Incremental Construction of Realizable Choreographies

Sarah Benyagoub, Meriem Ouederni, Yamine Aït-Ameur, Atif Mashkoor

Incremental Construction of Realizable Choreographies

Sarah Benyagoub[1,2], Meriem Ouederni[2], Yamine Aït-Ameur[2](✉), and Atif Mashkoor[3]

[1] University of Mostaganem, Mostaganem, Algeria
benyagoub.sarah@univ-mosta.dz

[2] INP-ENSEEIHT/IRIT, Université de Toulouse, Toulouse, France
{ouederni,yamine}@enseeiht.fr

[3] SCCH GmbH and Johannes Kepler University, Linz, Austria
atif.mashkoor@scch.at, atif.mashkoor@jku.at

Abstract. This paper proposes a correct-by-construction method to build realizable choreographies described using conversation protocols (*CPs*). We define a new language consisting of an operators set for incremental construction of CPs. We suggest an asynchronous model described with the Event-B method and its refinement strategy, ensuring the scalability of our approach.

Keywords: Realisability · Conversation protocols
Correct-by-construction method proof and refinement · Event-B

1 Introduction

Distributed systems are pervasive in areas like embedded systems, Cyber Physical systems, medical devices and Web applications. In a top-down design of such systems, the interaction among peers is usually defined using a global specification called conversation protocols (CP), aka choreography in SOC [9]. These CPs specify interactions among peers as the allowed sequences of sent messages.

A main concern, already addressed by research community, is the verification of CP realizability i.e., verification whether *there exists a set of peers whose composition generates the same sequences of sending messages as specified by the CP*. Considering asynchronous communication, this realizability problem is undecidable in general [8] due to possible ever-increasing queuing mechanism and unbounded buffers. The work of [5] proposed a necessary and sufficient condition for verifying whether a CP can be implemented by a set of peers communicating asynchronously using FIFO buffers with no buffer sizes restrictions. This

The research reported in this paper has been partly supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

https://doi.org/10.1007/978-3-319-77935-5_1

work solves the realizability issue for a subclass of asynchronously communicating peers (synchronizable systems) *i.e.*, systems composed of interacting peers behaving equivalently either with synchronous or asynchronous communication.

A CP is *realizable* if there exists a set of peers implementing this CP, *i.e.*, the peers send messages to each other in the same order as the CP does, and their composition is synchronizable. In [5], checking CP realizability applies three steps: (i) *peer projection* from CP; (ii) checking *synchronizability;* and (iii) checking *equivalence* between CP and its distributed system obtained after projection.

The work given in [5] relies on model checking for systems with reasonable sizes (*i.e.*, number of states, transitions and communicating peers). This verification procedure is global and a posteriori. It considers the whole CP and its projection, and does not handle compositional verification.

This paper proposes a *compositional* and *incremental* formal verification procedure that scales to systems of arbitrary sizes. It promotes a top-down design of realizable CPs following a correct-by-construction method which decreases the complexity of the verification task and supports real-world complex systems. We define a compositional language using an algebra of operators (sequence, branching, and loop). From an initial basic CP, we inductively (incrementally) build a realizable CP by composing other realizable ones, using these composition operators while preserving realizability [5] w.r.t identified conditions. The informal definition of these operators were originally introduced in [6,7] the feasibility of the approach on toy case studies is shown. [6,7] did not give the formal proof of correctness of realizability preservation of the defined operators. Consequently, in this paper, we provide a correctness support for the results sketched in [6,7]. An inductive proof, based on realizability invariant preservation, is set up with Event-B [2] on Rodin [19] platform. Refinement is used to decompose this invariant in order to ease the proof and development processes. The generic model we define is scalable and its parameters have arbitrary values (*i.e.*, numbers of peers, buffer sizes, number of states and transitions can take any value in their corresponding sets of possible values). Furthermore, this model can be instantiated to describe any CP by incremental application of the composition operators we defined.

In the remainder, Sect. 2 introduces the formal definitions and the background our proposal relies on. Section 3 presents the set of composition operators together with the set of identified sufficient conditions that ensure realizability of the built CPs. The formal Event-B development based on the refinement strategy we have set up is shown in Sect. 4. Finally, Sect. 5 overviews related work Sect. 6 concludes this work.

2 Background and Notations

2.1 Model

We use labeled transition systems (LTSs) for modeling CP and peers included in that specification. This model defines messages order being sent in CP.

Definition 1 (Peer). *A peer is an LTS $\mathcal{P} = (S, s^0, \Sigma, T)$ where S is a finite set of states, $s^0 \in S$ is the initial state, $\Sigma = \Sigma^! \cup \Sigma^? \cup \{\tau\}$ is a finite alphabet partitioned into a set of send messages, receive messages, and the internal action, and $T \subseteq S \times \Sigma \times S$ is a transition relation.*

We write $m!$ for a send message $m \in \Sigma^!$ and $m?$ for a receive message $m \in \Sigma^?$. We use the symbol τ for representing internal activities. A transition is represented as $s \xrightarrow{l} s'$ where $l \in \Sigma$. Notice that we refer to a state $s^f \in S$ as final if there is no outgoing transition at that state.

Definition 2 (CP). *A conversation protocol CP for a set of peers $\{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ is a LTS $CP = (S_{CP}, s^0_{CP}, L_{CP}, T_{CP})$ where S_{CP} is a finite set of states and $s^0_{CP} \in S_{CP}$ is the initial state; L_{CP} is a set of labels where a label $l \in L_{CP}$ is denoted $m^{\mathcal{P}_i, \mathcal{P}_j}$ such that $\mathcal{P}_i$ and $\mathcal{P}_j$ are the sending and receiving peers, respectively, $\mathcal{P}_i \neq \mathcal{P}_j$, and m is a message on which those peers interact; finally, $T_{CP} \subseteq S_{CP} \times L_{CP} \times S_{CP}$ is the transition relation. We require that each message has a unique sender and receiver: $\forall \{m^{\mathcal{P}_i, \mathcal{P}_j}, m'^{\mathcal{P}'_i, \mathcal{P}'_j}\} \subseteq L_{CP} : m = m' \implies \mathcal{P}_i = \mathcal{P}'_i \wedge \mathcal{P}_j = \mathcal{P}'_j$.*

In the remainder of this paper, we denote a transition $t \in T_{CP}$ as $s \xrightarrow{m^{\mathcal{P}_i, \mathcal{P}_j}} s'$ where s and s' are source and target states and m^{P_i, P_j} is the transition label. We refer to a basic $CP =< S_{CP}, s^0_{CP}, L_{CP}, T_{CP} >$ as CP_b if and only if $T_{CP} = \{s_{CP} \xrightarrow{m^{\mathcal{P}_i, \mathcal{P}_j}} s'_{CP}\}$. We refer to the set of final states as S^f where the system can terminate its execution. It is worth noticing that the peers' LTSs are computed by projection from CP as follows:

Definition 3 (Projection). *Let the projection function $\downarrow CP$ which returns the set of peers LTSs $\mathcal{P}_i = <S_i, s^0_i, \Sigma_i, T_i>$ obtained by replacing in $CP = <S_{CP}, s^0_{CP}, L_{CP}, T_{CP}>$ each label $(\mathcal{P}_j, m, \mathcal{P}_k) \in L_{CP}$ with $m!$ if $j = i$ with $m?$ if $k = i$ and with τ (internal action) otherwise; and finally removing the τ-transitions by applying standard minimization algorithms [14].*

Definition 4 (Synchronous System). *The synchronous system denoted as $Sys_{sync}(\mathcal{P}_1, \dots, \mathcal{P}_n) = (S_s, s^0_s, L_s, T_s)$ corresponds to the product of peer LTSs composed under synchronous communication semantics.*

In this context, a communication between two peers occurs if both agree on a synchronization label, *i.e.*, if one peer is in a state in which a message can be sent, then the other peer must be in a state in which that message can be received. A peer can evolve independently from others through internal actions.

