Towards Compliance Verification Between Global and Local Process Models

Pieter M. Kwantes[1](✉), Pieter Van Gorp[2], Jetty Kleijn[1], and Arend Rensink[3]

[1] LIACS, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands
p.m.kwantes@liacs.leidenuniv.nl
[2] Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[3] Department of Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract. This paper addresses the question how to verify that the local workflow of an organisation participating in a cross-organisational collaboration is in compliance with the globally specified rules of that collaboration. We assume that the collaborative workflow is specified as a BPMN Collaboration Diagram and the local workflows as BPMN Process Diagrams. We then employ existing LTL semantics of the former and token semantics of the latter to verify conformance. We use the graph transformation tool GROOVE to automate the verification, and exemplify our approach with a case study from the financial markets domain.

1 Introduction

The development of computer network technology and distributed systems running on top of those networks has enabled a tighter integration between automated operations across organisational boundaries. Any organisation aiming to participate effectively in a cross-organisational collaborative workflow must ensure that the design of its internal operations complies with the rules of that collaboration. This paper addresses the question how to verify such compliance. We propose an approach starting from Business Process Modelling Notation (BPMN) specifications (version 2.0, [20]) in which inter-organisational workflows (or global behaviour) are specified as BPMN Collaboration Diagrams (BPMN CD, for short) while intra-organisational workflows (local behaviour) are specified as BPMN Process Diagrams (BPMN PD). Note that the global behaviour of a collaboration is the public, communicating, behaviour collectively exhibited by all participants. The local behaviour of a participant consists of its communicating behaviour and possibly additional, private (non-communicating), behaviour. The GROOVE — GRaphs for Object-Oriented VErification —, tool [13,23] can be used to automate the verification process. GROOVE includes a model checker for automated verification of state spaces against a Linear-time Temporal Logic (LTL) formula [22]. In order to leverage that for our verification scenario, we have to translate the BPMN CD into an LTL formula which represents a

© Springer International Publishing Switzerland 2015
F. Parisi-Presicce and B. Westfechtel (Eds.): ICGT 2015, LNCS 9151, pp. 221–236, 2015.
DOI: 10.1007/978-3-319-21145-9_14

behavioural constraint on the participants of the inter-organisational collaboration. For the translation of collaboration diagrams into LTL we follow the set-up of [4] where BPMN workflow specifications are considered as possible visual alternatives for LTL formulae and an LTL semantics for BPMN 2.0 is provided. On the other hand, in [14], a formal semantics of BPMN 2.0 is provided in the form of graph transformation rules. In order to answer our research question, we have implemented in GROOVE the rules from [14] and we have added some rules specifically for message-driven collaborations between partner organizations. This rule set enables GROOVE to compute the state space representing the behaviour of a participant and verify it against an LTL formula. Finally, we apply our proposed approach to an example from the financial markets domain.

Paper outline. In Sect. 2 we discuss the syntax and semantics of BPMN Collaboration Diagrams and BPMN Process Diagrams to specify global and local process models respectively. In Sect. 3 we describe an implementation allowing automated verification of local process models against LTL-formulae derived from global process models using the GROOVE tool. In Sect. 4 we test the proposed implementation using a case study from the financial markets domain. In Sect. 5 we discuss related work. In Sect. 6 we discuss a number of issues encountered during our research, and future work.

2 Process Modelling in BPMN

2.1 Global Behaviour

In this paper, the global aspects of an inter-organisational collaboration are specified as a BPMN Collaboration Diagram. Such a diagram describes the communicating behaviour of all participating organizations.

Syntax of BPMN Collaboration Diagrams. We discuss here only the subset of available BPMN elements used in the example diagrams in Sect. 4. This subset of elements is shown in Fig. 1. A BPMN CD consists of pools each delineating the workflow of an individual participating organisation. Events and tasks are the active elements in a workflow. Each workflow begins with a start event and finishes with an end event. There are two types of intermediate events: a message event (marked with a small envelope) represents the receipt of a message and a timer event (with a clock) indicates a timing requirement or delay. In diagrams, instances of events and tasks are usually labelled with a name describing the activity they represent. Gateways model the flow of control. Both the exclusive-or gateway (marked with an "X") and the event-based gateway (displayed as a pentagon inside a circle) indicate an exclusive choice. In the first case, the choice is coincidental, whereas the choice of an event-based gateway is triggered by events. Within the workflow of an organisation, active elements and gateways are connected by sequence flows (arrows) indicating the flow of control. Message flows represent the exchange of messages between organisations and connect a (sending) task of one workflow with a (receiving) message event in another workflow.

