Automatic Generation and Translation of Process Collaboration Models

Frederik Bischoff and Walid Fdhila

Faculty of Computer Science, University of Vienna, Austria

Abstract. In the research field of business process models and techniques, researchers can only rely on a repository of centralized, intra-organizational processes to use as support of their work. But regarding decentralized, cross-organizational models, they face the problem that there is a lack of available model examples. Within this work this lack is tried to be tackled. Thereby, a concept is introduced how to generate collaborative business processes randomly by following a *top-down approach*, which first generates the choreography model and then derives the public and private models of each process participant from it. Additionally, a possibility for specifying and imposing global compliance rules onto the collaboration is elaborated. The conception of this random collaboration process generator is prototypically implemented within an existing framework in order to evaluate the process. During the generation, the models are maintained as a Refined Process Structure Tree. In order to utilize the generated processes for the support of for further research, the models are exported in BPMN notation.

Keywords: Process Collaboration · Automatic Generation · Process Models · Compliance Rules.

1 Introduction

In the research field of business process models and techniques, researchers can only rely on a repository of centralized, intra-organizational processes to support their work. But regarding decentralized, cross-organizational models, they face the problem that there is a lack of available model examples. Within the scope of this work, this lack is aimed to be tackled. Thereby, the main objective is to build a repository of distributed and collaborative process models by developing and implementing an automatic generation process for business process collaborations. The generation process must thereby ensure soundness, consistency and compatibility of the resulting models. Additionally, it should also allow the generation of models which comply to imposed compliance rules. At last, to ensure the executability of the auto-generated models by process engines, an additional goal of this work is to develop a transformation of the utilized RPST[1] representation to BPMN[2]. These requirements ensure that the repository can then be

[1] Refined Process Structure Tree

[2] Business Process Model and Notation - http://www.bpmn.org/

used for further research, such as change management or process mining of collaborative processes. Based on those objectives, the following research questions are derived:

- *RQ-1*: How to build a repository of collaboration process models that can be used as support for further research in this field?
- *RQ-2*: How to ensure that the resulting process models in this repository are correct in terms of consistency, compatibility and compliability? Which process flow perspectives and compliance rule patterns should be supported regarding compliability?
- *RQ-3*: How to transform a collaborative process represented as an RPST into an executable form?

This work is part of the CRISP[3] project (ICT15-072) funded by the Vienna Science and Technology Fund (WWTF). The implementation of the prototype is integrated in an already existing framework. The main research within the project is to analyze flexibility and adaptivity of collaborative business processes at design time as well as at runtime. Regarding consistency and correctness of collaborative business processes, the propagation of a process change over all process participants is one of the main challenges the project tries to tackle. Furthermore, the impact of compliance rules imposed on collaborative business scenarios is analyzed as well as the issue of ensuring business compliance by considering the privacy and autonomy of the involved business partners.

This work is structured as followed. First, a brief blablabla.

2 Process Collaborations

This chapter provides the basics of business process collaborations, represented in BPMN. In BPMN, collaborative processes are represented from different perspectives, whereas each perspective is represented by a different BPMN model type. In the following, the different model types and their represented perspective are explained with the help of a collaborative business process example.

2.1 Different Perspectives of Process Collaborations

Blablabla... TODO: smaller and simpler example collaboration

Private Model The private model is modeled from the perspective of a single participant of a collaborative process. In a collaborative scenario, it describes the complete internal business logic of one partner as well as the messages exchanged with other partners. Activities which are only for the participant are called *private activities*, whereas activities which involve participation of other partners are called *public activities*.

[3] http://gruppe.wst.univie.ac.at/projects/crisp/

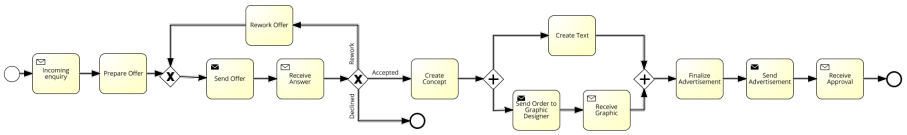

Fig. 1. Private Model Example

For example, in the private model shown in Figure 1, the task *Create Concept* represents a private activity, whereas the task *Send Offer*, involves message exchange with another participant and therefore represents a public activity. A private model is a process model in the classical sense. It's fundamental modeling objects are unchanged since BPMN 1.0. Is a private process modeled and attributed in detail, it also represents an executable process that can be executed by a process engine.