Definition 5 (Asynchronous System). *In the asynchronous system denoted as $Sys_{async}(\mathcal{P}_1, \dots, \mathcal{P}_n) = (S_a, s^0_a, L_a, T_a)$, peers communication holds through FIFO buffers. Each peer $\mathcal{P}_i$ is equipped with an unbounded message buffer Q_i.*

Where a peer can either send a message $m \in \Sigma^!$ to the tail of the receiver buffer Q_j at any state where this sent message is available, read a message $m \in \Sigma^?$ from its buffer Q_i if the message is available at the buffer head, or evolve independently through an internal action. Reading from the buffer is non observable, and it is presented by internal action in the asynchronous system.

2.2 Realizability

The definition of realizability we use in this paper is borrowed from [5]. A CP is realizable if there exists a set of peers where their composition generates the same sequences of sending messages as specified in CP. In [5] a defined sufficient and necessary condition characterizes the set $R \subseteq CP$ of realizable CPs. A deterministic $cp \in R$ is realizable iff the system obtained from the composition of the projected peers of cp is *synchronizable*, *well-formed*, and *equivalent* to the initial CP. A proof of correctness of global system realizablity using Event-B is available in [13].

Definition 6 (Deterministic Choice). *Let DC be the set of deterministic CPs, thus $\forall CP \in DC : \forall s_{CP} \in S_{CP}$, $\nexists\{s_{CP} \xrightarrow{m^{P_i,P_j}} s'_{CP}, s_{CP} \xrightarrow{m^{P_i,P_j}} s''_{CP}\} \subseteq T_{CP}$ where $s'_{CP} \neq s''_{CP}$.*

Definition 7 (Equivalence). *CP is equivalent to $Sys_{sync}(\downarrow CP)$, denoted $CP \equiv Sys_{sync}(\downarrow CP)$, if they have equal message sequences, i.e., trace equivalence [16].*

A system is synchronizable when its behavior remains the same under both synchronous and asynchronous communication semantics.

Definition 8 (Synchronizability). *Given a set of peers $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, the synchronous system $Sys_{sync}(\mathcal{P}_1, \ldots, \mathcal{P}_n) = (S_s, s^0_s, L_s, T_s)$, and the asynchronous system $Sys_{async}(\mathcal{P}_1, \ldots, \mathcal{P}_n) = (S_a, s^0_a, L_a, T_a)$, two states $r \in S_s$ and $s \in S_a$ are synchronizable if there exists a relation $Sync\_st$ between states such that $Sync\_st(r, s)$ and:*

- *for each $r \xrightarrow{m} r' \in T_s$, there exists $s \xrightarrow{m!} s' \in T_a$, such that $Sync\_st(r', s')$;*
- *for each $s \xrightarrow{m!} s' \in T_a$, there exists $r \xrightarrow{m} r' \in T_s$, such that $Sync\_st(r', s')$;*
- *for each $s \xrightarrow{m?} s' \in T_a$, $Sync\_st(r, s')$.*

Synchronizability is the set of synchronizable systems such that $Sys_{async}(\mathcal{P}_1, \ldots, \mathcal{P}_n) \in Synchronizability \Leftrightarrow Sync\_st(s^0_s, s^0_a)$.

Well-formedness states that whenever the size of a receive queue, Q_i, of the i^{th} peer is greater than 0 (*i.e.*, Q_i is non-empty), the asynchronous system can eventually move to a state where Q_i is empty.

Definition 9 (Well-formedness). *Let WF be the set of well formed system. An asynchronous system $Sys_{async} = (S_a, s^0_a, \Sigma_a, T_a)$ defined over a set of peers $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ is well-formed, i.e., $Sys_{async} \in WF$, if and only if $\forall s_a = (s_1, Q_1, \ldots, s_n, Q_n) \in S_a$, where s_a is reachable from $s^0_a = (s^0_1, \epsilon, \ldots, s^0_n, \epsilon)$, the following holds: if there exists Q_i such that $\mid Q_i \mid > 0$, then there exists $s_a \Rightarrow^* s'_a \subseteq T_a$ where $s'_a = (s'_1, Q'_1, \ldots, s'_n, Q'_n) \in S_a$ and $\forall Q'_i, \mid Q'_i \mid = 0$.*

Note that $\Rightarrow^*$ means that there exists one or more transitions in the asynchronous system (Definition 5) leading into the state s'_a.

Definition 10 (Realizability). *$\forall CP \in DC : CP \in R \Longleftrightarrow (CP \equiv Sys_{sync}(\downarrow CP)) \wedge (Sys_{async}(\downarrow CP) \in Synchronizability) \wedge (Sys_{async}(\downarrow CP) \in WF)$.*

3 CCP Language for Realisable CPs

In this section, we define our composition operators and identify the conditions sufficient to build CP realizable CPs.

3.1 Composition Operators

We present the proposed composition operators $\otimes_{(\gg, s^f_{CP})}$ (sequence), $\otimes_{(+, s^f_{CP})}$ (branching), and $\otimes_{(\circlearrowleft, s^f_{CP})}$ (iteration) where $s^f_{CP} \in S^f_{CP}$. Each expression of the form $\otimes_{(op, s^f_{CP})}(CP, CP_b)$ assumes that the initial state of CP_b is fused with the final state s^f_{CP}. In the other word, CP_b is appended to CP at state s^f_{CP}.

Definition 11. *Sequential Composition* $\otimes_{(\gg, s^f_{CP})}$. *Given a CP, a state* $s_{CP} \in S^f_{CP}$, *and a* CP_b *where* $T_{CP_b} = \{s_{CP_b} \xrightarrow{l_{CP_b}} s'_{CP_b}\}$, *the sequential composition* $CP_{\gg} = \otimes_{(\gg, s_{CP})}(CP, CP_b)$ *means that* CP_b *must be executed after* CP *starting from* s_{CP}, *and:*

- $S_{CP_{\gg}} = S_{CP} \cup \{s'_{CP_b} | \, s_{CP_b} \xrightarrow{l_{CP_b}} s'_{CP_b} \in T_{CP_b}\}$
- $L_{CP_{\gg}} = L_{CP} \cup \{l_{CP_b}\}$
- $T_{CP_{\gg}} = T_{CP} \cup \{s_{CP} \xrightarrow{l_{CP_b}} s'_{CP_b}\}$
- $S^f_{CP_{\gg}} = (S^f_{CP} \setminus \{s_{CP}\}) \cup \{s'_{CP_b}\}$

