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

Flavio Corradini, Andrea Morichetta, Andrea Polini, Barbara Re, Francesco Tiezzi
University of Camerino, School of Science and Technology, Camerino, Italy

***Abstract*—The BPMN 2.0 standard is nowadays largely used to model distributed informative systems in both academic and industrial contexts. The notation makes possible to represent these systems from different perspectives. A local perspective, using *collaboration* diagrams, to describe the internal behaviour of each component of the systems, and a global perspective, using *choreography* diagrams, where the interactions between system components are highlighted without exposing their internal structure. In this paper, we propose a formal approach for checking conformance of collaborations, representing possible system implementations, with respect to choreographies, representing global constraints concerning components' interactions. In particular, we provide a direct formal operational semantics for both BPMN collaboration and choreography diagrams, and we formalise the conformance concept by means of two relations defined on top of the semantics. To support the approach into practice we have developed the C^4 tool. Its main characteristic is to make the exploited formal methods transparent to systems designers, thus fostering a wider adoption of them in the development of distributed informative systems. We illustrate the benefits of our approach by means of a simple, yet realistic, example concerning a traveling scenario.**

1. Introduction

Distributed informative systems are characterised by interacting components that agree on communication patterns. The OMG standard BPMN 2.0 [1] (in the following just BPMN) is more and more adopted by academia and industry as modelling language for these systems. This is mainly due to its graphical notation and capability of describing systems at different perspectives. In particular, a BPMN *collaboration* diagram describes the implementation of each single component, possibly deployed and managed by different organizations, in terms of exchanged messages and internal behaviour, while a BPMN *choreography* diagram provides a global specification focusing on component interactions.

In such a setting organizations that are willing to cooperate can refer to, possibly predefined, choreography specifications detailing how different parties should interact to reach specific objectives. On the other hand involved organizations can put in place the cooperation deploying software systems behaving according to specific internal processes. The integration of such processes leads to the so called collaboration that nevertheless should show a behaviour related to the global specification. Indeed in this distributed setting, the *conformance* of a given collaboration with respect to a pre-established choreography is then crucial. This permits to ensure that the system components are able to successfully collaborate without invalidating the communication constraints imposed by the global specification. Despite the effort devoted to the study of this concept in the general context of service-oriented systems [2]–[7], there is a lack of approaches and tools supporting the conformance checking between collaboration and choreography models when the BPMN notation is considered. The effects of this issue are intensified in those contexts, such as the industrial ones, where system designers are not familiar with formalisms and verification techniques but are accustomed to standard graphical notations like BPMN.

To fill this gap we provide in this paper **a novel solution for directly checking the conformance of BPMN collaborations with respect to BPMN choreography models without using intermediate languages**. Our approach is based on a direct semantics describing the behaviour of both models, taking into account the specificities of the BPMN standard when used to model distributed systems (e.g., asynchronous communication among components). More specifically, the operational semantics associates to each BPMN model a Labelled Transition System (LTS) formally describing its behaviour. The conformance between a collaboration and a choreography thus boils down to compare their LTSs according to behavioural relations. In particular, we rely on a conformance relation (based on bisimulation [8, Sec. 5]) that is sensitive to deadlocks and different forms of non-determinism, and on another relation (based on traces [8, Sec. 9.4]) that is relaxed on this respect. These relations allow the system designer to find the desired tradeoff between the strength of the properties ensured by the system and the breadth of choice among available system components.

The proposed theoretical framework has been implemented as the C^4 *(Collaboration vs Choreography Conformance Checker for BPMN)* tool. It uses standard input formats for the BPMN models, thus enabling the interoperability with external BPMN modelling environments (e.g., Camunda, Signavio and Eclipse BPMN2 Modeler), hence permitting the systems designer to use the preferred one. It results that the usage of **the underlying formal methods are completely transparent to the system designer**, which

is our driving objective. Summing up, the major contributions of this paper is threefold: (i) the definition, and the Java implementation, of a formal operational semantics for BPMN collaborations and choreographies; (ii) the definition, and implementation, of two conformance relations; (iii) the implementation of the C^4 tool supporting our conformance checking solution.

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

2. Related Works

To clarify how the proposed approach advances the state of the art here we relate the distinctive aspects of our work to the literature.

