View-Based Reverse Engineering Approach for Enhancing Model Interoperability and Reusability in Process-Driven SOAs

Huy Tran, Uwe Zdun, and Schahram Dustdar

Distributed Systems Group, Information Systems Institute
Vienna University of Technology, Austria
{htran,zdun,dustdar}@infosys.tuwien.ac.at

Abstract. In many companies, process-driven SOAs are introduced using technical process languages, such as BPEL, to orchestrate services. The process models developed using this approach are often too complex and hard to reuse because all process-related concerns are tangled in only one type of model. To make the models more understandable for non-technical stakeholders, many companies additionally introduce high-level process descriptions, e.g., specified in BPMN or EPCs, to offer a non-technical view of the processes. This divergence of process languages often leads to inconsistencies after a few evolution steps. We propose a novel approach based on architectural views that not only offers models tailored to the various stakeholders' concerns but also provides an automated integration of models at different abstraction levels. In particular, we propose an extensible reverse-engineering tool-chain to automatically populate various view models with information from existing process descriptions and generate executable code from these view models.

1 Introduction

In a process-driven, service-oriented architecture (SOA), business functionality is accomplished by executing business processes invoking various services. A typical business process includes a number of activities and a control flow. Each activity corresponds to a communication task (e.g., it invokes other services or processes) or a data processing task. The control flow describes how these activities are orchestrated. A process is typically represented either in an executable language, such as BPEL [7] or XPDL [24], or in a high-level modeling language such as BPMN [15], EPC [10], or UML Activity Diagrams [14].

Nowadays, business process developers have to deal with increasing needs for change, for instance, concerning business requirement changes or IT technology changes. Therefore, the process models should enable a quicker reaction on business changes in the IT by manipulating business process models instead of code. Unfortunately, most of the existing business processes are developed and maintained by technical experts (aka the IT experts) in low-level, executable languages. It is difficult for the business analysts to get involved in process development and maintenance because for these tasks an understanding of many

H. Mei (Ed.): ICSR 2008, LNCS 5030, pp. 233–244, 2008.

technical details is required. Hence, technical experts are required for many task in managing, developing, and maintaining the process models. At the same time, the process models become too complex and the various process concerns are hard to reuse. In addition, there is a lack of adaptation of process models to suit the needs of particular stakeholders, e.g. business analysts or technical experts.

As a solution to these problems, some companies introduce high-level process descriptions, for instance, specified in BPMN or EPCs, to offer a non-technical view of the processes. This practice leads to yet another problem, namely, the divergence of process representations. That is, various more or less abstract descriptions of each business process are created, which might quickly become inconsistent as changes occur. As a consequence, neither the information in the high-level models is reused for defining the technical models, nor vice versa.

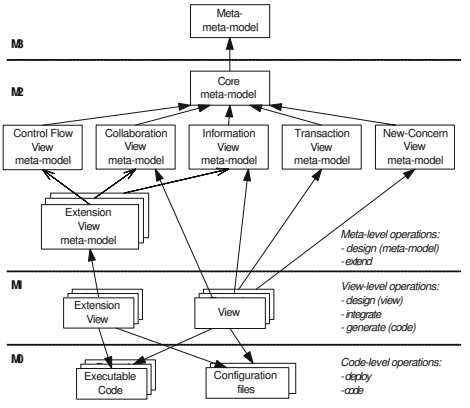
The aforementioned challenges have not been resolved in the context of process-driven SOAs yet. We present in this paper a novel view-based reverse engineering approach for addressing these challenges. Our approach harnesses the *concept of architectural views* and the *partial interpreter* pattern [25] to adapt process models to suit the requirements of particular stakeholders. Using the partial interpreter pattern, we devise a number of interpreters to extract more and less abstract views from process descriptions. The relationships between these views are maintained via our view-based modeling framework (VbMF) [20]. Using extension and view integration mechanisms, the views can be manipulated to produce more appropriate representations according to the stakeholders' requirements, and code in executable languages can be (re-)generated. VbMF not only supports the reuse of information in process models at different abstraction levels and in different process concerns, but also the reuse of information in existing process models, e.g. written in BPEL.

