February 2007

Integration of Conceptual Process Models by the Example of Event-driven Process Chains

Carlo Simon
Universität Koblenz-Landau, simon@uni-koblenz.de

Jan Mendling
Wirtschaftsuniversität Wien, jan.mendling@wu-wien.ac.at

In: Oberweis, Andreas, u.a. (Hg.) 2007. *eOrganisation: Service-, Prozess-, Market-Engineering*; 8. Internationale Tagung Wirtschaftsinformatik 2007. Karlsruhe: Universitätsverlag Karlsruhe

ISBN: 978-3-86644-094-4 (Band 1)
ISBN: 978-3-86644-095-1 (Band 2)
ISBN: 978-3-86644-093-7 (set)

Integration of Conceptual Process Models

by the Example of Event-driven Process Chains

Carlo Simon

Institut für Management
Universität Koblenz-Landau
56016 Koblenz, Deutschland
simon@uni-koblenz.de

Jan Mendling

Institut für Wirtschaftsinformatik und Neue Medien
Wirtschaftsuniversität Wien
1090 Wien, Österreich
jan.mendling@wu-wien.ac.at

Abstract

It has become common place in business life that companies with related operations engage in a so-called merger in order to benefit from synergies or from combined products and services. In order to handle the complexity of such an endeavour, it is important to utilise a structured approach for finding similarities and contradictions in business process models of both partners. In this paper, we present a suitable procedure for this task. Furthermore, we demonstrate how to identify those specific activities within the overall business processes which must be adapted. In particular, we discuss how such integration can be conducted if the processes of both parties are modelled with Event-driven Process Chains, one of the most popular conceptual business process modelling languages. By the help of a running example we illustrate the join operator for the integration of these models and the interpretation of the result.

1 Introduction

It has become common place in business life that companies with related operations engage in a so-called merger in order to benefit from synergies or from combined products and services. The rationale behind such a merger is that the combination of both companies' operations is expected to result in lower total cost as for the sum of both and a wider range of capabilities. It is a prerequisite for the leveraging of these benefits that the operational infrastructure of the business processes of the merging partners is integrated. Since the complexity of such an endeavour is a considerable challenge, the enterprise model repositories of both companies play an important role to guide and structure this integration procedure.

Several perspectives have been proposed to document information systems of enterprises. The control flow perspective of the business process is the most important one mentioned e.g. by [ÖBH91, p. 173, Sch00, p. 41]. Other views are organisation, function and data in accordance with the St. Galler information system's architecture of Österle et al. [ÖBH91, p. 173]. Scheer extends this model by output and control within the architecture of integrated information systems (ARIS) [Sch00, p. 41]. Axenath et al. [AKR05, p. 6] add authorisation and authentication as well as assignment as further perspectives. These rather technical views on businesses may be extended by strategies as proposed by Frank [Fra94, p. 170] or Krcmar [Krc05, p. 43].

With respect to the integration of information models in case of a merger, there is extensive work reported in the database community on *view* and *schema integration of data models*. In the 1980s, Batini et al. [BLN86] provide a comparative analysis of schema integration methodologies. They distinguish *preintegration*, *comparing*, *conforming*, *merging*, and *restructuring* as schema integration activities. Several contributions focus on specific activities of this integration process. Rahm and Bernstein provide a survey on how matches across different schemas can be identified automatically [RB01]. Rizopoulos and McBrien discuss a hypergraph data model (HDM) with a set of semantic relationships to support the merge operation [RM05]. A comprehensive integration method is provided by [SS05].

While the integration of data models is a rather mature research discipline, surprisingly little work has been conducted on the integration of process models in theory and practice. Most of these contributions offer integration procedures on the level of Petri nets with only a few like [GRSS05a, GRSS05b] covering generic aspects. For this paper, we adopt the integration process for a conceptual process modelling language as presented in [MS06] since it provides a straight forward support for Event-driven Process Chains (EPCs). For further related work on

behaviour integration, we also refer to [MS06]. The approach which is presented in the following builds on general insights from database schema integration and integration operators that borrow ideas from the Semantic Process Language [Sim06]. Moreover, it focuses on the control flow aspect of business process models. The remainder of the paper is structured as follows. Section 2 gives a theoretical foundation of the process integration methodology applied in this paper. Section 3 provides a formal syntax definition of EPCs and introduces a running example of two procurement EPCs. Section 4 applies the integration process as defined in Section 2 to the two example EPCs. The paper closes with a conclusion in Section 5.

2 The Process Model Integration Process

The integration of two business process models needs to consider those parts of the processes that coincide or contradict each other. This bears the following challenges:

- If the business processes are described using different modelling languages, then a translation of these models into a unique representation must be made first. For this reason *a meta-model* or *concept-ontology* must be defined for each language, i.e. the source languages and the target language which, however, do not necessarily have to be different. Beside the syntactical concepts also semantics must be described and mapped to each other.

- A *domain ontology* is needed to identify those actions which can be used for the synchronisation of the business process models. Therefore, the domain ontology must describe the language of the various domain experts in order to match their views of the modeled field of application.

- A formal *integration operator* must be defined to join two process models into a single model which can then be used to advice mergers concerning their mutual adoption of business processes. For conceptual process models, only heuristics can be formulated for this task, since without a state semantic a formal equivalence relationship cannot be defined.

