Business Process Design by View Integration

Jan Mendling[1] and Carlo Simon[2]

[1] Vienna University of Economics and Business Administration
Augasse 2-6, 1090 Vienna, Austria
jan.mendling@wu-wien.ac.at
[2] University of Koblenz-Landau, Germany
simon@uni-koblenz.de

Abstract. Even though the design of business processes most often has to consolidate the knowledge of several process stakeholders, this fact is utilized only to a limited extent by existing modeling methodologies. We address this shortcoming in this paper by building an analogy between database schema design by view integration on the one hand and process modeling on the other hand. In particular, we specify a method for business process design by view integration starting from two views of a process as input. We identify formal semantic relationships between elements of the two process views which are then used to calculate the integrated process model applying the merge operator. Finally, the integrated model is optimized using reduction rules. A case study with two EPC business process models from the SAP reference model demonstrates the applicability of our approach.

1 Introduction

Business process design and in particular the design of business process models that capture real-world process semantics is a difficult task. While work procedures that are executed by one person are easy to document, business processes often span several departments of a company and include several activities performed by different persons. This implies a considerable complexity of the design task and calls for a structured approach. In this paper, we build on insight from database design theory, in particular view integration. View integration is a classical technique for database schema design. The idea is to identify the different views on the data of every person that is supposed to work with the database. Each person is interviewed and her view is documented in a separate so-called input schema. Then, the matching parts of the input schemas are identified. Based on these matches, the integrated schema is derived as a merge of the input schemas.

For so far, this technique has attracted only little attention in the context of business process modeling, basically, due to two reasons. First, the conceptual difference of process models and data models hinders a direct application of database schema integration for process design. An approach is needed that is analogous to data schema integration, but which addresses the specific nature of business process models, i.e., control flow defined between activities. Second,

J. Eder, S. Dustdar et al. (Eds.): BPM 2006 Workshops, LNCS 4103, pp. 55–64, 2006.

dedicated techniques for behavior integration have been defined for Petri nets (cf. [12,16,17]), but not for conceptual languages such as EPCs. As EPCs are frequently used in process modeling practice (see e.g. the SAP reference model [4,7]) and EPCs offer OR-joins which cannot be mapped to Petri nets without loosing readability, there is a more general approach needed.

The contribution of this paper is threefold. First, we identify semantic relationships between activities of different business process models. Second, we define a merge operator for EPCs that takes as input two EPCs and semantic relationships between their activities to calculate an integrated EPC. Third, we propose a set of restructuring rules in order to arrive at an integrated EPC that does not include unnecessary structure. The availability of view integration techniques for conceptual business process modeling languages provides several advantages for business process design. If a business process designer conducts interviews with process stakeholders, she can document each view in an input EPC and use a merge operation to integrate them. This is less prone to errors and more time efficient than building an integrated model manually. Furthermore, this procedure provides traceability: changes to the input EPCs can be studied with respect to their impact on the resulting integrated process model. If all interviews would be directly documented in one process model, the individual views are lost. Beyond that, our approach can also support a merger scenario where business process models of two companies with overlapping semantics have to be integrated into one repository.

Following this line of argumentation, the remainder of this paper is structured as follows: in Section 2 we give a definition of EPCs and an overview of our integration approach. Furthermore, we introduce semantic relationships between activities of different business process models, we define the merge operator for EPCs, and we identify restructuring rules. In Section 3, we apply this integration technique to two EPC business process models from the SAP reference model. The example models have the same name and share several activities. Section 4 gives an overview of related research before Section 5 concludes the paper.

2 Preliminaries

In this section, we first introduce Event-driven Process Chain (EPC) as a business process modeling language (Sect. 2.1). The subsequent sections introduce the steps of our integration approach, i.e. definition of semantic relationships (Sect. 2.2), the merge operator (Sect. 2.3), and restructuring rules (Sect. 2.4).