Definition 12. *Choice Composition* $\otimes_{(+, s^f_{CP})}$. *Given a* CP, *a state* $s_{CP} \in S^f_{CP}$, *a set* $\{CP_{bi} \mid i = [1..n], n \in \mathbb{N}\}$ *such that* $\forall T_{CP_{bi}}$, $T_{CP_{bi}} = \{s_{CP_{bi}} \xrightarrow{l_{CP_{bi}}} s'_{CP_{bi}}\}$, *the branching composition* $CP_+ = \otimes_{(+, s_{CP})}(CP, \{CP_{bi}\})$ *means that* CP *must be executed before* $\{CP_{bi}\}$ *and there is a choice between all* $\{CP_{bi}\}$ *at* s_{CP}, *and:*

- $S_{CP_+} = S_{CP} \cup \{s'_{CP_b 1}, \ldots, s_{CP'_{bn}} | \, s_{CP_{bi}} \xrightarrow{l_{CP_{bi}}} s'_{CP_{bi}} \in T_{CP_{bi}}\}$
- $L_{CP_+} = L_{CP} \cup \{l_{CP_{bi}}, \ldots, l_{CP_{bn}}\}$
- $T_{CP_+} = T_{CP} \cup \{s_{CP} \xrightarrow{l_{CP_{b1}}} s'_{CP_{b1}}, \ldots, s_{CP} \xrightarrow{l_{CP_{bn}}} s'_{CP_{bn}}\}$
- $S^f_{CP_+} = (S^f_{CP} \setminus \{s_{CP}\}) \cup \{s'_{CP_{b1}}, \ldots, s'_{CP_{bn}}\}$

Definition 13. *Loop Composition* $\otimes_{(\circlearrowleft, s^f_{CP})}$. *Given* CP, *a state* $s_{CP} \in S^f_{CP}$, *and a set* CP_b, *such that* $T_{CP_b} = \{s_{CP_b} \xrightarrow{l_{CP_b}} s_{CP_b}\}$, *the loop composition* $CP_{\circlearrowleft} = \otimes_{(\circlearrowleft, s^f_{CP})}(CP, CP_b)$ *means that* CP *must be executed before* CP'_b *and every* CP_{bi} *can be repeated 0 or more times, and:*

- $S_{CP_{\circlearrowleft}} = S_{CP}$
- $L_{CP_{\circlearrowleft}} = L_{CP} \cup \{l_{CP_b}\}$
- $T_{CP_{\circlearrowleft}} = T_{CP} \cup \{s_{CP} \xrightarrow{l_{CP_{b1}}} s_{CP'_b}\}$
- $S^f_{CP_{\circlearrowleft}} = S^f_{CP}$

3.2 Realizable-by-Construction CP

As mentioned in the introduction, our intention is to avoid a posteriori global verification of realisability. We set up an incremental verification of realisability using a correct by construction approach. Building CPs using the aforementioned operators does not guarantee its realisability. Indeed, the definitions of the previous operators rely on syntactic conditions mainly by gluing final and initial states of the composed CPs.

Sufficient Conditions. We identified a set of sufficient conditions (*i.e.*, Conditions 1, 2, and 3 which entail realisability when the CPs are built using the operators we have previously defined. These conditions are based on the semantics of the messages ordering and exchange.

Condition 1 (Deterministic Choice (DC)). *See Definition 6.*

Condition 2 (Parallel-Choice Freeness (PCF)). *Let PCF be the set of CPs free of parallel choice. Then,* $CP \in PCF$ *iff* $\forall s_{CP} \in S_{CP}, \nexists \{s_{CP} \xrightarrow{m^{P_i,P_j}} s'_{CP},\ s_{CP} \xrightarrow{m'^{P_k,P_q}} s''_{CP}\} \subseteq T_{CP}$ *such that* $P_i \neq P_k$ *and* $s'_{CP} \neq s''_{CP}$.

Condition 3 (Independent Sequences Freeness (IseqF)). *Let ISeqF be the set of CPs free of independent sequences. Then,* $CP \in$ *ISeqF iff* $\forall s_{CP} \in S_{CP}$, $\nexists \{s_{CP} \xrightarrow{m^{P_i,P_j}} s'_{CP},\ s'_{CP} \xrightarrow{m'^{P_k,P_q}} s''_{CP}\} \subseteq T_{CP}$ *such that* $P_i \neq P_k$ *and* $P_j \neq P_k$.

All these conditions are structural conditions defined at the CP level. They do not involve conditions on the synchronous nor on the asynchronous projections.

Realizable-by-Construction CP Theorems. Table 1 gives the theorems that ensure the realisability of a *CP* built incrementally using our composition operators. Each theorem relies on the previously introduced sufficient conditions.

Proof Sketch. To prove the theorems of Table 1 we rely on a generic proof pattern consisting in decomposing the realisability condition of Definition 10. According to this definition, we need to prove equivalence (Definition 7), synchronizability (Definition 8) and well formedness (WF in Definition 9).

Table 1. Theorems for realizable by construction CPs

Theorem 1	$CP_b \in R$
Theorem 2	$CP \in R \wedge CP_b \in R \wedge CP_{\gg} = \otimes_{(\gg, s^f_{CP})}(CP, CP_b) \in \text{ISeqF} \Rightarrow CP_{\gg} \in R$
Theorem 3	$CP \in R \wedge \{CP_{bi}\} \subseteq R \wedge CP_{+} = \otimes_{(+, s^f_{CP})}(CP, \{CP_{bi}\}) \wedge\ CP_{+} \in \text{DC}$ $\wedge\ CP_{+} \in \text{ISeqF} \wedge\ CP_{+} \in \text{PCF} \ \Rightarrow\ CP_{+} \in R$
Theorem 4	$CP \in R \wedge CP_b \in R \wedge CP_{\circlearrowleft} = \otimes_{(\circlearrowleft, s^f_{CP})}(CP, CP_b) \in \text{ISeqF} \ \Rightarrow\ CP_{\circlearrowleft} \in R$

The proof is a structural induction on the defined operators. Let $CP_b \in R$ and $CP \in R$ be a basic realizable CP and a realizable CP respectively. We need to prove that $CP_{op} \in R$ holds for each composition operator $op \in \{\gg, + \circlearrowright\}$ when the defined sufficient condition op_{cond} corresponding to conditions 1, 2 and 3 defined above and associated to each op holds.

When considering the equivalence, synchronisability and well formedness, this proof uses the projection $\downarrow CP_{op}$ of CP_{op}. It can be formalised using the following proof pattern.

$$CP \in R \ \wedge \ CP_b \in R \ \wedge \ \mathbf{Op_{cond}} \Longrightarrow \begin{cases} CP_{op} \equiv Sys_{sync}(\downarrow_{CP\mathbf{op}}) \\ \wedge \\ Sys_{async}(\downarrow_{CP_{op}}) \in Synchronizability \\ \wedge \\ Sys_{async}(\downarrow_{CP_{op}}) \in WF \end{cases} \qquad (1)$$

Theorem 1. *Any CP_b is realizable.*

Proof 1. CP_b is made of a single transition of the form $s \xrightarrow{m^{\mathcal{P}_i, \mathcal{P}_j}} s'$. Therefore, the projection will produce two peers $\mathcal{P}_i$ and $\mathcal{P}_j$ with a single transition where $\mathcal{P}_i$ sends the message m to the receiving peer $\mathcal{P}_j$. This projection is realizable.