On the Choice of the Modelling Notation. A lot of effort [2]–[7] has been devoted by the research community in the past few years to study modelling languages for collaborations (e.g., the OASIS standard WS-BPEL [9]) and choreographies (e.g., the W3C standard WS-CDL [10]). However, more recently the focus of this study is shifting towards the OMG standard BPMN [1]. Indeed this notation is becoming one of the most prominent modelling languages for distributed information systems [11]. This is also testified by EU research projects (e.g. CHOReOS [12] and CHOReVOLUTION [13]), linking academia and industry, that have adopted BPMN as the reference modelling notation. The recent attention devoted to BPMN motivates our choice of selecting it as modelling language.

From Choreographies to Code. Using model-driven approaches to develop distributed systems, component stubs can be derived from a choreography model. For example, the authors of [14] provide a semi-automatic RESTful implementation of BPMN choreographies basing their methods on natural language analysis. Similarly, the authors of [15] propose an approach that permits to derive WS-BPEL processes from choreography models. However, in these works no formal guarantees are provided to ensure that the developed system conforms to the prescribed interaction strategy. On the other hand, purely theoretical works (e.g., those based on global/local session types [16]) formally ensure a correct-by-construction derivation, but they only deal with simple formalisms for describing choreographies and collaborations, which are very far from the notations used in practice, like BPMN. In our work we do not aim at deriving components from a choreography, but we compare the behaviour resulting from a collaboration of components with respect to a given choreography. In particular, we figure out a software development/integration context in which different organizations let their informative systems cooperate in order to reach the objectives of a choreography.

Conformance. Our work focuses on the notion of conformance, which sometimes in the literature is referred with different terminologies depending on the context, like compliance or compatibility. Most works in the literature [17]–[24] aim at comparing processes forming a collaboration with respect to domain-specific regulations and rules. These works, however, express these global rules by means of logics or other formalisms, rather than using the standard choreography notation provided by BPMN. Thus, they require the system designer to directly deal with formal technicalities of the used checking technique. Other works [25]–[27], instead, only focus on a local view. They apply behavioural equivalences to find processes with compatible behaviours, according to the actions they can execute. However, these works completely lack a global perspective. Our work differs from the ones mentioned above since it fully relies on BPMN models, at both global and local level, and it is implemented in a conformance checking tool.

Direct Semantics. In the literature many proposals are based on BPMN semantics given in terms of a translation to other languages or formalisms. These semantics differ for the target language of the translation: process algebras [28]–[30], more complex formal specification languages (e.g., LOTOS) [31]–[34], transition-based models (e.g., Petri Nets) [35]–[38], or session types [39].

In our work we rely on a direct semantics for both collaboration and choreography models. Our semantics is given in terms of features and constructs of BPMN, rather than in terms of their low-level encoding into another formalism that is equipped with its own syntax and semantics. This permits to formalise the BPMN features as close as possible to their definition in the standard specification, without any bias from the use of another formalism, thus ensuring a more effective verification. The direct semantics proposed in this paper is inspired by [40], and by its extended version in [41], but its technical definition is significantly different. In particular, configuration states are here defined according to a global perspective, and the formalisation now includes choreography diagrams, which were overlooked in the previous semantics definition.

Concerning conformance checking, our direct approach permits to focus on specific features of BPMN that would be ignored by using available Petri Nets-based semantics. In particular, in the BPMN to Petri Nets translation reported in [35], it is not possible to distinguish different types of non-determinism resulting from event-based or exclusive gateways. Indeed these two BPMN elements have different effects: the event-based gateway produces non-dominated non-determinism (roughly, no one in the model has complete knowledge on the decision that will be taken), while the exclusive gateway produces dominated non-determinism (roughly, the decision is taken by one party and followed by the others). Our approach, instead, permits to distinguish the dominated and non-dominated non-determinism produced by the gateways, as prescribed by the BPMN standard. This is somehow similar to [28], [42], which rely on the concept of internal and external choice defined in the CSP process

algebra. Notably the different kinds of non-determinism have an impact on the conformance relations, as detailed in Table 1.

Tool Support. A distinctive contribution of our work is the development of the C^4 tool that incorporates all the defined formal concepts, like the BPMN collaboration/choreography semantics, and the related conformance relations. The tool makes such features easily accessible to non-expert users by means of a GUI. A similar endeavour has led to the realisation of the VerChor tool [34]. However, VerChor objective is rather different since its purpose to use conformance to check the realisability of a set of peers obtained from a projection of a given choreography. The analysis tool VBPMN, proposed in [43], aims instad at checking properties of business processes using the model checker CADP. This is achieved, on the one hand, by transforming BPMN models into PIF ones and then into LNT process algebraic descriptions, and, on the other hand, by generating specific SVL verification scripts from UI inputs. In comparison to VBPMN, C^4 relies on a direct semantics of BPMN. Moreover, C^4 enables conformance checking of collaborations w.r.t. choreographies, while VBPMN only deals with the analysis of single processes, thus it is not suitable to support the development of distributed information systems.

