Converting DEMO PSI Transaction Pattern into BPMN: A Complete Method

Ondřej Mráz, Pavel Náplava, Robert Pergl(✉), and Marek Skotnica

Czech Technical University in Prague, Prague, Czech Republic
mraz.ondra@gmail.com, naplava@fel.cvut.cz,
{perglr,skotnicam}@fit.cvut.cz

Abstract. The goal of this paper is to contribute to efforts of improving the Business Process Modelling (BPM) practice. We present an original method for converting 0enterprise ontology Design & Engineering Method for Organisations (DEMO) process models into a BPMN 2.0 notation. By this approach, we are able to mitigate certain methodological deficiencies of BPMN. The method exhibits the following qualities: Implementation of the complete transaction pattern formulated by the PSI-theory, correct managing of multiple child transaction instances, and executability of the resulting BPMN model.

Keywords: PSI-theory · BPMN · DEMO · Business Process Modelling · Enterprise ontology · Conceptual modelling

1 Introduction

BPMN (Business Process Model and Notation) [1] is a graphical notation that is used for modelling business processes. Key characteristics of BPMN are simplicity of the underlying theory (flowchart), standardised notation and a large number of tools. This makes BPMN one of the most wide-spread process modelling notation in practice, in spite of its limitations and flaws. BPMN offers three different types of diagrams: Choreography, Conversation and Collaboration diagrams. For this work, only the Collaboration Diagram will be considered. This diagram expresses the process flow in achieving participants' goals.

One of the BPMN weaknesses is the absence of a methodology for constructing diagrams, which is addressed for example by Silver [2]. Nevertheless, the design freedom is still too broad, which results in different modelling styles of individual analysts and different models depicting the same situation, which complicates enterprise engineering tasks like mergers and reorganisations.

DEMO (Design & Engineering Method for Organisations) [3] is a leading modelling method used in the discipline of Enterprise Engineering [4] based on deep and sound theoretical basis (the PSI-theory) and high ontological relevance. Its benefits for the practical use has been proven, as documented for example in [5] or [6]. It does not limit itself just to process modelling, but it also deals

D. Aveiro et al. (Eds.): EEWC 2017, LNBIP 284, pp. 85–98, 2017.
DOI: 10.1007/978-3-319-57955-9_7

with capturing structural (factual) knowledge and business rules, thus delivering a complete enterprise ontology exhibiting certain criteria (C4E). However, compared to BPMN, DEMO is still a niche approach and relatively demanding to master. Also, a limited number of tools is available today.

For a brief description of DEMO, we take a help of Op't Land and Dietz [5]:

A complete, so-called essential model of an organization consists of four aspect models: *Construction Model (CM)*, *Process Model (PM)*, *Action Model (AM)*, and *State Model (SM)*. The CM specifies the composition, the environment and the structure of the organization. It contains the identified *transaction types*, the associated *actor roles* as well as the information links between actor roles and transaction banks (the conceptual containers of the process history). The PM details each transaction type according to the *universal transaction pattern*. In addition, it shows the structure of the identified business processes, which are trees of transactions. The AM specifies the imperatively formulated *business rules* that serve as guidelines for the actors in dealing with their agenda. The SM specifies the *object classes*, the *fact types* and the declarative formulations of the *business rules*.

The DEMO Process Model reveals details of the transactions with the respect to universal transaction pattern. The basis is the "happy flow" consisting of `request`, `promise`, `state` and `accept`, which is also called the *basic transaction pattern*. In the so-called *standard transaction pattern* (not depicted), `decline` may happen instead of `promise` and `reject` may happen instead of accept. Then, a new attempt may be made, or `quit`, resp. `stop` may end the transaction unsuccessfully. Real situations may become even more complicated, which is addressed by the *complete transaction pattern* in Fig. 1. It incorporates the notion of *revocation* – an actor may want to "take back" their act done before[1]. If that is allowed by the other party, the transaction "rolls back" to the desired state.

The logic of the complete transaction pattern is automatically included in all DEMO transactions, which is one of the reasons why the models are compact.

The main goal of this paper is to combine the simplicity of the BPMN and ontological qualities of the DEMO. The result is the method that converts enterprise ontology Design & Engineering Method for Organisations (DEMO) process models into a BPMN 2.0 notation. This approach mitigates the mentioned absence of a sound methodological approach for BPMN. The BPMN models resulting from the described method converge, similarly to DEMO, to one essential model, thus eliminating different modelling styles of individual analysts leading to comparable models. Our other requirements are: implementation of the complete transaction pattern formulated by the PSI-theory, correct managing of multiple child transaction instances, and executability of the resulting BPMN model.

