Transformation from EPC to BPMN

Willi Tscheschner

Hasso-Plattner-Institute, Potsdam, Germany
willi.tscheschner@student.hpi.uni-potsdam.de

Abstract. Business process modeling is a big part in the industry, mainly to document, analyze, and optimize workflows. Currently, the EPC process modeling notation is used very wide, because of the excellent integration in the ARIS Toolset and the long existence of this process language. But as a change of time, BPMN gets popular and the interest in the industry and companies gets growing up. It is standardized, has more expressiveness than EPC and the tool support increase very rapidly. With having tons of existing EPC process models; a big need from the industry is to have an automated transformation from EPC to BPMN. This paper specified a direct approach of a transformation from EPC process model elements to BPMN. Thereby it is tried to map every construct in EPC fully automated to BPMN. But as it is described, not for every process element works this out, so in addition, some extensions and semantics rules are defined.

1 Introduction

As a current situation, EPC are used very often in the industry and it exists a lot of process models, not least because of the strong impact of using EPC in the ARIS Toolset. With this wide usage, EPC is seen as a de-facto standard. As a change of time, with the standardization of BPMN, which is more expressive, and the starting of having a better tool support, BPMN becomes more acceptance and usage in the industry. A hugh need out of this is to have a fully automated transformation from existing process models in EPC to process models in BPMN.

Event-driven Process Chain (EPC) is invented and defined in the year 1992 at the Institute for Information Systems in Saarbruecken, Germany, from Keller et al. [6] in cooperation with the SAP AG. The main purpose was to have a clear defined documentation of business process models. EPC is mainly used in the SAP R/3 reference model of SAP AG and in the ARIS Toolset of IDS Scheer AG. EPC process models are event-driven, or in other words state-based, that means that the main focus for one process model is more on the system states and their behaviour, rather than on the interaction and communication between organizations. Processes modeled in EPC are basically used within one company or organization.

The Business Process Modeling Notation (BPMN) is a standardized business process notation which is defined and specified from the Object Management Group (OMG) [1] and available in the Version 1.1. The main focus for BPMN is a powerful and understandable documentation of business process models which is used for all business user groups. From the technical developer, who has more focus on the technical aspects up to the business person who is deeper into the managing and optimizing of business processes. BPMN supports control flow behavior as well as interaction behaviour and data flow. Therewith it is not only used for processes within one company, it is also used for modeling interaction processes of multiple companies or organizations.

In this paper, I describe how to automatically transform the core and extended EPC process model to a BPMN process model. The paper is structured as followed: Section 2 gives a quick overview of already exists mapping approaches. Section 3 explains which EPC model constructs are available and which BPMN constructs are used for the transformation. Section 4 specifies the actual transformation. Section 5 introduces some notes and extensions. Finally Section 6 and 7 present an example and the conclusion.

2 Related Work

A transformation can be done with different approaches, this is discussed in this section.

One approach could be an indirect mapping, that means a mapping which uses already available transformations and does the actual transformation over these with a third or a fourth modeling language. So it would be possible to map EPCs to Petri nets, a formalized modeling language for distributed systems, like its already done from W.M.P. van der Aalst [8]. Take the transformation from BPMN to Petri nets which is described from Dijkman et al. [5] and just map both Petri nets together. The transformation itself would be easier, because already exist mappings can be reused. Only the mapping from one process model to another process model in the same language should be described. But, this approach would not be effective, because Petri nets used in [8, 5] are less expressive compared to BPMN and EPC, so that from the particular process languages the important structural and semantical information gets lost.

Another approach would be a direct mapping. It can be done directly from the structure and from the core model data, or over an equivalent representation of this model, like for EPC the EPC Markup Language (EPML) [7] or for BPMN the Business Process Modeling Language (BPML) [2]. Both abstract process languages are XML based [3] and can be transformed via XSLT [4]. A generic concept of a XML-based transformation of business process models is specified in the paper from Vanderhaeghen et al. [9]. There it is described, how a mapping over the particular abstract XML-based process languages could look like. As an example they use the process languages EPC and BPMN, but does not explain the actual mapping. They only explain the individual steps of the XML mapping. As a conclusion, a direct mapping is very effective, because no information is hidden, the whole structual information is still there and the semantics is also still available, because the transformation is done directly on the basis of both process languages.