In this paper, first we give a short introduction to VbMF in Section 2. Section 3 describes the view-based reverse engineering approach. In Section 4 we present the details of using view-based interpreters to analyze existing business processes and extract various architectural views from the processes. Finally, in Section 5 we discuss related work and conclude in Section 6.

2 The View-Based Modeling Framework

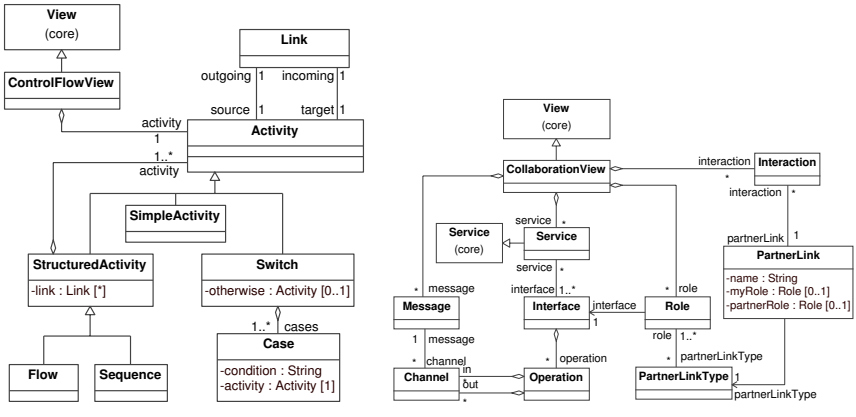
2.1 Overview of the View-Based Modeling Framework

The view-based modeling framework [20] is based on the concept of architectural views. An architectural view is a representation of a system from the perspective of a related set of concerns [8]. Each particular concern is (semi-)formalized by a respective meta-model. VbMF defines a number of meta-models (see Figure 1(a)), one for each architectural view. A meta-model at a lower abstraction level is defined as an extension of the meta-models at higher levels. VbMF's meta-models are either directly or indirectly derived a Core meta-model (see Figure 1(b)). The relationships between meta-models are used to bridge the gaps between meta-models at different abstraction levels and to propagate changes.



(a) View-based model-driven framework (b) The core meta-model

Fig. 1. The VbMF modeling framework and the *Core* meta-model



(c) Information View Meta-model (d) BPEL extension of the Collaboration View

Fig. 2. Three basic concern meta-models and a BPEL extension meta-model example

Example meta-models that we have derived from the Core meta-model are: Control Flow, Collaboration and Information View (see Figures 2(a), 2(b) and 2(c)). For particular technologies, e.g. BPEL/WSDL, the extension mechanisms can be used to enrich the abstract meta-models with the specifics of those technologies. To illustrate the extension mechanisms, we present a BPEL-specific collaboration view meta-model in Figure 2(d), which is defined by extending the elements from Figures 1(b) and 2(b). We use the distinction of the Core meta-model, generic view meta-models, and extension meta-models to represent different abstraction levels, such as business level and technical level.

In our implementation of these concepts, we exploit the model-driven software development (MDSD) paradigm [22] to separate the platform-neutral views from the platform-specific views. Code can be generated from the views by using model-to-code transformations. We have realized VbMF in openArchitectureWare (oAW) [16], a model-driven software development tool, and all meta-models are defined using the Eclipse Modeling Framework [5]. To demonstrate our approach, we have exemplified it using BPEL and WSDL, which are likely the most popular process/service modeling descriptions used by numerous companies today. Nevertheless, in general, the same approach can be taken for any other process-driven SOA technologies by defining respective meta-models.

2.2 View-Based Reverse Engineering Tool-Chain

VbMF mainly consists of a forward engineering tool-chain (see Figure 3) in which the stakeholders can develop process-driven view models, can generate process code from these views, or can extend the modeling framework with other process concerns by adding new meta-models or by enhancing existing meta-models.