3. BPMN 2.0 Overview

This section presents the relevant elements of choreography and collaboration diagrams we use in the paper, and introduces a scenario that will be used as a running example.

The BPMN Standard. Fig.1.b depicts the most used modelling elements that can be 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 message 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 to specify the execution flow.

In a collaboration diagram, also the elements in Fig.1.a can be included. **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 during the flow of the process, 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 exchanges between two or more participants. This is done by means of **Choreography Tasks** in Fig.1.c. 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).

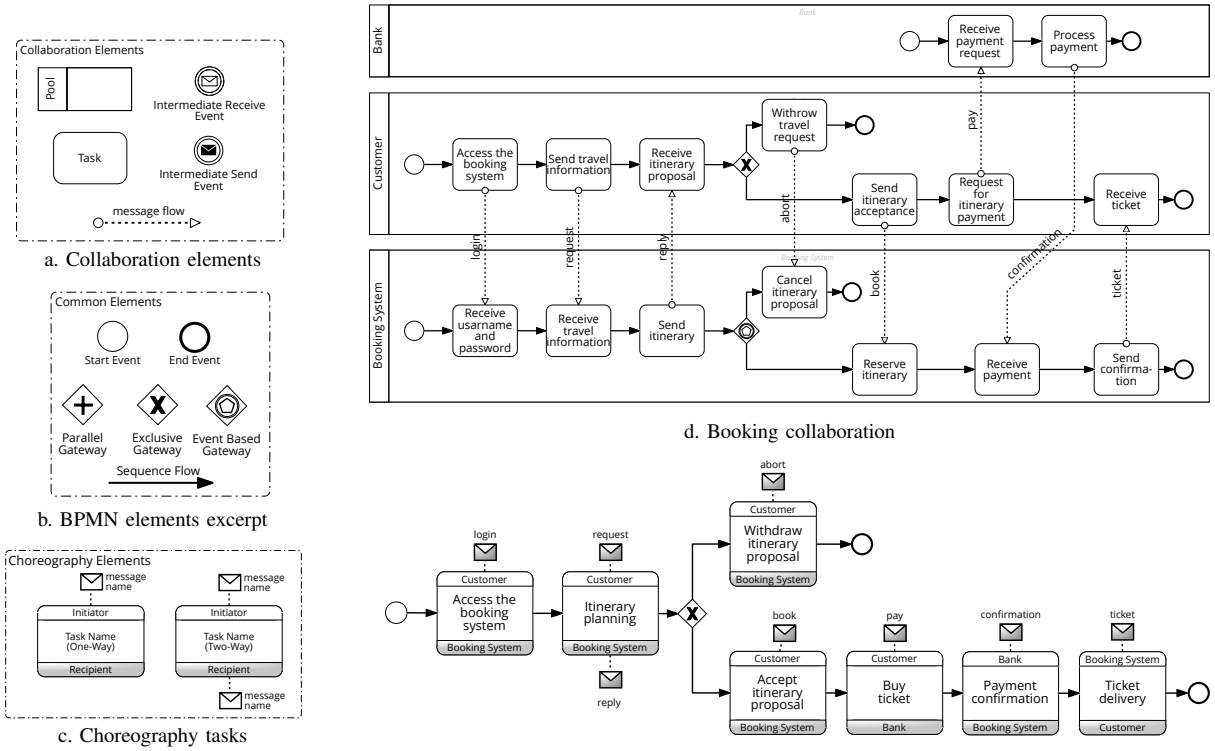
Running Example. A collaboration and a choreography model regarding a booking system are here presented.

Collaboration Example. The collaboration in Fig.1.d combines a customer and a bank that have to interact in order to book a travel. After the customer login into the booking system, she requests some travel information and she receives a proposal from the booking system. The customer then decides whether to withdraw or accept the proposal; this is represented by means of an 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.

Choreography Example. The choreography in Fig.1.e combines the work-activities of the same participants of the collaboration. 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 triggered 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. The latter completes the procedure by sending the ticket to the customer.

4. Formal Framework: Semantics

This section presents our formalization of the BPMN semantics at the base of the proposed framework. Specifically, we first summarize its distinctive aspects in relation to the BPMN modeling principles, and then we illustrate its formal definition.



a. Collaboration elements

b. BPMN elements excerpt

c. Choreography tasks

d. Booking collaboration

e. Booking choreography

Figure 1. BPMN 2.0 Elements and Booking Example.