Theorem 2. *Given an $CP = <S_{CP}, s^0_{CP}, L_{CP}, T_{CP}>$ and a CP_b such that $CP \in R$ and $CP_b \in R$, $s_{CP} \in S^f_{CP}$, then $CP_{\gg} = \otimes_{(\gg, s_{CP})}(CP, CP_b) \in R$.*

Proof 2. The proof is inductive. It follows the previous proof pattern. When this pattern is instantiated for the sequence operator, we obtain.

$$CP \in R \ \wedge \ CP_b \in R \ \wedge \ CP_{\gg} \in \mathbf{ISeqF} \Longrightarrow \begin{cases} CP_{\gg} \equiv Sys_{sync}(\downarrow_{CP_{\gg}}) & (2.a) \\ \wedge \\ Sys_{async}(\downarrow_{CP_{\gg}}) \in Synchronizability & (2.b) \\ \wedge \\ Sys_{async}(\downarrow_{CP_{\gg}}) \in WF & (2.c) \end{cases} \qquad (2)$$

Basic case. Let $CP = \varnothing$ and a CP_b then $CP_{\gg} = \otimes_{(\gg, s^0_{CP})}(\varnothing, CP_b) \in R$. So $CP_{\gg} = CP_b$. $CP_{\gg} \in R$ holds by Theorem 1 of Table 1.
Inductive Case. Let $CP = <S_{CP}, s^0_{CP}, L_{CP}, T_{CP}>$ and a CP_b such that $CP \in R$ and $CP_b \in R$. Let $s_{CP} \in S^f_{CP}$ be the gluing state (i.e. both the final state of CP and the initial state of CP_b). Let s^q_i denote the i^{th} state in the LTS associated to peer P_q.

According to the proof schema of Eq. 2, we need to prove the Properties 2.a, 2.b and 2.c

2.a Equivalence property. By recurrence hypotheses we write $CP \equiv Sys_{sync}(\downarrow CP)$, $CP_b \equiv Sys_{sync}(\downarrow CP_b)$. Let us assume that the sufficient condition for sequence holds i.e. $CP_{\gg} \in ISeqF$. We need to prove now that $CP_{\gg} \equiv Sys_{sync}(\downarrow CP_{\gg})$ (Eq. (1.a)).
Let us consider

- any trace $T_{CP} = \{s_0 \xrightarrow{m^{P_i \to P_j}} s_1, \ldots, s_n \xrightarrow{m'^{P_k \to P_q}} s_{n+1}\}$ in the realizable CP
- and the trace $T_{CP_b} = \{s_{b0} \xrightarrow{m''^{P_t \to P_z}} s_{b1}\}$ in the realizable CP_b

Since the $ISeqF$ condition holds, two cases are distinguished.

1. **Either** $P_k = P_t$, then the following suffixes of the traces occur for peers $P_k = P_t$, P_q and P_z
   - $\{\ldots, s_n^k \xrightarrow{m'!} s_{n+1}^k, s_{n+1}^k \xrightarrow{m''!} s_{n+2}^k\} \subseteq T_k$
   - $\{\ldots, s_n^q \xrightarrow{m'?} s_{n+1}^q\} \subseteq T_q$
   - $\{\ldots, s_n^z \xrightarrow{m''?} s_{n+1}^z\} \subseteq T_z$.
2. **or** $P_q = P_t$, then the following traces occurs for peers $P_q = P_t$, P_k and P_z
   - $\{\ldots, s_n^k \xrightarrow{m'!} s_{n+1}^k\} \subseteq T_k$
   - $\{\ldots, s_n^q \xrightarrow{m'?} s_{n+1}^q, s_{n+1}^q \xrightarrow{m''!} s_{n+2}^q\} \subseteq T_q$
   - $\{\ldots, s_n^z \xrightarrow{m''?} s_{n+1}^z\} \subseteq T_z$

Thanks to the ISeqF property, the sending-receiving transition (synchronous transition) of CP_b requires that either the sending peer or the receiving peer of the CP_b are used by the previous transition or the realizable CP. Moreover, it is always performed once the sending-receiving transitions of the synchronous projection of CP are completed. The sending-receiving transition of CP_b becomes the last transition of $Sys_{sync}(\downarrow CP_{\gg})$.

2.b Synchronisability condition. By the recurrence hypotheses, we write $Sys_{async}(\downarrow CP) \in Synchronizability$, $Sys_{async}(\downarrow CP_b) \in Synchronizability$. Synchronisability is deduced from equivalence and from the $ISeqF$ condition. The last transition of the traces of $\downarrow CP_{\gg}$ corresponds to $Sys_{sync}(\downarrow CP_b) = \{s_{b0} \xrightarrow{m''} s_{b1}\}$ and $Sys_{async}(\downarrow CP_b) = \{s_{b0} \xrightarrow{m'?} s_b, s_b \xrightarrow{m''?} s_{b1}\}$ where S_b is an intermediate state in the asynchronous projection In this intermediate state, in which the queues related to the peers contain the message m''.

2.c Well-formedness condition. Again, as recurrence hyptheses, we write $Sys_{async}(\downarrow CP) \in WF$, $Sys_{async}(\downarrow CP_b) \in WF$. This means that by hypotheses, the queues are empty in the final state of $Sys_{async}(\downarrow CP)$ since it is realizable (thus well formed). We have to show that the queue is still empty after running message exchanges of CP_b.
When adding a sequence $\otimes_{(\gg, s_{CP}^f)}(CP, CP_b) \in ISeqF$, the sending transition of m'' gives $Q_i = \varnothing, Q_j = \varnothing, Q_k = \varnothing, Q_q = \varnothing, Q_t = \varnothing, Q_z = \{m''\}$. It and the consumption of the m'' empties the queue Q_z such that $Q_i = \varnothing, Q_j = \varnothing, Q_k = \varnothing, Q_q = \varnothing, Q_t = \varnothing, Q_z = \varnothing$.

At this level we can conclude that the defined sequence composition operator preserves realizability.

The proofs for the choice and loop operators follow the same inductive schema. We do not present these proofs due to space limitations. A sketch of these proofs is given in [6].

4 CCP Model: Refinement-Based Realizability

The proofs reported in the previous section are handmade. In order to give full confidence in our results on correct-by-construction realizability, we designed a whole formal development of this proof using refinement. The Event-B method has been set up as follows.

4.1 The Refinement Strategy

The refinement operator offered by the Event-B method proved efficient to handle the complex proofs associated to each operators. This operator allowed us to handle the realizability property incrementally by introducing first equivalence, then synchronizability and finally well formedness in specific refinements. Therefore, the following refinement strategy has been set up:

Root Model. The root model defines the conversation protocols. It introduces basic CP. Each composition operator is defined as an event which incrementally builds the final CP obtained by introducing a final state. All the built CP satisfy an invariant requiring DC (deterministic choice, Condition 1). This model also declares a prophecy variable [1] as a state variable. This variable defines an arbitrary numbers of exchanged messages and is used to define a variant in order to further prove well formedness.