The purpose of an ontology for the development of information systems is to describe human (domain specific) language in such a way that it can be represented and processed by software [EGHS05, p. 204]. For the particular modelling problem discussed here we need such a formali-

sation of the domain which justifies each merge operation on event and function level. A possible representation of ontologies ranges from *unstructured* over *semi-structured* to *fully structured* [GFC04, p. 169]. An unstructured ontology is described in natural language, a semi-structured ontology in natural language which is restricted to a specific form, and a structured one is defined with the aid of a formal language which supports proofs concerning soundness and completeness.

In this paper, we use a semi-structured approach to define the domain ontologies of the examples focusing on two aspects: first, the purpose of each process function is described by natural language. Second, the relation to other functions within the process is taken down. While formal integration of (business) processes has been reported by Simon [Sim06] for a Semantic Process Language (SPL), for Petri nets in general by Best et al. [BDK01], and for Process Algebra [BPS01, Fok00], we define and apply a process model integration process that is especially tailored for conceptual business process modelling languages such as EPCs. We specify the merge operator on the level of the formal syntax of EPCs. In this way, we extend previous work on the integration of Petri nets (see e.g. [PCS01, Sim06]).

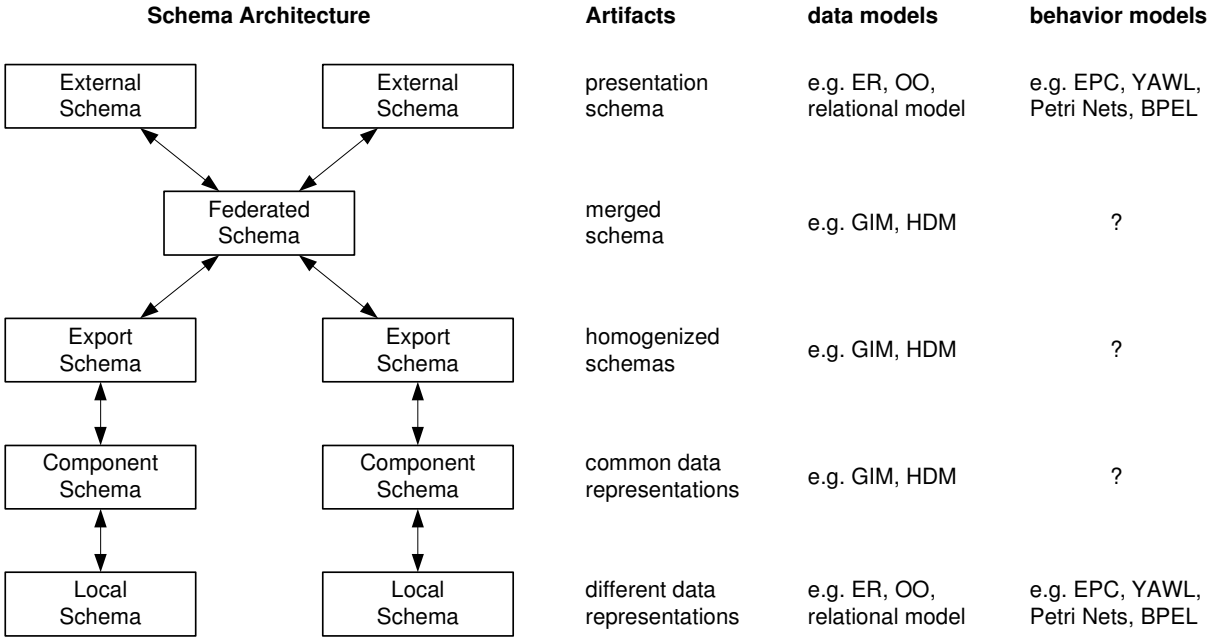

Fig. 1: Database Schema Integration and Process Model Integration [MPZ05]

In essence, the process model integration process can be specified analogously to the database schema integration process as defined in [SL90, SS05], compare Figure 1. While explaining the integration steps of the process model integration process, we give the terms from database integration according to [SL90] in brackets to establish the link. The process takes two process

models as input (“local schema”). These models might comply with two different business process modelling languages, e.g. one EPC and one BPEL model. In a first step, the models have to be mapped to a common business process modelling language that is utilised throughout the further integration. As a result, this yields the two models in the same language (“component schema”). In a second step, the elements of the process models have to be matched. In particular, potential homonyms and synonyms have to be analysed with special attention. The resulting models are called “export schemas” according to [SL90]. The third step represents the application of the merge operator and we achieve an integrated process model (called “federated schema” in [SL90]). Depending on user requirements, the integrated process model could be mapped to another process modelling language for presentation purposes (“external schema”).
In this paper, we consider two business process models that are both modelled as EPCs. Therefore, we do not have to apply the mapping to a common process modelling language. Furthermore, we have to select the right variant of the merge operator: if the process models capture different views on the same business process, the integration is a kind of conjunction of the models (“integration by specialization” in [PCS01]). If the models represent process variants, the integration is a kind of disjunction (“integration by generalization” in [PCS01]). For a conjunction based integration of EPC business process models and further related work, refer to [MS06].

