Checking Realizability of a Timed Business Processes Choreography

Manuel I. Capel
Software Engineering Department, Granada University,
Periodista Daniel Saucedo Aranda, 18071 Granada, Spain

Keywords: Business Process Management, Business Process as a Service, BPMN, Process Modeling, Temporal Constraints and Dependencies, Temporal Semantics.

Abstract: A business process (BP) can be understood as a set of related, structured, interacting services acting as peers, according to an intended choreography that is capable of giving complex functionality to customers. Several authors have made progress in solving the "choreography realization" problem. The research work carried out in this paper amounts to analyzing and automatically checking the realizability of the defined choreography for services that communicate through messages in a general, distributed, and highly parallel system.

1 INTRODUCTION

Busines Process Model and Notation (BPMN) was extended into BPMN 2.0 to provide an interaction model for the business processes (BP) which is based on specification of *choreographies* as opposed to "interconnected interface models" (Poizat P., 2012) that promote *orchestration-oriented* specifications, which is a bottom-up and a local approach of service composition.

Realizability of choreographies consists of to formally determine if the individual peers obtained from a choreography, independently from the kind of communication or their relative execution order, are capable of interacting as prescribed in the choreography requirement specification. We can say that a choreography is realizable if all the interactions specified in BPMN 2.0 choreography-diagrams are equivalent to those that can be executed by the interacting peers when the BP model is implemented in a Web-services description language. In this way, a formalization of behavioral and temporal aspects of BPMN 2.0 models through a *provably-realizable choreography* will provide important benefits at analysis and design of complex applications built over distributed and highly parallel platforms.

There are already contributions to the resolution of the *choreography realization problem* (Rozinat A., 2006), (Dongen B., 2004). The research work up until now can be divided into two categories. The first approach aims at transforming BPMN models into executable environments (Capel M.I., 2014) and performing formal analysis (Aalst, 2009). The second one includes formal methods for BPMN models verification, which are based on Π-calculus (Milner, Hall) or Petri Nets (Cerone, 2002), which can debug grammatical errors and can transform business processes diagrams (BPD) into BP Execution Language (BPEL) code (Arkin A., 2005; OASIS, 2007), or representing BPM system properties with CCTL. In this second group, the central problem to tackle consists of proving the soundness of BPMN model transformation.

BPMN transformation is more than just individually converting the model's entities. In many cases, a model cannot be verified because its representation in an executable environment does not react to external events in the same way.

We propose here a solution to that problem based on the transformation of the BPMN and the peer-system models into a formal specification, realized in the timed process calculus named CSP+T, thereby allowing their complete verification with state-of-the-art model checking tools.

In Section 2 we introduce the use of BPMN 2.0 for choreography specification. Section 3 presents our formal model transformation from BPMN 2.0 choreographies into CSP+T process calculus. Section 4 shows how this encoding can be used to reify choreographies with LTS, and check their realizability. Model checking of choreographies is addressed in Section 5, and tool support is discussed in section 6. Finally, the conclusions and future leads of our work.

Capel, M.
Checking Realizability of a Timed Business Processes Choreography.
DOI: 10.5220/0006300404410448
In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pages 413-420
ISBN: 978-989-758-243-1

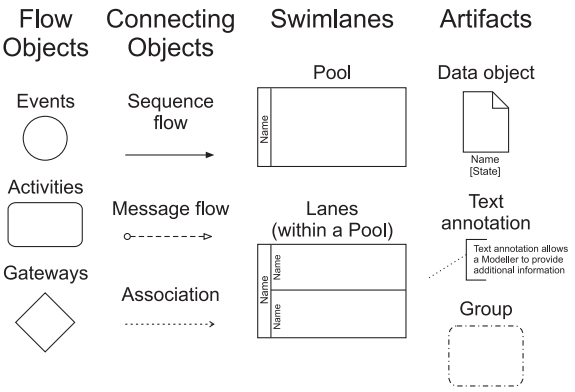

Figure 1: Graphical representation of BPMN elements.

2 BPMN CHOREOGRAPHIES FORMALIZATION

The graphical standardized notation Business Process Model and Notation (BPMN) (OMG, 2011) provides representation elements for business processes with an emphasis on control–flow. BPMN introduces a specific type of flowchart, named *Business Process Diagram* (BPD), which provides graphical constructs tailored to model BPMN elements, such as: (a)gateways, (b) events *start, stop*, (c)tasks, and (d)control flows. Non-technical personnel, usually management-oriented, can easily understand a BPD diagram and use it for modeling BPs. The apparent simplicity of BPD, however, offers the necessary expressiveness power to model very complex BPs and it can be mapped to different business execution languages such as BPEL (Arkin A., 2005) or XLANG (Thatte, 2001). Figure 1 shows BPMN elements as they are represented in BPD.