Linguistic Aspects and Design Choices. Concerning 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 [1, 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 [1, p. 315], a choreography task completes when the receiver participant reads the message. 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. This means that a message sent by one participant is enqueued by the receiving one, which can then consume and process

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

Figure 2. Syntax of BPMN Choreography Structures.

it subsequently, while the sender is free to proceed with its execution. This reflects the distributed nature of collaborations. The use of two different communication models also impacts on the definition of the conformance relations as illustrated in Sec. 5.

Semantics of BPMN Choreographies. To enable a formal treatment of a BPMN choreography we defined a Backus Normal Form (BNF) syntax of its model structure (Fig. 2). In 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. With respect to the graphical notation, the textual one is more manageable for supporting the formal definition of the semantics and its implementation.

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 the 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.

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. 3, formalizes the execution of a choreography in terms of marking evolution and message exchanges. 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 configurations of transitions. Thus, a transition $\langle Ch, \sigma \rangle \xrightarrow{l} \langle Ch, \sigma' \rangle$ is written as $\langle Ch, \sigma \rangle \xrightarrow{l} \sigma'$.

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

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

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

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

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

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

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

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

$$\frac{\langle Ch_1, \sigma \rangle \to \sigma'}{\langle Ch_1 | Ch_2, \sigma \rangle \to \sigma'} (\mathit{Ch\text{-}Int_1}) \qquad \frac{\langle Ch_2, \sigma \rangle \to \sigma'}{\langle Ch_1 | Ch_2, \sigma \rangle \to \sigma'} (\mathit{Ch\text{-}Int_2})$$

Figure 3. Choreography Semantics (τ labels are omitted).

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.

We describe some rules in Fig. 3, the meaning of the others can be easily deduced. In particular rule $\mathit{Ch\text{-}Start}$ starts the execution of a choreography 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 $\mathit{Ch\text{-}AndJoin}$ 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 $\mathit{Ch\text{-}XorSplit}$ is applied when a token is available in the incoming edge of an XOR split gateway, the rule decrements the token in the incoming edge and increments the tokens in one of the outgoing edges. Rule $\mathit{Ch\text{-}Task}$ 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 $\mathit{Ch\text{-}Int_1}$ and $\mathit{Ch\text{-}Int_2}$ deal with interleaving.

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. 4. The non-terminal symbol C represents *Collaboration Structures*, while terminal symbols denote, as usual, the considered BPMN elements. The exchange of messages 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 two intermediate events, i.e. receiving (interRcv) and sending (interSnd).

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

Figure 4. 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 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 number 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. 5 (for the sake of presentation, we focus on rules concerning the exchange of messages). As usual, we omit the collaboration structure from the target configuration of transitions.

We now briefly comment on the operational rules. Rule $C\text{-}EventG$ is activated when there is a token in the incoming edge of an event-based gateway 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 message tokens in the meantime is decreased (i.e., a message from the corresponding queue is consumed). Rule $C\text{-}Task$ deals with simple tasks, acting as a pass through. Rule $C\text{-}TaskRcv$ is activated not only when there is a token

$$
\begin{array}{ll}
(C\text{-}EventG) & \\
\langle \mathsf{eventBased}(\mathsf{e}_i, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m}, \mathsf{e}_o) \cup M), \sigma, \delta \rangle & \sigma(\mathsf{e}_i) > 0, \\
\xrightarrow{\mathsf{o}_1 \to \mathsf{o}_2 : \mathsf{m}} \langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), dec(\delta, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \rangle & \delta(\mathsf{o}_1, \mathsf{o}_2, \mathsf{m}) > 0 \\
(C\text{-}Task) & \\
\langle \mathsf{task}(\mathsf{e}_i, \mathsf{e}_o), \sigma, \delta \rangle \to \langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), \delta \rangle & \sigma(\mathsf{e}_i) > 0 \\
(C\text{-}TaskRcv) & \\
\langle \mathsf{taskRcv}(\mathsf{e}_i, \mathsf{e}_o, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})), \sigma, \delta \rangle \xrightarrow{\mathsf{o}_1 \to \mathsf{o}_2 : \mathsf{m}} & \sigma(\mathsf{e}_i) > 0, \\
\langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), dec(\delta, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \rangle & \delta(\mathsf{o}_1, \mathsf{o}_2, \mathsf{m}) > 0 \\
(C\text{-}TaskSnd) & \\
\langle \mathsf{taskSnd}(\mathsf{e}_i, \mathsf{e}_o, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})), \sigma, \delta \rangle \to & \sigma(\mathsf{e}_i) > 0 \\
\langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), inc(\delta, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \rangle & \\
(C\text{-}InterRcv) & \\
\langle \mathsf{interRcv}(\mathsf{e}_i, \mathsf{e}_o, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})), \sigma, \delta \rangle \xrightarrow{\mathsf{o}_1 \to \mathsf{o}_2 : \mathsf{m}} & \sigma(\mathsf{e}_i) > 0, \\
\langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), dec(\delta, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \rangle & \delta(\mathsf{o}_1, \mathsf{o}_2, \mathsf{m}) > 0 \\
(C\text{-}InterSnd) & \\
\langle \mathsf{interSnd}(\mathsf{e}_i, \mathsf{e}_o, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})), \sigma, \delta \rangle \to & \sigma(\mathsf{e}_i) > 0 \\
\langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), inc(\delta, (\mathsf{o}_1, \mathsf{o}_2, \mathsf{m})) \rangle &
\end{array}
$$