First Refinement: The Synchronous Model. The second model is obtained by refining each event (composition operator) to define the synchronous projection. A gluing invariant linking the CP to the synchronous projection is introduced. The equivalence property is proved at this level. It is defined as an invariant preserved by all the events encoding a composition operator. This projection represents the synchronous system, it preserves the message exchanges order between peers and hides the asynchronous exchanges.

Second Refinement: The Asynchronous Model. The last model introduces the asynchronous projection. Each event (composition operator) is refined to handle the asynchronous communication. Synchronous and asynchronous projections are linked by another gluing invariant. Sending and receiving actions together with queue handling actions and variant decreasing of the prophecy variable are introduced. They are necessary to prove synchronizability and well formedness expressed as invariants. The refinement of the synchronous models in an asynchronous model eases the proof process.

At the last refinement, realizability is proved thanks to invariants preservation and to the inductive proof process handled by Event-B using the Rodin platform.

Next sections sketches this development. For each refinement step, we introduce the relevant definitions, axioms and theorems needed to build the model.

4.2 The Root Model

It describes the notion of CP and introduces the definition of each operator at the CP level. Each introduced Event-B event corresponds to the formalisation of one operator defined in Sect. 3.1.

Table 2. An excerpt of the LTS_CONTEXT.

```
LTS_CONTEXT
SETS PEERS, MESSAGES , CP_STATES.
CONSTANTS CPs_B, DC, ISeqF, NDC, ...
AXIOMS
    axm1: CPs_B ⊆ CP_STATES × PEERS × MESSAGES× PEERS × CP_STATES×ℕ
        – Determinstic CP definition DC
    axm2_Cond1: NDC ⊆ CPs_B
    axm3_Cond1: ∀Trans2, Trans1·(Trans1 ∈ CPs_B ∧ Trans2 ∈ CPs_B∧
                    SOURCE_STATE(Trans1) = SOURCE_STATE(Trans2)∧
                    LABEL(Trans1) = LABEL(Trans2)∧
                    DESTINATION_STATE(Trans1) ≠ DESTINATION_STATE(Trans2))
                            ⇒{Trans1, Trans2} ⊆ NDC
    axm4_Cond1: DC = CPs_B \ NDC
        – Independent sequence freeness definition ISEQF
    axm5_Cond3: ISeqF ⊆ CPs_B
    axm6_Cond3: ∀ cp_b · ( cp_b ∈ CPs_B ∧
                    (PEER_SOURCE(cp_b) = LAST_SENDER_PEERS(SOURCE_STATE(cp_b)) ∨
                    PEER_SOURCE(cp_b) = LAST_RECEIVER_PEERS(SOURCE_STATE(cp_b))))
                            ⇒  {cp_b} ⊆ ISeqF
     ...
End
```

Required Properties for CPs (cf. Table 2). Table 2 presents part of the Event-B context used at the abstract level. We introduce, using sets and constants, the whole basic definitions of messages, CP states, basic CPs, etc. A set of axioms is used to define the relevant properties of these definitions.
For example, in axiom $axm1$, a CP is defined as a set of transitions with a source and target state, a message and a source and target peers. $axm3\_Cond1$ defines what a non deterministic CP is using the NDC set. This NDC set characterises all the non deterministic choices in a CP. Observe that axiom $axm4\_Cond1$ defines the DC property in Definition 10 of Sect. 2.2.

The Root Machine (cf. Table 4). This model corresponds to the definition of the CP LTS. Each operator corresponds to one event and contributes to build a given CP represented in the state variable $BUILT\_CP$ which shall define deterministic CP only (see invariant $inv1$ in Table 3).

Table 3. An excerpt of the invariants of the LTS_model.

```
        Invariants
   inv1:  BUILT_CP ⊆ DC
             ...
```

The *Add_Seq* event corresponds to the sequence operator of Definition 11 of Sect. 3.1. Its effect is to add a given basic CP, namely *Some_cp_b* to the currently built CP (union operation in action $act1$) and sets up the new final states in action $act3$. This event is triggered only if the relevant conditions identified in Sect. 3.1 holds (guards). For example, it is clearly stated that the independent sequence property $ISeqF$ shall hold before adding another CP in sequence. This condition is given by guard $grd3$ (see Table 4).

Table 4. An excerpt of the LTS_model.

```
INITIALISATION≜
EVENTS
    Add_Seq ≜
        Any Some_cp_b
        Where
            grd1: $Some\_cp\_b \in cps\_b$
            grd2: $MESSAGE(Some\_cp\_b) \neq End$
            grd3: $Some\_cp\_b \in ISeqF$
            grd4: $SOURCE\_STATE(Some\_cp\_b) \in CP\_Final\_states$
            ...
        Then
            act1: $BUILT\_CP := BUILT\_CP \cup \{Some\_cp\_b\}$
            act3: $CP\_Final\_states :=$ (CP_Final_states $\cup$
                    $\{DESTINATION\_STATE(Some\_cp\_b)\})\setminus$
                    $\{SOURCE\_STATE(Some\_cp\_b)\}$
            ...
    End
    Add_Choice ≜ ...
    Add_Loop ≜ ...
    Add_End ≜ ...
End
```

Up to now, no proof related to realizability is performed. We have just stated that all the built CPs are deterministic (they belong to the DC set of CPs which represent a condition for the ralizability property of Definition 10 in Sect. 2.2.

4.3 First Refinement: Synchronous Model

The objective of the first refinement is to build the synchronous projection corresponding to Definition 4. Here again, before building this projection, some property definitions are required, in particular for equivalence ($\equiv$), denoted $EQUIV$ in Event_B models.

Required Properties for Synchronous Projection (cf. Table 5). The definition of the state-transitions system corresponding to the synchronous projection is given by the set CPs_SYNC_B defined by axiom $axm1$ of Table 5. Actions (send ! and receive ?) are introduced. Then, two other important axioms, namely $axm\_1.a$ and $axm\_1.a1$, are given to define the equivalence between a CP and its synchronous projection. The $EQUIV$ relation is introduced. It characterises the set of CPs that are equivalent to their synchronous projection. $axm\_1.a1$ formalises Definition 7 of Sect. 2.2.

Table 5. An excerpt of the LTS_SYNC_CONTEXT.

```
LTS_SYNC_CONTEXT, EXTENDS LTS_CONTEXT
SETS ACTIONS. CONSTANTS CPs_B , EQUIV, ...
AXIOMS
    axm1: $CPs\_SYNC\_B \subseteq CP\_STATES \times ACTIONS \times MESSAGES \times PEERS \times$
                              $PEERS \times ACTIONS \times MESSAGES \times CP\_STATES \times \mathbb{N}$
       – Equivalence of CP and Synchronous projection
    axm_1.a: $EQUIV \in CPs\_B \twoheadrightarrow CPs\_SYNC\_B$
    axm_1.a1: $EQUIV = \{\ Trans \mapsto S\_Trans \mid Trans \in CPs\_B \land S\_Trans \in CPs\_SYNC\_B \land$
                    $SOURCE\_STATE(Trans) = S\_SOURCE\_STATE(S\_Trans) \land$
                    $DESTINATION\_STATE(Trans) = S\_DESTINATION\_STATE(S\_Trans) \land$
                    $PEER\_SOURCE(Trans) = S\_PEER\_SOURCE(S\_Trans) \land$
                    $PEER\_DESTINATION(Trans) = S\_PEER\_DESTINATION(S\_Trans) \land$
                    $MESSAGE(Trans) = S\_MESSAGE(S\_Trans) \land$
                    $INDEX(Trans) = S\_INDEX(S\_Trans)\ \}$
     ...
End
```