We start the paper by the discussion of the related work of efforts of improving BPM and BPMN, specifically the approaches based on applying the enterprise-engineering rigour (Sect. 2). We then briefly present results of a comparative analysis of DEMO and BPMN (Sect. 3), which led to formulating our

[1] In the DEMO theory, nothing can disappear, so the original fact remains in the fact bank, however, the transaction flow is changed.

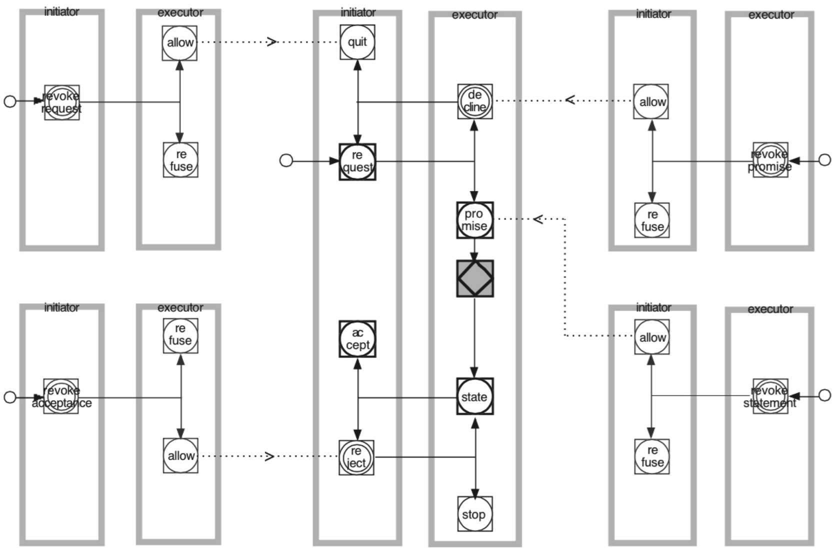

Fig. 1. DEMO complete transaction pattern [7]

method of conversion (Sect. 4). We demonstrate the method on an example (Sect. 5). Finally, we discuss the result and formulate conclusions (Sect. 6).

2 Related Work - Improving BPM and BPMN

A poor ontological quality of BPMN is generally known and documented [8]. The most practised remedy is exercising a methodological approach like the one proposed by Silver [2], who distinguishes three levels of BPMN: (i) Descriptive, (ii) Analytical and (iii) Executable and proposes several analysis patterns and anti-patterns.

The discipline of enterprise engineering (EE) [4] brought about a rigorous approach of building an enterprise ontology (EO) [3], DEMO being its modelling method. There are several foundational EE theories, the most notable being the PSI-theory. As on of the central concerns of EE is the business process management, the effort to apply EE theories (EET) to the existing (less formal approaches) is promising. The efforts in this area are twofold:

1. Applying EET for analysis of existing BPMN models of business processes: for example [9–11].
2. Enhancing the formal foundations of BPMN by EET: for example [9,10,12,13]

2.1 Applying EET for Analysis of Existing BPMN Models of Business Processes

Caetano et al. showed that applying the DEMO PSI-theory to improve business process modelling deserves attention [9]. The authors started by analysing existing BPMN models and identified missing DEMO transaction pattern steps in these models. It had been determined or each BPMN activity from the analysed models, whether this activity is an ontological, infological or datalogical part of a transaction. It had been also determined, which part of the transaction pattern each activity represents. Next, the authors created an ATD and a PSD diagram of DEMO and using a PSD diagram, they enriched existing BPMN models by adding missing parts of the transaction pattern into the BPMN models.

In the second part, the authors present results of applying this method to analysis of existing BPMN models of key processes of a big organization (more than 500 activities and 60 actors). The authors identified numerous missing act types in the original BPMN models. The results from this analysis were: (i) 25% of production C-acts missing in the original BPMN model, (ii) 25% of request C-acts missing in the original BPMN model, (iii) 50% of promise C-acts missing in the original BPMN model, (iv) 25% of state C-acts missing in the original BPMN model, (v) 40% of accept C-acts missing in the original BPMN model.

Results reported by Pergl and Náplava for an academic institution [11] state reduction of DEMO essential models complexity to 21% of the original BPMN size and several model quality improvements similar to [9].

2.2 Enhancing the Formal Foundations of BPMN by EET

These efforts aim to precisely express the EE ontological constructs using the standard BPMN notation. Two approaches have been followed. The first is to enhance the BPMN models by adding the missing C-(F)acts and other constructs from the PSI-theory. Caetano [9] is an example of this method.

The second way is generating BPMN models from the DEMO models. This method was discussed in the diploma theses [13], from which the approach in this paper was designed.

