, is defined as follows: $(\sigma \cdot \{\mathsf{e} \mapsto \mathsf{n}\})(\mathsf{e}')$ returns n if $\mathsf{e}' = \mathsf{e}$, otherwise it returns $\sigma(\mathsf{e}')$. The *initial state*, where all edges are unmarked is denoted by σ_0 formally, $\sigma_0(\mathsf{e}) = 0 \quad \forall \mathsf{e} \in \mathbb{E}$. The transition relation over configurations, written $\xrightarrow{l}$ and defined by the rules in Fig. 5, formalizes the execution of a choreography in terms of marking evolution and message exchange. Labels l represent computational steps and are defined as: τ, denoting internal computations (in the rules these labels are omitted for the sake of readability); and $\mathsf{o}_1 \to \mathsf{o}_2 : \mathsf{m}$, denoting an exchange of message m from organization o_1 to o_2. Notably, despite the presence of labels, this has to be thought of as a reduction semantics, because labels are not used for synchronization (as instead it usually happens in labeled semantics), but only for keeping track of the exchanged messages in order to enable the conformance checking discussed in Sec. 5. Since choreography execution only affects the current states, for the sake of presentation, we omit the choreography structure from the target configuration of the transition. Thus, a transition $\langle Ch, \sigma \rangle \xrightarrow{l} \langle Ch, \sigma' \rangle$ is written as $\langle Ch, \sigma \rangle \xrightarrow{l} \sigma'$.

Before commenting on the rules, we introduce the auxiliary functions they exploit. Specifically, function $inc : \mathbb{S} \times \mathbb{E} \to \mathbb{S}$ (resp. $dec : \mathbb{S} \times \mathbb{E} \to \mathbb{S}$), where $\mathbb{S}$ is the set of states, allows updating a state by incrementing (resp. decrementing) by one the number of tokens marking an edge in the state. Formally, they are defined as follows: $inc(\sigma, \mathsf{e}) = \sigma \cdot \{\mathsf{e} \mapsto \sigma(\mathsf{e}) + 1\}$ and $dec(\sigma, \mathsf{e}) = \sigma \cdot \{\mathsf{e} \mapsto \sigma(\mathsf{e}) - 1\}$. These functions extend in a natural ways to sets of edges as follows: $inc(\sigma, \emptyset) = \sigma$ and $inc(\sigma, \{\mathsf{e}\} \cup E)) = inc(inc(\sigma, \mathsf{e}), E)$; the cases for dec are similar.

For the sake of space we only describe some rules in Fig. 5, the meaning of the other can be easily deduced. In particular rule *Ch-Start* starts the execution of a process when it is in its initial state (i.e., all edges are unmarked). The effect of the rule is to increment the number of tokens in the edge outgoing from the start event. Rule *Ch-AndJoin* decrements the tokens in each incoming edge and increments the number of tokens of the outgoing edge, when each incoming edge has at least one token. Rule *Ch-XorSplit* is applied when a token is available in the incoming edge of a XOR split gateway, the rule decrements the token in the incoming edge and increments the tokens in one of the outgoing edges. Rule *Ch-Task* is activated when there is a token in the incoming edge of a choreography task, so that the application of the rule produces a message exchange label and moves the token from the incoming edge to the outgoing one. Finally, rules *Ch-Int$_1$* and *Ch-Int$_2$* deal with interleaving in a standard way.

$$(\textit{Ch-Start}) \quad \langle \mathsf{start}(\mathsf{e}_o), \sigma_0 \rangle \to inc(\sigma_0, \mathsf{e}_o)$$

$$(\textit{Ch-End}) \quad \langle \mathsf{end}(\mathsf{e}_i), \sigma \rangle \to dec(\sigma_0, \mathsf{e}_i) \qquad \sigma(\mathsf{e}_i) > 0$$

$$(\textit{Ch-AndSplit}) \quad \langle \mathsf{andSplit}(\mathsf{e}_i, E_o), \sigma \rangle \to inc(dec(\sigma, \mathsf{e}_i), E_o) \qquad \sigma(\mathsf{e}_i) > 0$$

$$(\textit{Ch-AndJoin}) \quad \langle \mathsf{andJoin}(E_i, \mathsf{e}_o), \sigma \rangle \to inc(dec(\sigma, E_i), \mathsf{e}_o) \qquad \forall \mathsf{e} \in E_i . \sigma(\mathsf{e}) > 0$$

$$(\textit{Ch-XorSplit}) \quad \langle \mathsf{xorSplit}(\mathsf{e}_i, \{\mathsf{e}\} \cup E_o), \sigma \rangle \to inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}) \qquad \sigma(\mathsf{e}_i) > 0$$

$$(\textit{Ch-XorJoin}) \quad \langle \mathsf{xorJoin}(\{\mathsf{e}\} \cup E_i, \mathsf{e}_o), \sigma \rangle \to inc(dec(\sigma, \mathsf{e}), \mathsf{e}_o) \qquad \sigma(\mathsf{e}) > 0$$