Figure 5. Collaboration Semantics (excerpt of rules - τ labels are omitted).

in the incoming edge, like the one related to simple tasks, but also when there is a message to be consumed. Similarly, rule $C\text{-}TaskSnd$, instead of consuming, adds a message in the corresponding queue. Rule $C\text{-}InterRcv$ (resp. $C\text{-}InterSnd$) follow the same behavior of rule $C\text{-}TaskRcv$ (resp. $C\text{-}TaskSnd$).

5. Formal Framework: Conformance

This section discusses about the relations we propose for checking the conformance between choreographies and collaborations. We then present how they work in practice.

Bisimulation-Based and Trace-Based Conformance. Here we present the Bisimulation-Based Conformance (BBC) and the Trace-Based Conformance (TBC) relations we have defined. The two relations are inspired by well-established behavioural equivalences [8], 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.

At the collaboration level the definition of conformance requires the use of the hiding operator C/L, defined by the

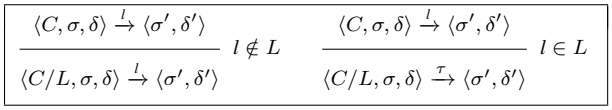


Figure 6. Hiding Operator.

rules in Fig. 6. This operator, as usual, transforms into τ all the actions in the set L, in order to consider them as internal actions in the conformance relation.

Definition 1. - **BBC 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 $\langle C, \sigma_c, \delta \rangle \xRightarrow{o_1 \to o_2 : m} \langle \sigma'_c, \delta' \rangle$ 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 $\langle Ch, \sigma_{ch} \rangle \xRightarrow{o_1 \to o_2 : m} \sigma'_{ch}$ 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 [44]). 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-EventG*, *C-TaskRcv* and *C-InterRcv* in Fig. 5). This permits to compare a choreography communication with the effective completion of the message exchange, defined by a message reception, in the collaboration.

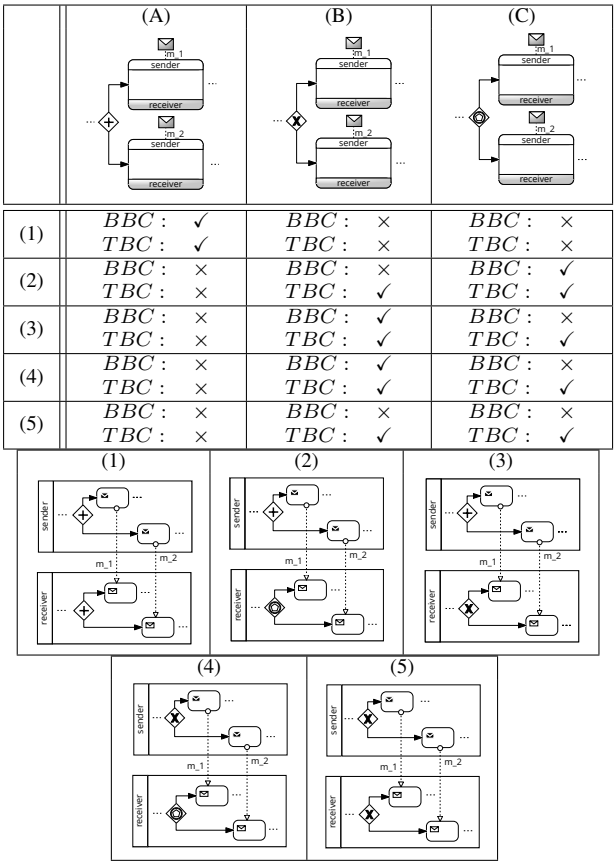
BBC guarantees that the collaboration takes decisions, concerning the execution flow, exactly as what is specified in the choreography. Sometimes this condition may be too restrictive and the system designer would prefer to adopt a weaker relation. To this aim, in our work we also introduce the more 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 $o_1 \to o_2 : m$. Notationally, the transition $\langle Ch, \sigma \rangle \overset{s}{\Rightarrow} \sigma'$, where s is a sequence of visible labels $l_1 l_2 \dots l_n$, denotes the sequence $\langle Ch, \sigma \rangle \overset{l_1}{\Longrightarrow} \langle Ch, \sigma_1 \rangle \overset{l_2}{\Longrightarrow} \langle Ch, \sigma_2 \rangle \dots \overset{l_n}{\Longrightarrow} \langle Ch, \sigma' \rangle$ of weak transitions. Transition $\langle C, \sigma, \delta \rangle \overset{s}{\Rightarrow} \langle \sigma', \delta' \rangle$ is similarly defined.