In this paper, a direct mapping from EPC to BPMN is described, with the main focus on the transformation itself, rather than on the underlying technology or data structure. It is pointed out which process elements in EPC can be mapped to which process elements in BPMN, including the peculiarities of the inidividual constructs.

3 Modeling Elements/Concepts of EPC and BPMN

In the rules for the transformation all EPC process elements and a subset of BPMN process elements are used and get introduced in this section.

3.1 EPC

In Figure 1 all defined EPC process elements are shown. The core EPC elements are formed by these ones which are used in the documentation from Keller et al. [6]. With these elements a basic process model can be specified and documented. Only the core set of elements are formalized [8] and documented. The extended elements of EPC are neither formalized nor well documented. Because of the main usage in ARIS the documentation and semantics of core EPC and extended EPC are strongly connected to this.

In the core set of EPC there is a function defined which is specified as a semantical transformation rule from an input system state to an output system state. An event is described as an occurred system state which affects the following process flow. Connectors describe how functions and events can be combined; options are exclusive disjunction (xor-connector), logical conjunction (and-connector), and inclusive disjunction (or-connector). Connecting the process elements to each other is done with a control flow. A process should start and end with an event and, in general, the graph should be directed, events and functions should alternate, and optional connectors can appear. Out of a xor-connector or or-connector where more than one control flow is following, functions must not appear, because then the semantics is not clear defined.

As an extension to the core set there are some more constructs specified. The main focus for the extensions are to add organizational structure and data flow to the process model, which cannot be described with the core EPC. Therefore following elements are specified: organizational unit, position, data, and system. Those can only be mapped to a function with a relation. For data the relation can also be directed, which means that the data has to be read or written. In addition a process link is introduced, that describes a hierarchical or linked (flat) concatenation to another process model. This can be used instead of an event or a function.

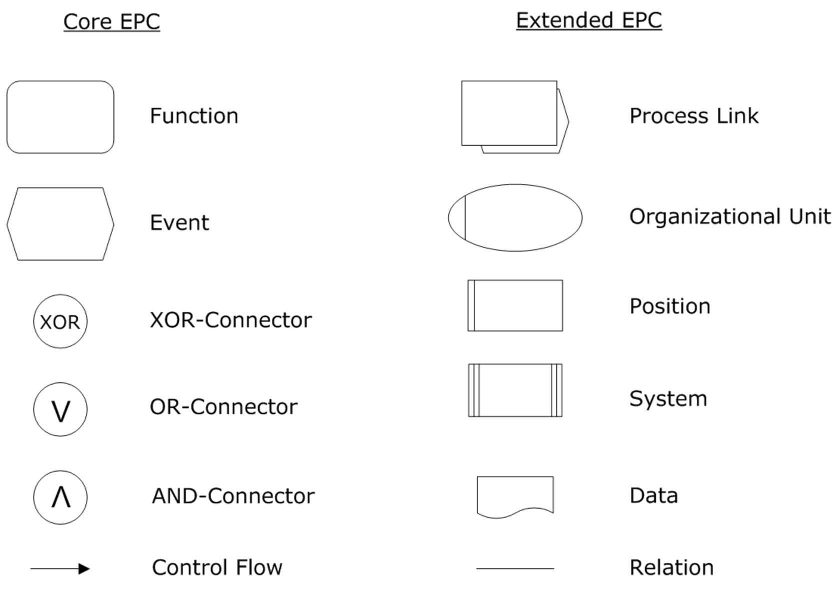

Fig. 1. Overview of EPC process elements

3.2 BPMN

The standardized Business Process Modeling Notation (BPMN) is defined as "readily understandable by all business users" (p.1, [1]) from the OMG. Compared to EPC, the constructs are well defined and specified. In Figure 2 all the BPMN elements are shown which are targeted in the transformation from EPC to BPMN. There are far more constructs but because of the different scope of both process languages, not every process element from BPMN is used for the transformation and therewith not in the focus of this paper.