$$(\textit{Ch-Task}) \quad \langle \mathsf{task}(\mathsf{e}_i, \mathsf{e}_o, \mathsf{o}_1, \mathsf{o}_2, \mathsf{m}, \mathsf{t}), \sigma \rangle \xrightarrow{\mathsf{o}_1 \to \mathsf{o}_2 : \mathsf{m}} inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o) \qquad \sigma(\mathsf{e}_i) > 0$$

$$(\textit{Ch-EventG}) \quad \langle \mathsf{eventBased}(\mathsf{e}_i, (\mathsf{e}_o, \mathsf{o}_1, \mathsf{o}_2, \mathsf{m}, \mathsf{t}) \cup T), \sigma \rangle \xrightarrow{\mathsf{o}_1 \to \mathsf{o}_2 : \mathsf{m}} inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o) \qquad \sigma(\mathsf{e}_i) > 0$$

$$\frac{\langle Ch_1, \sigma \rangle \to \sigma'}{\langle Ch_1 | Ch_2, \sigma \rangle \to \sigma'} (\textit{Ch-Int}_1) \qquad \frac{\langle Ch_2, \sigma \rangle \to \sigma'}{\langle Ch_1 | Ch_2, \sigma \rangle \to \sigma'} (\textit{Ch-Int}_2)$$

Figure 5. BPMN Choreography Semantics (τ labels are omitted).

4.3. Semantics of BPMN Collaborations

The formal treatment of collaborations is similar to that of choreographies, therefore, we will focus here only on the main differences. The BNF syntax of the collaboration model structure is given in Fig. 6. The non-terminal symbol C represents *Collaboration Structures*, while terminal symbols denote, as usual, the considered BPMN elements. Differently from a choreography, the message exchange in a collaboration is modeled by means of *message edges*. Here, they are represented by triples of the form $(\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})$ indicating, in order, the sending organization, the receiving organization and the message; we use $\mathbb{M}$ to denote the set of message edges. Accordingly, an event-based gateway specifies a list of (at least two) message edges, each one enriched with the outgoing edge enabled by the message reception. Moreover, in a collaboration model there are three types of tasks, i.e. non-communicating (task), receiving (taskRcv) and sending (taskSnd), and also receiving and sending intermediate events (interRcv and interSnd, respectively).

$$\begin{array}{lcl}
C & ::= & \mathsf{start}(\mathsf{e}_o) \mid \mathsf{end}(\mathsf{e}_i) \mid \mathsf{andJoin}(E_i, \mathsf{e}_o) \mid \mathsf{xorSplit}(\mathsf{e}_i, E_o) \\
 & \mid & \mathsf{andSplit}(\mathsf{e}_i, E_o)\mathsf{xorJoin}(E_i, \mathsf{e}_o) \mid \mathsf{task}(\mathsf{e}_i, \mathsf{e}_o) \\
 & \mid & \mathsf{taskRcv}(\mathsf{e}_i, \mathsf{e}_o, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \mid \mathsf{taskSnd}(\mathsf{e}_i, \mathsf{e}_o, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \\
 & \mid & \mathsf{eventBased}(\mathsf{e}_i, M_1, M_2) \mid \mathsf{interRcv}(\mathsf{e}_i, \mathsf{e}_o, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \\
 & \mid & \mathsf{interSnd}(\mathsf{e}_i, \mathsf{e}_o, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \mid C_1 | C_2 \\
M & ::= & (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m}, \mathsf{e}_o) \mid M_1, M_2
\end{array}$$

Figure 6. Syntax of BPMN Collaboration Structures.

The operational semantics we propose is given in terms of configurations of the form $\langle C, \sigma, \delta \rangle$, where: C is a collaboration structure; σ is the first part of the execution

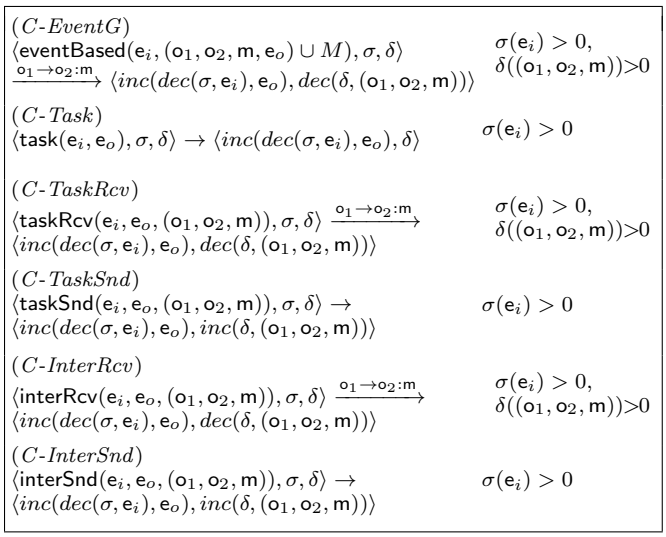


Figure 7. BPMN Collaboration Semantics (excerpt of rules).