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

- $\langle Ch, \sigma_{ch} \rangle \overset{s}{\Rightarrow} \sigma'_{ch}$ implies $\langle C', \sigma_c, \delta \rangle \overset{s}{\Rightarrow} \langle \sigma'_c, \delta' \rangle$ for some σ'_c and δ';
- $\langle C', \sigma_c, \delta \rangle \overset{s}{\Rightarrow} \langle \sigma'_c, \delta' \rangle$ implies $\langle Ch, \sigma_{ch} \rangle \overset{s}{\Rightarrow} \sigma'_{ch}$ 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 to BBC also conform according to TBC.

Conformance at work. To demonstrate in 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).



	(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 COLLABORATIONS AND CHOREOGRAPHIES.

Checking all possible conformance combinations, we realize that for each considered 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 an XOR gateway (as in collaboration 4). This becomes clearer if we generalise the collaboration 2 by considering more than one sender 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 designer can select the more appropriate relation that fits more his needs, taking into account that BBC provides more guarantees on the correct behaviour between the two models, while TBC ensures only that both models produce the same sequences of messages.

6. C^4 Supporting Tool

The C^4 formal framework presented so far is implemented as a Java tool (available at *http://pros.unicam.it/c4/*) supporting system designers in automatically checking whether a collaboration conforms to a prescribed choreography. A distinctive aspect of the tool is that designers

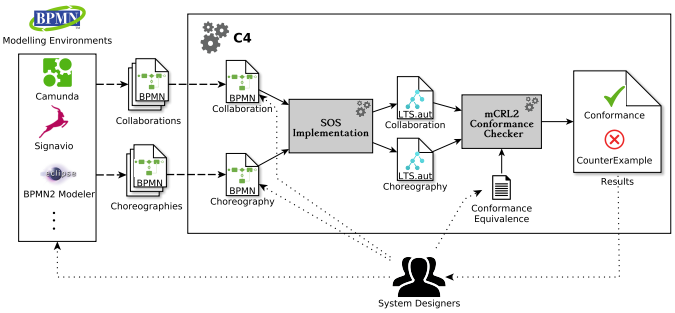
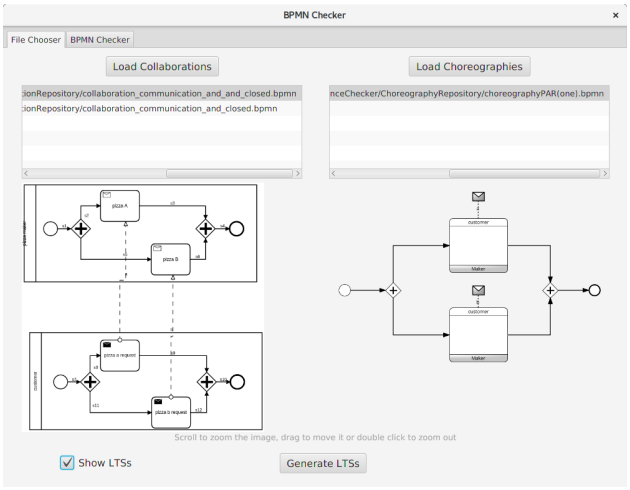


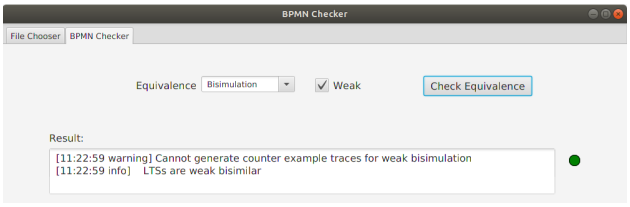
Figure 7. C^4 Supporting Tool.