Table 6. An excerpt of the invariants of the LTS_Synchronous_model.

```
Invariants
        inv1: $BUILT\_SYNC \subseteq CPs\_SYNC\_B$
        inv_1.a: $\forall Trans \cdot \exists S\_Trans \cdot (Trans \in BUILT\_CP \land S\_Trans \in BUILT\_SYNC \land$
                 $BUILT\_CP \neq \varnothing) \Rightarrow \mathbf{Trans \mapsto S\_Trans \in EQUIV}$
```

The Synchronous Projection (cf. Table 7). The first refinement introduces the synchronous projection of the $BUILT\_CP$ defined by variable $BUILT\_SYNC$ in Table 7. Table 6 introduces through invariant $inv\_1.a$. The equivalence ($\equiv$) property corresponding to Condition 2.a in Eq. 2. The invariant requires equivalence between a CP and its synchronous projection. Invariant $inv2$ of Table 6 describes the equivalence property using the $EQUIV$ relation

Table 7. An excerpt of the LTS_Synchronous_model.

```
INITIALISATION
    ...
EVENTS
    $Add\_Seq$ Refines $Add\_Seq \triangleq$
        Any
            $S\_Some\_cp\_b, Some\_cp\_sync\_b$
        Where
            grd1: $Some\_cp\_sync\_b \in cps\_sync\_b$
            grd3: $S\_SOURCE\_STATE(Some\_cp\_sync\_b) \in CP\_Final\_states$
            grd4: $S\_Some\_cp\_b \in ISeq$
            grd8: $MESSAGE(S\_Some\_cp\_b) \neq End$
            grd9: $MESSAGE(S\_Some\_cp\_b) = S\_MESSAGE(Some\_cp\_sync\_b)$
            ...
        With Some_cp_b: $Some\_cp\_b = S\_Some\_cp\_b$
        Then
            act1: $BUILT\_CP := BUILT\_CP \cup \{S\_Some\_cp\_b\}$
            act2: $BUILT\_SYNC := BUILT\_SYNC \cup \{Some\_cp\_sync\_b\}$
        ...
    End
```

defined in the context of Table 5. So, one part of the realizability property (i.e. $CP \equiv Sys_{sync}$) of Definition 10 is already proved at this refinement level.
The event *Add_Seq* or sequence operator (Table 7) refines the same event of the root model. It introduces the *BUILT_SYNC* set corresponding to the synchronous projection as given in Definition 4. Here, again, the *Add_Seq* applies only if the conditions in the guards hold. The *With* clause provides a witness to glue *Some_cp_b* CP with its synchronous version.

4.4 Second Refinement: Asynchronous Model

The second refinement introduces the asynchronous projection with sending and receiving peers actions. Well formedenss and synchronizability remain to be proved in order to complete realizability preservation (Table 8).

Table 8. An excerpt of the LTS_ASYNC_CONTEXT.

```
CONTEXT LTS_ASYNC_CONTEXT EXTENDS LTS_SYNC_CONTEXT
SETS A_STATES, . . . ,
CONSTANTS CPs_ASYNC_B, SYNCHRONISABILITY, WF, . . .
AXIOMS
    axm1: CPs_ASYNC_B ∈ (A_STATES × ETIQ × ℕ) ⇸ A_STATES
        − Synchronisability property
    axm_1.b:  SYNCHRONISABILITY  ∈ CPs_SYNC_B ⤖ R_TRACE_B
    axm_1.b1:  SYNCHRONISABILITY  = {S_Trans↦ R_Trans | S_Trans ∈ CPs_SYNC_B ∧
                   R_Trans ∈ R_TRACE_B ∧ S_INDEX(S_Trans) = R_INDEX(R_Trans) ∧
                   S_SOURCE_STATE(S_Trans) = R_SOURCE_STATE(R_Trans) ∧
                   S_PEER_SOURCE(S_Trans) = R_PEER_SOURCE(R_Trans) ∧
                   S_MESSAGE(S_Trans) = R_MESSAGE(R_Trans) ∧
                   S_PEER_DESTINATION(S_Trans) = R_PEER_DESTINATION(R_Trans) ∧
                   S_DESTINATION_STATE(S_Trans) = R_DESTINATION_STATE(R_Trans)}
        − Well formedness property
    axm_1.c: WF  ∈ A_TRACES → QUEUE
    axm_1.c1: ∀ A_TR,queue · ( A_TR ∈ A_TRACES ∧ queue ∈ QUEUE ∧ queue = ∅ )
     ⇒ A_TR↦ queue ∈  WF
     . . .
End
```

The Asynchronous Projection (cf. Tables 10 and 11). The invariants associated to this model are presented in Table 9. In particular, the properties of synchronizability, expressed in invariant $axm\_1.b$ used in Definition 10 ($Sync(Sys_{sync}, Sys_{async})$), and of well formedness, expressed in invariant $axm\_1.c$ used in Definition 10 ($WF(Sys_{async})$) are introduced in the invariant of this refinement level. These two properties complete the proof of realizability.

At these level, each event corresponding to a composition operator is refined by three events: one to handle sending of messages (*Add_Seq_send*) on Table 10, one for receiving messages (*Add_Seq_receive*) and a third one (*Add_Seq_send_receive*) on Table 11 refining the abstract *Add_seq* event.

Tables 10 and 11 define these events. Sending and receiving events are interleaved in an asynchronous manner. Once a pair of send and receive events

Table 9. An excerpt of the invariants of the LTS_Asynchronous_model.

Invariants
inv1 $BUILT\_SYNC \subseteq CP\_SYNC\_B$
inv2 $REDUCED\_TRACE \subseteq R\_TRACE\_B$
inv3 $A\_TRACE \subseteq A\_TRACES$
inv_1.b $\forall S\_Trans \cdot \exists R\_Trans \cdot (S\_Trans \in BUILT\_SYNC \land R\_Trans \in REDUCED\_TRACE) \Rightarrow$ $\mathbf{S\_Trans} \mapsto \mathbf{R\_Trans} \in \mathbf{SYNCHRONISABILITY}$
inv_1.c $\forall A\_Trans \cdot (A\_Trans \in A\_TRACES \land MESSAGE(Last\_cp\_trans) = End \land A\_TRACE \neq \varnothing) \Rightarrow \mathbf{A\_Trans} \mapsto \mathbf{queue} \in \mathbf{WF}$
inv6 $BUILT\_ASYNC \subseteq CP\_ASYNC\_B$
...

Table 10. An excerpt of the LTS_Asynchronous_model.