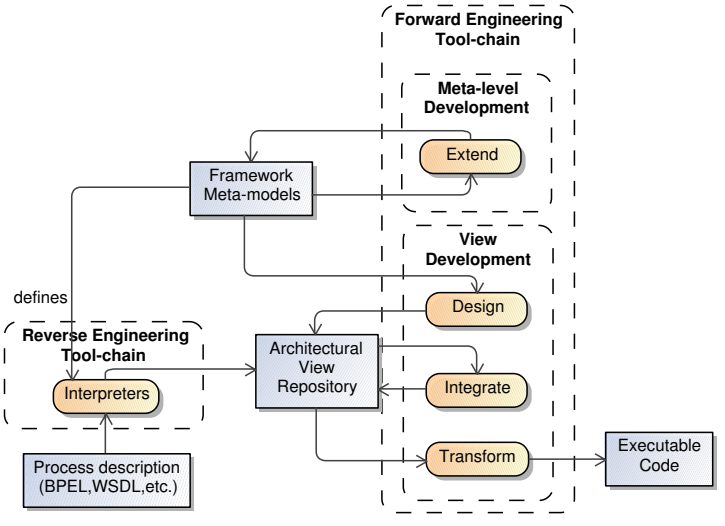


Fig. 3. The extended VbMF including the view-based reverse-engineering

Companies today have built up a vast amount of legacy process representations, either high-level or low-level, but there is no proper integration of these process descriptions, and no appropriate adaptation of process models to the stakeholders' needs and focus. Typically off-the-shelf process modeling tools, such as BPEL or BPMN tools, are used, and hence it is required to integrate them into VbMF. For these reasons, we extended VbMF with a reverse-engineering tool-chain for adapting process models and integrating various modeling representations. The outcome are tailored views that can be put into a common repository, and then be re-used in other processes or manipulated to re-generate new executable code, which corresponds to changes in the corresponding views (see Figure 3).

3 View-Based Reverse Engineering Approach

In the context of process-driven SOAs, many existing systems have built up an enormous repository of existing process code in executable languages, such as BPEL and WSDL. There are two important issues that have not been solved yet. Firstly, such process code integrates many tangled concerns such as message exchanges, data processing, service invocations, fault handling, transactions, and so forth. Secondly, these languages are rather technology-specific and therefore the abstract representations are not explicitly available at the code level. As a result, the process models become too complex for stakeholders to understand and maintain, to integrate, to cooperate with other processes, or to re-use process models from existing modeling tools.

Our view-based approach can potentially resolve these issues. However, for budgetary reasons, developing the view models, required in our approach, from scratch is a costly option. The alternative is an (automated) re-engineering approach comprising two activities: *reverse-engineering* for building more appropriate and relevant representations of the legacy code; *forward-engineering* for manipulating the process models and for re-generating certain parts of the process code. During the reverse engineering process, high-level, abstract and low-level, technology-specific views on the process models are recovered from the existing code. This way, the reverse engineering approach helps stakeholders to get involved in process re-development and maintenance at different abstraction levels. Reverse engineering of business processes should not only help to adapt process models to stakeholder needs but also offer the ability to integrate various process models to enhance the interoperability of process models. The view-based reverse engineering approach we propose in this paper aims at achieving these goals.

3.1 The Reverse Engineering Tool-Chain

The reverse engineering tool-chain (see Figure 4) consists of a number of view-based interpreters, such as *control flow interpreter*, *collaboration view interpreter*, and so forth. Each interpreter is used for interpreting and extracting the corresponding view from the process descriptions. An interpreter of a certain view must be defined based on the meta-model which that view conforms

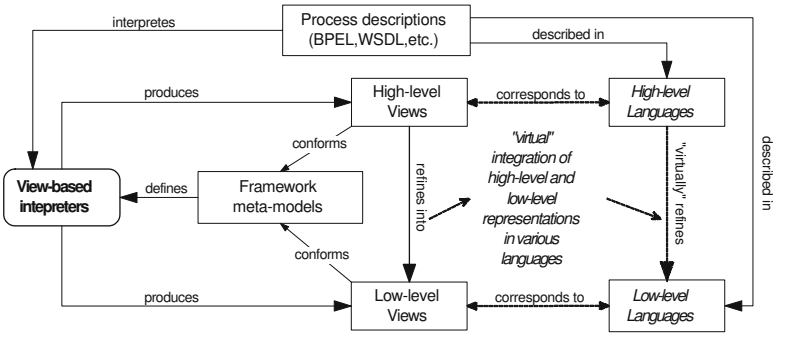


Fig. 4. The view-based reverse engineering tool-chain

to. For instance, the *control flow view* consists of elements such as *Activity*, *Flow*, *Sequence*, *Switch*, *Case* according to the *control flow view meta-model* (see Figure 2(a)). In order to extract the control flow view from process descriptions, the interpreter walks through the input descriptions to pick the above-mentioned elements. Other elements are ignored.