2.1 Event-Driven Process Chains (EPCs)

Event-driven Process Chain (EPC) is a business process modeling language representing temporal and logical dependencies between activities of a process [6]. EPCs offer *function type* elements to capture activities of a process, *event type* elements which 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 arcs

(join connectors) or one incoming and multiple outgoing arcs (split connectors). As a syntax rule, functions and events alternate but may be separated by connectors. For more details on EPCs, we refer to [8,9]. Formally, the structure of EPC models is defined as follows:

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

Definition 1 (EPC). An $EPC = (E, F, C, l, A)$ is a directed graph consisting of three pairwise disjoint sets of nodes E called *events*, F called *functions*, and C called *connectors*, a mapping $l : C \to \{and, or, xor\}$ specifying the connectors' types, and a binary relation $A \subseteq (E \cup F \cup C) \times (E \cup F \cup C)$ of the directed arcs between these nodes defining the intended control flow of the EPC such that

- $|\bullet e| \leq 1$ and $|e\bullet| \leq 1$ for each $e \in E$.
- $|\bullet f| = 1$ and $|f\bullet| = 1$ for each $f \in F$.
- Either $|\bullet c| = 1$ and $|c\bullet| > 1$ or $|\bullet c| > 1$ and $|c\bullet| = 1$ for each $c \in C$.

Figure 1 illustrates this definition showing two EPCs. Both describe similar processes of how a customer inquiry about products is received, processed, and how a quotation is created from the inquiry. The left EPC is taken from the Project Management branch of the SAP reference model and it is called *Customer Inquiry and Quotation Processing*. The second process EPC stems from the Sales and Distribution branch and its name is *Customer Inquiry*. The processes share two events and one function indicated by equal names. In the following, we elaborate how these two process models can be integrated.

2.2 Semantic Relationships

In the following, we define two kinds of semantic relationships between functions and events of two distinct EPCs, namely equivalence and sequence.

Definition 2 (Equivalence). 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 and $Eq \subseteq (E \times E) \cup (F \times F)$ a binary relation.

- If $e_1 \in E_1$ and $e_2 \in E_2$ describe the same real-world events, we write $(e_1, e_2) \in Eq$.
- If $f_1 \in F_1$ and $f_2 \in F_2$ describe the same real-world functions, we write $(f_1, f_2) \in Eq$.

Definition 3 (Sequence). Let EPC_1 and EPC_2 be two EPCs and $Seq \subseteq (E \times F) \cup (F \times E)$ a binary relation.

- If $e_1 \in E_1$ is always followed by $f_2 \in F_2$, we write $(e_1, f_2) \in Seq$.
- If $f_1 \in F_1$ is always followed by $e_2 \in E_2$, we write $(f_1, e_2) \in Seq$.