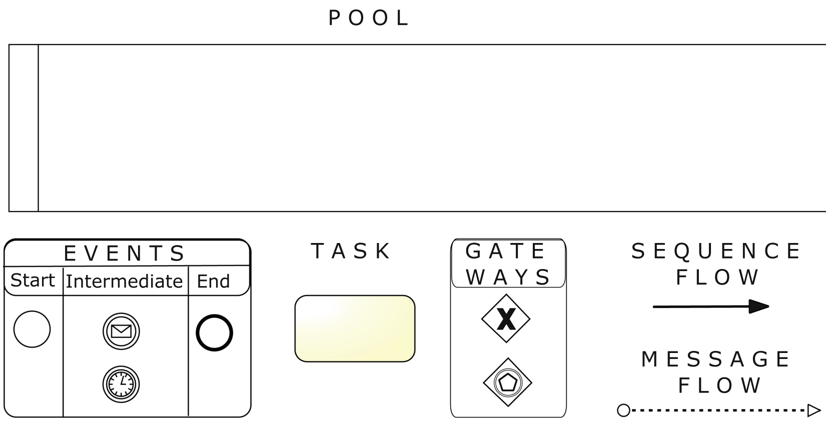


Fig. 1. BPMN Symbols used in this paper

LTL-Semantics of BPMN Collaboration Diagrams. We follow the approach of [4] to translate a BPMN Collaboration Diagram into an LTL formula [5,21,22]. We will use propositional symbols as atomic propositions, the usual Boolean combinators ($\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$), and Until (U), Eventually (F) and Global (G) as temporal combinators. We do not need the Next combinator [4]. The Boolean combinator "exclusive or" denoted by xor is used as a shorthand with $\Phi\ xor\ \Psi$ semantically equivalent with $\neg(\Phi \leftrightarrow \Psi)$. The less known past LTL-combinator Before (B) as it appears in the translation rules described in [4] can be replaced by an Until construct (see [4,12]). To avoid confusion and because B is not supported by GROOVE, rather than $\Phi\ B\ \Psi$, we will use the semantical equivalent $\neg(\neg\Phi\ U \neg\Psi)$ which expresses that either Ψ will always hold or Φ will hold some time before Ψ becomes false. The syntax of the LTL-fragment used in this paper is summarized below:

$$\begin{aligned} \Phi, \Psi ::= & P_1 | P_2 | \ldots && (\textit{atomic propositions}) \\ & | \neg\Phi \,|\, \Phi \wedge \Psi \,|\, \Phi\, xor\, \Psi | \Phi \vee \Psi | \Phi \rightarrow \Psi | \Phi \leftrightarrow \Psi && (\textit{boolean combinators}) \\ & | \Phi\ U\ \Psi | F\ \Phi | G\ \Phi && (\textit{temporal combinators}) \end{aligned}$$

Following [4], tasks and intermediate events are activities that define atomic propositions. The status of these activities is of interest: they are *active* or *completed*. In this way, every activity A has two atomic propositions as its counterparts: atomic proposition Aa standing for A being active and atomic proposition Ac standing for A being completed. We also have atomic propositions for gateways to be able to explicitly indicate the flow of control. For Gateways no distinction is made between active or completed. For readability we use square brackets in the atomic propositions. Sequence flows are used to identify meaningful fragments (relating tasks, events, and gateways) and form the basis of the translation. As in [4], the translation is not based on single elements, but on meaningful fragments of the diagram (connected by sequence flows, see

Table 1). The LTL formulae derived from these fragments are combined using conjunction. A Sequence (representing a sequence flow) combines two activities or gateways and is translated into a formula indicating that either the second activity (gateway) never becomes active or the first one has been completed first. Our gateways represent exclusive choice and as such can occur as splitting or as merging the flow of control. The start event and the end event translated in LTL formulae indicate that the workflow will eventually begin and eventually finish. All this gives us the set of translation rules shown in Table 1. The translation of a BPMN-collaboration diagram into an LTL-formula, using the rules in Table 1, involves the following steps:

1. Select the relevant part of the Collaboration Diagram: i.e. the part that corresponds to the local workflow that is verified.
2. Identify the BPMN model fragments included in the selected part of the Collaboration Diagram.
3. Translate each identified BPMN model fragment into a corresponding LTL-formula using the translation rules mentioned above.
4. The conjunction of the LTL-formulae resulting from step 3 provides us with one single LTL-formulae, which completes the translation.

In Sect. 4.2 we give an example of this translation process.

2.2 Modelling Local Behaviour in BPMN

Syntax of BPMN Process Diagrams. The symbols and syntactical rules to create BPMN Process Diagrams are largely the same as those given in Sect. 2.1 for BPMN Collaboration Diagrams. There are some differences however. The number of Pools is restricted to one, as a Process Diagram represents the workflow of one participant and there are *no* Message Flows, because these always connect two Pools. An extension is that there *are* non-communicating or *private* activities present, represented by BPMN Tasks which are not associated with a Message Flow. Examples of BPMN Process Diagrams are discussed in Sect. 4.

Token Based Semantics of BPMN Process Diagrams. The BPMN specification [20] contains an informal semantics definition in terms of tokens. Conceptually, this is similar to Petri Nets, where executions are also modeled as tokens that travel across net elements. A big difference though, is that Petri Nets contain only one type of active element (i.e., the transition) while BPMN has a multitude of elements (e.g. Gateways, Events and Tasks), all with their own behavioural characteristics. Additionally, beyond tokens, the BPMN semantics is defined in terms of process instances, which have their own lifecycle information. Therefore, while the semantics of Petri Nets can be defined with just one graph transformation rule, it requires a multitude of rules to define the BPMN semantics formally. In [14], the largest subset of BPMN process elements so far was formalised as visual, in-place graph transformation rules. For each supported

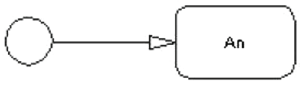
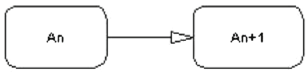
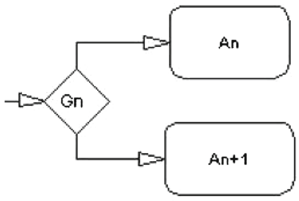
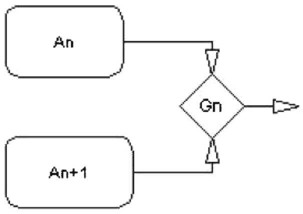
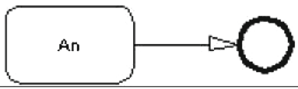
BPMN element, two rules were defined: one rule which activates the BPMN element and a second rule for modeling the completion of the BPMN element. This leads to rules with names such as "enterTask" "leaveTask", "enterSubProcess", "leaveSubProcess", etc. With this rule set, every valid execution of a specific BPMN process can be represented as a sequence of occurrences of these rules.