3 Event-driven Process Chains (EPCs)

Event-driven Process Chains (EPCs) are a conceptual business process modelling language. EPCs capture the temporal and logical dependencies of activities of a business process [KNS92]. So called function type elements represent activities of a business process. Event type elements describe pre- and post-conditions of functions, and three kinds of connector types including AND, OR, and XOR. Control flow arcs are used to link these elements. Connectors have either multiple incoming and one outgoing arc (join connectors) or one incoming and multiple outgoing arcs (split connectors). As a syntax rule, functions and events have to alternate, either directly or indirectly when they are linked via one or more connectors. The syntax of EPCs can be formally defined as follows (cf. [MS06]):

Notation 1 (Predecessor and Successor Nodes) Let N be a set of nodes and $A \subseteq N \times N$ a binary relation over N defining the arcs. For each node $n \in N$, we define the set of *predecessor nodes* $\bullet n = \{x \in N \mid (x, n) \in A\}$, and the set of *successor nodes* $n\bullet = \{x \in N \mid (n, x) \in A\}$.

Definition 1 (EPC) An EPC = (E, F, C, l, A) consists of three pair wise disjoint sets E, F, C of nodes, a mapping $l: C \rightarrow \{AND, OR, XOR\}$, and a binary relation $A \subseteq (E \cup F \cup C) \times (E \cup F \cup C)$ such that

- $|\bullet e| \leq 1$ and $|e\bullet| \leq 1$ for each $e \in E$. An element of E is called *event*.
- $|\bullet f| = 1$ and $|f\bullet| = 1$ for each $f \in F$. An element of F is called *function*.
- Either $|\bullet c| = 1$ and $|c\bullet| > 1$ or $|\bullet c| > 1$ and $|c\bullet| = 1$ for each $c \in C$. An element of C is called *connector*.
- The mapping l specifies the *type of a connector* $c \in C$ as AND, OR, or XOR.
- A defines the control flow as a coherent, directed graph. An element of A is called *arc*.

We define the semantics of the different EPC connector types in an informal manner since we only consider the structure of an EPC in the integration process. For an overview of formalisation approaches refer to [Ki06]. An AND split activates all subsequent branches in concurrency while the XOR split activates one of the subsequent branches. The OR split triggers at least one and at most all of multiple branches. The AND join synchronises all incoming branches, then it activates the subsequent EPC element. The OR join sychronises all active branches. This feature has been debated as non-local semantics (see e.g. [Ki06]). The XOR split has also non-local semantics. Either there is one input branch active (which is the expected case) and it activates the subsequent EPC element, or there are multiple branches active and it blocks.

In the remainder of this section, we discuss two example EPCs that both represent a standard procurement process. The labels of the second EPC were translated from German. The first procurement process considered here is taken from [Roh96, p.49]. The start event represents the readiness to calculate procurement requirements on base of a master requirements plan and inventory data. The corresponding calculations are repeated until all open items are handled. Afterwards, these calculated net requirement orders are taken to generate and release an order and the actual procurement is conducted. Figure 2 shows the EPC model of this procurement process.

