Automated Model Transformation Method from BORM to BPMN

Jakub Tůma and Petr Hanzlík

Department of Information Engineering
Faculty of Economics and Management
Czech University of Life Sciences
Kamýcká 129, 165 21 Praha 6 - Suchdol, Czech Republic

Abstract

This paper presents an automated model-to-model transformation method between two popular business modelling notations, namely BORM-II (Business Object Relation Modelling, second generation) and BPMN (Business Process Modelling Notation). Our transformation method is based on unique combination of Petri nets and final state machine, which is convergent.

Mathematics Subject Classification: 68U35, 68R10

Keywords: Model Transformation, Business Object Relation Modelling, Business Process Modelling Notation, Finite State Machine, Petri Net, Transformation Method Design

1 Introduction

This paper describes two models used to represent business processes and presents a transformation method between them. The first model under investigation is BPMN (Business Process Modelling Notation) [2] and the second model is BORM-II (Business Object Relation Modelling, second generation) [11], [12]. This research aims to connect popular BPMN Process-State diagram (PSD) with older yet innovative BORM Object Relation Diagram (ORD) [13],
in order to bridge the gap between engineers and business analysts, who are
typical users of these methods. This complies with the principles of transfor-
mation tool described within Model Driven Architecture (MDA).

This paper is divided into two main parts. The first part describes ba-
sic principles of BORM ORD and BPMN PSD and explains the theoretical
background of both models. The second part explains the proposed transfor-
mation method from BORM to BPMN-based notation. The transformation is
thoroughly described with syntax and semantics of Petri-nets and Final State
Machine (FSM).

This contribution follows a previously published paper on model to text
transformation, where a transformation method between BORM diagram in-
put and textual output structured in compliance with the Semantics of Busi-
ness Vocabulary and Rules (SBVR) [12] has been presented. Several different
approaches have been used for mapping one business modelling notation onto
another. Paper [4] shows how DEMO process is transformed into BPMN pro-
Petri net. Paper [14] uses direct transformation approach through CFG (Con-
text Free Grammar) and FSM. In comparison, our approach uses a unique
combination of Petri Nets and FSM. To the best of our knowledge, such ap-
proach has not been presented so far.

1.1 BORM

Business Object Relationship Modelling [9], [19] is an object-oriented software
engineering methodology, which has been proven to be very effective in develop-
ment of information and knowledge systems. It was designed as a method
capable of covering all phases of software development; however, it is primarily
used in the initial phase of project, also known as business analysis. It makes
extensive use of business process modelling [8]. The effectiveness of BORM is
achieved by unified and simple methods for presenting all relevant aspects of
model and use of a limited, easily comprehensible group of concepts for every
life-cycle phase. BORM represents every concept with the same set of symbols
for data structure and communication. This all makes it easy to understand
even for the first-time users with limited to no knowledge of business analysis.
In order to visualize information, BORM uses simple diagrams that contain
only the necessary number of concepts and symbols. These concepts and sym-
bols cover most of the needs for initial description of a model and its processes.
The following core symbols of ORD belong to the core elements used for the
initial model description:

- Participant is an object representing a stakeholder involved in one of
  processes modelled. It is identified during analysis.
State describes a sequential change of participant in time.

Activity represents an atomic step of behaviour of an object recognised during analysis.

Communication represents a data flow and dependencies between activities. Bidirectional data flows may be present.

Transition describes a sequence state-activity-state, thus represents changes of states through activities.

Condition describes constraint that holds for communication or activity, [9]

1.2 BPMN

The Business Process Modelling Notation (BPMN) [6] has emerged as a standard notation for capturing business processes, especially at the level of domain analysis and high-level systems design. This notation inherits and combines elements from a number of previously proposed notations for business process modelling, including XML Process Definition Language (XPDL) [20] and Activity Diagrams component of Unified Modelling Notation (UML) [5]. A BPMN process model is composed of:

- Activity nodes
- Business events (items of work performed by humans or by software applications)
- Control nodes capturing the flow [3]

1.3 Goal

This article aims to show the continual development process of BORM, and its associations with other popular modelling methods. The main goal is to design an automated transformation tool, capable of converting BORM ORD model to BPMN PSD model. Partial goals include verification and validation of the transformation method proposed. The method will be verified by a mathematical proof of transformation.

1.4 Method

This research has been conducted following a stepwise methodology.
1. Analysis of both notations, BORM ORD and BPMN PSD, is conducted. Particular notation objects and the elements they consist of are analysed, described, and dropped (if relevant).

2. Mappings between particular notation objects are set up using identified notation elements.

3. Definition of mapping rules that contains both syntax and semantics of mapping rules is formalised. Mapping is done from notation model onto Petri-net.