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 through a RESTful interface, or integrable as a plug-in in existing modelling tools. In this regards, even if we support any BPMN modelling environments, we widely tested its compatibility with Eclipse BPMN Modelling, Camunda and Signavio. Fig. 7 depicts the internal components of the C^4 tool and the interfaces with the user. Specifically, C^4 takes as input a choreography and a collaboration in the *.bpmn* format. Input models can be generated by the designer 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. 8(a)). The user 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. Such 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 of a Java implementation of the direct semantics defined in Sec. 4.

Once the LTSs are generated, C^4 saves the results in two *.aut* files [45] and automatically opens the BPMN Checker (Fig. 8(b)) where the desired conformance relations can be checked. The conformance checking is achieved by resorting to the mCRL2 equivalence checker [46], 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



(a) Models Selection.



(b) Conformance Checker.

Figure 8. C^4 Graphical User Interface.

back a counterexample. Notably, the usage of the *.aut* format for storing the LTS graphs enables 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.1.d can be considered a valid implementation of the choreography in Fig.1.e, 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.1.d cannot guarantee

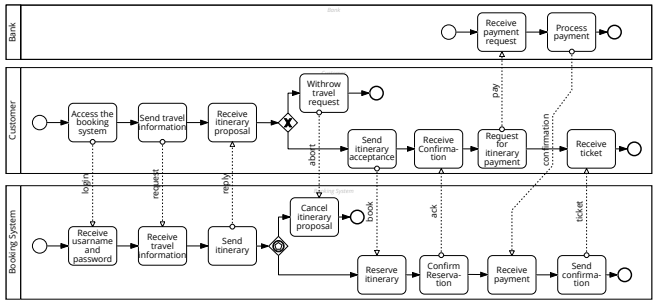


Figure 9. Repaired Collaboration.

the correct order in which the messages shoul be handled. To solve such an issue, we can revise the collaboration as shown in Fig.9, where an ack message between the $book$ and pay messages have been added. This guarantees that the booking phase completes before giving to the customer the possibility to proceed with the payment. By checking again the conformance between the revised collaboration and the choreography in Fig.1.e, 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. Notably, the added message is not foreseen by the choreography specification, nonetheless it permits to further constrain the collaboration so to obtain a behaviour satisfying both conformance relations. In such a case the hiding operator will substitute the ack message with a τ action in the composition of the various participants in the collaboration, and before checking the two conformance relations.

7. Conclusions and Future Work

In this paper we propose a novel approach for checking conformance between BPMN choreographies and collaborations. We define a formal operational semantics for choreographies and collaborations, following descriptions provided by the BPMN standard. On top of that, we define the notion of conformance in terms of a trace-based and a bisimulation-based relation. As proof of concept, the semantics and the conformance relations have been implemented and tested on the C^4 tool.

In the next future we intend to further develop the C^4 tool, integrating it in different platforms and providing a visual support for counterexamples, so to enlarge the usability of the tool. We also plan to use our approach in the process mining field, in order to check the conformance of collaborations derived from logs with given choreographies.

References

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

[2] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, 2007.

[3] 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.

[4] Y. Liu, S. Muller, and K. Xu, "A static compliance-checking framework for business process models," *IBM Systems Journal*, vol. 46, no. 2, pp. 335–361, 2007.

[5] M. El Kharbili, A. K. A. de Medeiros, S. Stein, and W. M. van der Aalst, "Business process compliance checking: Current state and future challenges." *MobIS*, vol. 141, pp. 107–113, 2008.

[6] M. Rouached, W. Fdhila, and C. Godart, "Web services compositions modelling and choreographies analysis," *International Journal of Web Services Research (IJWSR)*, vol. 7, no. 2, pp. 87–110, 2010.

[7] A. Martens, "On compatibility of web services," *Petri Net Newsletter*, vol. 65, no. 12-20, p. 100, 2003.

[8] R. Milner, *Communication and Concurrency*. Prentice-Hall, Inc., 1989, vol. 84.

[9] OASIS WSBPEL TC, "Web Services Business Process Execution Language Version 2.0," OASIS, Tech. Rep., April 2007.

[10] N. Kavantzas, D. Burdett, and G. Ritzinger, "Web Services Choreography Description Language version 1.0." W3C, Tech. Rep., 2004.