The main elements in BPMN are tasks (an atomic activity), gateways ("used to control the divergence and convergence of Sequence Flow" (p.18, [1])) and sequence flow to determine the control flow of the process. To signal a start and the end of one process, start events and end events are used. With these constructs, a basic process model can be modeled.

To extend the "normal" behavior of a process flow, special constructs are defined. For example, a sub-process (non-atomic activity) can compound another process, pools and lanes (sub-portions of a pool) can define available organizations or roles, data objects specify the usage of the data in the process, and annotations give a comment or note to a process element. Data objects and annotations are linked via an association to the particular construct. For data objects the association can be directed, which defines the flow of the data. In addition, intermediate events are defined, like message intermediate event which can be used for sending a message ("throwing") or receiving a message ("catching").

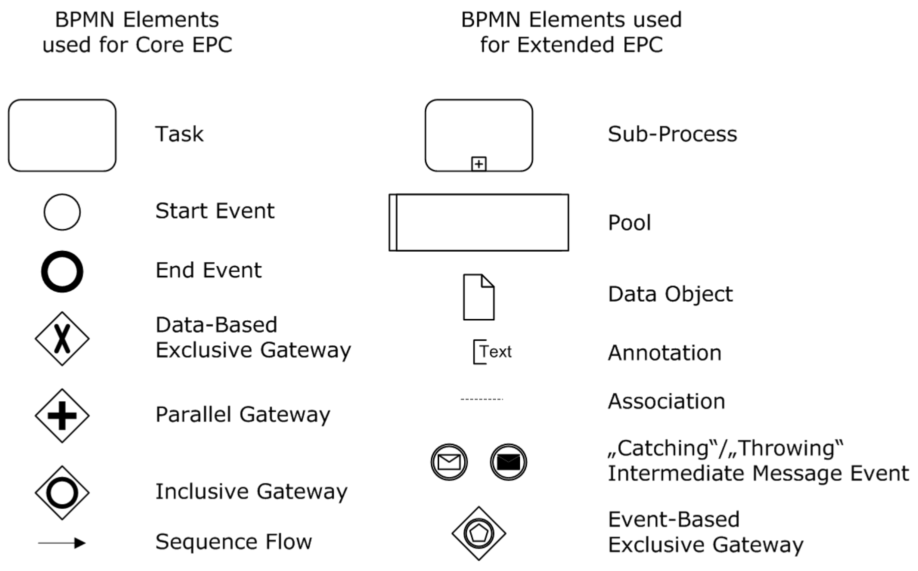

Fig. 2. Overview of used BPMN elements

In BPMN an exclusive gateway is separated in data-based, where the incoming data decide the following control flow, and event-based, where the following intermediate events define the behavior of the gateway.

4 Transformation

As already discussed, the transformation from EPC to BPMN is very interesting and important, and in this section this is going to be explained in detail on the basis of several rules.

The transformation is specified as a direct mapping from EPC constructs to BPMN process elements. It is done mostly as a structural mapping. But because of the difference of both notations, I also introduce some additional concepts, like name matching, and semantics rules to enlarge the number of available transformations; this is described in the Section 5. The transformation is grouped to the core set of EPC process elements (function, event, and connector) [6] defined in Rule 1 to 3 and to the extended set of EPC elements (organization, position, data, system, and process link) defined in Rule 4 to 7.

4.1 Transformation - Core EPC

Followed in this subsection the description of the transformation from the core EPC process elements to BPMN is given. Additionally, in Figure 3 there is an overview illustrated which shows all rules of the core EPC and the corresponding transformation of the process elements.