3.2 General Approach for View Extraction

The process descriptions comprise the specification of business functionality in a modeling language, for instance, as we exemplify in this paper, BPEL [7]. Moreover, the process functionality also exposes service interfaces, for instance, expressed in WSDL [23]. To demonstrate the extraction of appropriate views from process descriptions, we developed a number of interpreters such as *control flow interpreter*, *collaboration view interpreter*, as well as a *BPEL-specific extension view interpreter*.

Our general approach to define view interpreters is based on the Partial Interpreter pattern [25]. This pattern is typically applied when the relevant information to be interpreted from a language is only a (small) sub-set of the source document's language, and thus, the complexity of the whole language should be avoided in the subsequent interpretation. The approach based on Partial Interpreter enables us to define modular, pluggable interpreters, and the framework to be easily extensible with new views and view extraction interpreters. The solution is to provide a Partial Interpreter for view extraction, which only understands the specific language elements required for one view. There is a generic parser that is responsible for parsing the process descriptions. The parsing events generated by this generic parser are interpreted by the Partial Interpreters, which only interprets the language elements relevant for a particular view.

The Partial Interpreter's mapping specification and view-specific interpretation specification are both defined generically on basis of the meta-models. Hence, they can be reused for many concrete view models. In the subsequent sections, we present the details of the realization of the mapping specifications for basic process concerns, i.e., control flow interpreter, information view interpreter

and collaboration view interpreter to illustrate our general approach. Other view interpreters can be implemented following the same approach.

4 Details of the View-Based Reverse Engineering Approach: Three Empirical Analyses

In this section, we empirically analyze the capabilities of the view-based reverse engineering approach, such as the adaptation of process models to stakeholders' needs and the integration of models at different levels of abstraction, by investigating three typical cases in which the view-based reverse engineering approach can get applied. In doing so, we also introduce the details of our approach for applying it to BPEL/WSDL as an exemplary process-driven SOA technology.

These empirical analyses have been carried out on an industrial case study, namely, customer care, billing and provisioning systems of an Austrian Internet Service Provider (see [6] for more details). In the following, we use the Billing Renewal process as an example. The billing platform includes a wide variety of services provided by various partners such as financial services, domain services, physical hosting services, retail/wholesale services, and so on. These services are exposed in WSDL interfaces and integrated by using BPEL processes.

4.1 Extracting Relevant Views

The basic analysis, we performed, was to deal with the extraction of the control flow view from BPEL code. The control flow interpreter walks through the process description in BPEL and collects necessary information of atomic and structured activities. Then, it creates the elements in the Control Flow View and assigns their attributes with relevant values as specified by the Control Flow View meta-model (see Figure 2(a)). We demonstrate the mapping of Billing Renewal specification in BPEL onto the Control Flow View in Figure 5.

4.2 Extracting Views at Different Abstraction Levels

To illustrate the ability of adapting views at different levels of abstraction, we devise two interpreters to extract the Collaboration View and the BPEL-specific extension of the Collaboration view. These interpreters are realized using the same approach as used for the control flow interpreter. However, these views comprise not only elements from the BPEL descriptions but also elements of the process interfaces specified in WSDL files. That is, the interpreters firstly collect information from WSDL descriptions, then walk through the BPEL specifications to the extract relevant elements, and finally create relevant elements on the views according to the Collaboration View meta-model in Figure 2(b). Figures 5 and 6 illustrate the extraction of the Collaboration View from BPEL descriptions of the Billing Renewal process.

The Collaboration View is a high-level representation compared to the BPEL extension of the Collaboration View, which is at a lower level of abstraction.