Semi-formal specification of task workflows can be done with BPMN notation.

BPMN 2.0 supports the collaboration between analysis entities in BP models, which brings forward a *choreographic* model based on peer interactions, instead of following a design model based on services orchestration. Thus, it promotes a collaborative and abstract description of software systems that allows for focusing more on what services do in a composition than on how they do it. Interactions between system's components or *peers* should be more precisely described now than in "interconnected interface" models, within which the interactions are defined internally to each peer only. Contrary to interface description–based, "interaction–based" models consider the description of "conversations" between peers as the basic building blocks of any BP system design, whereas the specification of interfaces then

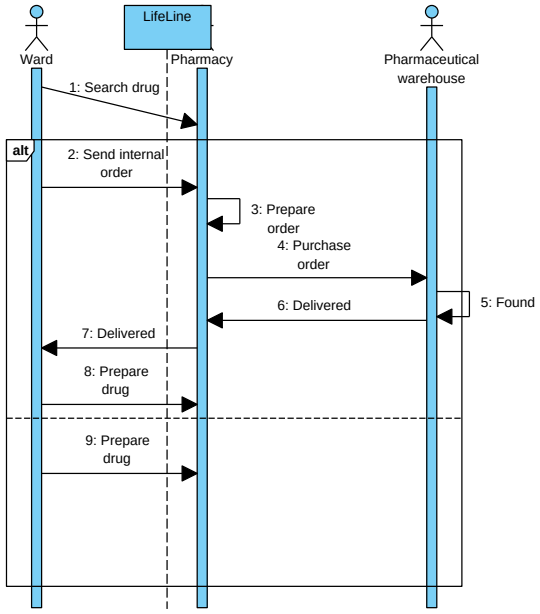

Figure 2: Ward-Pharmacy interaction as a UML sequence diagram.

becomes secondary to the system's properties analysis.

Case Study. We will use a Pharmacy Hospital logistic process as a simple running example. The UML sequence-diagram in Figure 2 depicts the message flows between two participants, the *Ward* and the *Pharmacy*, which are independent BP and may have been constructed separately. Clearly, the *synchtronization* between both participants is a necessary behavioural property for successful collaboration. The Ward-Pharmacy interaction choreography is realizable if that property is individually proved for each participant.

2.1 Coreography Description Language

BPMN 2.0 has introduced the Choreography Description Language (CDL) for specifying the requirements of a business process choreography. CDL is capable of defining, from an integrated point of view, the common and complementary observable behavior of different services that exchange messages according to some ordered protocol for accomplishing a common business goal (W3C, 2005).

Peers interactions are the basic *building blocks* of any CDL specification. Peers A and B are represented by the upper and lower bands, respectively, in the round boxes of Figure 3. One–way and two–way interactions between peers are easily described by BPMN 2.0 Choreography Diagrams (BCD).

In a choreography, two-way interactions of peers (tasks) are represented by message exchanging. A is called the initiating peer and is represented by a white band as opposite to the dark filled one for the the receiving peer B.

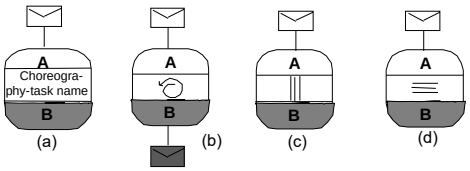

Figure 3: BPMN 2.0 choreography diagram tasks.

There are one-way interactions (Figure 3, a,c,d) and two-way interactions (Figure 3, b). Two-way interactions are represented by an intiating message (white envelope) and a return message (black envelope). Tasks can also include internal markers, such as the *standard loop* (Figure 3, b), in which the interaction is performed several times depending on a boolean condition. In *Multi–instance* parallel loops (Figure 3, c,d), the interactions are performed by several instances of the choreography task, which can execute the actions in parallel$\|\ |$ or sequentially $\equiv$. If the message exchange need not be repeated, no marker is used to describe a task.

Business decisions and flow branching are modelled using *gateways*, which are similar to decision symbols in a flowchart. Gateways describe how sequence flows join (multiple outgoing sequence flows and at most one incoming sequence flow), and fork (multiple incoming sequence flows and at most one outgoing sequence flow). In our description language we take into account (Figure 4) the following symbols for gateways: XOR exclusive gateways (decision, alternative paths), inclusive gateways (inclusive decision, but also parallel paths), parallel gateways (creating and merging parallel flows) and event-based gateways (choices based on events). Diagrams needing both converging and diverging choices can be described by a sequence of single-converging/diverging gateways.