3 Analysis of DEMO and BPMN

Here follows observations of comparing various aspects of DEMO with respect to BPMN, from which follows the conversion principles and decisions made. These were formulated based on the DEMO theory axioms and models definitions related to the BPMN elements definitions, as introduced in Sect. 1.

- Similar parts of methods that can be simply transformed from the DEMO to BPMN:
  - The *Process Structure Diagram (PSD)* of DEMO contains process information, which can be related to a BPMN process diagram.
  - The *Action Model (AM)* of DEMO expresses complex decision rules for Coordination acts (C-acts)[2]. The contained information can be used for branching in BPMN.

[2] Apart from containing all the information from the other models.

  - BPMN does not distinguish the three key human abilities (forma, informa, performa), however applying this distinction can be introduced straightforwardly, as shown for example in [11]. As this concern is orthogonal to our effort, we do not discuss the distinction axiom here.
  - Related to the point above, the (atomic) actor roles in DEMO are executors of exactly one transaction, while swimlanes may contain many different actions.
- Different parts of methods that require deep analysis before transformation from DEMO to BPMN:
  - The DEMO *Transaction Axiom* concept does not exist in BPMN. Only happy flows and the most obvious unhappy flows are expressed in models.
  - The *Object Fact Diagram (OFD)* being a factual model does not have an analogy in BPMN.
  - DEMO and BPMN employ different execution models. While BPMN is flow-based, DEMO operates on the basis of a so-called CRISP model [3], which may be characterised as an event-driven, or more precisely, an agenda-driven execution model.
  - The *Construction Model (CM)* of DEMO is an abstraction that does not specify process, it provides just structural information.

4 Converting DEMO into BPMN

The goal is to convert the complete transaction axiom into BPMN, including all revoke types. Sections 4.1 to 4.4 describe all the necessary pieces and Sect. 4.7 presents the result. We used BPMN 2.0 and leveraged the newly available *Data Store* construct.

4.1 C-acts

C-acts are essentially activities that take place in order specified by the transaction pattern. BPMN has the concept of *activities* and the order is specified by *sequence flows*. As C-acts are atomic, the appropriate activity type is *task*.

4.2 C-facts

As mentioned in Sect. 1, a C-fact becomes existent in the world as a consequence of performing a C-act. Heller in his thesis [13] lists three possibilities of expressing C-facts using BPMN:

1. Not explicitly expressed – the existence of the fact-C is not explicitly expressed. It is indirectly realised by a sequence flow. This option is sufficient if revokes are not considered (see further).
2. Using a BPMN message – the actor, who performs the given C-act sends a BPMN message with the C-fact to the other actor (transaction participant).

3. Using a BPMN signal – the actor, who performs the given C-act emits a BPMN signal on creating a C-fact. This has the benefit that apart from the other actor, any other actor may subscribe to the signal reception, which is aligned with the PSI-theory, where facts are present in the world, not only in the transaction, thus available also outside the transaction (modelled by interstriction links).

However, under a closer consideration, none of the above solutions are completely sufficient for a correct handling of revokes. For each revoke act, the PSI-theory specifies a certain *state* in which the transaction must be. The state is formulated like "X or further": request(ed) or further, promise(d) or further and so on. This is why we decided for another representation: the BPMN 2.0 *data store*, into which the state of the transaction is stored. This data store is connected to every C-act activity by an association.

4.3 P-(F)acts

It is not necessary to store information about them creating a P-(f)act into the data store, because they can be derived from C-(f)acts: According to the PSI-theory, the P-fact starts to exist based on acceptance of the product, so P-(f)acts can be expressed by an activity only. If need be (optimisation of an implementation), they can be stored similarly to the C-(f)acts described above.

4.4 Actors

Swimlanes in BPMN are isomorphic to actors in DEMO [10]. BPMN lacks a higher abstraction level of actor roles, being the logical sum of responsibility, authority and competence necessary to carry out the product [3]. There are generally two approaches: (i) abstracting the swimlanes to actor roles (like Decider or Concluder), (ii) remaining on the BPMN's low level of abstraction and using swimlanes to represent actors – company functional roles – like CEO or specific people like Jane.

Another possibility for representing actor roles is using BPMN *pools*, where each pool represents one actor. The resulting BPMN models will be very similar to models using swimlanes, however we have not chosen this representation because: (i) The correspondence of actor roles and transactions is not explicit, (ii) sequence flow cannot be used between pools, which would result in using messages, further complicating the diagrams.

4.5 The Composition Axiom

A composition of transactions may be dealt with in two ways: (i) to model all the transactions in one diagram, (ii) to separate diagram for every transaction. Generally, both approaches are valid, but (ii) may lead to huge diagrams, as can be seen in Figs. 8 and 9. As (ii) guarantees the limit of the diagram size, we preferred it. On the other hand, it may render understanding of the big picture harder.