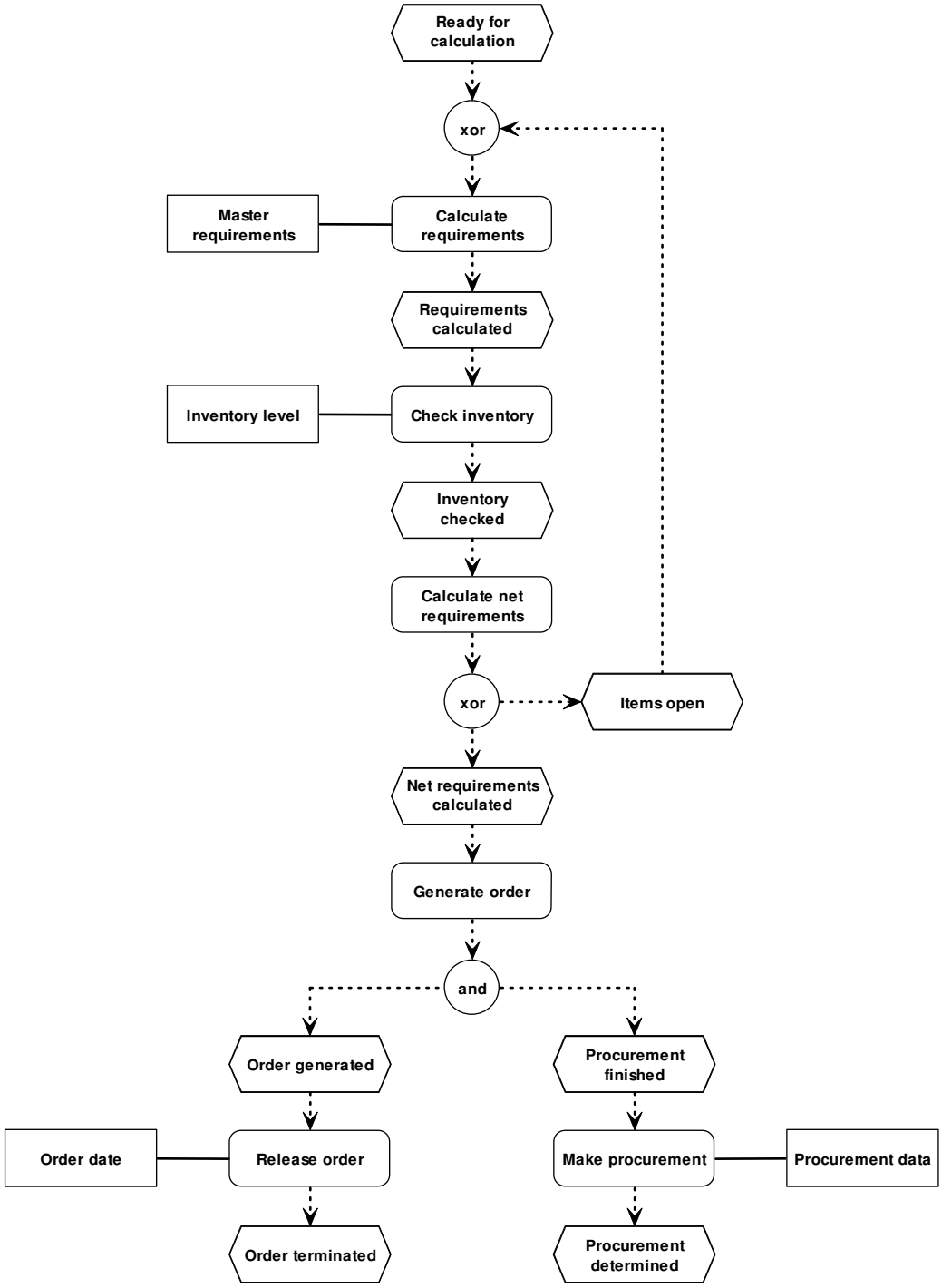

Fig. 2: Procurement process in adoption of [Roh96, p. 49]

The second procurement process is taken from [BS96, p. 65]. A fund manager starts a procurement process on behalf of a recognised demand with stocktaking of the requested products in the warehouse or in sourced out third-party warehouses. On base of the currently available amount of goods the purchase order quantity is calculated and an order is initiated which increases the warehouses. Figure 3 depicts this business process in a diagram.

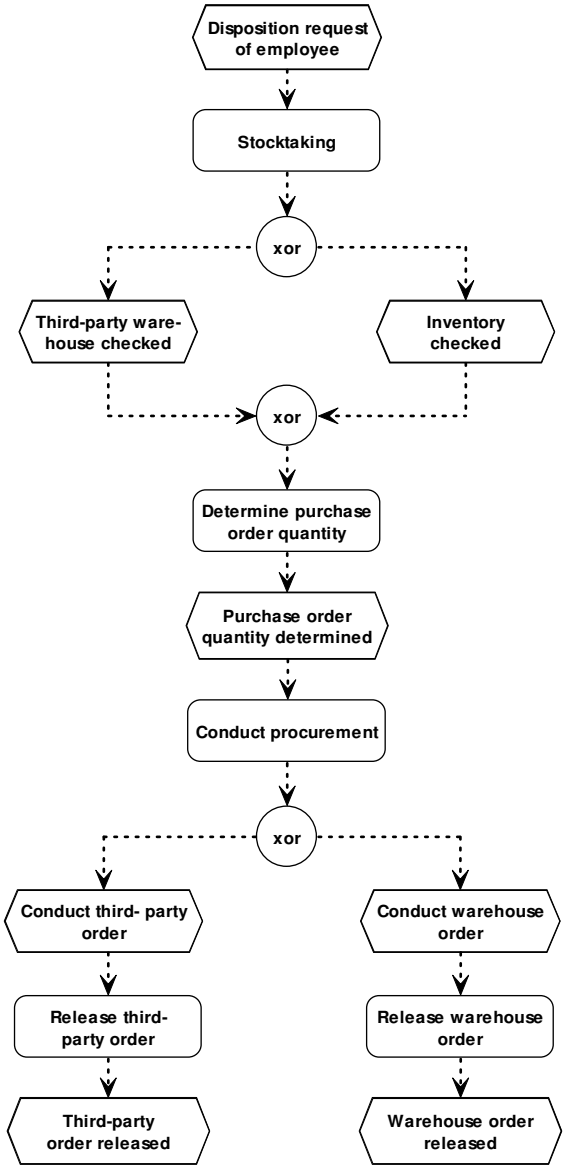

Fig. 3: Procurement process in adoption of [BS96, p. 65]

4 Merging EPCs based on Process Model Integration

Since both models are developed as EPCs, a concept ontology is not needed for an integration if this operation can be applied immediately to the EPC models. In the subsequent section the theoretical foundations will be laid for this and the integration operator is applied to the examples.

Domain ontologies for the examples must explain all used functions and events. As an input for their specification, we utilise both the (narrative) description of the functions in the EPC models

and their sequence relation to other functions. These facts provide adequate information to draw conclusions concerning the similarity of functions in different models. The events of the models play a subordinate role within these examples since they primarily describe intrinsic states of the process progress which can also be derived from the functions and the sequence in which they occur.