Rule 1: Function

In EPC a function is descriped as an activity which supports the completion of a business objective [6]. It is semantically a process rule for transforming an input system state to a following output system state. In BPMN an activity is a generic term for work that a company performs [1]. A task is specified as an atomic activity which cannot be broken down to several activities. With both definitions a function in EPC is mapped to a task in BPMN.

Rule 2: Event

Events in EPC are not easy to map to any BPMN process elements, because there are differently specified. Events in EPC are defined in the article of Keller et al. [6] as: events can trigger functions, events can get triggered by functions, events define an occurred business situation, and events specify the business conditions. That means an event is an occurred state in the information system which can determine the following process flow. It is also described as a passive component in the information system. An event in BPMN "is something that 'happens' during the course of a business process." (p.35, [1]). Same as in EPC it has no time consumption and can affect the flow of the process. The difference is, an event in BPMN is defined more as a trigger or a consumption of "something" which causes or has an impact on the following process flow. "Something" could be a message, a signal, or even an error that came up. Hence, it is not possible to map an event in EPC directly to an event in BPMN. Because of that, an event in EPC is hidden in the transformed BPMN process model, except of some events, which are defined in the following sub sections. In addition, in Section 5 an advanced concept is introduced, which looks on the semantics of the particular events, and defines some extra rules for mapping an event to BPMN.

Rule 2a: Event without incoming Control Flow

A process model, described in EPC, is determined by one or more events (an explicit entrance state of this process). Similar to this, a process model in BPMN is also implicitly or explicitly started through a start event. Out of this, an event in EPC which has no incoming control flow can be mapped to a start event in BPMN.

Rule 2b: Event without outgoing Control Flow

Process models defined in EPC are terminated by events (an explicit end state of this process). In BPMN a process model is also implicit or explicit terminated through an end event. With this, an event in EPC which has no outgoing control flows can be mapped to an end event in BPMN.

Rule 3: Connector

Connectors (xor-connector, and-connector, and or-connector) in EPC are defined as concatenation points in the process for events and functions [6]. Gateways in BPMN are defined for controlling the divergence and convergence of the sequence flow [1]. With this, the several connectors can be mapped to the same gateway (xor-connector to data-based exclusive gateway, and-connector to parallel gateway, and or-connector to inclusive gateway) independent from the splitting or joining behavior.

In addition, if a xor-connector or an or-connector occurs which has splitting behavior (by concatenation of events) the process flow is affected by the states of the following events. Depending on the previous function and the output situation, this event gets chosen which represents the appropriated state. Similar to this, in BPMN the process flow after a gateway is determined by the condition of the following sequence flow. Therewith, those events in EPC can be mapped to the condition expression of the following sequence flows (shown in Figure 3 - Rule 3).

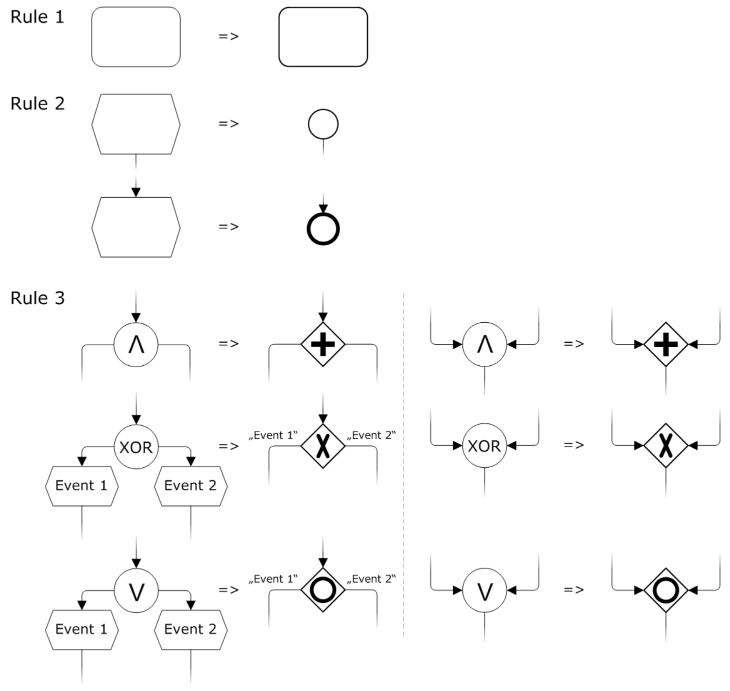

Fig. 3. Overview of the transformation from core EPC to BPMN

4.2 Transformation - Extended EPC

In the following sub sections the transformation from the extended EPC constructs to BPMN is described. In Figure 4 the transformation rules for extended EPC are visual shown, to give a better overview of the transformation rules.