state, storing for each sequence edge the current number of tokens marking it; and δ is the second part of the execution state, storing for each message edge the current number of message tokens marking it. Specifically, $\delta : \mathbb{M} \to \mathbb{N}$ is a function mapping message edges to numbers of message tokens; so that $\delta((\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) = \mathsf{n}$ means that there are n messages of type m sent by o_1 and stored in the o_2's queue. Notably, to deal with decidability issues, in the implementation we fixed the maximum of admissible tokens in a message edge. Update and initial state for δ are defined in a way similar to σ's definitions.

The transition relation $\xrightarrow{l}$ over collaboration configurations formalizes the execution of a collaboration in terms of edge and message marking evolution. It is defined by the rules in Fig. 7 (for the sake of presentation, we report only the rules concerning message exchange, as the other rules are similar to those for choreography diagrams). As usual, we omit the collaboration structure from the target configuration of transitions.

We now briefly comment on some of the operational rules. Rule *C-EventBased* is activated when there is a token in the incoming edge and there is a message m to be consumed, so that the application of the rule moves the token from the incoming edge to the outgoing edge corresponding to the received message, whose number of tokens in the meantime is decreased (i.e., a message from the corresponding queue is consumed). Rules *C-Task* deals with simple tasks, acting as a pass through. It is activated only when there is a token in the incoming edge, which is then moved to the outgoing edge. Rule *C-TaskRcv* is activated not only when there is a token in the incoming edge, like the one related to simple tasks, but also when there is a message to be consumed. Similarly, rule *C-TaskSnd*, instead of consuming, adds a message in the corresponding queue. Rule *C-InterRcv* (resp. *C-InterSnd*) follow the same behavior of rule *C-TaskRcv* (resp. *C-TaskSnd*).

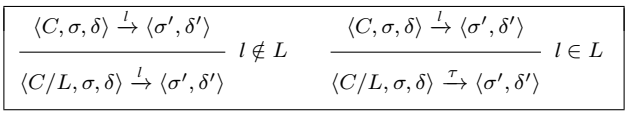


Figure 8. Hiding Operator.

5. C^4 Formal Framework: Conformance-checking

In this section we first discuss about the relations we propose for checking the conformance between choreographies and collaborations, then we present how they work into practice.

Bisimulation-Based and Trace-Based Conformances. Here we present the Bisimulation-Based Conformance (BBC) and the Trace-Based Conformance (TBC) relations we have defined to check if an implementation (collaboration) respects a given global specification (choreography). The two conformances are inspired to well-established behavioral equivalences [30], largely used in the literature and revised to deal with BPMN characteristics. Before providing the formal definition of BBC, we introduce the necessary notation. $\mathbb{C}h$ and $\mathbb{C}$ represents the sets of all choreography and collaboration configurations, respectively. Moreover, weak transitions are defined as follows: $\Rightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}$, i.e. zero or more τ-transitions; $\stackrel{l}{\Rightarrow}$ denotes $\Rightarrow\xrightarrow{l}\Rightarrow$. We exploit functions $labels(C)$ and $labels(Ch)$ returning the sets of all communication labels that can be potentially generated by the collaboration C and the choreography Ch, respectively. These functions are inductively defined on the syntax of collaboration and choreography structures in a straightforward way. For example, in case of choreographies we have the definition case $labels(\mathsf{task}(\mathsf{e}_i, \mathsf{e}_o, \mathsf{o}_1, \mathsf{o}_2, \mathsf{m}, \mathsf{t})) = \{o_1 \to o_2 : m\}$, meaning that if a choreography contains a task element, then its label set contains the label corresponding the message exchange described by the task.

Finally, at the collaboration level the definition of conformance requires the use of the hiding operator */*, defined by the rules in Fig. 8. This operator, as usual, transforms in τ all the actions in the set L, in order to consider them as internal actions in the conformance relation.

Definition 1. - **Bisimulation-Based Conformance Relation.**
A relation $\mathcal{R} \subseteq (\mathbb{C}h \times \mathbb{C})$ is a weak Bisimulation Conformance if, for any $\langle Ch, \sigma_{ch}\rangle \in \mathbb{C}h$ and $\langle C, \sigma_c, \delta\rangle \in \mathbb{C}$ such that $\langle Ch, \sigma_{ch}\rangle \ \mathcal{R} \ \langle C, \sigma_c, \delta\rangle$, it holds:

- for all o_1, o_2, m and σ'_{ch}, if $\langle Ch, \sigma_{ch}\rangle \xrightarrow{o_1 \to o_2 : m} \sigma'_{ch}$ then $\langle C, \sigma_c, \delta\rangle \stackrel{o_1 \to o_2 : m}{\Longrightarrow} \langle \sigma'_c, \delta'\rangle$ for some σ'_c, δ' s.t. $\langle Ch, \sigma'_{ch}\rangle \ \mathcal{R} \ \langle C, \sigma'_c, \delta'\rangle$;
- for all o_1, o_2, m, σ'_c and δ', if $\langle C, \sigma_c, \delta\rangle \xrightarrow{o_1 \to o_2 : m} \langle \sigma'_c, \delta'\rangle$ then $\langle Ch, \sigma_{ch}\rangle \stackrel{o_1 \to o_2 : m}{\Longrightarrow} \sigma'_{ch}$ for some σ'_{ch} s.t. $\langle Ch, \sigma'_{ch}\rangle \ \mathcal{R} \ \langle C, \sigma'_c, \delta'\rangle$;

- for all σ'_{ch}, if $\langle Ch, \sigma_{ch}\rangle \xrightarrow{\tau} \sigma'_{ch}$ then $\langle C, \sigma_c, \delta\rangle \Longrightarrow \langle \sigma'_c, \delta'\rangle$ for some σ'_c, δ' s.t. $\langle Ch, \sigma'_{ch}\rangle \; \mathcal{R} \; \langle C, \sigma'_c, \delta'\rangle$;
- for all σ'_c and δ', if $\langle C, \sigma_c, \delta\rangle \xrightarrow{\tau} \langle \sigma'_c, \delta'\rangle$ then $\langle Ch, \sigma_{ch}\rangle \Longrightarrow \sigma'_{ch}$ for some σ'_{ch} s.t. $\langle Ch, \sigma'_{ch}\rangle \; \mathcal{R} \; \langle C, \sigma'_c, \delta'\rangle$.

A choreography $\langle Ch, \sigma_{ch}\rangle$ and a collaboration $\langle C, \sigma_c, \delta\rangle$ *conform* if there exists a weak Bisimulation Conformance relation $\mathcal{R}$ such that $\langle Ch, \sigma_{ch}\rangle \; \mathcal{R} \; \langle C/(labels(C) \backslash labels(Ch)), \sigma_c, \delta\rangle$.

The proposed BBC relation considers to conform collaborations that are able to simulate step by step choreographies, and vice versa. In particular, if the choreography performs a message exchange, in the collaboration we expect to observe the reception of the message, possibly preceded or followed by any number of internal actions, and then the two continuations have to be in relation. Analogously, if we observe a message reception in the collaboration, the choreography has to reply with the corresponding weak transition. Moreover, if one of the two models performs an internal action, the counterpart can react with a weak transition $\Rightarrow$. The definition of conformance is quite close to a standard bisimulation relation, except for the use of the hiding operator at the collaboration level. Specifically, the hiding is used to ignore all additional behaviors in the collaboration that are not explicitly expressed, and hence regulated, in the choreography. In this way, even if a collaboration performs some additional communications, if it is able to (bi)simulate with the given choreography, they do conform. The different communication models defined in the semantics of choreographies and collaborations significantly affects the conformance checking. Considering that collaborations rely on an asynchronous communication model, one may think that the collaboration actions to be observed should be the sending ones (as, e.g., in the labeled bisimulation introduced for asynchronous π-calculus [31]). However, our aim here is to check the conformance with respect to a choreography model that, at an higher level of abstraction, prescribes that all interactions are synchronous. Since the non-blocking nature of message sending in the asynchronous collaborations may generate misalignment with the message exchanges defined in the synchronous choreography, we focus only on the message reception in the collaboration (see rules $C\text{-}EventG$, $C\text{-}TaskRcv$ and $C\text{-}InterRcv$ in Fig. 7). This guarantees the comparison of a choreography communication with the effective completion of the message exchange, defined by a message reception, in the collaboration.

BBC guarantees that the collaboration takes decisions, concerning the execution flow, exactly as what is specified in the choreography. Sometimes this condition may result too restrictive and the modeler would prefer to adopt a weaker relation. To this aim, in our work we also introduced the more relaxed TBC relation. Intuitively, in this case two models conform if and only if they can perform exactly the same weak sequences of actions. In the definition below, we deem a label to be *visible* if it is of the form $o_1 \rightarrow o_2 : m$. Notationally, the transition $\langle Ch, \sigma\rangle \stackrel{s}{\Rightarrow} \sigma'$, where s is a sequence of visible labels $l_1 l_2 \ldots l_n$, denotes the sequence $\langle Ch, \sigma\rangle \stackrel{l_1}{\Longrightarrow} \langle Ch, \sigma_1\rangle \stackrel{l_2}{\Longrightarrow} \langle Ch, \sigma_2\rangle \ldots \stackrel{l_n}{\Longrightarrow} \langle Ch, \sigma'\rangle$ of weak transitions. Transition $\langle C, \sigma, \delta\rangle \stackrel{s}{\Rightarrow} \langle \sigma', \delta'\rangle$ is similarly defined.