We propose the following 2-part expression of the composite axiom:

1. **Launching a child transaction** in a specific place in the parent transaction. The child transaction must be started just after creating a specific C-fact. A *message-throwing event* may be used in case of initiating a single child transaction. In case of firing multiple child transactions, signals are appropriate, similar to the C-acts above. Moreover, it is needed to ensure the multiplicity. In case that it is greater than one, we need to initiate several child transaction instances. This is achieved either by using a *cycle* for creating child transactions or a *loop activity*. Modelling by cycle (Fig. 2) means, that the model contains an activity counting, how many times the activity was run. After this activity, there is a gateway. If the counter has not reached the number of child transaction instances to spawn, the process goes into message throwing event to start a child transaction instance and then the process returns to the counting activity. This happens $0 \ldots N$ times, as required. When multiplicity is modelled by a loop activity (Fig. 3), the activity is in the form of a subprocess (with parallel loop), which sends a signal[3] that starts a child transaction. In the examples described below, the first (counter) variant is used because the model is more explicit. At the same time for models with a multitude of child transactions, the more concise loop variant is recommended. Also, from the execution point, the implementation variant may be driven by the vendor, as correlation of instances must be ensured (more discussed in Sect. 4.8).
2. **Blocking execution of the parent process** until the child process has not reached the given state (creating a C-fact being waited on). This blocking can be realized by a BPMN *catching event condition* in the parent process waiting for a specific condition before the given C-act. Here, a conditional event must be used instead of a signal event, as we do not wait just for a signal, but for a specific instance in case of multiple child transaction instances. This situation is modelled in Fig. 4. Again, specific vendor correlation techniques may apply (Sect. 4.8).

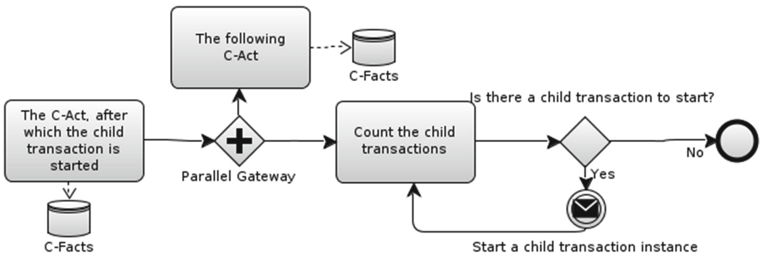
The following C-Act
C-Facts
Is there a child transaction to start?
The C-Act, after which the child transaction is started
Parallel Gateway
Count the child transactions
No
Yes
C-Facts
Start a child transaction instance

Fig. 2. Launching child transactions by using counter

[3] We cannot use a message send in this situation, because the encapsulation would be violated.

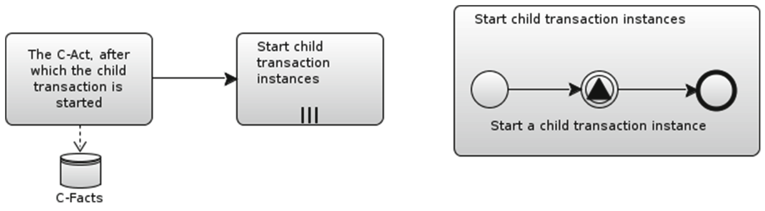
The C-Act, after which the child transaction is started
C-Facts
Start child transaction instances
Start child transaction instances
Start a child transaction instance

Fig. 3. Launching child transactions by using loop

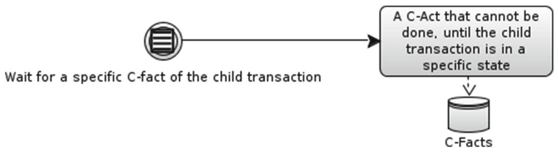
Wait for a specific C-fact of the child transaction
A C-Act that cannot be done, until the child transaction is in a specific state
C-Facts

Fig. 4. Waiting for a child transaction

4.6 Revokes

Revokes are the most challenging part of the conversion. Let us present the challenges and how we dealt with them:

- A revoke must be applied on a specific instance of the transaction; in a certain time, there can be several parallel transaction instances running. This must be ensured by the BPMN system (Sect. 4.8).
- A revoke can be fired independently on the running main process. Can be modelled straightforward, as BPMN allows several independent start events.
- A revoke can be fired only if the transaction is in an allowed state. This we ensure by an activity checking the state of the transaction, which was previously stored into a data store.
- When revoking a C-fact, after which a child transaction has been started, the child transaction must be completely revoked. This is done by calling a compensation throwing event by the revoke, followed by performing the compensation activity by the corresponding parent transaction.
- In the process flow, there can happen a situation, that a P-fact was already created (the P-act has been finished), while a revoke moves the process to a state preceding performing the P-act. In this case, it is necessary to "throw-away" the P-fact. We solve this using a BPMN compensation element and the respective compensation activity, similarly to the previous point.
- A revoke must be initiated by the actor who performed the respective C-act to be revoked. This is ensured by using the same identifier for the swimlane of the actor role initiating the revoke as for the actor role of the respective transaction.