Rule 4: Organization Unit/Position

The organization unit in EPC is defined as a structural unit within a company. A position in EPC is specified as an occurred specific role within the process. A pool in BPMN represents a participant in the process (either an organization/company or a more specific business role). A lane is only defined as a sub-portion in a pool and can be specified by the user [1]. So it could be defined either as an organizational unit or as a specific role. With this, an organization and a position in EPC can be mapped to a lane in BPMN which is a child of a pool.

Note If at least one organization or position is modeled in the EPC model, a lane with an unspecified name should be added to the pool, where all those functions are added, that have no organization or position related to.

Rule 5: Data

In EPC a function can have manipulate, read, or write access to data or information (depending on the direction of the relation). This could have implicit influence of the process flow. If for example the information is not available the process flow would be on hold waiting for the access. The same is defined in BPMN with data objects. Data object do not have direct affect on the sequence flow or message flow in any kind, but these could be required to perform or to produce. Data object provide additional information for the process. Anyway, both have the same meaning and can be mapped directly to each other.

Rule 6: System

A system in EPC is a used system for a particular function. It means that a user should use this system to fulfill this function to get the specified output. In BPMN there is nothing like this. A system could appear as a participant in the process, but then the system is actively involved in the process and not just supporting the role. The better transformation is that a system in an EPC model is mapped to an annotation in BPMN, because than the information is not lost and it is still related to the particular task.

Rule 7: Process Link

A process link in EPC is used for splitting down the process into a sub-process or to refer to a following process. In other words, process links can be used for specialization of one process step or to link this process to a followed or previous process. In BPMN there is a sub-process construct which is defined as the same. There is an embedded sub-process and an independent sub-process [1]. Embedded means that the sub-process is defined in the same scope, but specializes this process step. Independent sub-process means that this sub-process refers to a different process within a different scope. With both concepts, a process link construct in EPC can be mapped to a sub-process in BPMN.

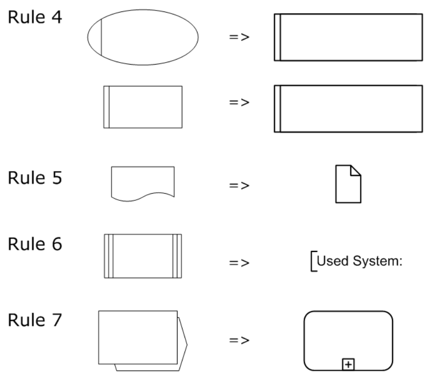

Fig. 4. Overview of the transformation from extended EPC to BPMN

5 Transformation Extension with Semantics Rules

As previously discussed, not every process element in EPC can get mapped directly to BPMN. To enlarge the quality of the transformation, in this section an approach for this problem is discussed.

Because of the different scope of both process languages and the different semantics of events in EPC and BPMN, a direct mapping for this construct cannot be performed. As aforementioned, this paper introduces some rules where events are mapped to constructs in BPMN. But sometimes, events in EPC really behave like events that occur in the information system. Often it is used as e.g. “Received a Message” or “Message is sent”. With the current mapping, those events would not come up in the transformed BPMN process model, because the transformation rules are just looking on the structure and not on the semantics of the process. Not to lose those events, followed are some semantics rules defined.