In Sect. 3, we first demonstrate how a GROOVE implementation of the rule set can be used as the basis for evaluating LTL expressions on graphs that represent all possible occurrences of the rules, for an input BPMN model. With that tool infrastructure in place, evaluating the LTL expression imposed by a global collaboration diagram is just one of many possible applications.

3 Implementation in GROOVE

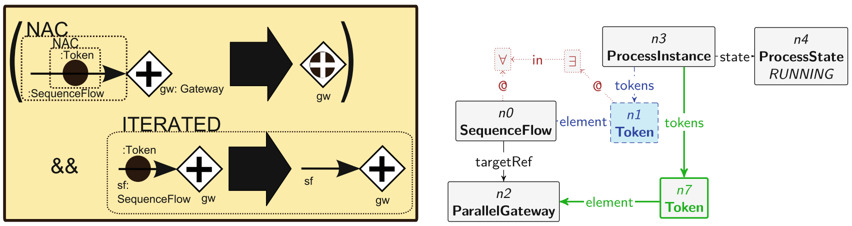
GROOVE is a graph transformation tool with unique verification capabilities. It is particularly strong in evaluating LTL, CTL and even PROLOG expressions on statespaces. Statespaces are produced by applying a graph transformation

Table 1. Translation rules based on [4]

	BPMN Fragment	**LTL Rules**
Start Event		$F\,[A_n a]$
Sequence		$\neg(\neg[A_n c]\,U\,[A_{n+1}a])$
XOR-Split Gateway		$\neg(\neg[G_n]\;U\;(F[A_n a]\,xor\,F[A_{n+1}a]))$
XOR-Merge Gateway		$\neg(\neg([A_n c]\,xor\,[A_{n+1}c])\;U\;[G_n])$
End Event		$F\,[A_n c]$

rule set non-deterministically on a given input graph. In this paper, we rely on the LTL capabilities only.

In order to leverage the GROOVE tool for the envisioned BPMN verification support, the rules from Sect. 2.2 have been implemented in GROOVE's graph transformation language. Figure 2(a) shows one of the various rules from [14] while Fig. 2(b) shows the implementation of this rule in GROOVE syntax. The example rule expresses when and how a token can enter a BPMN AND gateway: when each of the incoming sequence flows hold at least one token, the rule's preconditions are satisfied. Upon applying the rule, one token should be removed from each incoming sequence flow. Additionally, a token should be added to the AND gateway.



(a) Conceptual rule definition from [14] (b) GROOVE rule implementation

Fig. 2. Implementing the "enterParallel" rule from [14] in GROOVE.

Figures 2(a) and 2(b) demonstrate some key differences in rule specification style. First of all, Fig. 2(a) is a rewrite rule in concrete syntax while Fig. 2(b) is in abstract syntax. Second, the conceptual rule from Fig. 2(a) explicitly separates the left- and right-hand sides. In contrast, the GROOVE rule from Fig. 2(b) combines the left- and right-hand sides in one rule graph. Blue elements are parts of the left-hand side which are no part of the implicit right-hand side (i.e., they should be removed upon a match) while green elements are parts of the implicit right-hand side which are no part of the left-hand side (i.e., they should be created upon a match). Third, Fig. 2(a) shows the use of an embedded subrule. Finally, it also relies on a nested double Negative Application Condition (NAC) to express the "for each incoming flow" condition, while the rule from Fig. 2(b) relies on the built-in GROOVE ∀ operator. Further details are outside the scope of this paper since the focus here is on what these rules enable rather than on how they are realized.

Figure 3 shows an example process model which we can give as input to GROOVE and to which we can apply our GROOVE implementations of the rules from [14]. The example model includes four tasks. Due to the BPMN AND split and join (resp. the branching and merging gateway with the "+" sign), tasks T2a and T2b are allowed to be executed in parallel, so they can be activated and completed in any locally interleaved order. However, first T1 needs to be completed and only when both T2a and T2b are completed can task

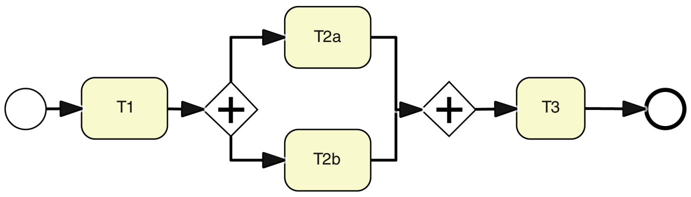


Fig. 3. Example BPMN 2.0 model for checking LTL formulae.