A revoke works in the following steps according to the transaction pattern. First, the revoking actor asks the other actor for granting the revoke. The other

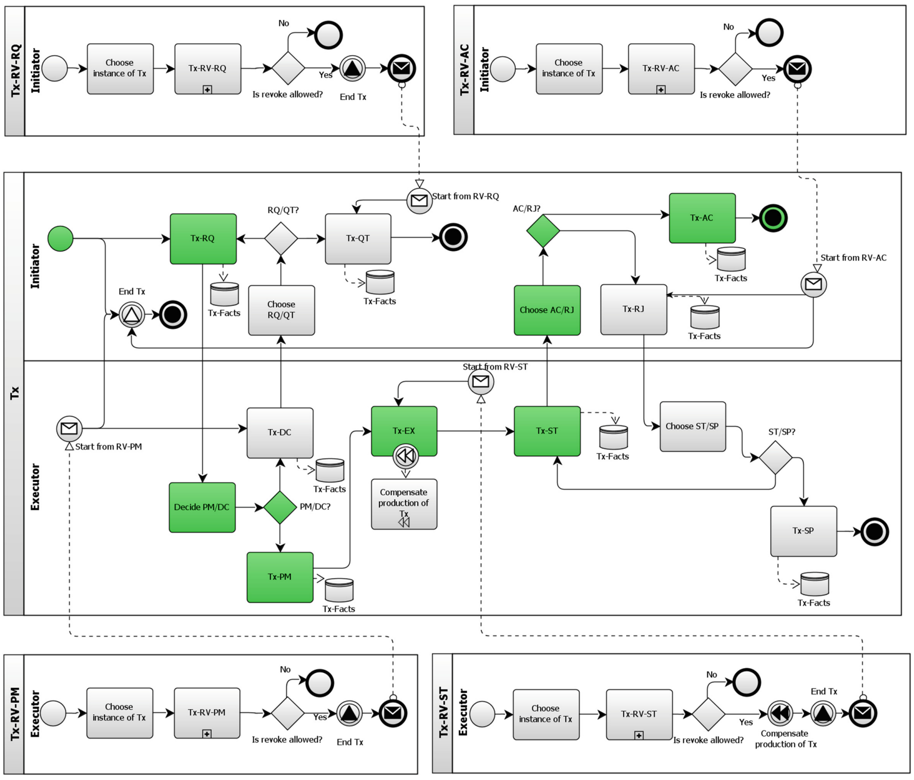

Fig. 5. Transaction in BPMN, Happy flow is marked by green colour (Color figure online)

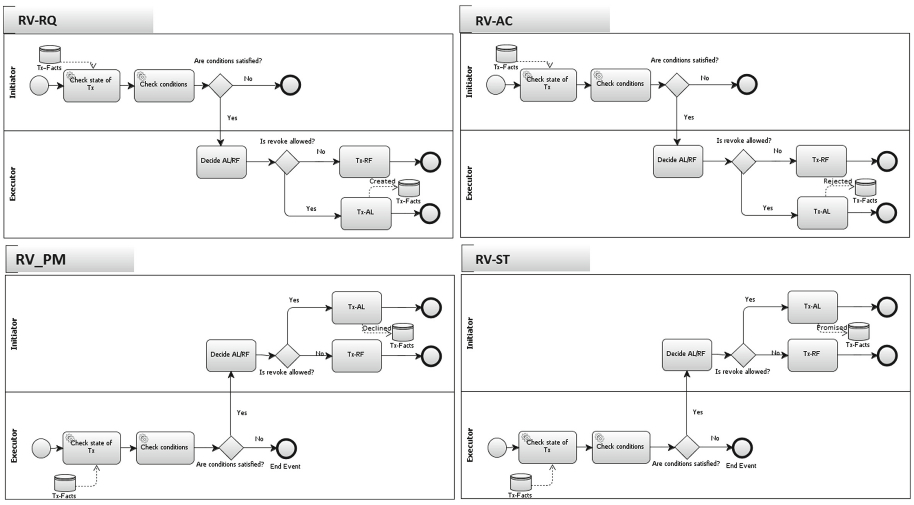

Fig. 6. Revokes in BPMN

actor allows or refuses. If the revoke is allowed, the main process returns to the appropriate state. We model this by using simple BPMN *subprocess* with a set of appropriate activities (Fig. 6).