Table 1 shows the domain ontology for the model of Rohloff in tabular form explaining the functions and relates them to their predecessor and successor functions.

Action	**Predecessor**	**Successor**	**Description**
Calculate requirements		*Check inventory* or *Calculate net requirements*	Determine the need
Check inventory	*Calculate requirements*	*Calculate net requirements*	Check current stock amount
Calculate net requirements	*Check inventory*	*Release order* and *Make procurement*	Calculate procure-ment amount
Generate order	*Calculate net requirements*		Formulate request
Release orders	*Generate order*		Distribute order
Make procurement	*Generate order*		Control procurement

Tab. 1: Domain ontology for the functions in the model of Rohloff

Tabular 2 shows the domain ontology of Becker and Schütte in a similar format. Both representations are used to compare the domain ontologies and to find similarities.

The first function in the model of Rohloff is the calculation of requirements while in the model of Becker and Schütte the process starts with stocktaking followed by determining the purchase order quantity. This indicates that the required amount of goods must have been calculated before a process described by Becker and Schütte starts. And indeed, processes of Becker and Schütte start with an event *Disposition request of employee* which indicates that the demand has already been calculated. It is therefore not possible to identify an analogue function for *Calculate requirements* in the second model.

Action	Predecessor	Successor	Description
Stocktaking		*Determine purchase order quantity*	Determine current inventory
Determine purchase order quantity	*Stocktaking*	*Conduct procurement*	Determine required quantities
Conduct procurement	*Determine purchase order quantity*	*Release third-party order* or *Release warehouse order*	Generate order
Release third-party order	*Conduct procurement*		Order for external warehouse
Release warehouse order	*Conduct procurement*		Order for internal warehouse

Tab. 2: Domain ontology for the functions in the model of Becker and Schütte

The next function in the model of Rohloff is *Check inventory* which can be seen as similar to *Stocktaking* due to their descriptions. A difference between these two functions can only be observed, if the following events are considered. Becker and Schütte explicitly distinguish between internal and external warehouses while such a differentiation cannot be identified in the description of Rohloff. Despite of this distinction, both functions in principle describe the same kind of activity.

The next following function in the model of Rohloff is *Calculate net requirements* and in the model of Becker and Schütte *Determine purchase order quantities*. Both functions can be seen as similar due to their description. For the same reasons, also *Generate order* and *Conduct procurement* are diagnosed as similar.

At the end of the procurement process of Rohloff, two functions occur (*Release order* and *Make procurement*) while the process of Becker and Schütte ends with an alternative (between *Release third-party order* and *Release warehouse order*). This differentiation results from the two different warehouses (internal and external) and does not occur in the model of Rohloff. It can therefore be concluded that these two functions are specialisations of *Release order*.

5 Integration of the EPC Models

In this section, a heuristic is developed for the join of EPC models such that the resulting model represents the intersection of the processes of the input models. Due to the absence of a formal state semantics, it cannot be proved that this is formally true like in the Semantic Process Language. Nonetheless, we provide evidence for the usefulness of the chosen definition by explaining the outcome of each integration step. We then apply the heuristic to our example.
Since it is the goal of a merger to integrate the former individual views into a single one, the following join operator is conceptualised as a conjunction of models. The resulting EPC in principle describes the intersection of the input schemas.

Definition 2 (Joined EPC) Let $EPC_1 = (E_1, F_1, C_1, l_1, A_1)$ and $EPC_2 = (E_2, F_2, C_2, l_2, A_2)$ be two EPCs. The *Joined Event-driven Process Chain* $EPC_J = (E_J, F_J, C_J, l_J, A_J)$ – the conjunction of EPC_1 and EPC_2 – is defined in three consecutive steps as follows:

1. Basically, the elements of EPC_1 and EPC_2 are combined in a single diagram:
   $E_J'' := E_1 \cup E_2$ and $F_J'' := F_1 \cup F_2$,
   $C_J'' := C_1 \cup C_2$ and $l_J'' := l_1 \cup l_2$,
   $A_J'' := A_1 \cup A_2$
2. Each pair (e_1, e_2) of similar event elements $e_1 \in E_1$ and $e_2 \in E_2$ describing the same real-world events is fused into a single one. Former incoming and outgoing control flow arcs are synchronised with the aid of two new connectors c_{split} and c_{join}:
   $E_J' := E_J'' - \{e_2\}$ and $F_J' := F_J''$
   $C_J' := C_J'' \cup \{c_{split}, c_{join}\}$
   $l_J' := l_J'' \cup \{(c_{split}, \text{and}), (c_{join}, \text{and})\}$ and
   $\forall x_1 \in {}^\bullet e_1, \forall x_2 \in {}^\bullet e_2, \forall y_1 \in {}^\bullet e_1, \forall y_2 \in e_2{}^\bullet$:
   $A_J' := A_J'' - \{(x1, e_1), (x_2, e_2), (e_1, y_1), (e_2, y_2)\} \cup$
   $\{(x_1, c_{join}), (x_2, c_{join}), (c_{join}, e_1), (e_1, c_{split}), (c_{split}, y_1), (c_{split}, y_2)\}$
   Incomplete tuples due to missing predecessor or successor nodes are omitted.
   The result of this operation is that each merged event occurs after all its successor functions and that it triggers all subsequent functions.