Before the semantics rules can get defined, we have to look on the semantics of “Received a Message” and “Message is sent”. Receiving a message means, that “something” or “somebody” is waiting or pending till this message occurs. No action in advance is needed, but when message comes up, it should be handled somehow. Sending out a message is different, because “something” or “somebody” has to be actively involved in this process step. Before the process state achieve that a message was sent out, somehow the message must be actively sent out. This clearly shows the differences between both events and the need for seperating them. When looking at available EPC process models, there are clear coherences: Receiving a message is normally specified only with one occurred event, but followed with several functions that handle this message. In difference, sending out a message is normally declared as a function with “Sending out a message” with a following event that symbolizes “Message sent”. But because of having a function specified, this obviously demonstrates the active performing of sending a message.

To find those entire events, the user should specify several keywords, separated by the two different relevancies. With these keywords, during the transformation, the event can be identified and considered. To enhance the number of identified events and functions, the search algorith should not look only on the same words rather than on radical of the words. Therewith you can specify a keyword “send” for a “sending” event and you will find an event which is called “Message sent” as well.

Coming to BPMN, there exist the same behavior and the two different meanings. On the one hand, there is a “catching” event that consume an event or wait of it. On the other hand, there is a “throwing” event defined that is specified as trigger or spawn of an event. Combined both principles with an intermediate message event, there are “catching” and “throwing” intermediate message events defined [1], which fits perfectly to the semantic of the behaviour above.

With this cognition, there are two semantics rules defined:

Sematics Rule 1: “Sending” Events

Every found “send” function, followed (maybe indirectly) by a “sending” event, in the EPC process model can be mapped to a “throwing” intermediate message event (shown in Figure 5 - A). If this event has no following process element, the intermediate message event must be an end message event, shown in Figure 5 - C. The title of the found function defines the action and should title the intermediate event.

As a clarification, a “throwing” message event cannot be a start event. This is nor defined in BPMN and it is nether defined in EPC, because process models in EPC starts with an event and not with a function. If nevertheless this happens, the transformation can be done normally, but a start event has to add to the BPMN process which is preceding the intermediate message event.

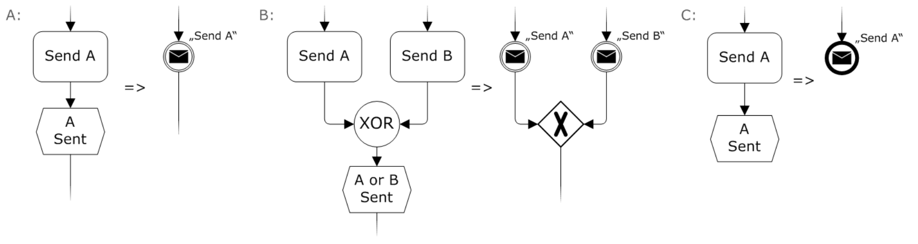

Fig. 5. Semantics Rule 1

Sematics Rule 2: "Receiving" Events

Every available "receiving" event, in the EPC process model can be mapped to a "catching" intermediate message event (shown in Figure 5 - A). Events which have no incoming control flow must be mapped to start message events. An end message event with the semantics of "catching" is not defined in BPMN, because somehow the message should be handled. If nevertheless a "receiving" event occurs of the end of an EPC process model, an intermediate message event followed by a normal end event should be generated as shown in Figure 6 - C.

"Receiving" events which have a preceding xor-connector must be paid more attention. In BPMN there is the distinguishing between data-based exclusive gateways and event-based exclusive gateways. As already discussed in Section 3, data-based means that this gateway checks the condition expression of the following sequence flows by using the incoming data. For event-based exclusive gateways this sequence flow will be chosen where the first following event occurs. These events must be an intermediate message, timer, or signal event. That means, if after a xor-connector more than one "receiving" event are following, this xor-connector can be mapped to an event-based exclusive gateway and not to a data-based exclusive gateway (see Figure 6 - B). But be careful, because every followed event should be a "receiving" event, if not, it should handle this exception as the following way:

Map every event which comes up directly after the event-based xor-connector to a "receiving" event, independent if this is a found "receiving" event.