Definition 2. - **Trace-Based Conformance Relation.** A choreography $\langle Ch, \sigma_{ch}\rangle$ and a collaboration $\langle C, \sigma_c, \delta\rangle$ *trace conform* if, given $C' = C/(labels(C) \setminus labels(Ch))$, for any sequence s of visible labels it holds:

- $\langle Ch, \sigma_{ch}\rangle \stackrel{s}{\Rightarrow} \sigma'_{ch}$ implies $\langle C', \sigma_c, \delta\rangle \stackrel{s}{\Rightarrow} \langle \sigma'_c, \delta'\rangle$ for some σ'_c and δ';
- $\langle C', \sigma_c, \delta\rangle \stackrel{s}{\Rightarrow} \langle \sigma'_c, \delta'\rangle$ implies $\langle Ch, \sigma_{ch}\rangle \stackrel{s}{\Rightarrow} \sigma'_{ch}$ for some σ'_{ch}.

The TBC relation guarantees that the collaboration is able to produce the same sequences of messages of the choreography, and vice versa, without controlling presence of deadlock states and distinguishing different decision points and non-determinism forms. Concerning this latter point, BBC can recognize dominated non-determinism, where a participant (non-deterministically) takes a decision using a XOR gateway and the other behaves accordingly, from non-dominated non-determinism, based on a race condition among the messages managed by an event-based gateway. As it usually happens for these classes of behavioral relations, models that conform according BBC also conform according to TBC.

Conformances at work. To demonstrate into practice the characteristics of the conformance relations focusing on the management of non-determinism, we test them considering various model fragments in a simple scenario, where two participants are involved. Table 1 depicts in the rows the three gateways (i.e., in order, parallel, exclusive, and event-based) that can be used in a choreography model, and in the columns the possible combinations of participants in collaborations (i.e., in order, parallel-parallel, parallel-event, parallel-exclusive, exclusive-event, and exclusive-exclusive).

Checking all possible conformance combinations, we realize that for each choreography we have at least one BBC implementation. In particular, the choreography A can be implemented by a bisimilar collaboration 1, the choreography B is bisimilar to collaborations 3-4, and C to the collaboration 2. This last case results from the non-dominated non-deterministic behavior characterizing the event-based gateway, which is properly implemented by a sender using an AND gateway and not a XOR gateway (as in collaboration 4). This becomes clearer if we generalize the sender using an AND gateway to different senders in a race condition, each one sending a single message.

The conformance checking results reported in the table show in detail the differences between BBC and TBC. The modeler can select the more appropriate relation that fits more his needs, taking into account that BBC provides more guarantees on the correct behavior between the two models, while TBC ensures only that both models produce the same sequences of messages.

	(A)	(B)	(C)
(1)	$BBC:$ ✓ $TBC:$ ✓	$BBC:$ × $TBC:$ ×	$BBC:$ × $TBC:$ ×
(2)	$BBC:$ × $TBC:$ ×	$BBC:$ × $TBC:$ ✓	$BBC:$ ✓ $TBC:$ ✓
(3)	$BBC:$ × $TBC:$ ×	$BBC:$ ✓ $TBC:$ ✓	$BBC:$ × $TBC:$ ✓
(4)	$BBC:$ × $TBC:$ ×	$BBC:$ ✓ $TBC:$ ✓	$BBC:$ × $TBC:$ ✓
(5)	$BBC:$ × $TBC:$ ×	$BBC:$ × $TBC:$ ✓	$BBC:$ × $TBC:$ ✓

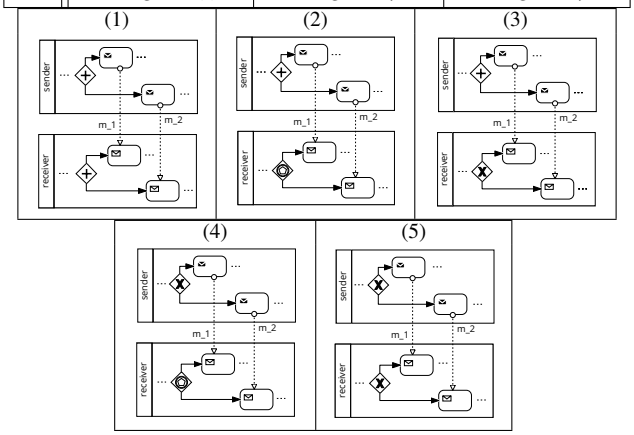


TABLE 1. CONFORMANCE BETWEEN COLLABORATION AND CHOREOGRAPHY.