3. Each pair (f_1, f_2) of similar function elements $f_1 \in F_1$ and $f_2 \in F_2$ describing the same real-world events is fused into a single one. Former incoming and outgoing control flow arcs are synchronised with the aid of two new connectors c_{split} and c_{join}:

   $E_J := E_J'$ and $F_J := F_J' - \{f_2\}$

   $C_J := C_J' \cup \{c_{split}, c_{join}\}$

   $l_J := l_J' \cup \{(c_{split}, and), (c_{join}, and)\}$ and

   $\forall x_1 \in \bullet f_1, \forall x_2 \in \bullet f_2, \forall y_1 \in \bullet f_1, \forall y_2 \in f_2\bullet$:

   $A_J := A_J' - \{(x1, f_1), (x_2, f_2), (f_1, y_1), (f_2, y_2)\} \cup$

   $\{(x_1, c_{join}), (x_2, c_{join}), (c_{join}, f_1), (f_1, c_{split}), (c_{split}, y_1), (c_{split}, y_2)\}$

   Incomplete tuples due to missing predecessor or successor nodes are omitted.

   The result of this operation is that each merged function is only executed if all its successor events have occurred and that it activates all possible follower events.

The first step results in a single EPC model which contains both input EPCs without any connections between them. The second step joins events. Most of the functions have similar events according to the fact that they only describe internal states of the processes. Similarity can be assumed between those events for which a similarity has be identified concerning their adjacent functions (*Requirements calculated* and *Disposition request of employee* as well as for *Net requirements calculated* and *Purchase order quantity determined*). The joined events are labelled using the terminology of Rohloff. Figure 4 exemplarily depicts the join of *Net requirements calculated* and *Purchase order quantity determined.*

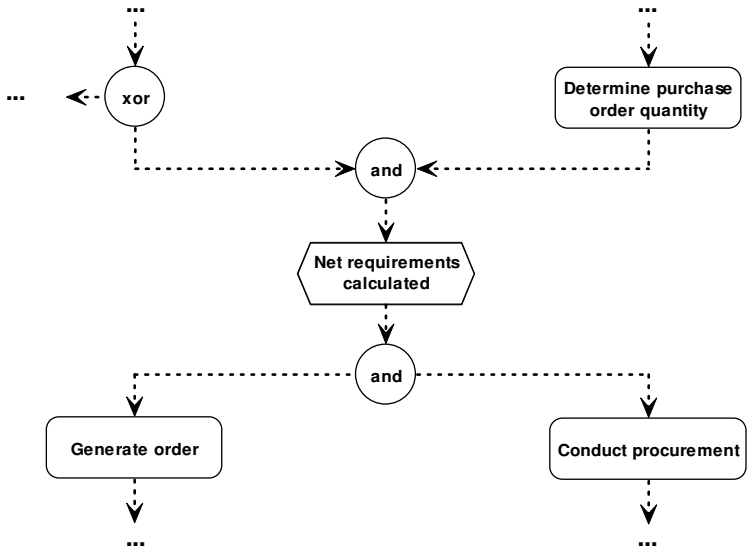

Fig. 4: Result of joining two events

Moreover, *Inventory checked* of Rohloff can be seen as a generalisation of *Third-party warehouse checked* and *Inventory checked* of Becker/Schütte. These events may be joined as well and the resulting events are labelled using the more specific terminology of Becker/Schütte. Definition 2, however, only provides means to join one event of the first EPC with one event of the other. If more than one join makes sense (due to a generalisation/specialisation relationship) each of the specialised joined functions must be enabled with respect to the preceding functions in the more general model i.e., further OR-connectors have to be added. The corresponding formal graph transformation operation can be specified similar to the previously defined operations.

The join of functions uses the ontologies defined in the previous section. Although *Check inventory* and *Stocktaking* are in principle identical, the latter one takes into account the existence of different warehouses which means that this term is a specialisation of *Check inventory*. The joined function is therefore labelled *Stocktaking*. Concerning the in each model following two functions, the join result is labelled using the terminology of Rohloff.

Also concerning the functions a generalisation/specialisation relationship can be observed. The corresponding functions can be merged with the same operation described for events already, i.e. by adding additional OR-connectors.

In the joined model, we observe an inflation of connector symbols. Many of them can be omitted since they are redundant or because they only have one incoming and outgoing arc. Figure 5 shows the resulting Joined EPC of the input schemas.

The resulting EPC model can be interpreted as follows: in the joined model, we still find valid processes. Consequently, a merge is possible. Nonetheless, also deadlocks can be observed at the end of the process. These deadlocks result from the differentiation of warehouse types made in one model but not in the other. A deadlock, however, does not represent a weakness of the presented approach. In contrary, such contradictions are typical for mergers. They help to find those parts of businesses which need to be restructured and where operations have to be aligned.