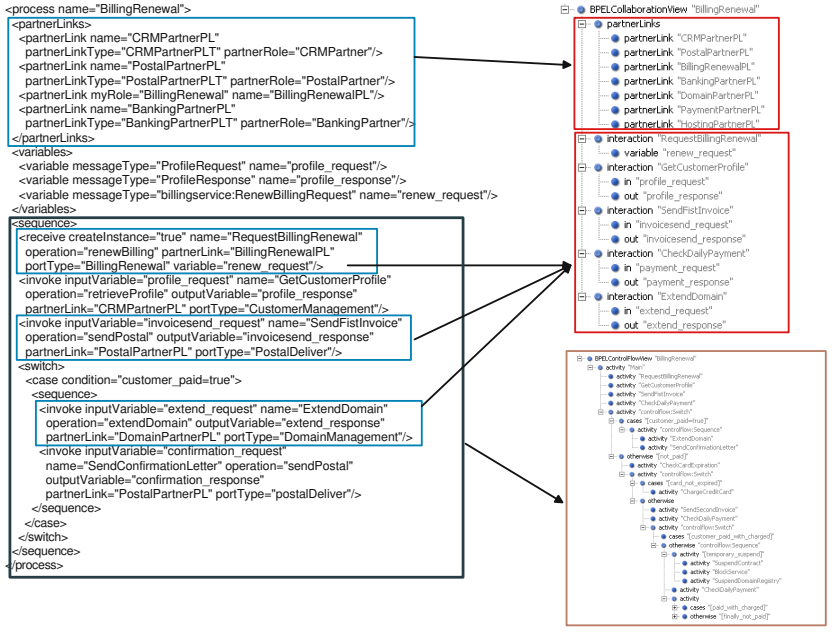


Fig. 5. Mapping the Billing Renewal process (left-hand side) onto the VbMF's views including the Collaboration View (top-right) and the Control Flow View (bottom-right)

Therefore, the BPEL extension view consists of additional elements and some of these elements have extra properties compared to those of the Collaboration View. This way, other process-driven modeling languages, either high-level or low-level, can be handled and integrated by using the view-based reverse engineering tool-chain and VbMF.

4.3 Enhancing the Adaptability of the Process Models

The adaptability of process models to the requirements of a certain stakeholder can be enhanced using two methods developed in VbMF: extension mechanisms and view integration. View extension mechanisms [20] allow us to enrich existing meta-models with additional elements and/or extra attributes for the existing elements of the original meta-models. This way, the abstract views can be gradually refined into less abstract views by increasing their granularity with added technology-specific features until the resulting views are well suited for a particular stakeholder's needs. Next, we define respective interpreters for these views and use the interpreters to extract the corresponding views from the existing process code. An example of view extension is the BPEL-specific extension of the Collaboration View shown in the previous analysis.

View integration [20] is another method to produce new richer views by merging existing views. For instance, in [20], we have developed a simple name-based

Fig. 6. Mapping the Billing Renewal process (left-hand side) onto the Collaboration View (right-hand side)

matching algorithm and presented an example of integrating the control flow view and the collaboration view. The matching algorithm searches the input views for integration points, which are, in this case, the conformable elements with the same name. Afterward, the two views are merged together at these integration points. The resulting view inherits the control flow that defines the execution order of activities. In addition, the activities in the resulting view that are responsible for invoking services inherit a number of additional attributes from the corresponding activities defined in the collaboration view.

5 Related Work

Our work presented in this paper is a *reverse engineering approach* [3] based on the concept of architectural views [8]. The whole VbMF tool-chain provides support for *reengineering* [17] as well. That is, in addition to the reverse engineering parts of the tool chain, means for re-structuring, modification, and forward engineering are provided to yield new system structures and functionality.

In the context of reverse engineering, view-based approaches are an emerging area of interest. For instance, the approaches reported in [2, 4, 19] focus on inter-organizational processes (in term of cross-organizational workflows) and use views to separate the abstract process representations (aka public processes) from the internal processes (aka private processes). Bobrik et al. [1] present an approach to process visualization using personalized views and a number of operations to customize the views. Zou et al. [27] propose an approach for extracting business logic, also in terms of workflows, from existing e-commerce applications. All these approaches aim at providing perspectives on business processes

at a high level of abstraction and maintaining the relationships among different abstraction levels to quickly re-act to changes in business requirements. These approaches have in common that only the control flow of process activities (aka the workflows) is considered. Other process concerns, such as service/process interaction, data processing, etc. have only been partially exploited or not targeted. In addition, these approaches do not support enhancing process views or propagating changes as supported in our approach, for instance, through view integration, view extension and code generation.