Event $Add\_Seq\_Send \triangleq$
Any
$send, lts\_s, lts\_d, msg, index$
Where
grd1: $\exists send\_st\_src, send\_st\_dest \cdot ((lts\_s \mapsto send\_st\_src) \in A\_GS \land ((send\_st\_src \mapsto (Send \mapsto msg \mapsto lts\_d) \mapsto index) \mapsto send\_st\_dest) \in CPs\_ASYNC\_B \land \ldots$
...
Then
act1: $A\_TRACE := A\_TRACE \cup \{Reduces\_Trace\_states \mapsto St\_Num \mapsto Send \mapsto lts\_s \mapsto msg \mapsto lts\_d \mapsto Reduces\_Trace\_states \mapsto (St\_Num + 1) \mapsto A\_Trace\_index\}$
act2: $queue, back := queue \cup \{lts\_d \mapsto msg \mapsto back\}, back + 1$
act3: $A\_GS := A\_Next\_States(\{send\} \mapsto A\_GS \mapsto queue)$
...
End

has been triggered, the event *Add_Seq_send_receive* records that the emission-reception is completed. This event increases the number of received messages (action *act*5). Traces are updated accordingly by the events, they are used for proving the invariants.

4.5 Instantiation and Axiom Validation

To illustrate our approach, we have instantiated our model on a toy example corresponding to the CP depicted on Fig. 1. The labels of the transitions of the form $m^{p \longrightarrow p'}$ denote a message m sent by peer p to the peer p'.

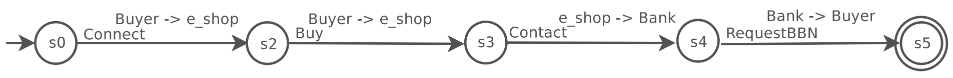

Fig. 1. Four messages exchanges in sequence for a electronic commerce system

The whole Event-B model has been instantiated. The context of Table 12 shows the instantiation of the model for the CP of Fig. 1. It also shows that the axioms

Table 11. An excerpt of the LTS_Asynchronous_model.

```
    Event $Add\_Seq\_Receive \triangleq$
      Any
        $send, receive, lts\_s, lts\_d, msg, index$
      Where
        grd1: $queue \neq \varnothing \wedge lts\_d \mapsto msg \mapsto front \in queue$
        grd2: $\exists receive\_st\_src, receive\_st\_dest \cdot (((lts\_d \mapsto receive\_st\_src) \in A\_GS) \wedge$
         $((receive\_st\_src \mapsto (Receive \mapsto msg \mapsto lts\_s) \mapsto index) \mapsto receive\_st\_dest)$
         $\in CPs\_ASYNC\_B \wedge \ldots$
         ...
       Then
        act1: $A\_TRACE := A\_TRACE \cup \{Reduces\_Trace\_states \mapsto St\_Num \mapsto$
         $Receive \mapsto lts\_s \mapsto msg \mapsto lts\_d \mapsto Reduces\_Trace\_states \mapsto (St\_Num + 1)$
         $\mapsto A\_Trace\_index\}$
        act2: $queue := queue \setminus \{lts\_d \mapsto msg \mapsto front\}$
         ...
    End
    Event $Add\_Seq\_Send - Receive$ Refines $Add\_Seq \triangleq$
      Any
        $A\_Some\_cp\_b, A\_Some\_cp\_sync\_b, Send\_cp\_async\_b, Receive\_cp\_async\_b, R\_trace\_b$
      Where
        grd1: $A\_MESSAGE(Send\_cp\_async\_b) = A\_MESSAGE(Receive\_cp\_async\_b)$
        grd2: $ACTION(Receive\_cp\_async\_b) = Receive \wedge ACTION(Send\_cp\_async\_b) = Send$
        grd3: $A\_Some\_cp\_b \in ISeq$
        grd4: $MESSAGE(A\_Some\_cp\_b) = A\_MESSAGE(Send\_cp\_async\_b)$
         ...
            With $S\_Some\_cp\_b : S\_Some\_cp\_b = A\_Some\_cp\_b,$
                 $Some\_cp\_sync\_b : Some\_cp\_sync\_b = A\_Some\_cp\_sync\_b$
       Then
        act1: $BUILT\_CP := BUILT\_CP \cup \{A\_Some\_cp\_b\}$
        act2: $BUILT\_SYNC := BUILT\_SYNC \cup \{A\_Some\_cp\_sync\_b\}$
        act3: $BUILT\_ASYNC := BUILT\_ASYNC \cup \{Send\_cp\_async\_b\} \cup \{Receive\_cp\_async\_b\}$
        act4: $REDUCED\_TRACE := REDUCED\_TRACE \cup \{R\_trace\_b\}$
         ...
    End
       ...
  End
```

defined in the model are inhabited. The ProB [15] model checker associated to Event-B on the Rodin platform has been used for automatic validation.

Other case studies borrowed from the research community dealing with realizability have been used to instantiate our model. These case studies use the whole composition operators we defined.

4.6 Assessment

Table 13 gives the results of our experiments. One can observe that all the proof obligations (POs) have been proved. A large amount of these POs has been proved automatically using the different provers associated to the Rodin platform. Interactive proofs of POs required to combine some interactive deduction rules and the automatic provers of Rodin. Few steps were required in most of the cases, and a maximum of 10 steps was reached.

Table 12. An excerpt of the LTS_CONTEXT_instantiation.

```
LTS_CONTEXT_instantiation      EXTENDS LTS_CONTEXT
CONSTANTS s0, s1, s2, s3, s4, s5, Connect, Buy, Contact, Request_BBN, End, Buyer, ...
AXIOMS
    axm1: partition(PEERS,{Buyer},{e_shop},{Bank},{Pend})
    axm2: partition(MESSAGES,{Connect},{Buy},{Contact},{Request_BBN},{End})
    axm3: partition(CP_STATES, {s0} ,{s1} ,{s2} ,{s3} ,{s4}, {s5})
    axm4: CPs_B= {s0 ↦ Buyer ↦ Connect ↦ e_shop ↦ s1 ↦ 1, . . . ,
    axm5: CPs_SYNC_B = {s0 ↦ Send ↦ Connect ↦ e_shop ↦ Buyer ↦ Receive ↦ . . .
    axm6: partition(A_STATES, {B_s0}, {B_s1}, {B_s2}, {B_s3}, {e_s0}, {e_s1}, . . . )
    axm7: CPs_ASYNC_B = {((B_s0 ↦ (Send ↦ Connect ↦ e_shop) ↦ 1) ↦ B_s1), . . . }
    axm8: A_TRACES = {s ↦ 0 ↦ Send ↦ Buyer ↦ Connect ↦ e_shop ↦ s ↦ 1 ↦ 1, . . . }
    axm9: R_TRACE_B = {s0 ↦ Buyer ↦ Connect ↦ e_shop ↦ s1 ↦ 1, . . . }
    axm10: S_Next_States = {{((B_s0 ↦ (Send ↦ Connect ↦ e_shop) ↦ 1) ↦ B_s1)} ↦
                           {(Buyer ↦ B_s0), (e_shop ↦ e_s0), (Bank ↦ Bk_s0)} ↦
                           {(Buyer ↦ B_s1), (e_shop ↦ e_s0), (Bank ↦ Bk_s0)}, . . . }
    axm11: A_Next_States = {{((B_s0 ↦ (Send ↦ Connect ↦ e_shop) ↦ 1) ↦ B_s1)} ↦
                            {(Buyer ↦ B_s0), (e_shop ↦ e_s0), (Bank ↦ Bk_s0)} ↦ ∅ ↦
                            {(Buyer ↦ B_s1), (e_shop ↦ e_s0), (Bank ↦ Bk_s0)}, . . . }
    . . .
END
```