4.7 The Resulting BPMN Model

The complete transaction pattern described by the BPMN notation illustrates Fig. 5[4]. Although it describes only one transaction, it is very complex and complicated. As it is presented in Sect. 5 and discussed in Sect. 6, models containing more than one transaction are not easily readable by usual readers and it is recommended to use them for the process execution in BPM systems.

4.8 The Execution

Apart from documentation purposes, BPMN models can be simulated and/or executed. While designing the conversion, we tried to make the resulting BPMN model precisely following the required behaviour. Unfortunately, the BPMN standard does not specify the execution implementation details. Each company developing BPM system (system for modelling, simulation and execution of processes), as Intalio, BizAgi or IBM, has their specific implementation, which requires various additional modelling and programming steps necessary to make the model executable. At the same time, some of the BPMN constructs may not be supported or they are implemented differently. All these aspects must be taken into consideration for turning the resulting BPMN models into an executable form. Generally, here are the things that must be implemented:

- Agenda handling. The possibility to start a process and providing a "task inbox" of the required reactions on the originating C-facts. This requires developing some sort of user interface (UI).
- Allowing the participants to make their choices. Again, some sort of UI solves this. Also, some choices may be determined by complex facts evaluation specified in the Action Model. There are two possible approaches:
    1. Leaving the evaluation to users, which means the users have the rules in their head or consult the Action Model or any other codification of the rules.
    2. Programming the BPM system to (help) evaluate the rules. The extent to which the automation may happen depends on the possibilities of the BPM system used and also on the context (the availability of the necessary data in the company technological systems and their accessibility).
- Signals handling.
- Implementation of reading and writing data to data stores.
- Instance matching. Specific instances of transactions must be matched in some situations as child transactions (Sect. 4.5) and revokes (Sect. 4.6). Intalio and Oracle call this concept a "correlation".

[4] This and the following models may not be legible in the printed version. We recommend obtaining the electronic (zoomable) version. The source models may be downloaded from https://ccmi.fit.cvut.cz/methodologies/bpmn/.

5 Example – Case Voley

As an example for the demonstration of our method, the traditional Case Voley example [7] was selected because of its simplicity, yet including the substantial constructs. In Fig. 7 there is the OCD diagram of this example.

The process has two transactions and three actors. The transformed BPMN model converted by the described method is in Figs. 8 and 9. Subprocesses depicted in Fig. 6 are not shown here, as they are generally the same.

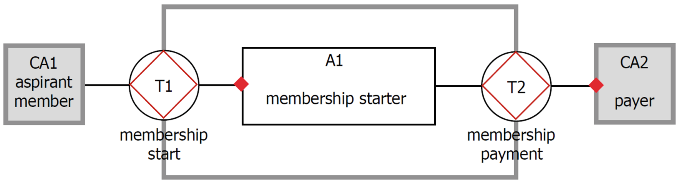

Fig. 7. OCD of Case Voley [7]