Kazman et al. [9] describe the Dali workbench, an approach for understanding and analysis the system architecture. The extraction process begins with extracting views from source code using lexical analyzers, parsers or profilers. Next, the relationships among views are established by view fusion to improve the quality and the correctness of views. However, because of the complexity of typical process models, this approach is hardly applicable to capture the whole process description in a unique view.

In the context of process-driven modeling, there are a number of standard languages in which some provide high-level descriptions, for instance, BPMN [15], EPC [21, 10] and Abstract BPEL in WS-BPEL 2.0 [13]. There is no explicit link between these languages and the executable languages. This has led to a number of recent research approaches. For instance, Mendling et al. [12] discuss the transformation of BPEL to EPCs. Ziemann et al. [26] present an approach to model BPEL processes using EPC-based models. Recker et al. [18] translate between BPMN and BPEL. Mendling et al. [11] report on efforts in X-to-BPEL and BPEL-to-Y transformations. These transformation-based approaches mostly focus on one concern of the process models, namely, the control flow, which describes the execution order of process activities. They offer no support for extension of process models or integrating other concerns of process models, such as service interactions, data processing, transaction handling, etc. Hence, during the transformation from process code to abstract representations, necessary information required to re-generate executable code gets lost.

WS-BPEL 2.0, the newly revised standard, provides the concept of an Abstract BPEL process, which is represented by the same structures as an Executable BPEL process. Developers can explicitly hide some syntactic constructs in an Abstract BPEL process using predefined opaque tokens as explicit placeholders for the omitted details. An abstract process is often associated with a profile which specifies the semantics of the opaque tokens. Hence, one could use an approach akin to our approach where the high-level view is the abstract process profile, and low-level representations are respective profiles. Then our reverse engineering tool-chain could be used to extract the relevant views.

All the above-mentioned approaches and standards have difficulties in handling the complexity of process models: Because the business process integrates numerous concerns, the complexity of process model increases as the number of process elements, such as message exchanges, service invocations, data processing tasks, etc. grows. Hence, these approaches are less efficient than our

approach in dealing with pretty huge existing process repositories, developed in other languages or dialects, or integrating arbitrary process modeling tools.

6 Conclusion

The view-based reverse engineering approach, presented in this paper, can help the various stakeholders of a process-driven SOA to overcome two important issues. Firstly, it exploits the concept of architectural view to deal with the complexity of existing process repositories and to adapt the process representations to the stakeholders' needs and focus. Secondly, it provides the ability of integrating diverse process models and offers explicit relationships for understanding and maintaining process models and for propagating changes. Hence, process models at different abstraction levels and different process concerns can be reused to populate the other. This has been achieved by developing a novel concept for a reverse engineering tool chain, based on partial interpreters and view models, and by seamlessly integrating this reverse engineering tool chain into our view-based modeling framework, which also supports means for forward engineering, such as view integration, view extension and code generation. The reverse engineering tool chain enables the reuse of existing process code, e.g. written in BPEL/WSDL, in the view-based modeling framework.