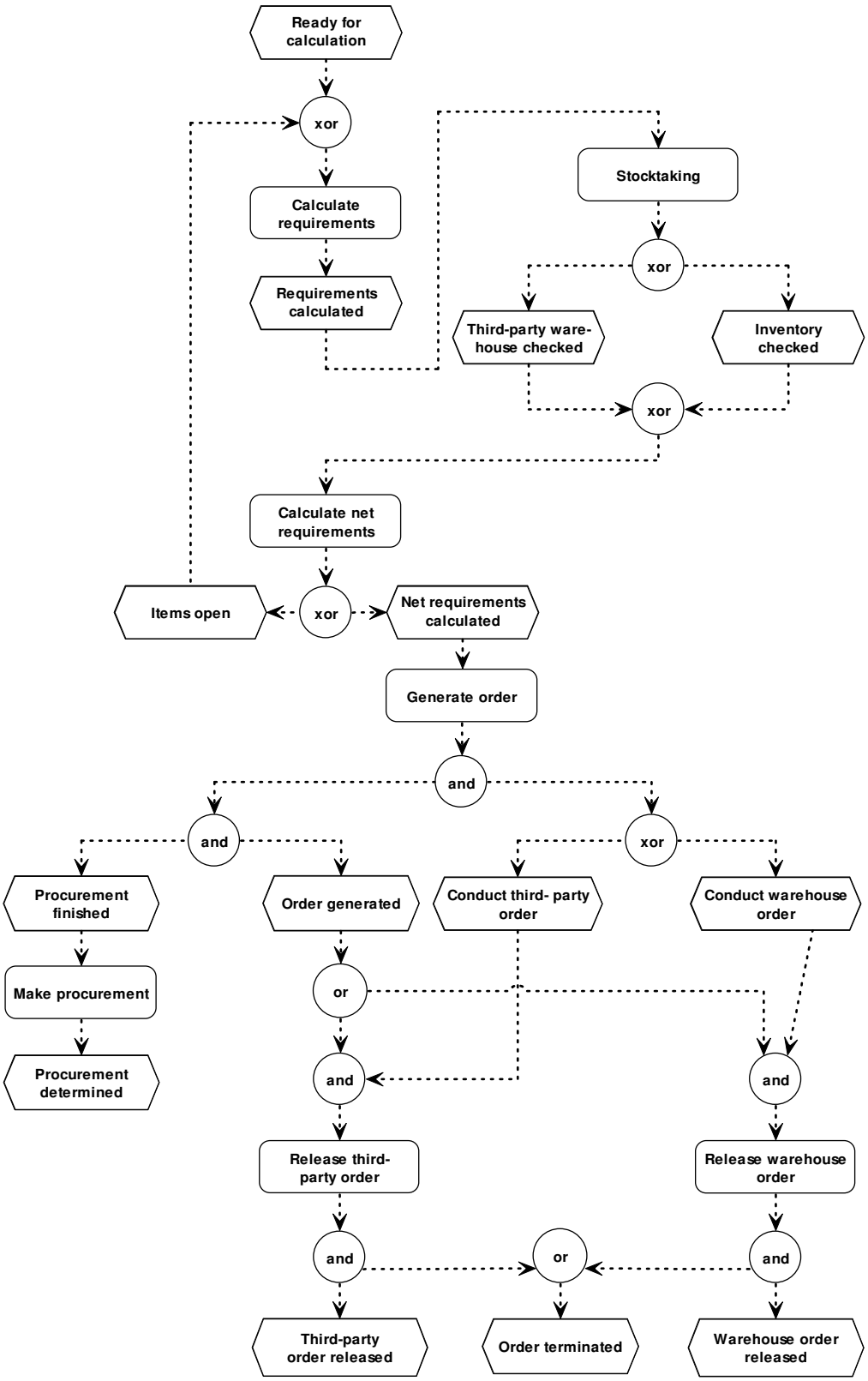

Fig. 5: Joined EPC of Rohloff and Becker/Schütte

6 Conclusion

Support for the conceptual integration of business process models is a key requirement for combining business processes in a merger scenario. Within this paper, we introduce a novel integration operator for conceptual process models (in particular EPCs). We follow an integration process which is adapted from database integration. Within this integration process, a merge operator specific for conceptual business process models is used to combine the models. In order to demonstrate the feasibility of our integration approach, we integrate two reference procurement processes taken from literature. Moreover, the actual merge ontologies are formulated building on the structure of the input EPCs.

The resulting EPC demonstrates that – although the input EPCs have some similarities – in this merger scenario a partial restructuring of the processes would be necessary. The method guides to these critical subsequences and detects those parts of the process models that differ. Since the method is applied immediately to the EPC input models, in opposite to former work [SM06] which required an intermediary transformation into Petri nets, the presented approach takes both into account, functions and events.

The integration operator that we use results in a conjunction of the input schemas. From database integration, also scenarios are known where the result equals a disjunction. The usefulness of a transfer of this integration type to process models will be considered in future research. Furthermore, we aim to provide tool support for the integration process. Beyond that, the process view is only one of many perspectives on information systems of a company. Future work will have to combine the data and the process integration into a multi-perspective integration approach.

References

[Aal05] Aalst, van der, W. M. P.: Pi calculus versus Petri nets: Let us eat “humble pie” rather than further inflate the “Pi hype”. BPTrends, 3(5), 2005, pp. 1-11

[AKR05] Axenath, B.; Kindler, E.; Rubin, V.: The Aspects of Business Processes: An Open and Formalism Independent Ontology. Fachberichte Informatik tr-ri-05-256, Universität Paderborn.