6. C^4 Supporting Tool

The C^4 formal framework presented so far is implemented as a Java tool[1] supporting modelers in automatically checking whether a collaboration conforms to a prescribed choreography. A distinctive aspect of the tool is that modelers do not need to know the formal notions underlying its functionalities. The tool was developed as a stand-alone solution, but it is also available as a service accessible, or integrable as a plug-in in existing modeling tools, through a RESTful interface. In this regards, even if we support any BPMN modeling environments, we widely tested its compatibility with Eclipse BPMN Modeling, Camunda and Signavio. Fig. 9 depicts the internal components of the C^4 tool and the interfaces with the modeler. Specifically, C^4 takes as input a choreography and a collaboration in the *.bpmn* format. Input models can be generated by the modeler using different BPMN modeling environments, or can be retrieved from public repositories. The input files can be loaded in the C^4 tool using a dedicated GUI (Fig. 10(a)). The modeler can load multiple files, both for choreographies and collaborations. The inclusion of this feature was driven by the necessity of checking the conformance between different versions of the same model, avoiding to load each time a new file. The loaded models are listed in two text-areas and by clicking in one of them, a graphical preview of the model is showed automatically. Once the input files have been selected the C^4 tool parses the models and generates the corresponding LTS graphs for both the choreography and the collaboration. The parsing of the input files is based on the Camunda API. The API has been used as it is for the collaboration models, while it has been extended (to include choreography tasks) for the choreographies. The LTSs are computed by means a Java implementation of the direct semantics defined in Sec. 4.

Once generated the LTSs, C^4 saves the results in two *.aut* files [32] and automatically open the BPMN Checker (Fig. 10(b)) where the desired conformance relations can be checked. The conformance checking is achieved by resorting to the mCRL2 equivalence checker [4], that is fully integrated in the C^4 tool. Notably, the standard bisimulation and trace equivalences supported by mCRL2 can be directly used at this stage, as all the specific characteristics of our conformance relations (e.g., the use of hiding), have been already taken into account during the LTS generation. The verification results are visualized using a green/red indicator that states the satisfiability/unsatisfiability of the conformance relation. In case of dissatisfaction, C^4 returns back a counterexample. Notably, the usage of the *.aut* format for storing the LTS graphs enable the integration with other checkers that could be used in the future for further analysis (e.g., stochastic verification).

C^4 tool at work on the booking example. To check if the booking collaboration in Fig. 3 can be considered a valid implementation of the choreography in Fig. 2, we used the C^4 tool with both BBC and TBC relations. These analyses returned violations for both conformance relations. In particular, considering TBC the following counterexample is produced:

c→bs:login, c→bs:request, bs→c:reply, c→bk:pay

where c, bs and bk stand for the customer, booking system and bank organization names, respectively. This trace is allowed by the collaboration and not by the choreography. It shows that the expected flow 'booking and then payment' is not respected in the collaboration, which indeed permits to pay the reservation before booking it. This undesired behavior is due to the non-blocking nature of the collaboration sending task, which permits the customer to send the payment immediately after the booking request, without waiting for any acknowledgment from the booking system. This would not be a problem in case of a collaboration with only two participants, or more generally when the receiver of the two messages is the same participant, since the order in which the messages are processed is managed by the behavior of the receiver. Instead, in our running scenario the *book* and the *pay* messages are received by two different participants. The collaboration in Fig. 3 cannot guarantee the correct order in which the messages are handled. To solve such an issue, we can revise the collaboration in

1. The C^4 tool is available at https://goo.gl/ZWDMbs

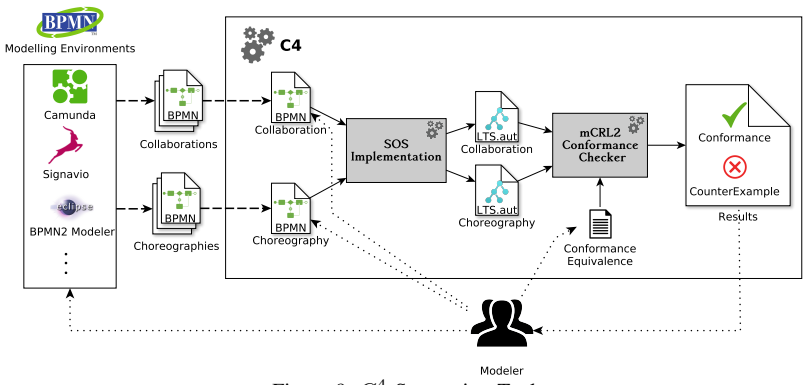
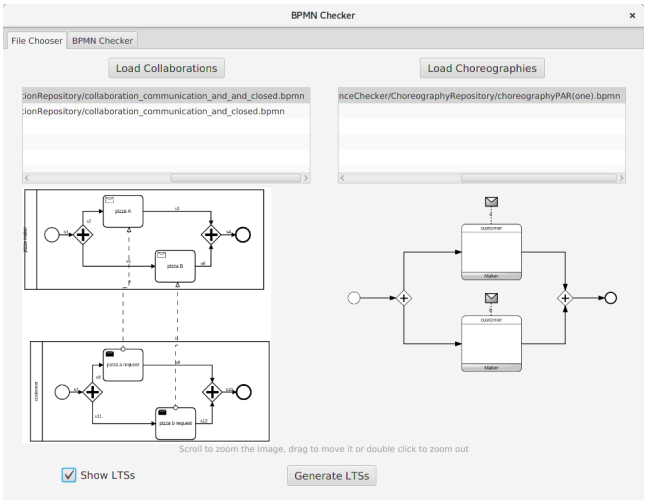
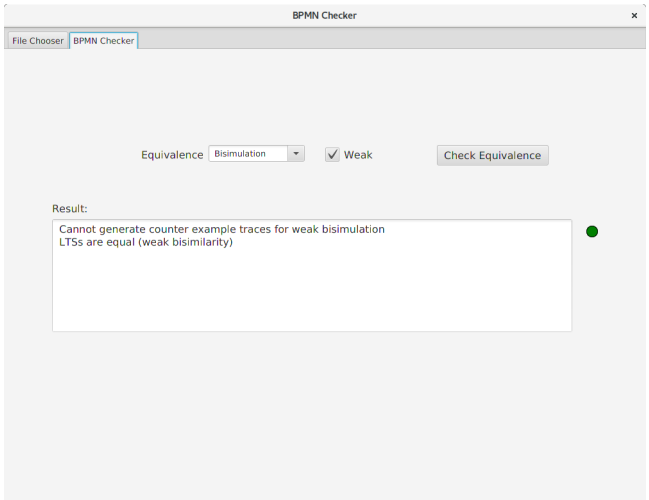