Public Model The public model is also modeled from the perspective of a single participant of a collaborative process. It is a reduced view on the private model of a partner. It can also be described as a projection of the whole collaboration process focusing on one participant. It only includes public activities, involving message exchange with other partners. Private activities which are not relevant for other partners and which don't want be to be shared with the partners are omitted deliberately. Figure 2 shows the corresponding of the already introduced private model.

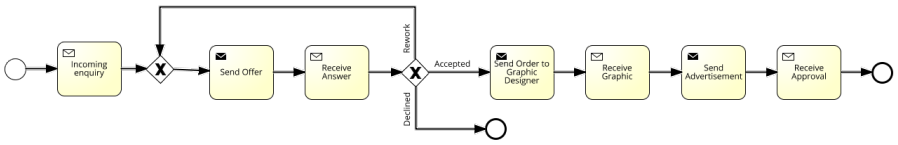

Fig. 2. Public Model Example

Collaboration Model The collaboration model is the interconnection between the public models of all participants. The thereby formed model gives a holistic view on the collaborative process and does not focus on one partner. Each partner is represented as a pool and the message exchange between them is shown by a message flow that connects two public activities or just the pools. Figure 3 shows an example collaboration model.

Each partner's public process is modeled inside a pool. It is also allowed for a process to not include the public model inside an participant's pool. If a pool

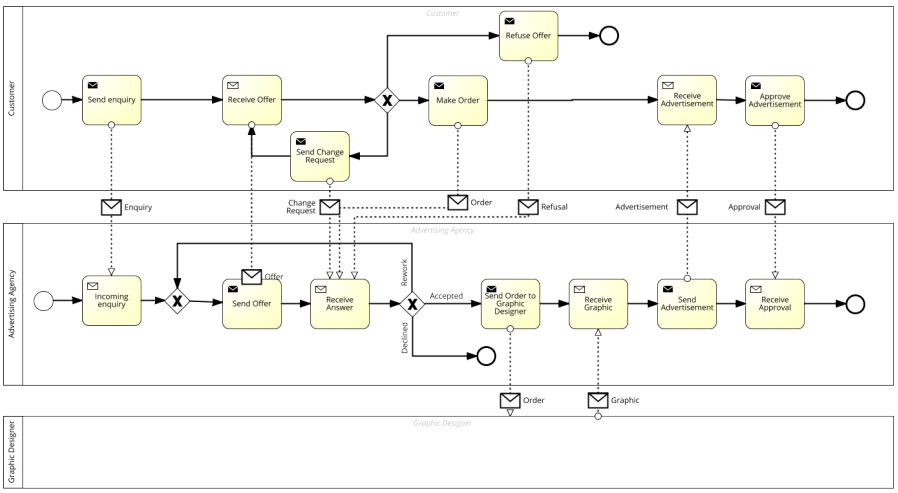

Fig. 3. Collaboration Model Example

contains a process, it is called a "white box". If a pool is empty, it's called a "black box". For Example in figure 3 the pools of the partners *Customer* and *Advertisement Agency* are modeled as "white box" and the pool of the *Graphic Designer* as a "black box" [**?**].

Choreography Model The choreography model is available since BPMN 2.0 and focuses solely on the sequence of message exchanges between the partners. Each message exchange is represented as an interaction with an initiating partner, a receiving partner (shaded in grey) and the message exchanged. In contrast to the collaboration model, which also focuses on the message flow, the choreography model additionally displays the exact sequence flow (i.e. conditional message flows or parallel message flows), which is not always evident form the former (i.e. black box pools).

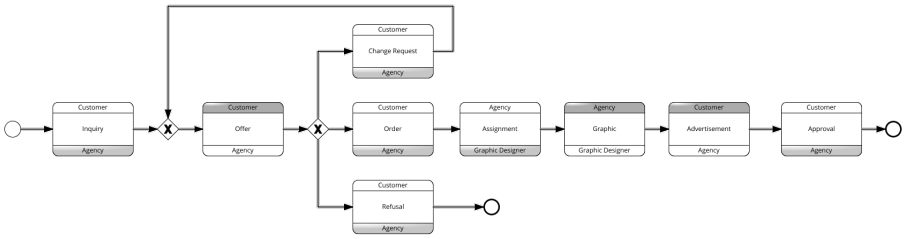

Fig. 4. Choreography Model Example

2.2 Correctness Levels of Process Models

maybe just cite [**?**]