T3 be activated. The following LTL formulae can be executed on the GROOVE statespace, to demonstrate that our tool supports the automatic verification of some related temporal properties:

1. $G('leaveTask(``T2a")' \rightarrow F'leaveTask(``T3")')$ is an LTL expression to check whether in the statespace it holds that for every application of the rule "leave-Task" to BPMN element named "T2a" it holds that some time afterwards the rule "leaveTask" can be applied to element "T3". When executing this expression in GROOVE, we get the guarantee that the property is satisfied for the input model.
2. $G('leaveTask(``T2a")' \rightarrow F'leaveTask(``T2b")')$ is almost the same as the previous expression yet takes T2b as the second task. In this case, GROOVE detects that the property is not satisfied and it gives as a counter-example a sequence in terms of parameterised rule applications (e.g., $enterTask(``T1'')$, $leaveTask(``T1'')$, $enterParallel()$, $leaveParallel(\,)$, $enterTask(``T2a'')$, $enterTask(``T2b''), leaveTask(``T2b''), leaveTask(``T2a''), enterParallel(\,)$, $leaveParallel(), enterTask(``T3''), leaveTask(``T3''))$.

In Sect. 4, we apply this set-up for the envisioned verification of global collaboration constraints against locally defined process diagrams to a more realistic example from the financial markets domain.

4 A Case Study: The Settlement Process

In this Section we discuss a case study demonstrating the approach presented in the previous sections. In Sect. 4.1 we provide a short introduction into the *Settlement process* and a BPMN Collaboration Diagram representing this Settlement process. The translation of this Collaboration Diagram into an LTL-formula is given in Sect. 4.2. The Process Diagrams representing local behaviour in Sects. 4.3 and 4.4 are respectively in conformance and in violation of the global behaviour represented by the LTL-formula. These Process Diagrams are subsequently used to demonstrate our implementation, which is discussed in Sect. 4.5.

4.1 BPMN Collaboration Diagram of the Settlement Process

The settlement process is concerned with the processing of transactions on secondary capital markets. While primary capital markets are involved in the creation or issuing of financial assets, secondary capital markets are markets where already existing financial assets are traded. The exchange of financial assets in secondary markets is a process that is composed of a number of clearly defined stages. The first stage is the "trading stage", where market participants try to close a deal. The next stage is the "clearing stage", in which the accountability for the exchange of funds and financial assets is determined. This might, for instance, involve the confirmation between the trading parties of the conditions of a transaction, or, for efficiency reasons, the netting of several transactions over a longer period, to reduce the actual exchange of funds and assets. A third stage is the "settlement stage", which involves the actual exchange of funds and assets. After the settlement stage, if all goes well, the financial asset involved is in the possession of the rightful owner. In most cases the safe keeping of the asset is left to a specialized financial institution called a *Custodian*. The settlement of a transaction involves at least three parties: the two parties (eg. *Investment Firms*, which we will use for our example) involved in the transaction and a Custodian. Execution of the settlement process crosses the boundaries of these parties and involves the exchange of standardized messages[1] between these parties. A detailed description of the settlement process is far beyond the scope and space of this paper. A simplified and stylized account of the settlement process, represented by the BPMN Collaboration diagram in Fig. 4, is sufficient to serve as a useful example. For more information about the settlement process see eg. [17] or [24].

One of these simplifications include the fact that Fig. 4 shows only *two* in stead of the *three* parties you might expect from the explanation above. The Custodian will expect *both* Investment Firms participating in a Financial markets transaction to send a *Settlement Instruction* (SI). Adding the second Investment Firm in Fig. 4 would change the process model for the Custodian and make it more complex, but this would not affect the interaction between each of the Investment Firms and the Custodian. As we will focus on the behaviour of the Investment Firm in our tool demonstration, this simplification will not affect our conclusions.

The settlement process is initiated by one of the *Investment Firms* involved in the transaction that has to be settled, by sending a *Settlement Instruction* to the custodian (Task "SSI" in Fig. 4). The Custodian will expect the other Investment Firm also to send an instruction, but as already mentioned, this is not shown in Fig. 4. After receiving an instruction (Intermediate Message Event "S2" in Fig. 4), the custodian will, after a certain delay (Timer Event "TE2" in Fig. 4), try to *match* it against instructions that have been received from other Investment Firms (not shown). If there are two matching instructions, the exchange of securities will be effectuated. This will subsequently be reported to the Investment Firm(s) in question with a *Settlement Confirmation* (Task "SSC" in Fig. 4). Another simplification introduced here is that we assume here that there will always be two matching instructions. Before matching occurs, each of the Investment Firms can send a *Cancellation* (Task "SC" in Fig. 4) to cancel the Settlement Instruction it sent earlier. In that case the Custodian will cancel the instruction and send a *Cancellation*

[1] Typically ISO15022 [25] or ISO20222 [26] standards.

Confirmation (Task "SCC" in Fig. 4) to the Investment Firm that sent the cancellation. Cancellation is *not* allowed when matching has already occurred, because a matched instruction involves a legally binding commitment to the transfer of the securities.

4.2 Translation of the BPMN Collaboration Diagram into an LTL-Formula

In this Section we discuss the translation of the BPMN Collaboration diagram shown in Fig. 4 into an LTL-formula following the steps of the translation process given in Sect. 2.1. In step 1 we select the part of the Collaboration Diagram representing the public behaviour of the *Investment Firm* for our case study. In step 2 we identify the BPMN-fragments included in that part of the Collaboration Diagram. The result of step 1 and 2 is shown in Fig. 5.