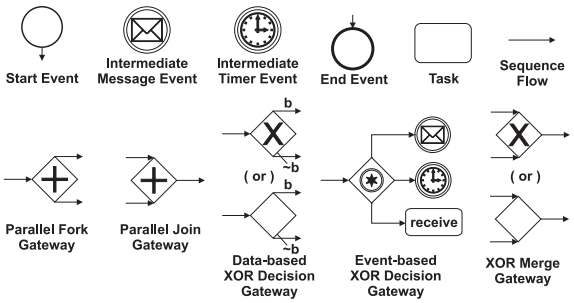

Figure 4: BPMN Elements Extended Graphical Representation.

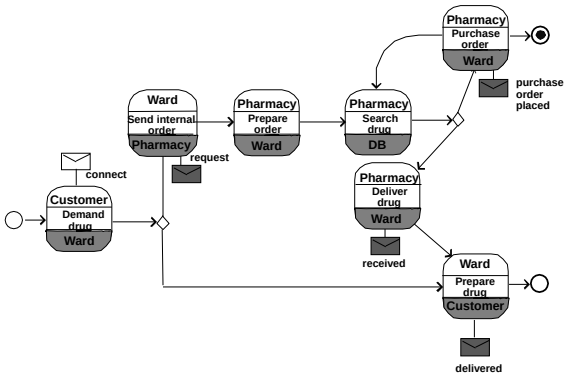

Figure 5: BCD of the Ward-Pharmacy example.

BCD of the Case Study. From the pharmacy-participant perspective, a `deliver drug` message is sent to the ward-participant for guaranteeing that a prior demanded drug is now available; and an `internal order` received prior a `purchase order` for a specific drug is made or sent. The ward-participant has to send the `demand drug` message with enough time in advance to guarantee one drug availability to begin a medical treatment.

The specification in the BCD (Figure 5) shows the interaction between the peers (`customer`,`ward`,`pharmacy`,`db`) of the choreography, translated from the business process model of 'Logistic Process in Hospitals' (Baacke L., 2009) that represents the behaviour of tasks within the 2 *pools* (BPMN notation to define task sequences): Ward and Pharmacy of the example. In this specification, we can firstly see that the customer interacts with the `ward` (`connect`), then the ward sends the prescription to the `pharmacy` (`request`) and eventually receives the drug when it is available (`received`). If that drug is not in stock, the `pharmacy` makes an order (`purchase order`), communicates (`purchase order placed`) to the ward and when the drug is available, it delivers the drug to the `ward`. Finally, the prescription will be prepared and given to the customer (`delivered`), thereby terminating the complete protocol.

WS-CDL Description. The Web Services Choreography Description Language (WS-CDL) is an XML-based language for describing Web services interactions (W3C, 2005). In WS-CDL there is no centralized control to coordinate different services; and therefore, there are no global variables, conditions or workunits either. In order to give the illusion of a global or shared state among the choreographed services, the variables located in one service can be aligned (synchronized) via message passing with other variables located in a distinct service. Message

sequences that do not follow the ordering rules are considered *out of sequence messages* and the language shows an error of conformance with respect to the description of the intended choreography. The lack of formal semantics of WS-CDL is currently considered an issue since it hinders the development of tool support for verifying message conformance to a specific choreography description. In addition to conformance checking, we need to determine whether a given choreography is *realizable*, i.e., if its individual peers obtained through projection from a choreography interact as prescribed in the specification of choreography requirements.

3 ENCODING BPMN 2.0 INTO CSP+T

CSP+T (Zic, 1994) is a real–time specification language which extends *Communicating Sequential Processes* (CSP) allowing the description of complex event timings, within a single sequential process, for use in the behavioural specification of any critical communicating process. A CSP+T process term $\mathcal{P}$ is defined as a tuple $(\alpha P, P)$, where $\alpha P = Comm\_act(P) \cup Interface(P)$ is the *communication alphabet* of P.

BPMN 2.0 into CSP. There have been several proposals to formalize BPMN 2.0 non–temporal constructs with process algebras, (Puhlmann, 2007; Ma S., 2008; Mendoza L.E., 2012).

In the CCS–based notation proposed in (Wong P.Y.H., 2008), which we will follow in the sequel, each BPMN entity has associated attributes describing its properties; for example the number of loops of a sequence multiple instance is recorded by the natural number in the constructor *miseq*.