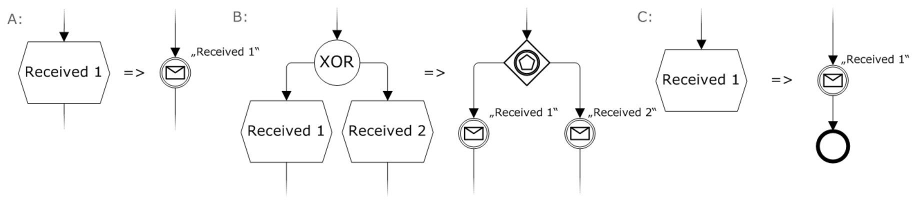

Fig. 6. Semantics Rule 2

6 Transformation Example

In the previous sections I define the transformation from EPC to BPMN. In this section an example is shown which should clarify all open issues and should show the importance of all transformation rules.

In Figure 7, a sales process is shown which is originally from the paper "Formalization and Verification of Event-driven Process Chains" from W.M.P. van der Aalst [8]. It is enlarged with the process elements of extended EPC and reduced of some functions and events which are not relevant for this example. The transformation shows all aspects of the given transformation including the extension of the transformation. For the mapping following keywords are defined: 'receive' for receiving a message and 'send' for sending a message. With this you see, that e.g. the start event of the process is a message start event, because of the title from the event "Customer order received". Same with "Send bill" and "Bill sent", using the keyword 'send' both are mapped to a "throwing" message event. Futhermore, organizations are mapped to lanes, e.g. for the organization "Sales" a lane with the title "Sales" is create where all functions and following constructs are added. Systems are mapped to annotations, shown in the example of an "ERP-System". Information, like "Customer data", is mapped to a data object. And, a process link, demonstrated in "Produce articles", is transformed to a sub-process in the BPMN process model.

7 Conclusion

In this paper a transformation is given from EPC [6] to BPMN 1.1 [1]. First I discuss different mapping approaches, subsequently I define the transformation on the basis of seven rules. After that, I introduce and discuss some extensions and semantics rules; and in the Section 6, I show an example of the transformation.

As already discussed not every construct in EPC can directly be mapped to BPMN, because of the different semantics and formalization. There is e.g. the event in EPC which is defined as a system state in the process, but in BPMN there is no construct like this. The most similar construct in BPMN would be the event as well, but only for some special defined event in EPC. In Section 4 there are these special events shown using the structural pattern and in Section 5 there is discussed which events can occure, when you look on the semantics.

Another construct, which can not be mapped directly is the System in EPC. Simply in BPMN there is nothing familar to that, so the best way for those constructs is it to map this to an Annotation in BPMN.

As a proof of concept, this transformation is implemented as a plugin in the Oryx-Editor, an open source web-based editor for business process modeling (available under http://www.oryx-editor.org). This implementation can give an idea, how this transformation can be implemented and realized. In Oryx available EPC processes can be imported and transformed to a BPMN process model. If needed, an already existent auto-layouter will reposition the process elements. Especially when transforming the organizations to lanes the contained elements