To proceed with step 3 we follow the notation as discussed in Sect. 2.1. So, for example, $[SSIa]$ is the LTL proposition to represent the active status of the activity "SSI" (Send Settlement Instruction). The BPMN fragments are marked with the labels Φ_1 through Φ_7. The translation of the fragments into LTL-formulae is listed in Table 2. The complete LTL formula representing the public behaviour of the *Investment Firm* shown in Fig. 5 is the conjunction of the sub formulae Φ_1 through Φ_7 given in Table 2. This formula is a formalization of the constraint, defined by the Collaboration Diagram in Fig. 4, on the local behaviour of the *Investment Firm*. It can be used to verify models of local behaviour of the *Investment Firm*. In Sects. 4.3 and 4.4 we propose two models for the local behaviour of the Investment Firm, that can be verified for compliance against the LTL-formula just derived. The actual verification is discussed in Sect. 4.5.

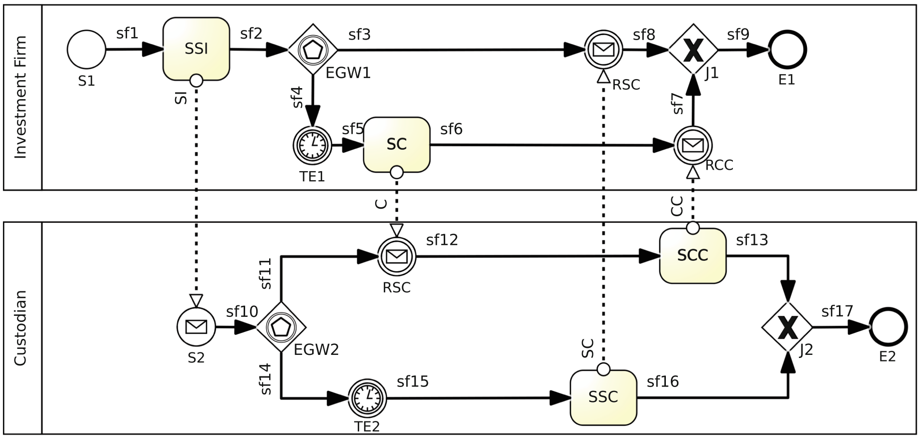


Fig. 4. Collaboration diagram of the settlement process

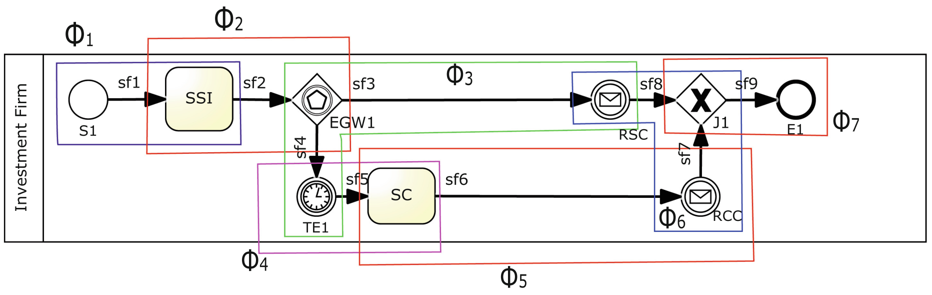


Fig. 5. Identification of BPMN-fragments for translation into LTL

Table 2. Translation of BPMN-fragments (see Fig. 5) in LTL-formulae

BPMN-fragment	LTL-formula
Φ_1	$F\,[SSIa]$
Φ_2	$\neg(\neg[SSIc]\,U\,[EGW1])$
Φ_3	$\neg(\neg[EGW1]U(F\,[RSCa]\,xor\,F\,[TE1a]))$
Φ_4	$\neg(\neg[TE1c]\,U\,[SCa])$
Φ_5	$\neg(\neg[SCc]\,U\,[RCCa])$
Φ_6	$\neg(\neg([RSCa]\,xor\,[RCCa])\,U\,[J1])$
Φ_7	$F[J1]$

4.3 Example of a Correct Specification of Local Behaviour

Figure 6 shows the Process Diagram representing the local behaviour of the *Investment Firm* that satisfies the required global (public) behaviour as specified by the Collaboration Diagram given in Fig. 4. The only difference between the public behaviour of the Investment Firm represented in the BPMN Collaboration diagram in Fig. 4 and its behaviour represented by the BPMN Process Diagram in Fig. 6 is that the latter includes two additional *internal* or *private* activities: "PC" (*prepare cancellation*) and "PSC" (*process settlement confirmation*). These additional activities are compliant with the public behaviour of the participant specified in Fig. 4 and therefore should not be considered as a violation.

4.4 Example of an Incorrect Specification of Local Behaviour

Figure 7 shows a specification of the process of the *Investment Firm* that violates the LTL formula given in Sect. 4.2.

The local behaviour specified in Fig. 7 is a violation of the public behaviour in Fig. 4 because it allows to send a Cancellation of a Settlement Instruction (Task "SC") after receiving a Settlement Confirmation (Intermediate Message Event "RSC"), i.e. after the custodian has matched both instructions of the Investment Firms, which is not allowed.

4.5 Test Results

The LTL formula that defines the public behaviour cannot be evaluated directly by GROOVE. Events such as [SSIc] are defined in terms of parameterised rule applications, such as leaveTask("SSI"), and the XOR operator is rewritten since it is not supported by GROOVE. In Fig. 8, the LTL expression for our running example, as derived in Sect. 4.2, can be seen as it is implemented in GROOVE. Evaluating the expression on the violating process flow from Sect. 4.4 yields the results one is expecting: GROOVE detects that the property is not satisfied for the statespace of the BPMN model and demonstrates this by means of the counter-example shown in Fig. 9.