The formal semantics of BPMN abstracts (partial function *hide*) the internal flow of the modeling entity named *state* and only describes the sequence of initializations and terminations with the semantic function *bpmn*.

The type of a sequence flow or an exception flow is given by the following schema definition:

$$Transition \mathrel{\widehat=} [guard : Guard;\ line : Line]$$

and the type of message flow:

$$Message\_flow \mathrel{\widehat=} [message : Message;\ channel : Channel]$$

If the sequence flow has no guard or the message flow contains an empty message, then the values of Transition and Message_flow record the default values "*tt*" and empty respectively. There are five sets of message flows (*send, receive, reply, accept, break*) associated to the *state* entity, which are syntactically defined in the next type and whose function follows from their names,

$$\begin{aligned} State \mathrel{\widehat=} [&type : Type;\ in, out, error : \mathbb{P}\, Transition;\ exit : \\ &\mathbb{P}(\mathbb{N} \times Transition);\ send, receive, reply, accept, break : \\ &\mathbb{P}\, Message\_flow;\ link : \mathbb{P}(Transition \times Message\_flow); \\ &depend : \\ &\mathbb{P}(Message\_flow \times Message\_flow);\ loopMax : \mathbb{N}] \end{aligned}$$

Each state also incorporates the variable *loopMax* to limit the number of state instances that each process can invoke. The state's component *link* pairs the incoming message flow which initiates or interrupts the execution of the state with either an incoming transition or an exception flow. The component *depend* pairs each incoming message that initializes the state's execution with its corresponding outgoing message flow.

$$\begin{array}{|l} [bpmn] \\ \hline bpmn : \mathbb{P}\, Name \nrightarrow Local \nrightarrow Process \\ hide : \mathbb{P}\, Name \nrightarrow Local \nrightarrow \mathbb{P}\, Event \\ \hline \mathsf{function\_name} = (Y \mid [\alpha Y]X) \backslash hide(\mid S \mid)) \\ \mathsf{Where}\, X = \Box_i : \alpha Y \backslash hide(fin, abt) \bullet \\ (i \rightarrow X \,\Box\, fin \rightarrow null \,\Box\, abt \rightarrow stop) \\ \mathsf{And}\, Y = (\|_i : Process\_set \bullet \alpha(P_i) \circ P(i)) \end{array}$$

The partial function *bpmn* maps a syntactic description of a BPMN diagram encapsulated by a *pool* or BPMN–*subprocess* into a parallel composition of CSP+T subprocesses, which correspond to the diagrams of the basic elements of BPMN shown in Figure 1. The set αY represents the communication alphabet that includes all the messages types and events that may affect the execution of processes. These communications represent the events that a process P receives from its *environment* (made up of all the other processes in the system) or those events that occur internally, i.e., which are not externally visible.

3.1 BPMN 2.0 Temporal Extension

In many models of choreographies, constraints on time and resources appear that may cause the violation of the system's safety properties.

In BPMN 2.0 the intermediate event (*timer*) admits the definition of a delay period, but the minimum and maximum execution time allowed for any activity cannot be specified as a task element.

We propose to extend BPMN 2.0, so that includes temporal attributes for specifying temporal constraints for *task* and other modelling elements. An intermediate event (*timer*) and a sub–process (*timeout*) represent a temporal constraint on the task flow. Notice that in BPMN 2.0 we can use the attribute of the *timer* element to define the task delay period.

Start Event. The schema labelled *Start event* represents the necessary instantiation of an activity prior to the start of its execution. To allow this in CSP+T, the specification of this event is represented by means of the ★ *instantiation event* according to the following pattern,

$$P(start) = \bigstar \bowtie v_{\bigstar} \rightarrow SKIP \fatsemi (P(S1) \Box \varepsilon_{end} \rightarrow SKIP)$$

Minimum and Maximum Duration Time of an Activity. The invocation of activities that make up a BP must be performed timely.

Let *bpmn S*1 be the activity that comes before activity *bpmn S*2 (figure 6). Thus, it should be guaranteed that the execution time of *bpmn S*1 activity does not overrun its maximum range of duration ($S1.ran.max$) and it does not occur before its minimum starting time ($S1.ran.min$) elapses. If the above times are not controlled, the activity fails and thus we cannot certify the temporal properties of the business process.

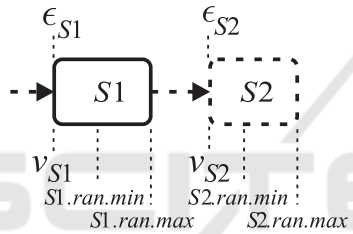

Figure 6: Temporal Annotations of the Activity.