4. Verification is done by synthesis. A BORM ORD is mapped to Petri-net. Petri-net is further specified to FSM. The same approach is repeated with BPMN PSD. Finally, equivalency of BORM ORD FSM and BPMN PSD FSM is proven.

2 Main Results

In order to perform transformation between BORM ORD and BPMN PSD elements, both notations must be analyzed. As a result of this analysis, a set of mapping rules has been established [2], [8], [7]. These mapping rules are presented in the formula 1, where there are elements of BORM on the left hand side and elements of BPMN on the right hand side.

\[
\begin{align*}
\text{Participant} & \Leftrightarrow \text{Lane (Swimlane)} \\
\text{Start} & \text{ of } \text{role} \Leftrightarrow \text{Start} \\
\text{End} & \text{ of } \text{role} \Leftrightarrow \text{End} \\
\text{Activity} & + \text{ State} \Leftrightarrow \text{Event} \\
\text{Communication}, \text{ Transition} & \Leftrightarrow \text{Sequence flow} \\
\text{Data flow} & \Leftrightarrow \text{Message flow}
\end{align*}
\]

These rules are used as a base for the formal definition, which covers both syntax and semantics of mapping. In order to fill the gap between notation models, the Petri-nets are employed as an intermediary representation. We thus further focus on mapping from notation models onto Petri-nets.
Definition 2.1. Let the BPMN process P have core elements syntax:

\[ P = \{O, F, Cond, A, E, G, T\} \]

\[ O \in \{A, E, G\}; \]

\[ A \text{...activities, } E \text{...events, } G \text{...gateways} \]

\[ A \in \{T\}; \quad T \text{...task} \]

\[ E \in \{e^s, e^i, e^f\}; \]

\[ e^s \text{...startevent} (\text{only one in diagram}), \]

\[ e^i \text{...intermediate event}, \]

\[ e^f \text{...final} \text{ (end)} \text{ event} (\text{only one}) \]

\[ G \in \{G^x, G^m\}; \]

\[ G^x \text{...XOR decision gateway}, \]

\[ G^m \text{...XOR merge gateway} \]

\[ F \subseteq \{O \times O\}; \]

\[ F \text{...control flow relation, i.e. set of sequenceflows} \]

\[ \text{Cond} : F \bigcap (G^x \times O) \rightarrow C, \text{where } C \text{ is} \]

\[ \text{boolean function : } C \in \{\text{true, false}\}. \]

Definition 2.2. Let the BPMN model M have core elements syntax:

\[ M \text{ is triple : } M = (Q, HR, F^M) \]

\[ Q \text{...set of processes(P), } HR \text{...connected graph} \]

\[ F^M \text{...set of message flows between processes} \]

\[ HR = ((P_1, P_2) \in Q \times Q) \]

\[ F^M \subseteq (\bigcup_{P \in Q} (T_P \cup e^f_P \cup e^i_P)) \times (\bigcup_{P \in Q} (T_P \cup e^f_P \cup e^i_P))/U_{P \in Q}(O_P \times O_P) \]

Definition 2.3. Let the BPMN process P be compatible with BORM ORD iff

\[ F \in \]

\[ \forall s \in e^s, in(s) = 0 \land | out(s) | = 1; \]

\[ \forall f \in e^f, out(f) = 0 \land | in(f) | = 1; \]

\[ \forall g \in G^x, | in(g) | = 1 \land | out(s) | = 2; \]

\[ \forall g \in G^m, | out(g) | = 1 \land | in(s) | = 2; \]

\[ \forall x \in O, \exists s \in e^s, \exists s \in e^e, sF^x \wedge sF^e. \]
Definition 2.4. BPMN model \( M \) is compatible with BORM ORD model iff \( M \) is set of compatible BPMN processes \( P \) and \( HR \) is directed acyclic graph (DAG).

Theorem 2.5. Complete Petri net formalizes semantics of standalone BPMN (see formulas 12, 13, and 14). BPMN semantics with communication between interacting processes is then formalized by Petri net in the formula 15. This definition was inspired by Dijkman et al. [3].