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

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

[13] 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.

[14] A. Nikaj, M. Weske, and J. Mendling, "Semi-automatic derivation of restful choreographies from business process choreographies," *Software & Systems Modeling*, pp. 1–14, 2017.

[15] B. Hofreiter and C. Huemer, "A model-driven top-down approach to inter-organizational systems: From global choreography models to executable BPEL," in *CEC/EEE*, IEEE. IEEE Computer Society, 2008, pp. 136–145.

[16] K. Honda, N. Yoshida, and M. Carbone, "Multiparty asynchronous session types," *J. ACM*, vol. 63, no. 1, pp. 9:1–9:67, 2016.

[17] G. Governatori, Z. Milosevic, and S. Sadiq, "Compliance checking between business processes and business contracts," in *EDOC*, IEEE. Computer Society, 2006, pp. 221–232.

[18] D. J. Mandell and S. A. McIlraith, "A bottom-up approach to automating web service discovery, customization, and semantic translation," in *WWW Workshop on E-Services and the Semantic Web*, 2003.

[19] D. Schumm, F. Leymann, Z. Ma, T. Scheibler, and S. Strauch, "Integrating compliance into business processes," *Multikonferenz Wirtschaftsinformatik 2010*, p. 421, 2010.

[20] A. Awad, G. Decker, and M. Weske, "Efficient compliance checking using bpmn-q and temporal logic," in *BPM*, ser. LNCS, vol. 5240. Springer, 2008, pp. 326–341.

[21] D. Knuplesch, M. Reichert, W. Fdhila, and S. Rinderle-Ma, "On enabling compliance of cross-organizational business processes," in *BPM*, ser. LNCS. Springer, 2013, vol. 8094, pp. 146–154.

[22] L. T. Ly, S. Rinderle-Ma, K. Göser, and P. Dadam, "On enabling integrated process compliance with semantic constraints in process management systems," *Information Systems Frontiers*, vol. 14, no. 2, pp. 195–219, 2012.

[23] D. Knuplesch and M. Reichert, "Ensuring business process compliance along the process life cycle," Universität Ulm, Tech. Rep., 2012.

[24] D. Knuplesch, M. Reichert, J. Mangler, S. Rinderle-Ma, and W. Fdhila, "Towards compliance of cross-organizational processes and their changes," in *BPM*, ser. LNCS, vol. 132. Springer, 2012, pp. 649–661.

[25] M. Weidlich, R. Dijkman, and M. Weske, "Behaviour equivalence and compatibility of business process models with complex correspondences," *The Computer Journal*, vol. 55, no. 11, pp. 1398–1418, 2012.

[26] J. Hidders, M. Dumas, W. M. van der Aalst, A. H. ter Hofstede, and J. Verelst, "When are two workflows the same?" in *Australasian symposium on theory of computing*, vol. 41. Australian Computer Society, Inc., 2005, pp. 3–11.

[27] R. Dijkman, M. Dumas, and L. García-Bañuelos, "Graph matching algorithms for business process model similarity search," in *BPM*, ser. LNCS, vol. 5701. Springer, 2009, pp. 48–63.

[28] 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.

[29] G. Salaün and T. Bultan, "Realizability of choreographies using process algebra encodings," in *iFM*, ser. LNCS, vol. 5423. Springer, 2009, pp. 167–182.

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

[31] 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.

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

[33] C. Molina and S. Shrivastava, "Establishing conformance between contracts and choreographies," in *CBI*. IEEE, 2013, pp. 69–78.

[34] 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.

[35] 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.

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

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

[38] F. Corradini, A. Polini, and B. Re, "Inter-organizational Business Process Verification in Public Administration," *Business Process Management Journal*, vol. 21, no. 5, 2015.

[39] 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.

[40] F. Corradini, A. Polini, B. Re, and F. Tiezzi, "An operational semantics of BPMN collaboration," in *FACS*, ser. LNCS, vol. 9539. Springer, 2015, pp. 161–180.

[41] F. Corradini, F. Fornari, A. Polini, B. Re, and F. Tiezzi, "A formal approach to modeling and verification of business process collaborations," *Science of Computer Programming*, vol. 166, pp. 35–70, 2018.

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

[43] A. Krishna, P. Poizat, and G. Salaün, "Vbpmn: Automated verification of bpmn processes," in *iFM*. Springer, 2017, pp. 323–331.

[44] 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.

[45] J. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu, "CADP a protocol validation and verification toolbox," in *CAV*, ser. LNCS, vol. 1102. Springer, 1996.

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