Thus, the occurrence of ε_{S2} must satisfy the condition,

$$v_{S1} + S1.ran.min \leq s(\varepsilon_{S2}) \leq v_{S1} + S1.ran.max.$$

With CSP+T we are allowed to precisely specify the time frame for the execution of an activity. The CSP+T pattern corresponding to two consecutive activities (*S*1 and *S*2) is as it follows,

$$\begin{aligned} P(S1) = {} & \varepsilon_{S1} \bowtie v_{S1} \rightarrow SKIP \fatsemi \\ & (\mathrm{I}(Time, v_{S1} + S1.ran.min).\varepsilon_{S2} \rightarrow SKIP \fatsemi P(S2) \\ & \Box \varepsilon_{end} \rightarrow SKIP) \\ & Time = S1.ran.max - S1.ran.min \end{aligned}$$

The measured range values $S_x.ran.min$ and $S_x.ran.max$ depend on event ε_{Sx} occurrence.

Timed Exception Flow. BPMN proposes two versions of type boundary event to represent timeouts. *Timer boundary* represents the occurrence of an event ε_{exc} (see Figure 7) that either triggers a parallel thread or interrupts the execution of activity *bpmn S*1 at *itime.ran* time units after the inception of S_1. Hence, there are two instances of the modelling entity *timer boundary even*, depending on whether the activity started at ε_{S1} is interrupted or not when this event occurs. The *itime.ran* time period must be therefore constrained by the maximum time limit that we have associated to any activity.

$$\begin{aligned} & (itime.ran < S1.ran.max) \wedge (s(\varepsilon_{exc}) = \\ & v_{S1} + itime.ran) \wedge (s(\varepsilon_{exc}) \in [v_{S1}, S1.ran.max)) \end{aligned}$$

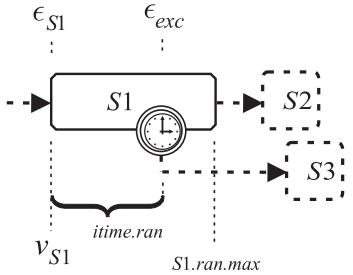

Figure 7: Timed Exception Flow.

To specify the behavior denoted by a *interrupting timed exception flow*, we use the *interrupt* operator ($\triangle$) in CSP+T, according to the following pattern,

$$\begin{aligned} P(S1) = {} & (\varepsilon_{S1} \bowtie v_{S1} \rightarrow SKIP \fatsemi \\ & (\mathrm{I}(Time, v_{S1} + S1.ran.min).\varepsilon_{S2} \rightarrow \\ & (SKIP \fatsemi P(S2) \bigtriangleup \mathrm{I}(itime.ran, v_{S1}) \rightarrow SKIP \fatsemi \varepsilon_{exc} \rightarrow \\ & SKIP \fatsemi P(S3)) \\ & \Box \varepsilon_{end} \rightarrow SKIP)) \\ & Time = S1.ran.max - S1.ran.min \end{aligned}$$

To represent a *non–interrupting exception flow*, we only need to substitute the *interrupt* operator ($\triangle$) for the parallel operator $\|$ or the interleaving $|||$ of activities in CSP+T.

4 CHOREOGRAPHY ANALYSIS

Any choreography can be considered *realizable* if all the interactions that we have specified in the BPMN 2.0 diagram (e.g., the one in Figure 5) are equivalent to those that can be executed by the interacting peers when we implement the model in a service-description language, such as WS-CDL.

Labelled transition system models (LTS) (Bezem M., 2003) *reify* choreographies and then allows the verifier to check its realizability w.r.t. the model of the system composed of interacting peers, which are also individually described as LTS (Figure 8). The *reification* of the case study choreography is shown in Figure 9.

4.1 Behavioral Equivalence

We encode both, the choreography and the interacting peers, into *bpmn* process terms that are semantically defined by the CSP+T (Zic, 1994) process calculus. In this way, we can count on a sound and well defined formalization of behavioral aspects of BPMN models in order to allow the designer to carry out an analysis of choregraphies.

The proposed analysis procedure for checking choreography realizability consists of the following integrated steps,

1. Generate the CSP+T encoding of a given choreography.
2. The derived choreography reification (LTS) is generated from the CSP+T description.
3. For the extracted interacting peers of the choreografy, each peer is encoded as one CSP+T process (using the *bpmn* pattern).
4. The distributed system model is built as a parallel composition of the structured process terms.
5. The choreography reification is *model–checked*. to prove behavioral equivalence with the peer-based and distributed system model (in 4).