Table 13. RODIN proofs statistics

Event-B Model	Interactive proofs	Automatic proofs	Proof obligations
Abstract context	06 (100%)	0 (0%)	06 (100%)
Synchronous context	02 (100%)	0 (0%)	02 (100%)
Asynchronous context	01 (33.33%)	02 (66.67%)	03 (100%)
Abstract model	28 (58.33%)	20 (41.67%)	48 (100%)
Synchronous model	39 (39%)	61 (61%)	100 (100%)
Asynchronous model	73 (38.83%)	115 (61.17%)	188 (100%)
Total	148 (100%)	198 (100%)	347 (100%)

5 Related Work

Several approaches addressed choreography realizability. In [10], the authors identify three principles for global descriptions under which a sound and complete end-point projection is defined. If these rules are respected, the projection will behave as specified in the choreography. This approach is applied to BPMN 2.0 choreographies [18]. [20] propose to modify their choreography language to include new constructs (choice and loop). During projection, particular communication is added to enforce the peers to respect the choreography specification. In [12], the authors propose a Petri Net-based formalism for choreographies and algorithms to check realizability and local enforceability. A choreography is locally enforceable if interacting peers are able to satisfy a subset of the requirements of the choreography. To ensure this, some message exchanges in the distributed system are disabled. In [21], the authors propose automated techniques to check the realizability of collaboration diagrams for different communication models.

Beyond advocating a solution for enforcing realizability, our contribution differs from these approaches as follows. We focus on asynchronous communication and choreographies involving loops. Our approach is non-intrusive; we do not add any constraints on the choreography language or specification, and the designer neither has to modify the original choreography specification, nor the peer models. We considerably reduce the verification complexity since there is no need to re-build the distributed system by composition of peers to check the realizability. Instead of that, we rely on a correct-by-construction approach based on sufficient conditions for realizability at the CP level. The technique we rely on here shares some similarities with counterexample-guided abstraction refinement (CEGAR) [11]. In CEGAR, an abstract system is analyzed for temporal logic properties. If a property holds, the abstraction mechanism guarantees that the property also holds in the concrete design. If the property does not hold, the reason may be a too coarse approximation by the abstraction. In this case, the counterexample generated by the model checker, is used to refine the system to a finer abstraction and this process is iterated.

To the best of our knowledge, our approach is the first correct-by-construction method which enables the designer to specify realizable CP avoiding behavioural errors in the distributed systems. By doing so, we propose an a priori verification method where the problems of state explosion and scalability are discarded. Other proof based techniques thant Event-B like Coq [3] or Isabelle [17] could have been used after defining the refinement operation. Our approach extensively uses built-in refinement operation and inductive proof schemes of Event-B.

6 Conclusion

This paper presents an a priori approach to build realizable CPs based on a correct-by-construction method. A language allowing to incrementally build complex realizable CPs from a set of basic realizable ones is defined. It offers a set of composition operators preserving realizability. Our proposal is proved to be sound and correct using the proof and refinement based formal method Event-B. Thanks to the use of arbitrary sets of values for parameters in our Event-B models, ou approach is scalable. Moreover, we have validated this model using several case studies. According to [4], this instantiation process is defined either using model checking to animate and test the CPs associated to each case study; or by explicitly supplying a witness to each parameter of the events in the Event-B model to build the CP associated to the case study.

As a short term perspective, we aim at extending our model with an operator enabling to compose entire CPs instead of requiring incremental composition of basic CP_b. Furthermore, we intend to define a set of patterns for realizable CPs and studying the completeness of our language in order to identify the class of real-world asynchronously communicating systems that can be specified. Last, we aim at providing the designers with an engine for automatic instantiation of realizable CPs.

References

1. Abadi, M., Lamport, L.: The existence of refinement mappings. Theor. Comput. Sci. **82**(2), 253–284 (1991)
2. Abrial, J.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge (2010)
3. Athalye, A.: CoqIOA: A formalization of IO automata in the Coq proof assistant, vol. 1019, pp. 1–53 (1995)
4. Babin, G., Ait-Ameur, Y., Pantel, M.: Correct instantiation of a system reconfiguration pattern: a proof and refinement-based approach. In: Proceedings of HASE 2016, pp. 31–38. IEEE Computer Society (2016)
5. Basu, S., Bultan, T., Ouederni, M.: Deciding choreography realizability. In: Proceedings of POPL 2012, pp. 191–202. ACM (2012)
6. Benyagoub, S., Ouederni, M., Ait-Ameur, Y.: Towards correct evolution of conversation protocols. In: Proceedings of VECOS 2016. CEUR Workshop Proceedings, vol. 1689, pp. 193–201. CEUR-WS.org (2016)
7. Benyagoub, S., Ouederni, M., Singh, N.K., Ait-Ameur, Y.: Correct-by-construction evolution of realisable conversation protocols. In: Bellatreche, L., Pastor, Ó., Almendros Jiménez, J.M., Aït-Ameur, Y. (eds.) MEDI 2016. LNCS, vol. 9893, pp. 260–273. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45547-1_21
8. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30**(2), 323–342 (1983)
9. Bultan, T.: Modeling interactions of web software. In: Proceedings of IEEE WWV 2006, pp. 45–52 (2006)
10. Carbone, M., Honda, K., Yoshida, N.: Structured communication-centred programming for web services. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 2–17. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71316-6_2
11. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 154–169. Springer, Heidelberg (2000). https://doi.org/10.1007/10722167_15
12. Decker, G., Weske, M.: Local enforceability in interaction petri nets. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 305–319. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75183-0_22
13. Farah, Z., Ait-Ameur, Y., Ouederni, M., Tari, K.: A correct-by-construction model for asynchronously communicating systems. Int. J. STTT **19**, 1–21 (2016)
14. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison Wesley, Boston (1979)
15. Bendisposto, J., Clark, J., Dobrikov, I., Karner, P., Krings, S., Ladenberger, L., Leuschel, M., Plagge, D.: Prob 2.0 tutorial. In: Proceedings of of 4th Rodin User and Developer Workshop, TUCS (2013)
16. Milner, R.: Communication and Concurrency. Prentice-Hall Inc., Upper Saddle River (1989)
17. Müller, O., Nipkow, T.: Combining model checking and deduction for I/O-automata. In: Brinksma, E., Cleaveland, W.R., Larsen, K.G., Margaria, T., Steffen, B. (eds.) TACAS 1995. LNCS, vol. 1019, pp. 1–16. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60630-0_1
18. OMG: Business Process Model and Notation (BPMN) - Version 2.0 (2011)
19. Project RODIN: Rigorous open development environment for complex systems (2004). http://rodin-b-sharp.sourceforge.net/

20. Qiu, Z., Zhao, X., Cai, C., Yang, H.: Towards the theoretical foundation of choreography. In: Proceedings of WWW 2007. ACM Press (2007)
21. Salaün, G., Bultan, T.: Realizability of choreographies using process algebra encodings. In: Leuschel, M., Wehrheim, H. (eds.) IFM 2009. LNCS, vol. 5423, pp. 167–182. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00255-7_12