Figure 9. C^4 Supporting Tool.



(a) Models Selection.



(b) Conformance Checker.

Figure 10. C^4 User Interface.

Fig. 11 adding an *ack* message between the *book* and *pay* message exchanges. This guarantees that the booking phase completes before giving to the customer the possibility to proceed with the payment. By checking again

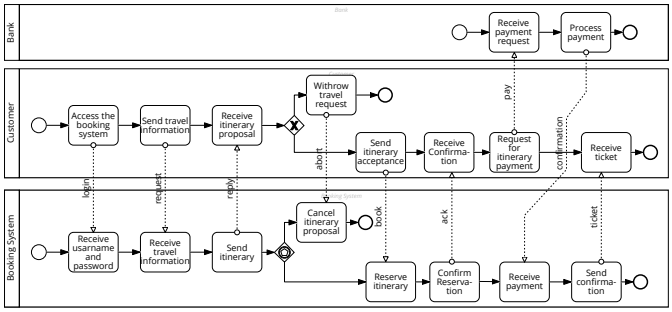


Figure 11. Repaired Collaboration.

the conformance between the revised collaboration and the choreography, in Fig. 2 C^4 tool states that the collaboration is a correct implementation of the choreography, as the two models conform according to both TBC and BBC. Notice that the *ack* message in the collaboration does not lead to a conformance violation, because it is not included in the choreography and, hence, is transformed into a τ by the hiding operator. In fact, this message exchange is a low-level implementation aspect of the collaboration necessary to conform with the given choreography.

7. Conclusions and Future Work

Here we propose a theory for checking conformance between BPMN choreographies and collaborations. We started defining the formal operational semantics for choreographies and collaborations, following step-by-step the behavior described by the BPMN standard, and on top of that we defined the notion of conformance in terms of a trace-based and a bisimulation-based relation. As proof of concept, the semantics and conformance relations have been implemented and tested on the C^4 tool using a running example.

In the next future we intend to further develop the C^4 tool, integrating it in different platforms and providing a user-friendly support for counterexamples, so to enlarge the possible interest on the tool.

References

[1] G. Castagna, M. Dezani, and L. Padovani, "On global types and multi-party sessions," in *FMOODS/FORTE*, ser. LNCS, vol. 6722. Springer, 2011, pp. 1–28.

[2] M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella, "Verifying the conformance of web services to global interaction protocols: A first step," in *Formal Techniques for Computer Systems and Business Processes*, ser. LNCS. Springer, 2005, vol. 3670, pp. 257–271.

[3] OMG, "Business Process Model and Notation (BPMN V 2.0)," 2011.

[4] J. F. Groote and M. R. Mousavi, *Modeling and analysis of communicating systems*. MIT press, 2014.

[5] R. Kazhamiakin and M. Pistore, "Choreography conformance analysis: Asynchronous communications and information alignment," in *International Workshop on Web Services and Formal Methods*, ser. LNCS, vol. 6. Springer, 2006, pp. 227–241.

[6] S. Basu and T. Bultan, "Choreography conformance via synchronizability," in *World wide web*. ACM, 2011, pp. 795–804.

[7] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro, "Choreography and orchestration conformance for system design," in *Coordination*, ser. LNCS, vol. 4038. Springer, 2006, pp. 63–81.

[8] Z. Qiu, X. Zhao, C. Cai, and H. Yang, "Towards the theoretical foundation of choreography," in *World Wide Web*. ACM, 2007, pp. 973–982.