3 Automatic Collaboration Generation Approach

3.1 Top-Down Approach

As already mentioned, a process collaboration involving several partners can be modeled from different perspectives (partner or global) through the use of different model types. The process collaboration generator, implemented in the context of this work, generates all three different model types as the output. In general, there exist two different approaches to build a process collaboration with all the described models [**?**]. In the *top-down approach*, first the choreography model is build, then the public and private models of each partner are derived and defined consistently. In comparison, in the *bottom-up approach*, each partner has already defined a private and public process. Then, the choreography model is constructed by connecting the public models via message exchange. The automatic generator process, presented in this work, follows the *top-down approach*, by first generating the choreography model and then deriving the collaboration, public and private models from it. Thereby, each interaction (choreography task) of the choreography model is converted into a send and receive task and then added to the involved partners processes, to build their public process models. In turn, each private model is derived from its corresponding public model by enriching the public model with abstract private tasks. The collaboration model is then built by composing the partner's public processes into one model.

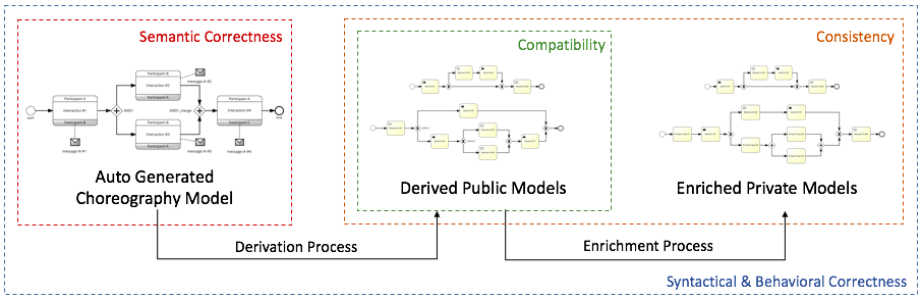

Fig. 5. Top-Down Approach

All three correctness levels are considered by the proposed algorithm for implementing an automatic process generator. The algorithm ensures that only model specific flow objects are used to build the processes and that they are connected appropriately (syntactical correctness). It also guarantees the absence of

deadlocks and lifelocks (behavioral correctness) and offers the possibility to define global compliance rules, to which the generated collaboration should comply (semantic correctness). Deriving all models from the before generated choreography model offers also the advantage, that if the deriving process is implemented correctly, it already ensures *consistency* and *compatibility* between the different models. In the context of collaborative processes, *consistency* means, that the private model of a partner has to be consistent with the corresponding public model, whereas *compatibility* requires the public models of the collaborating partners to be compatible with one another [**?**]. This ensures that the executed business process of one partner satisfies the behavior that is communicated to the partners through his public models [**?**].

3.2 Constraining the Collaboration

Despite the premise that the process collaborations should be generated randomly, it is reasonable to set some boundaries within which the random generation takes place. The implemented generator provides two different ways to influence the resulting choreography model and hence the whole collaboration. The first one provides the possibility to constrain the choreography model in terms of the employed flow objects and their exact quantity by specifying several input parameters. The second one enables the user to impose global compliance rules based on compliance patterns to which the resulting model must comply.

3.3 Parametric Constraints

The following input parameters are specified to influence the random generation of the choreography model and hence also the deriving models:

- Number of Partners:
  Determines the number of participants that are involved in the process collaboration.
- Number of Interactions:
  Determines the number of messages that are exchanged between the partners.
- Number of Exclusive Gateways:
  Determines the number of Exclusive Gateways that are put into the model.
- Number of Parallel Gateways:
  Determines the number of Parallel Gateways that are put into the model.
- Max. Branching:
  Determines the maximum possible number of paths created for each gateway.

3.4 Compliance Constraints

Generally, compliance rules can be defined for different process flow perspectives of a process. It can be distinguished between compliance rules that constrain the control flow (sequence of activities), the data associated with the activities, the

resources (specific user or role) that perform the activities or the time perspective. There exist several languages and approaches on how to define and specify compliance rules, including formal languages [**?**],[**?**], visual languages [**?**],[**?**] and pattern-based approaches [**?**], [**?**]. Both visual and pattern-based approaches aim at hiding formal details (e.g. temporal logic) and therefore simplifying the specification of compliance rules.