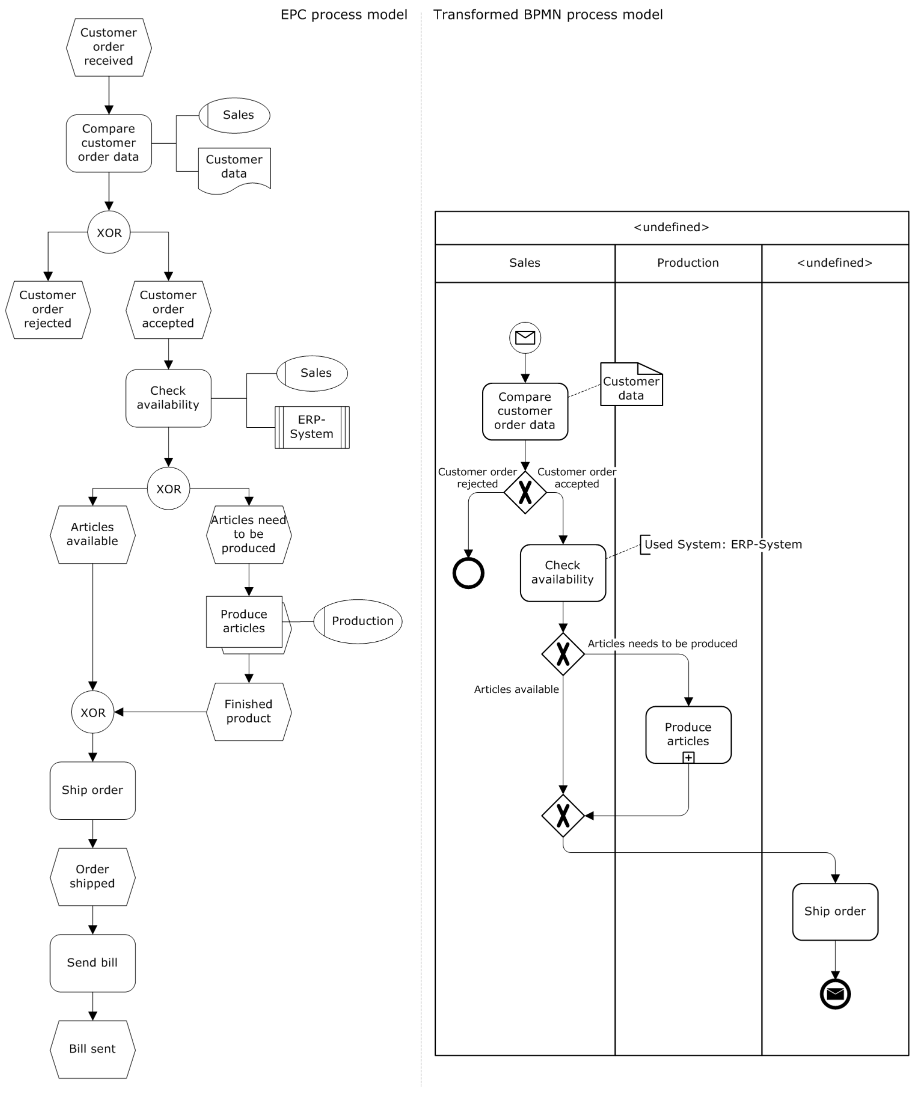

Fig. 7. Example of Transformation

have to be positioned correctly in the lanes. In addition, keywords for "sending" and "receiving" events can also be specified, where the message events get founded and mapped.

Next steps will be to test and validate this approach with real life processes in the industry and to enhance the implementation in Oryx.

References

1. Business Process Modeling Notation, V1.1. Technical report, Object Management Group (OMG), January 2008. http://www.omg.org/spec/BPMN/1.1/PDF.
2. A. Arkin et al. Business Process Modeling Language (BPML), Version 1.0. *BPML.org*, 2002.
3. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Franois Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition), W3C Candidate Recommendation. Technical report, February 2004. http://www.w3.org/TR/REC-xml.
4. J. Clark et al. XSL Transformations (XSLT) Version 1.0. *W3C Recommendation*, 16(11), 1999. http://www.w3.org/TR/xslt.
5. R.M. Dijkman, M. Dumas, and C. Ouyang. Formal Semantics and Analysis of BPMN Process Models using Petri Nets? Technical report, Technical Report 7115, Queensland University of Technology, Brisbane, 2007.
6. G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)". *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, 89, 1992.
7. J. Mendling and M. Nüttgens. EPC markup language (EPML): an XML-based interchange format for event-driven process chains (EPC). *Information Systems and E-Business Management*, 4(3):245–263, 2006.
8. WMP van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639–650, 1999.
9. D. Vanderhaeghen, S. Zang, A. Hofer, and O. Adam. XML-based Transformation of Business Process Models–Enabler for Collaborative Business Process Management. *XML4BPM 2005*.