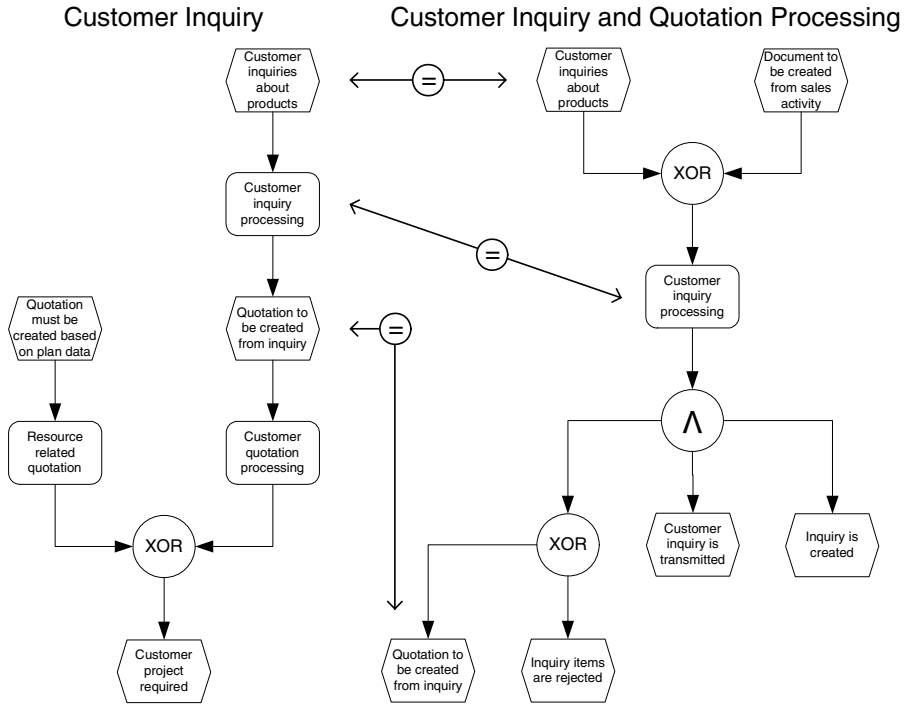

Fig. 1. *Customer Inquiry* and *Customer Inquiry and Quotation Processing* EPCs

WhenIf two views on the same business process have been documented as two EPC business process models, the business process designer has to identify semantic relationships in terms of equivalence and sequence between functions and events of the different models. Figure 1 might suggest that functions and events with similar labels are equivalent. Please note that EPC nodes can be also equivalent when syntactically different labels are used (analogous to synonyms) and that syntactically equivalent labels might relate to different business functions of another context (analogous to homonyms).

2.3 Merge Operator

The merge operator introduced in this section is novel. It takes two EPC views of the same business process plus a set of identified semantic relationships as input and produces an integrated EPC. As a first step, the integrated EPC includes all elements of the two input EPCs. Then, each pair of nodes (n_1, n_2) which describe the same real-world events or functions, i.e. $(n_1, n_2) \in Eq$, is merged into a single node and the former input and output arcs are joined and split with *and*-connectors, respectively. Finally, for each pair of nodes in the sequence relationship, an *and*-split is inserted after the predecessor node, followed by an arc to a new *and*-join before the successor node.

Definition 4 (Integrated EPC). Let EPC_1 and EPC_2 be two EPCs. The integrated EPC $EPC_i := (E_i, F_i, C_i, l_i, A_i)$ is defined in five consecutive steps as follows:

1. Basically, the elements of EPC_1 and EPC_2 are combined in a single diagram:

$$\begin{aligned} E_i &:= E_1 \cup E_2 \\ F_i &:= F_1 \cup F_2 \\ C_i &:= C_1 \cup C_2 \\ l_i &:= l_1 \cup l_2 \\ A_i &:= A_1 \cup A_2 \end{aligned}$$

2. Each pair $(e_1, e_2) \in Eq$ of event elements which describe the same real-world events is fused into a single event. The former incoming and outgoing control flow arcs are synchronized with the aid of two new connectors c_{split} and c_{join}:

$$\begin{aligned} E_i &:= E_i \setminus \{e_2\} \\ C_i &:= C_i \cup \{c_{split}, c_{join}\} \\ l_i &:= l_i \cup \{(c_{split}, and), (c_{join}, and)\} \\ A_i &:= A_i \setminus \{(x_1, e_1), (x_2, e_2), (e_1, y_1), (e_2, y_2)\} \cup \\ &\quad \{(x_1, c_{join}), (x_2, c_{join}), (c_{join}, e_1), (e_1, c_{split}), (c_{split}, y_1), (c_{split}, y_2)\} \end{aligned}$$

3. For each $(f_1, f_2) \in Eq$ of function elements which describe the same real-world functions is fused into a single event. The former incoming and outgoing control flow arcs are synchronized with the aid of two new connectors c_{split} and c_{join}:

$$\begin{aligned} F_i &:= F_i \setminus \{f_2\} \\ C_i &:= C_i \cup \{c_{split}, c_{join}\} \\ l_i &:= l_i \cup \{(c_{split}, and), (c_{join}, and)\} \\ A_i &:= A_i \setminus \{(x_1, f_1), (x_2, f_2), (f_1, y_1), (f_2, y_2)\} \cup \\ &\quad \{(x_1, c_{join}), (x_2, c_{join}), (c_{join}, f_1), (f_1, c_{split}), (c_{split}, y_1), (c_{split}, y_2)\} \end{aligned}$$

4. For each $(e_1, f_2) \in Seq$ of an event that is always followed by a function, two new connectors c_{split} and c_{join} are added and the arc from the new split after the event to the new join before the function makes the control flow explicit:

$$\begin{aligned} C_i &:= C_i \cup \{c_{split}, c_{join}\} \\ l_i &:= l_i \cup \{(c_{split}, and), (c_{join}, and)\} \\ A_i &:= A_i \setminus \{(e_1, y_1), (x_2, f_2)\} \cup \\ &\quad \{(e_1, c_{split}), (c_{split}, y_1), (c_{split}, c_{join}), (c_{join}, f_2), (x_2, c_{join})\} \end{aligned}$$