[9] T. Bultan, C. Ferguson, and X. Fu, "A tool for choreography analysis using collaboration diagrams," in *International Conference on Web Services. ICWS*. IEEE, 2009, pp. 856–863.

[10] E. Teicholz *et al.*, *Technology for Facility Managers: The Impact of Cutting-edge Technology on Facility Management*. John Wiley & Sons, 2012.

[11] H. Vincent, V. Issarny, N. Georgantas, E. Francesquini, A. Goldman, and F. Kon, "Choreos: scaling choreographies for the internet of the future," in *Middleware'10 Posters and Demos Track*. ACM, 2010, pp. 8–10.

[12] M. Autili, P. Inverardi, A. Perucci, and M. Tivoli, "Synthesis of distributed and adaptable coordinators to enable choreography evolution," in *Software Engineering for Self-Adaptive Systems 3*, ser. LNCS, vol. 9640. Springer, 2017, pp. 1–25.

[13] G. Salaün and T. Bultan, "Realizability of choreographies using process algebra encodings," in *International Conference on Integrated Formal Methods*, ser. LNCS, vol. 5423. Springer, 2009, pp. 167–182.

[14] G. Salaun, L. Bordeaux, and M. Schaerf, "Describing and reasoning on web services using process algebra," *International Journal of Business Process Integration and Management*, vol. 1, no. 2, pp. 116–128, 2006.

[15] S. Tasharofi and M. Sirjani, "Formal modeling and conformance validation for ws-cdl using reo and casm," *Electronic Notes in Theoretical Computer Science*, vol. 229, no. 2, pp. 155–174, 2009.

[16] H. N. Nguyen, P. Poizat, and F. Zaïdi, "A symbolic framework for the conformance checking of value-passing choreographies," in *ICSOC*, ser. LNCS, vol. 2012. Springer, 2012, pp. 525–532.

[17] P. Poizat and G. Salaün, "Checking the realizability of BPMN 2.0 choreographies," in *Symposium on Applied Computing*. ACM, 2012, pp. 1927–1934.

[18] C. Molina-Jimenez and S. Shrivastava, "Establishing conformance between contracts and choreographies," in *Business Informatics (CBI)*. IEEE, 2013, pp. 69–78.

[19] M. Güdemann, P. Poizat, G. Salaün, and L. Ye, "Verchor: a framework for the design and verification of choreographies," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 647–660, 2016.

[20] A. Dumont, "A lotos nt library for modelisation, analysis, and validation of distributed systems," *Internship Report, Ecole Nationale Suprieure d'Informatique et Mathmatiques Appliques de Grenoble (ENSIMAG), June 2012*.

[21] D. Reißner, R. Conforti, M. Dumas, M. L. Rosa, and A. Armas-Cervantes, "Scalable conformance checking of business processes," in *International Conference On Cooperative Information Systems*, 2017, pp. 607–627.

[22] Anonymous Authors, "Omitted due to double blind reviewing."

[23] R. M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN," *Information and Software Technology*, vol. 50, no. 12, pp. 1281–1294, 2008.

[24] R. Laue and J. Mendling, "The Impact of Structuredness on Error Probability of Process Models," in *Information Systems and e-Business Technologies*, ser. LNBIP. Springer, 2008, vol. 5, pp. 585–590.

[25] A. Polyvyanyy and C. Bussler, "The structured phase of concurrency," in *Seminal Contributions to Information Systems Engineering*. Springer, 2013, pp. 257–263.

[26] B. Kiepuszewski, A. H. M. ter Hofstede, and C. J. Bussler, "On structured workflow modelling," in *International Conference on Advanced Information Systems Engineering*, ser. LNCS, vol. 1789. Springer, 2000, pp. 431–445.

[27] A. Polyvyanyy, L. García-Bañuelos, and M. Dumas, "Structuring acyclic process models," *Information Systems*, vol. 37, no. 6, pp. 518–538, 2012.

[28] A. Polyvyanyy, L. Garcia-Banuelos, D. Fahland, and M. Weske, "Maximal Structuring of Acyclic Process Models," *The Computer Journal*, vol. 57, no. 1, pp. 12–35, 2014.

[29] M. Muehlen and J. Recker, "How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation," in *Advanced Information Systems Engineering*, ser. LNCS. Springer, 2008, vol. 5074, pp. 465–479.

[30] R. De Nicola, "A gentle introduction to process algebras," 2014.

[31] R. M. Amadio, I. Castellani, and D. Sangiorgi, "On bisimulations for the asynchronous pi-calculus," *Theor. Comput. Sci.*, vol. 195, no. 2, pp. 291–324, 1998.

[32] J. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu, "Cadp a protocol validation and verification toolbox," in *International Conference on Computer Aided Verification*, ser. LNCS, vol. 1102. Springer, 1996, pp. 437–440.