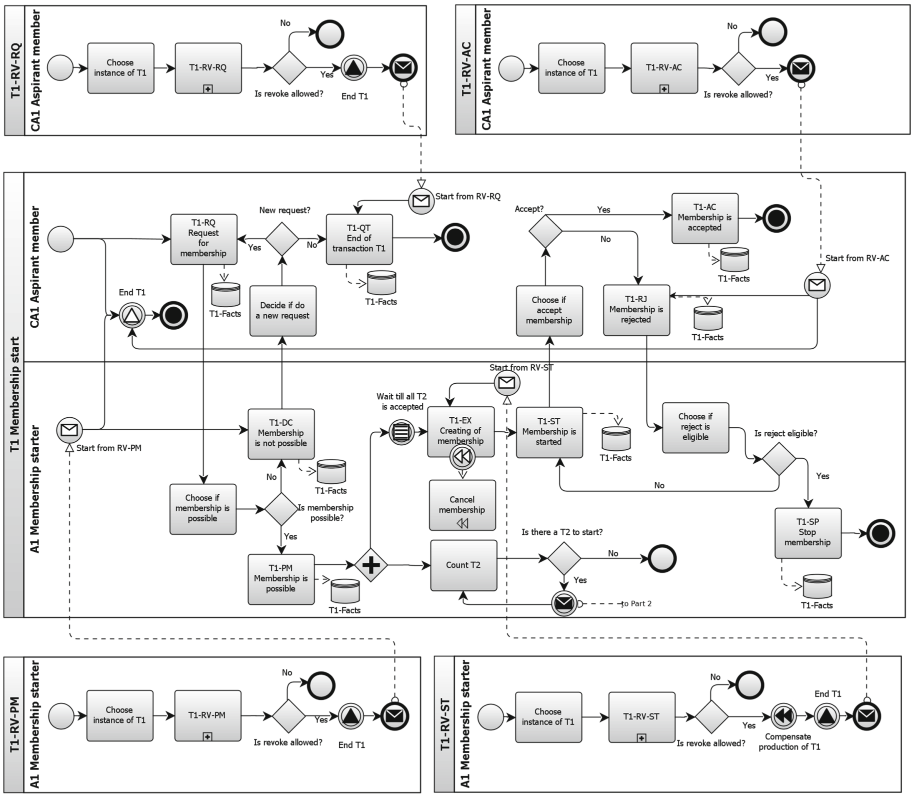

Fig. 8. Case Voley converted into BPMN – part 1

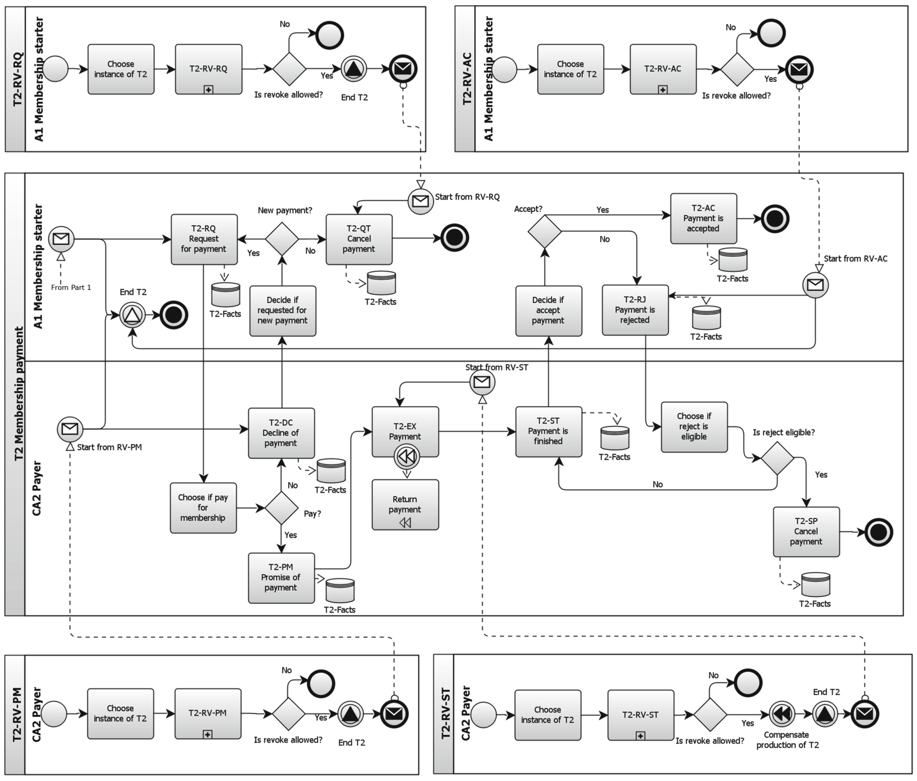

Fig. 9. Case Voley converted into BPMN – part 2

6 Discussion and Conclusions

The limitation of typical BPMN models from the view of the PSI-theory lie in their limited expression of reactions to unexpected situations. Many situations like decline, reject and especially revokes are not covered in the models, which causes operation troubles. The presented conversion method offers a remedy to this by bringing the *complete transaction pattern into BPMN*, which means including all revokes. Moreover, compared to the previous efforts, our method deals with spawning of *multiple child transaction instances* (initiation links with multiplicity $\neq 1$) and waiting for them in the parent transaction (waiting links with multiplicity $\neq 1$). Also, the resulting models are *executable*.

As for the DEMO models covered, the described conversion method covers the Construction Model plus the Process Model. Based on a concrete BPM system implementation, decision rules contained in the Action Model can be incorporated in the respective activities, as described in Sect. 4.8, which is also true for rules from the State Model.

The concept of interstriction has not been discussed, however a keen reader has probably realised that whenever an actor in its activity needs a specific information from another transaction, it is simply modelled by accessing the respective transaction data store.