The specific counter-example shown corresponds to the scenario where the custodian has sent a Settlement Confirmation (Task "SSC") and terminates gracefully, after which the Investment Firm receives the Confirmation (Intermediate Message Event "RSC") but still decides to send a Cancellation (Task

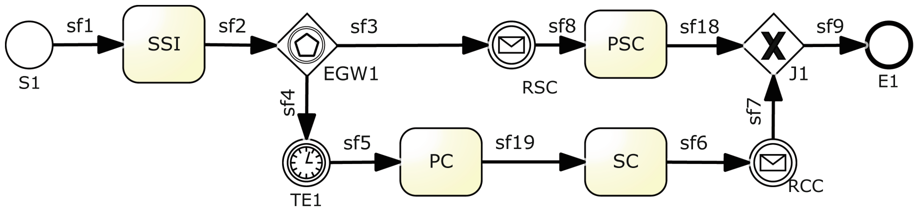


Fig. 6. Process diagram in conformance to collaboration

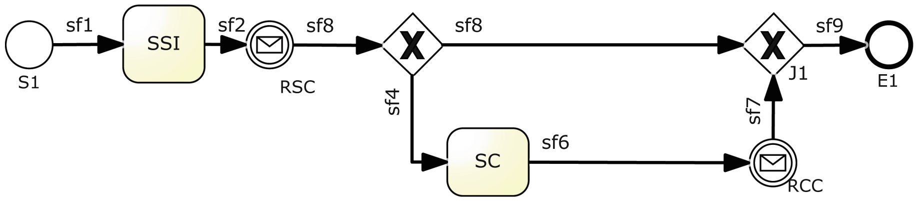
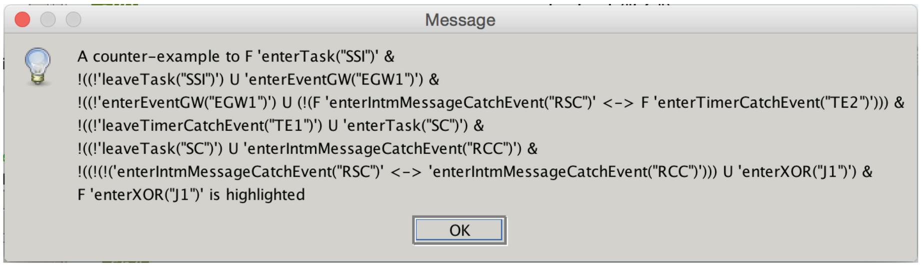
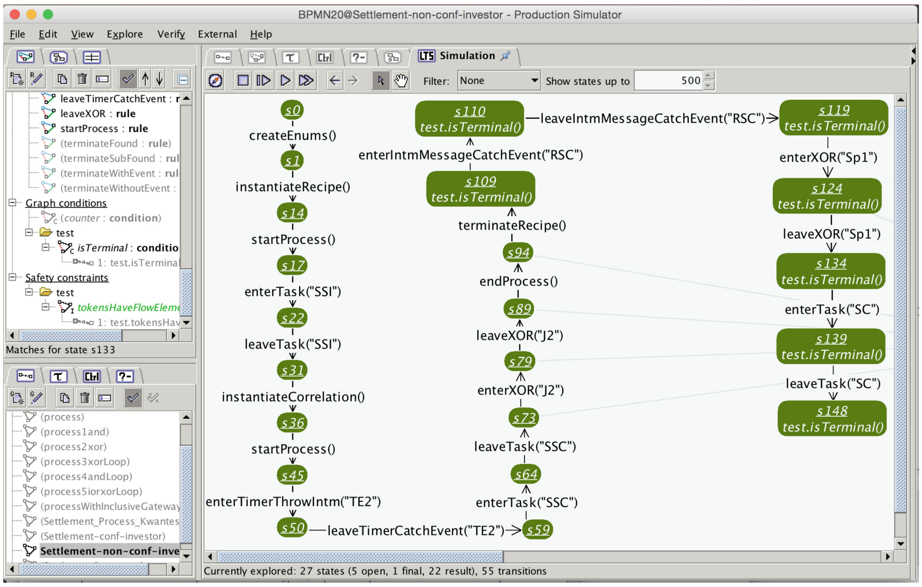


Fig. 7. Process diagram violating the collaboration diagram



```
A counter-example to F 'enterTask("SSI")' &
!((!'leaveTask("SSI")') U 'enterEventGW("EGW1")') &
!((!'enterEventGW("EGW1")') U (!(F 'enterIntmMessageCatchEvent("RSC")' <-> F 'enterTimerCatchEvent("TE2")'))) &
!((!'leaveTimerCatchEvent("TE1")') U 'enterTask("SC")') &
!((!'leaveTask("SC")') U 'enterIntmMessageCatchEvent("RCC")') &
!((!(!('enterIntmMessageCatchEvent("RSC")' <-> 'enterIntmMessageCatchEvent("RCC")'))) U 'enterXOR("J1")') &
F 'enterXOR("J1")' is highlighted
```

Fig. 8. The LTL formula as evaluated by GROOVE



Currently explored: 27 states (5 open, 1 final, 22 result), 55 transitions

Fig. 9. The process from Fig. 7 violates the conformance contract

"SC"). This leads to waiting in vain for a cancellation confirmation (at the Intermediate Message Event "RCC" in Fig. 7). Automatic verification results like this have been computed within a few seconds on a mainstream desktop computer.