\[
P = \bigcup_{p \in Q} \{ p_s \mid s \in e^S_p \} \cup \{ p_e \mid e \in e^E_p \} \cup \{ p(x,y) \mid (x, y) \in F_p \} \quad \text{— start event}
\]
\[
T = \bigcup_{p \in Q} \{ x \mid x \in T_p \cup e^i_p \wedge \text{in}(x) \neq 0 \} \cup \{ t(x,y) \mid x \in G^X_p \wedge y \in \text{out}(x) \} \cup \{ t(x,y) \mid x \in G^M_p \wedge y \in \text{in}(x) \} \cup \{ t_s \mid s \in e^S_p \} \cup \{ t_e \mid e \in e^E_p \} \quad \text{— end event}
\]
\[
F = \bigcup_{p \in Q} \{ (p(x,y), y) \mid y \in T_p \cup e^i_p \wedge \text{in}(y) \} \cup \{ (y, p(y,z)) \mid y \in T_p \cup e^i_p \wedge \text{in}(y) \neq 0 \wedge z \in \text{out}(y) \} \cup \{ (p(x,y), t(y,z)) \mid y \in G^X_p \wedge x \in \text{in}(y) \wedge z \in \text{out}(y) \} \cup \{ (t(y,z), p(y,z)) \mid y \in G^X_p \wedge z \in \text{out}(y) \} \cup \{ (p(x,y), p(y,z)) \mid y \in G^M_p \wedge x \in \text{in}(y) \} \cup \{ (t(y,z), p(y,z)) \mid y \in G^M_p \wedge x \in \text{in}(y) \wedge z \in \text{out}(y) \} \cup \{ (p_s, t_s) \mid s \in E^S_p \} \cup \{ (t_s, p(s,y)) \mid s \in E^S_p \land y \in \text{out}(s) \} \cup \{ (p(x,e), t_e) \mid e \in E^E_p \land e \in (e) \} \cup \{ (t_e, p_e) \mid s \in E^E_p \} \quad \text{— start a process}
\]
\[
PN = (P_M, T_M, F_M) \quad \text{— complete Petri net}
\]
\[
P_M = P \cup \{ p_{x,y} \mid (x, y) \in F^M \land y \notin \bigcup_{p \in Q} e^S_p \} \quad \text{— connecting place}
\]
\[
T_M = T \quad \text{— transition}
\]
\[
F_M = F \cup \{ (x, p_{x,y}) \mid (x, y) \in F^M \land x \notin \bigcup_{p \in Q} e^E_p \land y \notin \bigcup_{p \in Q} e^S_p \} \cup \{ (p_{x,y}, y) \mid (x, y) \in F^M \land x \notin \bigcup_{p \in Q} e^E_p \land y \notin \bigcup_{p \in Q} e^S_p \} \cup \{ (t_{x,y}, p_{x,y}) \mid (x, y) \in F^M \land x \in \bigcup_{p \in Q} e^E_p \land y \notin \bigcup_{p \in Q} e^S_p \} \quad \text{— flow}
\]
\[
(12)
\]
\[
(13)
\]
\[
(14)
\]
\[
(15)
\]
Lemma 2.6. Transformation method.

Conditions 9 and 10 from definition 2.3 are crucial for transforming Petri net to State machine (SM). In any state machine (SM), every transition has one incoming arc, and one outgoing arc, and all markings have exactly one token. Consequently, no concurrency can occur, but conflict may exist, i.e. non-determinism. Mathematically:

\[ \forall t \in T : |in(t)| = |out(t)| = 1. \] (16)

The transitions in our case fulfil this condition, because we only use data-based xor gateways (XOR GATE, definition 2.1, formula 3). Conditions 9 and 10 ensure the state machine cannot be nondeterministic. Therefore, the SM can be considered finite for selected BPMN elements. If BPMN core elements are mapped to Petri-net, there is one important rule for transformation - output function is dependent only on the state, a rule that also applies for Moore automaton (see definition [15]). The BORM objects can be mapped to Petri net with rules proposed by Podlucky and Pergl [18]. They use a prefix machine to formalise BORM ORD [17], specifically, for realizing decision forks (XOR gateway in BPMN). The relationship between BORM ORD model and Mealy automaton has been formalized in [16]. Eventually, transformation between Mealy and Moore automaton and vice versa is known issue discussed in several papers, e.g. [1]. All of these facts are used as a base for following prove:

Proof of transformation method.

Finite State Machine (FSM)

\[ A...input alphabet, B...inner states, C...output alphabet \]

\[ C = f(A, B) - Mealy automata, C = f(B) - Moore automata \]

\[ BORM ORD model \Leftrightarrow Mealy automata [16] \]

\[ BPMN PSD model \Leftrightarrow Moore automata [3] \]

\[ Mealy automata \Leftrightarrow Moore automata [1] \]

3 Conclusion

The paper makes a contribution to the area of transformation method design. Our method uses a unique combination of Petri Nets and FSM to realize the conversion between BORM ORD and BPMN PSD. This paper concludes that the gap between IS designers and domain experts who typically use different tools to express their needs, can be bridged by automated transformation of previously mentioned models (BORM and BPMN). This paper is a small step to realize this alignment.
Acknowledgements. We would like to express our sincere thanks to Professor Vojtěch Merunka, team leader Robert Pergl and many supporters.

References


Received: April 16, 2015; Published: September 16, 2015