The example shows that in spite of the simplicity of the DEMO model involved, the resulting BPMN model is complex. The reason is mostly the complete transaction pattern, which covers all the possible situations according to the PSI-theory. The question arises about the human readability. There are several points to this topic:

1. In practice, the model may be made smaller by leaving out the parts, which are not applicable (which means they (almost) never happen). These are typically the revoke patterns.
2. Yet, for complex models the resulting size may remain still unmanageable. In this case it would be advisable to cut the model into smaller pieces using some sort of decomposition and/or *link* BPMN elements. The concrete way how to do this may be explored in a future research.
3. It is questionable whether a human readability is required. If one wants human-readable diagrams according to the PSI-theory, the DEMO diagrams are the solution, as they have been tailored to it. It may be the case that learning and applying them comes at a lower cost than forcing the diagrams into a BPMN notation, just because "BPMN is the standard".

Our stance is that the greatest possibilities of our method lie in *machine readability*, which means generating BPMN models that can be fed into a BPMN execution system to implement an automated workflow that is able to react to every possible situation specified by the complete transaction pattern, not just a typical BPMN "happy path with a bit of branching".

Apart from converting the DEMO models, the conversion may be applied also for analysis of existing BPMN models of business processes as described in Sect. 2.1. The way of working would be to transform the BPMN models into DEMO and then generate the "supercharged" BPMN version by converting them back using our method.

As for the future work, a verification on a bigger models from practice is necessary. As such conversion will not be feasible by hand, an implementation of the conversion automation will be required.

Acknowledgements. This research has been funded by CTU SGS grant No. SGS16/120/OHK3/1T/18. The authors wish to deeply thank ForMetis BV and especially Dr. Steven van Kervel for the kind support of this research.

References

1. OMG: OMG: Business Process Model and Notation (BPMN) Version 2.0
2. Silver, B.: BPMN Method and Style, 2nd edn. with BPMN Implementer's Guide: A Structured Approach for Business Process Modeling and Implementation Using BPMN 2.0. Cody-Cassidy Press, New York, October 2011
3. Dietz, J.L.G.: Enterprise Ontology: Theory and Methodology. Springer, Berlin (2006)

4. Dietz, J.L.G., Hoogervorst, J.A.P., Albani, A., Aveiro, D., Babkin, E., Barjis, J., Caetano, A., Huysmans, P., Iijima, J., Kervel, S.J.V.: The discipline of enterprise engineering. Int. J. Organ. Des. Eng. **3**(1), 86–114 (2013)
5. Op 't Land, M., Dietz, J.L.G.: Benefits of enterprise ontology in governing complex enterprise transformations. In: Albani, A., Aveiro, D., Barjis, J. (eds.) EEWC 2012. LNBIP, vol. 110, pp. 77–92. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29903-2_6
6. Décosse, C., Molnar, W.A., Proper, H.A.: What does DEMO do? A qualitative analysis about demo in practice: founders, modellers and beneficiaries. In: Aveiro, D., Tribolet, J., Gouveia, D. (eds.) EEWC 2014. LNBIP, vol. 174, pp. 16–30. Springer, Cham (2014). doi:10.1007/978-3-319-06505-2_2
7. Dietz, J.L.: The Essence of Organization - An Introduction to Enterprise Engineering. Sapio bv, Voorburg (2012)
8. Guizzardi, G., Wagner, G.: Can BPMN be used for making simulation models? In: Barjis, J., Eldabi, T., Gupta, A. (eds.) EOMAS 2011. LNBIP, vol. 88, pp. 100–115. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24175-8_8
9. Caetano, A., Assis, A., Borbinha, J., Tribolet, J.: An application of the Ψ-theory to the analysis of business process models. In: Poels, G. (ed.) CONFENIS 2012. LNBIP, vol. 139, pp. 258–267. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36611-6_24
10. Nuffel, D., Mulder, H., Kervel, S.: Enhancing the formal foundations of BPMN by enterprise ontology. In: Albani, A., Barjis, J., Dietz, J.L.G. (eds.) CIAO!/EOMAS -2009. LNBIP, vol. 34, pp. 115–129. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01915-9_9
11. Naplava, P., Pergl, R.: Empirical study of applying the DEMO method for improving BPMN process models in academic environment. In: 2015 IEEE 17th Conference on Business Informatics, vol. 2, pp. 18–26, July 2015
12. Figueira, C., Aveiro, D.: A new action rule syntax for DEmo MOdels based automatic workflow procEss geneRation (DEMOBAKER). In: Aveiro, D., Tribolet, J., Gouveia, D. (eds.) EEWC 2014. LNBIP, vol. 174, pp. 46–60. Springer, Cham (2014). doi:10.1007/978-3-319-06505-2_4
13. Heller, S.: Usage of DEMO methods for BPMN models creation. Master thesis, Czech Technical University in Prague. Computing and Information Centre (2016). https://ccmi.fit.cvut.cz/wp-content/uploads/2017/03/Heller_thesis_2016.pdf