5 Related Work

Much of the research on inter-organisational workflows (see [2] for an overview) is concerned with the *construction* of such workflows. We distinguish between *top-down* and *bottom-up* approaches. An example of the first is the *Public-To-Private*(P2P) approach [1] which involves the construction of a local workflow as a *subclass* of a global workflow thereby *inheriting* the properties of the global workflow, including correctness. An example of the second is [18] involving the *composition* of local workflows represented as *workflow modules*, a kind of Petri nets. In [10] *service outsourcing* is presented as a bottom-up approach involving the construction and matching of *process views*. Bottom-up approaches are also concerned with *verification of general properties* (like soundness) of the global workflow. The problem addressed in this paper, i.e. whether the design of a local workflow is in compliance with the design of a global workflow, is not addressed in the above mentioned references. The concern for verification of soundness of the global workflow is a relevant issue we will discuss in Sect. 6

Another line of research involves the development of new *modelling languages* like *Let's Dance*,*Interaction Petri nets* and the *BPMN Choreography diagrams*, specifically designed to model collaborative behaviour and avoid modelling errors (eg. deadlocks) (see eg. [6,7]). The focus of our paper is on compliance verification using BPMN Collaboration diagrams but can easily be adapted to include other modelling languages.

In [2] a *service mining approach* is proposed. This includes *conformance checking* of event-logs against a choreography model. For a collaborative workflow in the design phase event-logs are not always available in which case our approach seems more appropriate.

Business Process Compliance [11] is another line of related research. There the aim is to automate compliance-checking of business process models against regulatory requirements. See e.g. [9] where a formal approach is presented to verify a specification of *local behaviour* in BPEL, against specifications in a dedicated *Compliance Request Language* representing legal constraints. In [16] this problem has been extended to include compliance of a *global workflow* with rules and regulations.

Another line of related research is involved in checking compliance of a (local) process model against its *refinement* or *implementation*. An example of the first is [19], which discusses the automated verification of low level UML activity diagrams against high level UML activity diagrams. The purpose is to establish behavioural containment such that the low-level diagram is a valid refinement of the high-level diagram. An example of the second problem is given in [4] which describes an approach involving derivation of the specification of a Web-application in WebML from a (local) BPMN Process Diagram and the subsequent verification of web execution logs against derived LTL formulae. The problem addressed by these approaches is different from the problem addressed in this paper, although we build on some of the techniques used by them.

In [3,15] the use of graph transformation to specify operational semantics of UML Activity Diagrams is described. In [14] a formal semantics of BPMN process diagrams is described using graph transformations. We extended this to include BPMN Collaboration diagrams and implemented it in GROOVE.

6 Discussion and Outlook

Organisations in the financial markets domain typically have to operate in a global operational context which often places complex and unyielding restrictions on the design of their business processes. Verifying the process design of an organisation against these restrictions is a costly, error prone and painful manual process. A real-world example that might illustrate this problem is the *Target2Securities* project [8]. This project involves a major effort (launched in 2006, spanning more than a decade and costing hundreds of millions of Euros) of the Eurosystem, the central banking system for the euro, to migrate the settlement process from a system of many collaborative workflows organized along national borders to one collaborative workflow on a European scale. The European Central Bank produces large quantities of BPMN models of the new collaborative workflow. The financial institutions involved in this new collaborative workflow are relatively autonomous in redesigning their own local workflows but they have to be compliant with the new global workflow to stay in business. As far as we know there is currently no approach available that directly addresses this problem. The approach described in this paper builds on and extends existing methods and technologies to address this problem. It is based on standard business process modelling notation, is quite generic and its application is not restricted to the financial markets domain. The evaluation of the test cases in Sect. 4.5 demonstrates that automated verification of local versus global process models, as proposed in this paper, in principle, is technically feasible.

There are however still a number of issues, which we will discuss below, that need to be addressed in our future work. The implementation described in Sects. 3 and 4 has not yet been tested beyond the complexity of the running example in this paper. However, since the verifications require only a few seconds of GROOVE computation time, they form a promising basis for further work. The translation of the BPMN Collaboration diagram into an LTL-formula described in Sect. 4.2 has been done manually. However, the procedure as described in Sects. 2.1 and 4.2 can be automated [4] and this is included in our agenda for future work. Another issue is, that we assign a formal semantics to BPMN Process models in two different ways: the first by interpreting BPMN as LTL-formula, as described in [4] and the second by assigning a token-based semantics according to [14]. Finding formal proof that these two different definitions of semantics are consistent is included in our future research.

Another issue is that we did not discuss checking the soundness of the global workflow in this paper, but this can be included easily. The reader might in fact have noticed that the global workflow presented in Fig. 4 is not sound. An undesirable situation for example occurs when the timer events of the Custodian and the Investment Firm occur concurrently. The Investment Firm then incorrectly decides to send a Cancellation, but ends up in a deadlock. In our evaluations this problem does show up as the process from Fig. 4 turns out to violate the derived LTL expression. The reason that the constraint is not satisfied resides in the final clause of the LTL expression, which requires that the derived processes reach the final XOR node “J1”. That is effectively not the case when both timer events are triggered. This means that we will have to extend our approach to include verification of the global workflow for soundness, and resolve any violations, before checking local workflows for compliance. Our approach can easily be extended to include soundness checking of the global workflow. Finally, an issue that needs to be addressed in our future work is that the LTL-formula derived following [4] seems only capable of capturing liveness requirements but not yet safety requirements.