References

1. Bobrik, R., Reichert, M., Bauer, T.: View-based process visualization. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 88–95. Springer, Heidelberg (2007)
2. Chebbi, I., Dustdar, S., Tata, S.: The view-based approach to dynamic inter-organizational workflow cooperation. Data Knowl. Eng. 56(2), 139–173 (2006)
3. Chikofsky, E.J., Cross, J.H.I.: Reverse engineering and design recovery: A taxonomy. IEEE Software 7(1), 13–17 (1990)
4. Chiu, D.K.W., Cheung, S.C., Till, S., Karlapalem, K., Li, Q., Kafeza, E.: Workflow view driven cross-organizational interoperability in a web service environment. Inf. Tech. and Management 5(3-4), 221–250 (2004)
5. Eclipse. Eclipse Modeling Framework (2006), `http://www.eclipse.org/emf/`
6. Evenson, M., Schreder, B.: D4.1 Use Case Definition and Functional Requirements Analysis. SemBiz Deliverable (August 2007)
7. IBM, B. Systems, Microsoft, SAP AG, and Siebel Systems. Business process execution language for web services (May 2003), `ftp://www6.software.ibm.com/software/developer/library/ws-bpel.eps`
8. IEEE. Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE-std-1471-2000, IEEE (2000)
9. Kazman, R., Carriere, S.J.: View Extraction and View Fusion in Architectural Understanding. In: ICSR 1998. Proc. of the 5th Int. Conference on Software Reuse, Washington, DC, USA, p. 290. IEEE Computer Society, Los Alamitos (1998)
10. Kindler, E.: On the semantics of EPCs: A framework for resolving the vicious circle. In: Business Process Management, pp. 82–97 (2004)

11. Mendling, J., Lassen, K.B., Zdun, U.: Transformation strategies between block-oriented and graph-oriented process modelling languages. Technical Report JM-200510 -10, WU Vienna (2005)
12. Mendling, J., Ziemann, J.: Transformation of BPEL processes to EPCs. In: Proc. of the 4th GI Workshop on Event-Driven Process Chains (EPK 2005), December 2005, vol. 167, pp. 41–53 (2005)
13. OASIS. Business Process Execution Language (WSBPEL) 2.0 (May 2007), http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.eps
14. OMG. Unified Modelling Language 2.0 (UML) (2004), http://www.uml.org
15. OMG. Business Process Modeling Notation (February 2006), http://www.bpmn.org/Documents/OMG-02-01.eps
16. openArchitectureWare.org (August 2002), http://www.openarchitectureware.org
17. Antonini, P., Canfora, G., Cimitile, A.: Reengineering legacy systems to meet quality requirements: An experience report. In: ICSM 1994. Proceedings of the International Conference on Software Maintenance, Washington, DC, USA, pp. 146–153. IEEE Computer Society, Los Alamitos (1994)
18. Recker, J., Mendling, J.: On the translation between BPMN and BPEL: Conceptual mismatch between process modeling languages. In: Eleventh Int. Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 2006), June 2006, pp. 521–532 (2006)
19. Schulz, K.A., Orlowska, M.E.: Facilitating cross-organisational workflows with a workflow view approach. Data Knowl. Eng. 51(1), 109–147 (2004)
20. Tran, H., Zdun, U., Dustdar, S.: View-based and Model-driven Approach for Reducing the Development Complexity in Process-Driven SOA. In: Intl. Working Conf. on Business Process and Services Computing (BPSC 2007), September 2007. Lecture Notes in Informatics, vol. 116, pp. 105–124. Springer, Heidelberg (2007)
21. van der Aalst, W.: On the verification of interorganizational workflows. Computing Science Reports 97/16, Eindhoven University of Technology (1997)
22. Völter, M., Stahl, T.: Model-Driven Software Development: Technology, Engineering, Management. Wiley, Chichester (2006)
23. W3C. Web Services Description Language 1.1 (March 2001)
24. WfMC. XML Process Definition Language (XPDL) (April 2005), http://www.wfmc.org/standards/XPDL.htm
25. Zdun, U.: Patterns of tracing software structures and dependencies. In: Proc. of 8th European Conference on Pattern Languages of Programs (EuroPLoP 2003), Irsee, Germany, June 2003, pp. 581–616 (2003)
26. Ziemann, J., Mendling, J.: EPC-based modelling of BPEL processes: a pragmatic transformation approach. In: Proc. of the 7th Int. Conference Modern Information Technology in the Innovation Processes of the Industrial Enterprises (MITIP 2005) (2005)
27. Zou, Y., Hung, M.: An approach for extracting workflows from e-commerce applications. In: ICPC 2006. Proc. of the 14th IEEE Int. Conf. on Program Comprehension (ICPC 2006), Washington, DC, USA, pp. 127–136. IEEE Computer Society, Los Alamitos (2006)