If the two models mentioned above are *behaviorally* equivalent then the model-checker response is void, meaning that the peer generation exactly satisfies the BPMN communication requirements. On the contrary, if the peers do not generate the same interactions as the ones specified in the choreography then a counter-example is returned by the model-checker, and we can say that this choreography is unrealizable.

5 CHOREOGRAPHY MODEL CHECKING

For each participant in the BCD, we specify a parallel composition of parallel CSP+T processes, which is given by a syntactically correct *bpmn* term. The proposed formal specification abstracts the internal interaction between the individual peer states and only represents the sequence of task initializations and terminations that occur in the choreography model. A compact model susceptible of being transformed into an LTS, and then verified by an automatic tool, is obtained.

The states of each two main interacting peers in the BCD of Figure 5 are represented by the following *bpmn* pattern of a CSP+T process,

$$
\begin{aligned}
&bpmn\\
&name = (Y \mid \alpha(P_i) \mid X)\ hide\{connect.Customer\}\\
&\quad Y = \|_{i=1}^{i=m} Tasks \bullet \alpha(P_i) \circ P(i)\\
&name = (Customer \mid Ward \mid Pharmacy \mid DB);\ (peers)\\
&Tasks = (Demand\,drug \mid Send\,internal\,order\\
&\quad \mid Prepare\,order \mid Search\,drug \mid Deliver\,drug\\
&\quad \mid Purchase\,order \mid Prepare\,drug);
\end{aligned}
$$

The parallel composition of CSP+T {P(i)} processes is mechanically obtained according with the language operators rules.

The process terms above are *reified* by the two LTS in Figure 8: (a) for *bpmn Ward* and (b) for *bpmn Pharmacy*.

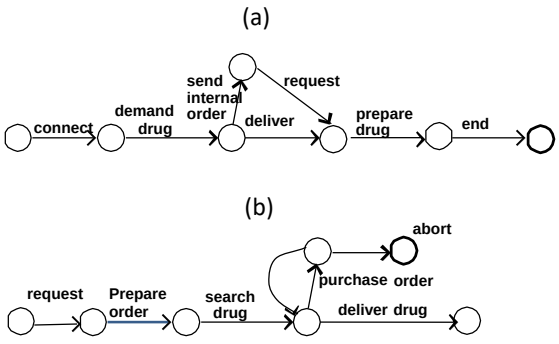

Figure 8: LTS models of peers.

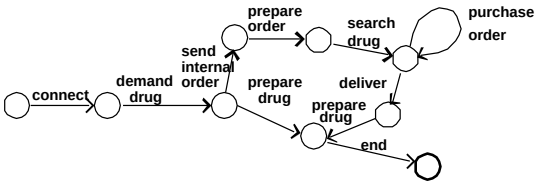

Figure 9: LTS model of the Ward–Pharmacy Example.

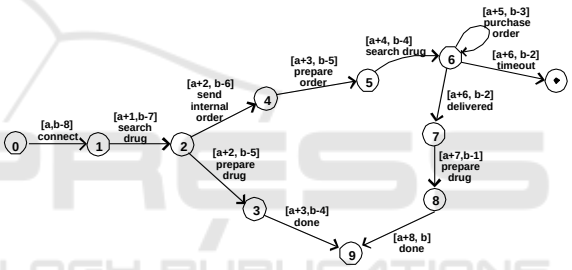

Figure 10: Timed LTS model of the Ward-Pharmacy.

5.1 Choreography Reification

$P(System) = Ward \| Pharmacy, \ \| Customer \| DB$ CSP+T term is the choreography encoding that should be tranformed into a timed LTS as the one shown in Figure 9, and then checked against the peer system-LTS (Figure 8) before starting the implementation of the distributed application system.

5.2 Verification

In our proposal, the reification of the choreography $PSystem(LTS)$ is considered realizable if the set of interactions specified by the process term $P(System)$ and those executed by the interacting peers in $P_{BPMN}(LTS)$, are the same. Thus, according to *traces and failures* semantics of CSP (Schneider, 2000), it must be ascertained that the following refining assertion is true,

$$PSystem(LTS) \sqsubseteq_F P_{BPMN}(LTS) \qquad (1)$$

The model-checker FDR2 (FormSys, 2005), however,

returned *false* since the trace $<$ `connect`, `demand drug`, `send order`, `prepare order`, `delivered`, `prepare drug` $>$ appears in both models, but the trace $<$ `connect`, `search drug`, `prepare order`, `send order`, `purchase order`, `abort` $>$ is present in the peers–based distributed system and not in the LTS of the choreography.