5. For each $(f_1, e_2) \in Seq$ of a function that is always followed by an event, two new connectors c_{split} and c_{join} are added and the arc from the new split

after the function to the new join before the event makes the control flow explicit:

$$\begin{aligned} C_i &:= C_i \cup \{c_{split}, c_{join}\} \\ l_i &:= l_i \cup \{(c_{split}, and), (c_{join}, and)\} \\ A_i &:= A_i \setminus \{(f_1, y_1), (x_2, e_2)\} \cup \\ &\quad \{(f_1, c_{split}), (c_{split}, y_1), (c_{split}, c_{join}), (c_{join}, e_2), (x_2, c_{join})\} \end{aligned}$$

2.4 Restructuring Rules

Deriving the integrated EPC according to Definition 4 may result in unnecessary structure of the process graph. In particular, we identify two reduction rules:

Definition 5 (Reduction Rules). Let $EPC = (E, F, C, l, A)$ be an (integrated) EPC. The following reduction rules can be applied without affecting the control flow in terms of the order of functions and events:

1. If there is a path $(c_1, p_1, \ldots, p_n, c_2)$ with $P = \{p_1, \ldots, p_n\} \in (E \cup F \cup C)$ and $(c_1, c_2) \in A$, then $A := A \setminus \{(c_1, c_2)\}$
2. If $c \in C \wedge |c\bullet| = |\bullet c| = 1$, then $A := A \setminus \{(x, c), (c, y)\} \cup \{(x, y)\}$ and $C := C \setminus \{c\}$.

The first rule eliminates redundant arcs between two connectors that represent control flow which is implicitly captured by an alternative path between these connectors. The second rule eliminates connectors that have only one input and one output arc. Such unnecessary connectors can result from applying the first reduction rule. Please note that the first rule can change the execution semantics of the EPC: if there is an *xor*-split or an *or*-split in the path between the *and*-split and the *and*-join, the *and*-join can run into a deadlock. As such a potential deadlock is introduced in the integration step, we argue that it should be eliminated using the first rule.

3 Application to the SAP Reference Model

In order to demonstrate the applicability of our process view integration approach, we use two EPC process models from the SAP reference model [4,7], namely the two processes *Customer Inquiry and Quotation Processing* and *Customer Inquiry* that were presented in Figure 1.

In Section 2, we have identified semantic equivalence relationships between the events *Customer inquiries about products* and *Quotation to be created* and the function *Customer inquiry processing* that appear in both input EPCs. Figure 2 shows the integrated EPC model after applying the merge operator. For each pair of equivalent functions and events, the respective *and*-joins and -splits are inserted following Definition 4. The first and the second pair of *and*-split and -join can be reduced according to the reduction rules of Definition 5. The restructured EPC model is given in Figure 3.