References

1. van der Aalst, W.M.P., Weske, M.: The P2P approach to interorganizational workflows. In: Dittrich, K.R., Geppert, A., Norrie, M. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 140–156. Springer, Heidelberg (2001). doi:10.1007/3-540-45341-5_10
2. van der Aalst, W.M.P., Weske, M.: Reflections on a decade of interorganizational workflow research. In: Bubenko, J., Krogstie, J., Pastor, O., Pernici, B., Rolland, C., Sølvberg, A. (eds.) Seminal Contributions to Information Systems Engineering: 25 Years of CAiSE, pp. 307–313. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36926-1_24
3. Bandener, N., Soltenborn, C., Engels, G.: Extending DMM behavior specifications for visual execution and debugging. In: Malloy, B., Staab, S., van den Brand, M. (eds.) SLE 2010. LNCS, vol. 6563, pp. 357–376. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19440-5_24
4. Brambilla, M., Deutsch, A., Sui, L., Vianu, V.: The role of visual tools in a web application design and verification framework: a visual notation for LTL formulae. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 557–568. Springer, Heidelberg (2005). doi:10.1007/11531371_70
5. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (2001). http://books.google.de/books?id=Nmc4wEaLXFEC
6. Decker, G., Barros, A.: Interaction modeling using BPMN. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 208–219. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78238-4_22
7. 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). doi:10.1007/978-3-540-75183-0_22
8. ECB: Target2securities. https://www.ecb.europa.eu/paym/t2s, Mar 2015
9. Elgammal, A., Turetken, O., van den Heuvel, W.J., Papazoglou, M.: Formalizing and appling compliance patterns for business process compliance. Softw. Syst. Model., 1–28 (2014). http://dx.doi.org/10.1007/s10270-014-0395-3
10. Eshuis, R., Norta, A., Kopp, O., Pitkanen, E.: Service outsourcing with process views. IEEE Trans. Serv. Comput. **8**(1), 136–154 (2015). http://doi.ieeecomputersociety.org/10.1109/TSC.2013.51
11. Fellmann, M., Zasada, A.: State-of-the-art of business process compliance approaches. In: 22st European Conference on Information Systems, ECIS 2014, Tel Aviv, Israel, 9–11 June 2014 (2014). http://aisel.aisnet.org/ecis2014/proceedings/track06/8
12. Gabbay, D.M.: The declarative past and imperative future: executable temporal logic for interactive systems. In: Temporal Logic in Specification, Altrincham, UK, 8–10 April 1987, Proceedings, pp. 409–448 (1987)

13. Ghamarian, A.H., de Mol, M., Rensink, A., Zambon, E., Zimakova, M.: Modelling and analysis using GROOVE. STTT **14**(1), 15–40 (2012). http://dx.doi.org/10.1007/s10009-011-0186-x
14. Gorp, P.V., Dijkman, R.M.: A visual token-based formalization of BPMN 2.0 based on in-place transformations. Inf. Softw. Technol. **55**(2), 365–394 (2013). http://dx.doi.org/10.1016/j.infsof.2012.08.014
15. Hausmann, J.H.: Dynamic META modeling: a semantics description technique for visual modeling languages. Ph.D. thesis, University of Paderborn (2005). http://ubdata.uni-paderborn.de/ediss/17/2005/hausmann/disserta.pdf
16. Knuplesch, D., Reichert, M., Fdhila, W., Rinderle-Ma, S.: On enabling compliance of cross-organizational business processes. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 146–154. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40176-3_12
17. Kwantes, P.M.: Design of clearing and settlement operations: a case study in business process modelling and evaluation with petri nets. In: 7th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN 2006) (2006)
18. Martens, A.: On compatibility of web services. Petri Net Newsletter **65**, 12–20 (2003)
19. Muram, F.U., Tran, H., Zdun, U.: Automated mapping of UML activity diagrams to formal specifications for supporting containment checking. In: Proceedings 11th International Workshop on Formal Engineering Approaches to Software Components and Architectures, FESCA 2014, Grenoble, France, 12th April 2014, pp. 93–107 (2014), http://dx.doi.org/10.4204/EPTCS.147.7
20. OMG: Business process model and notation (BPMN) version 2.0. Technical report, Jan 2011. http://taval.de/publications/BPMN20
21. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, pp. 46–57 (1977). http://dx.doi.org/10.1109/SFCS.1977.32
22. Pnueli, A.: The temporal semantics of concurrent programs. Theor. Comput. Sci. **13**, 45–60 (1981). doi:10.1016/0304-3975(81)90110-9
23. Rensink, A.: The GROOVE simulator: a tool for state space generation. In: Pfaltz, J.L., Nagl, M., Böhlen, B. (eds.) AGTIVE 2003. LNCS, vol. 3062, pp. 479–485. Springer, Heidelberg (2004). doi:10.1007/978-3-540-25959-6_40
24. SMPG: Securities markets practices group/market practices and documents/settlement and reconciliation. http://www.smpg.info, Mar 2015
25. S.W.I.F.T.: ISO15022 financial industry message scheme. http://www.iso15022.org, Mar 2015
26. S.W.I.F.T.: ISO20022 universal financial industry message scheme. http://www.iso20022.org, Mar 2015