The solution to this error in the LTS model is to make explicit an extra state in which the LTS is waiting for completing the purchase of the drug and add a timeout to this state. If that time period expires then the LTS will reach an `abort` state signifying that the purchase is cancelled, since probably the distributor's stock has been exhausted. The new LTS can be seen in Figure 10.

6 SUPPORTING TOOLS AND MEASUREMENTS

In order to fully support our approach the BTransformer (González A., 2011) tool has been implemented on top of the Eclipse platform (`http://www.eclipse.org`) as a plugin. BTransformer allows a business process designer to generate a specification in CSP+T from a BPMN model. The tool allows for the creation, the editing, the deleting and the storing of a set of the transformation rules of BPMN 2.0 into CSP+T. Since BTransformer tool is able to read input/output models written in standard XML files, it can be easily extended to encode BPMN 2.0 models into other notations of which can be defined transformation rules that are semantically sound. For instance, the set of transformations of BPMN 2.0 into LOTOS NT described in (Poizat P., 2012) have been recently used to extend the output models generated by BTransformer into LOTOS NT processes, which is one of the input formats accepted by CADP Toolbox (Garavel A., 2011) for checking BP models *realizability* and *verification* (Figure 11).

BTransformer permits to automatically generate formal specifications in several process languages, such as CSP+T and LOTOS NT, and business process orquestation languages as WS-BPEL. BTransformer is based on the *model tranformation language* ATLAS (ATL) (Jouault F., 2008) and is implemented as an Eclipse plugin, which is capable of trasforming BPMN 2.0 models designed with the editor Intalio (`http://www.intalio.com`) into output files of different formats. A UML class diagram that shows the components of the plugin that provides the functionality of BTransformer tool is shown in Figure 12. A detailed description of BTransformer implementation is given in (González A., 2011).

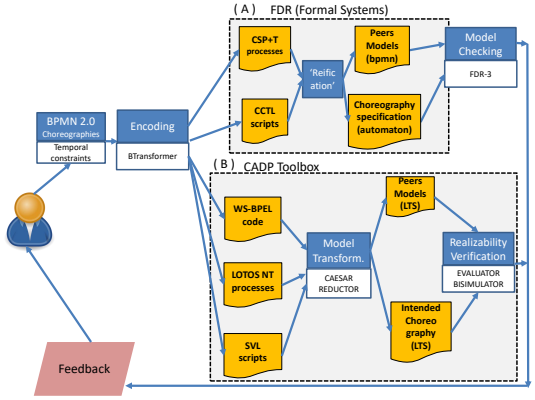

Figure 11: Software tools support.

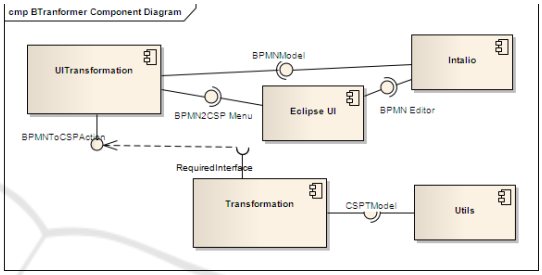

Figure 12: BTransformer Tool Components.

The BPMN model and case study used to assess the results on the choreography realization model checking and the operation of the implemented BTransformer tool were taken from the "Logistic Process in Hospitals' (Baacke L., 2009). A series of heuristics or criteria have been defined in order to detect inconsistencies between the notational BPMN elements in the source model and processes generated, according to our proposal, for the target model :

- *Completeness*: all elements in the source BPMN-model diagram appear reflected in their semantically equivalent specifications as process terms.
- *Number of processes*: there is at least one process activity defined in the diagram.
- *Completeness of relations*: it is possible to establish relationships between two processes as long as they present a relationship between two or more activities in the model.
- *Behavioral safety*: the set of execution sequences of the processes in the specification must be included (or coincide with) in the set of sequences of the model.

Table 1 shows the results obtained by applying the *consistency* criteria to the CSP+T encoding obtained with BTranformed tool for the complete case study.

Table 1: Results of tests obtained with BTransformer tool.

Model	Completeness of elements	Number of processes	Completeness of relations	Behavioural safety
Hospital Logistics	✓	22	✓	✓

7 CONCLUSION

In order to enforce the realization of feasible choreographies within the realm of business proceses, we have presented a feasible formalization of a subset of BPMN 2.0 constructs.