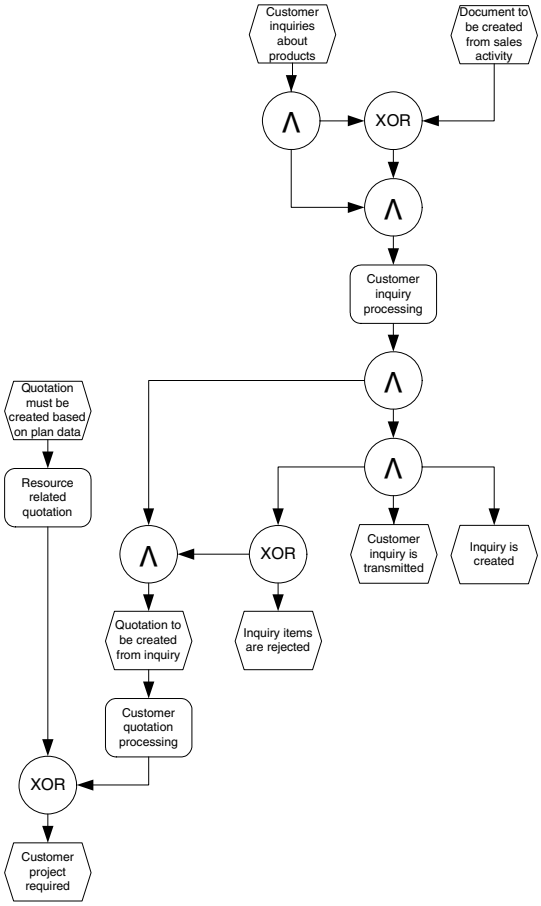

Fig. 2. Integrated EPC for *Customer Inquiry and Quotation Processing*

4 Related Research

There is extensive work in the database community on view integration and schema integration. Batini et al. [3] provide a comparative analysis of schema integration methodologies. They distinguish the schema integration activities of preintegration, comparing, conforming, merging, and restructuring. In our paper, we focus on comparing, merging, and restructuring EPCs. There are several contributions that focus on specific aspects of schema integration. Rahm and Bernstein provide a survey on how matches across different schemas can be identified automatically [13]. Rizopoulos and McBrien discuss the application of the hypergraph data model (HDM) with a wide set of semantic relationships for merging data schemas [14]. For a comprehensive integration method and a detailed overview of work on schema integration see [15].

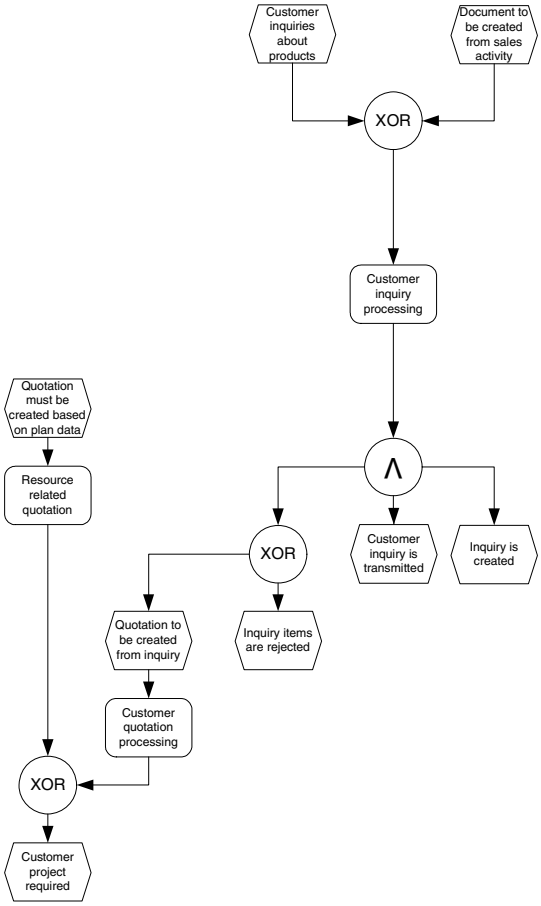

Fig. 3. Restructured EPC for *Customer Inquiry and Quotation Processing*

The heterogeneity of business process modeling languages is a notorious problem both for theory and practice [10]. Formalizing the metamodel of such a language as a schema makes schema integration applicable for process modeling language consolidation. In [11], the authors point to the problem of different control flow representations (graph-based versus block-oriented) as a specific source of heterogeneity. In [5] an integration process for business process metamodels is presented that is able to cope with such control flow heterogeneity.

There has been some work on joining and merging business process models with Petri net based formalisms. The integration of EPC models is conducted here in analogy with central concepts of the Semantic Process Language (SPL) [16,17]. In this language, formulas called modules can be formulated to specify process sets over elementary processes (in which an action occurs or is being forbidden) using operators for sequence building, alternatives, concurrency and synchronization, iteration, and negation. Canonical building rules take modules

as input and generate Module nets, i.e. Petri nets with explicit *start* and *goal* transitions. Processes of such a Module net are firing sequences which reproduce the empty initial marking. Per definition, they are also processes of the module from which the Module net was generated.

In principle, applying the *and*-operator of SPL (for concurrency and synchronization) yields the intersection of the process sets of the participating modules concerning common actions. For Module net implementations, all transitions interpreted by the same elementary process are fused into a single one. Also *start* and *goal* transitions are synchronized in order to enforce the co-execution of the entire processes. Finally, all contradicting elementary processes (in which an action occurs in one operand and is forbidden within the other) are prohibited from occurring together (or even synchronously) in a process by additional conflict places - a rule which is of no importance for our approach here since EPCs do not support prohibition.