For the specification of compliance rules for the automatic process generator, the pattern-based approach of Turetken et. al. [**?**] is utilized. In [**?**] a repository of *process control patterns* is introduced, which are high-level templates used to represent process properties which the process specification must satisfy. Because the generated process collaborations neglect the data, time and resource perspective and focus solely on the control flow, only compliance rules that constrain the sequence of activities are possible to impose. Following process control patterns are supported to constrain the automatic generated choreography:

Pattern	Description
P LeadsTo Q	Interaction P must lead to Interaction Q
P Precedes Q	Interaction Q must be preceded by Interaction P
P Exclusive Q	Interaction Q .. (not yet implemented!!!)
P Universal	Interaction P must always occur throughout execution
P Exists	Interaction P must be specified in process

Table 1. Overview of supported Compliance Patterns

Note that the *P LeadsTo Q* pattern does not demand an immediate succession of interaction Q on interaction P.

3.5 Random Process Generation

In this chapter, the needed components and their functionality for implementing a random process generator are explained. Each component encapsulates a step of the above described approach. Based on this approach, four components with single responsibilities can be derived: one component is responsible for randomly generating a choreography model, one for imposing compliance rules on the resulting process, another for deriving the remaining model types and the last component keeps control over the overall process. In the following the components and their functionalities will be explained in details.

3.6 Overall Process Controller

The *Overall Process Controller* represents the orchestration component for building random process collaborations, based on given constraints. The process follows the principle *'first build then check'*, which means that after a random

choreography model has been generated, it will then be checked if the interactions defined within the compliance rules, can be assigned into the already built model in such a way that the resulting interaction sequence complies to the imposed rules. If the interaction allocation is not possible without violating the compliance rules, new random models will be build until a compliant model has been generated. If the imposing of the compliance rules fails repeatedly, it's an indicator that the amount of interactions in the model is too small relative to the unique interactions specified within the compliance rules. To overcome this, the number of interactions is increased by 10 percent every 10th build. After a successful assignment of the compliance rules, the remaining public and private models are derived out of the generated choreography model. At last, all models will be translated into a valid BMPN/XML. The whole process of generating a random choreography by coordinating the different functions is outlined in Algorithm **??**.

Algorithm 1: Overall Collaboration Generation Controller

```
1  buildSuccess = false;
2  while buildSuccess ≠ true do
3  |  build new choreography model;
4  |  if compliance rules are defined then
5  |  |  assign interactions;
6  |  |  if assignment successful then
7  |  |  |  buildSuccess = true;
8  |  |  else if number of interaction mod 10 ≡ 0 then
9  |  |  |  increase number of interactions by factor 1.1;
10 |  |  end
11 |  else
12 |  |  buildSuccess = true;
13 |  end
14 end
15 generate whole collaboration;
16 export models to BPMN;
```

References

1. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic, pp. 326–341. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
2. Fdhila, W., Indiono, C., Rinderle-Ma, S., Reichert, M.: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Systems **49**, 1 – 24 (2015), http://www.sciencedirect.com/science/article/pii/S0306437914001550

3. Fdhila, W., Rinderle-Ma, S., Knuplesch, D., Reichert, M.: Change and compliance in collaborative processes. In: 12th IEEE International Conference on Services Computing (SCC 2015). pp. 162–169. IEEE Computer Society Press (June 2015), http://dbis.eprints.uni-ulm.de/1174/
4. Ghose, A., Koliadis, G.: Auditing Business Process Compliance, pp. 169–180. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
5. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance Checking Between Business Processes and Business Contracts. In: 10th International Enterprise Distributed Object Computing Conference (EDOC 2006). pp. 221–232. IEEE Computing Society (2006)
6. Knuplesch, D., Reichert, M., Ly, L.T., Kumar, A., Rinderle-Ma, S.: Visual modeling of business process compliance rules with the support of multiple perspectives. In: 32nd Int'l Conference on Conceptual Modeling (ER 2013). pp. 106–120. No. 8217 in LNCS, Springer (November 2013), http://dbis.eprints.uni-ulm.de/953/
7. Knuplesch, D., Reichert, M., Mangler, J., Rinderle-Ma, S., Fdhila, W.: Towards compliance of cross-organizational processes and their changes. In: 1st Int Workshop on Security in Business Processes (SBP?12), BPM'12 Workshops. pp. 649–661. No. 132 in LNBIP, Springer (September 2012), http://dbis.eprints.uni-ulm.de/848/
8. (OMG), O.M.G.: Business process model and notation (bpmn) version 2.0. Tech. rep., Object Management Group (OMG) (January 2011), http://www.omg.org/spec/BPMN/2.0
9. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where Did I Misbehave? Diagnostic Information in Compliance Checking, pp. 262–278. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
10. Turetken, O., Elgammal, A., van den Heuvel, W.J., Papazoglou, M.P.: Capturing compliance requirements: A pattern-based approach. IEEE Software **29**(3), 28–36 (May 2012)