[BDK01] Best, E.; Devillers, R.; Koutny, M.: Petri net Algebra. In: Brauer, W.; Rozenberg, G.; Salomaa, A. (Edt.): EATCS, Monographs in Theoretical Computer Science, Springer, Berlin, 2001

[BLN86] Batini, C.; Lenzerine, M.; Navathe, S. B.: A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys, 18(4), 1986, pp. 323-364

[BPS01] Bergstra, J. A.; Ponse, A.; Smolka, A. (Edt.): Handbook of Process Algebra, Elsevier, Amsterdam, 2001

[BS96] Becker, J.; Schütte, R.: Handelsinformationssysteme, Verlag Moderne Industrie, Landsberg/Lech, 1996

[EGHS05] Elzenheimer, M.; Grollius, T.; Heinemann, E.; Sternhuber, J.: Vergleichende Buchbesprechung: Ontologien. Wirtschaftsinformatik, 47(4), 2005, pp. 298-309

[Fok00] Fokkink, W.: Introduction to Process Algebra. Springer, Berlin, 2000

[Fra94] Frank, U.: Multiperspektivische Unternehmensmodellierung. Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung. Oldenbourg Verlag, München, 1994

[GFC04] Gómez-Pérez, A.; Fernández-López, M.; Corcho, O.: Ontological Engineering. Springer, Berlin, 2004

[GRSS05a] Georg Grossmann, Yikai Ren, Michael Schrefl, and Markus Stumptner. Behavior Based Integration of Composite Business Processes. In 3rd International Conference on Business Process Management, BPM 2005, September 5-8 2005. LNCS(3649/2005), pages 186-204, Springer-Verlag.

[GRSS05b] Georg Grossmann, Yikai Ren, Michael Schrefl, and Markus Stumptner. Definition of Business Process Integration Operators for Generalization. In 7th International Conference on Enterprise Information Systems, ICEIS 2005, May 24-28 2005.

[Ki06] Kindler, E.: On the semantics of EPCs: Resolving the vicious circle. Data & Knowledge Engineering, Volume 56, Number 1, January 2006, pages 23-40

[KNS92] G. Keller, M. Nüttgens, and A. W. Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK)". Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1992

[Krc05] Krcmar, H.: Informationsmanagement (4. edt). Springer, Berlin, 2005

[MPZ05] J. Mendling, C. Pérez de Laborda, U. Zdun: Towards an Integrated BPM Schema: Control Flow Heterogeneity of PNML and BPEL4WS. In: K.-D. Althoff, A. Dengel, R. Bergmann, M. Nick, T. Roth-Berghofer, eds.: Post-Proceedings of (WM 2005), Lecture Notes in Artificial Intelligence 3782, Kaiserslautern, Germany, pages 570–579, 2005.

[MS06] Mendling, J.; Simon, C.: Business Process Design by View Integration. In: Dustdar, S.; Eder, J.: BPM 2006 Workshops Proceedings. Lecture Notes in Computer Science (LNCS), Springer 2006

[PCS01] Preuner, G.; Conrad, S.; Schrefl, M.: View integration of behavior in object-oriented databases. Data Knowl. Eng. 36(2), 2001, pp. 153-183

[ÖBH91] Österle, H.; Brenner, W.; Hilbers, K.: Unternehmensführung und Informationssysteme: Der Ansatz des St. Galler Informationssystem-Managements. Teubner, Stuttgart, 2001

[RB01] Rahm, E.; Bernstein, P. A.: A Survey of Approaches to Automatic Schema Matching. VLDB Journal, 10(4), 2001, pp. 334-350

[RM05] Rizopoulos, N.; McBrien, P.: A General Approach to the Generation of Conceptual Model Transformations. In: Pastor, O.; e Cunha, J. F. (Edt.): Proceedings: Advanced Information Systems Engineering. 17th International Conference CAiSE 2005, Lecture Notes in Computer Science (LNCS), 3520, Porto, Portugal, Springer 2005, pp. 326-341

[Roh9605] Rohloff, M.: Reference Model and Object Oriented Approach for Business Process Design and Workflow Management. In: Proceedings: Information Systems Conference of New Zealand, 1996, pp. 43-52

[Sch00] Scheer, A.-W.: ARIS – Business Process Modeling (3rd. edt). Springer, Berlin, 2000

[Sim06] Simon, C.: Integration of Planning and Production Processes. In: Mathmod 2006, Special Session Petrinets: Current Research Topics and their Application in Traffic Safety and Automation Engineering, Wien, Austria, 2006

[SL90] Sheth, A.P.; Larson, J.A.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Comput. Surv. 22(3), 1990, pp. 183-236

[SM06] Simon, C., Mendling, J.: Verification of Forbidden Behavior in EPCs. In: Mayr, H. C.; Breu, R.: Modellierung 2006. Lecture Notes in Informatics (LNI P-82), Innsbruck, Austria, 2006, pp. 233-242

[SS05] Schmitt, I.; Saake, G.: A Comprehensive Database Schema Integration Method Based on the Theory of Formal Concepts. Acta Informatica, 41(7-8), 2005, pp. 475-524