Comparable work is reported in [12]. Calculating merges and joins of process models can also be related to problems and solutions on determining inheritance relationships between process models [2,1]. Still, the mentioned approaches have not yet been adopted for conceptual languages such as EPCs as we propose in our approach.

5 Contributions and Limitations

In this paper we have presented an approach to business process design by view integration. In particular, we have formalized semantic relationships between elements of different process models, we have specified a merge operator to integrate input process models, and have identified reduction rules to simplify the integrated process model. Furthermore, we have applied this approach to an example of two EPC business process models from the SAP reference model to demonstrate the applicability of our approach. It has to be mentioned that the approach, although defined for EPCs, can be adopted for other process modeling languages. In future work, we aim to provide tool support for the merging of EPC business process models.

References

1. Wil M. P. van der Aalst. Inheritance of business processes: A journey visiting four notorious problems. In Hartmut Ehrig, Wolfgang Reisig, Grzegorz Rozenberg, and Herbert Weber, editors, *Petri Net Technology for Communication-Based Systems - Advances in Petri Nets*, volume 2472 of *Lecture Notes in Computer Science*, pages 383–408. Springer, 2003.
2. T. Basten. *In Terms of Nets: System Design with Petri Nets and Process Algebra*. PhD thesis, Eindhoven University of Technology, The Netherlands, December 1998.
3. C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.

4. Thomas Curran, Gerhard Keller, and Andrew Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model.* Enterprise Resource Planning Series. Prentice Hall PTR, Upper Saddle River, 1997.
5. T. Hornung, A. Koschmider, and J. Mendling. Integration of heterogeneous BPM Schemas: The Case of XPDL and BPEL. Technical Report JM-2006-03-10, Vienna University of Economics and Business Administration, 2006.
6. G. Keller, M. Nüttgens, and A. W. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1992.
7. G. Keller and T. Teufel. *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping.* Addison-Wesley, 1998.
8. E. Kindler. On the semantics of EPCs: Resolving the vicious circle. In J. Desel and B. Pernici and M. Weske, editor, *Business Process Management, 2nd International Conference, BPM 2004*, volume 3080 of *Lecture Notes in Computer Science*, pages 82–97, 2004.
9. Jan Mendling and Markus Nüttgens. EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). *Information Systems and e-Business Management*, 4, 2006.
10. Jan Mendling, Markus Nüttgens, and Gustaf Neumann. A Comparison of XML Interchange Formats for Business Process Modelling. In F. Feltz, A. Oberweis, and B. Otjacques, editors, *Proceedings of EMISA 2004 - Information Systems in E-Business and E-Government*, volume 56 of *Lecture Notes in Informatics*, 2004.
11. Jan Mendling, Cristian Pérez de Laborda, and Uwe Zdun. Towards an Integrated BPM Schema: Control Flow Heterogeneity of PNML and BPEL4WS. In K.-D. Althoff, A. Dengel, R. Bergmann, M. Nick, and T. Roth-Berghofer, editors, *Post-Proceedings of the 3rd Conference Professional Knowledge Management (WM 2005)*, volume 3782 of *Lecture Notes in Artificial Intelligence*, pages 570–579. Springer Verlag, 2005.
12. Günter Preuner, Stefan Conrad, and Michael Schrefl. View integration of behavior in object-oriented databases. *Data Knowl. Eng.*, 36(2):153–183, 2001.
13. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
14. Nikolaos Rizopoulos and Peter McBrien. A general approach to the generation of conceptual model transformations. In Oscar Pastor and João Falcão e Cunha, editors, *Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings*, volume 3520 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2005.
15. Ingo Schmitt and Gunter Saake. A comprehensive database schema integration method based on the theory of formal concepts. *Acta Inf.*, 41(7-8):475–524, 2005.
16. Carlo Simon. *A Logic of Actions and Its Application to the Development of Programmable Controllers.* PhD thesis, University of Koblenz-Landau, Department of Computer Science, Germany, May 2002.
17. Carlo Simon. Incremental Development of Business Process Models. In Jörg Desel and Ullrich Frank, editors, *Proceedings of the Workshop Enterprise Modelling and Information Systems Architectures*, volume 75 of *Lecture Notes in Informatics*, pages 222–235, Klagenfurt, Austria, October 2005. German Informatics Society.