Consequently, a possible solution to the choreography realization problem is presented here by encoding BCD (*BPM 2.0 Choreography Diagrams*) modelling-entities into a proces calculus named CSP+T. In this way, we can analyze and automatically verify the realizability of a defined choreography into services that communicate through messages in a general, distributed, and highly parallel system.

In a longer term perspective, we also aspire to develop supporting tools for business analysts and modelers to find a way to improve the quality of their business models.

REFERENCES

Aalst, W. (2009). Challenges in business process analysis. In *In: Enterprise Information Systems. Lecture Notes in Business Information Processing, v. 12, 27:42.* Springer, Berlin, Heidelberg.

Arkin A., Askary S., e. (2005). Committee draft. In *Services Business Process Execution Language Version 2.0.* WS-BPEL TC OASIS.

Baacke L., Mettler T., R. R. (2009). Component-based process in health care. In *In:17th European Conference on Information Systems*.

Bezem M., Klop J.W., R. d. V. (2003). *"Terese". Term rewriting systems*. Cambridge University Press.

Capel M.I., M. L. (2014). Choreography modeling compliance for timed business models. In *In: Barjis J., Pergl R. (eds) Enterprise and Organizational Modeling and Simulation. EOMAS 2014. Lecture Notes in Business Information Processing, vol 191, 202:218.*

Cerone, A. (2002). From process algebra to visual language. In *Proceedings of the Conference on Application and Theory of Petri Nets: Formal Methods in Software Engineering and Defence Systems, v. 12*.

Dongen B., A. W. (2004). Multi–phase process mining: Building instance graphs. In *In: Conceptual Modeling–ER 2004. Lecture Notes in Computer Science, v. 3288, 362:376*. Heidelberg:Springer.

FormSys (2005). *Failures–Divergence Refinement – FDR2 User Manual*. Formal Systems Europe Ltd, Oxford, 2nd edition.

Garavel A., e. a. (2011). Cadp 2010: A toolbox for the construction and analysis of distributed processes. In *Proceedings of TACAS'11,v.6605 of LNCS, 372:387*. Springer.

González A., Mendoza L.E., C. M. (2011). Btransformer: A tool for bpmn to csp+t transformation. In *Proceedings of the 13th International Conference on Enterprise Information Systems (ICEIS), Volume 3, Beijing, China, 8-11 June, 2011*. ScitePress.

Jouault F., Allilaire F., e. a. (2008). Atl: A model tranformation tool. In *Science of Computer Programming, 72, 31:39*.

Ma S., Zhang L., H. J. (2008). Towards formalization and verification of unified business process model based on pi calculus. In *Proceedings ACIS International. Conference on Software Engineering Research, Management and Applications 1*.

Mendoza L.E., Capel M.I., P. M. (2012). *Conceptual framework for business processes compositional verification*. Information and Software Technology,54,149:161.

Milner, R. (Prentice-Hall). *Communication and Concurrency (International Series in Computer Science)*. The publishing company.

OASIS (2007). *Web Services Business Process Execution Language Version 2.0*. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

OMG (2011). *Business Process Model and Notation (BPMN)–version 2.0*.

Poizat P., S. G. (2012). Checking the realizability of bpmn 2.0 choreographies. In *TEMPLATE'06, 1st International Conference on Template Production.In: 27th Simposium of Applied Computing, 1927:1934, Riva del Garda(Italy), March 25-29*. ACM.

Puhlmann, F. (2007). Soundness verification of business processes specified in the pi-calculus. In *Lecture Notes in Computer Science, no.4803: 6-23*. Elsevier.

Rozinat A., A. W. (2006). Conformance testing: Measuring the fit and appropriateness of event logs and process models. In *In: Business Process Management Workshops. Lecture Notes in Computer Science*. Elsevier.

Schneider, S. (2000). *Concurrent and Real–Time Systems – The CSP Approach*. John Wiley & Sons, Ltd.

Thatte, S. (2001). *XLANG: Web Services for Business Process Design.Microsoft Corporation, 2001*.

W3C (November 9, 2005). W3c candidate recommendation. In *Web Services Choreography Description Language Version 1.0*. http://www.w3.org/TR/ws-cdl-10/.

Wong P.Y.H., G. J. (2008). A process semantics for bpmn. In *International Conference on Formal Engineering Methods, ICFEM 2008 (Kitakyushu-City), Japan, October 27-31, 2008. In* Lecture Notes in Computer Science *5256, 355:374*. Heidelberg:SpringerS.

Zic, J. (1994). Time–constrained buffer specifications in csp+t and timed csp. In *ACM TOPLAS,16(6), 1661:1674*. ACM.