Verification of Forbidden Behavior in EPCs

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

Jan Mendling
WU Vienna
jan.mendling@wu-wien.ac.at

Abstract: Event-driven process chains (EPCs) are frequently used as a modeling language for the representation of business processes. As such, business analysts are familiar with using EPC models in the context of business process management. Up to now, there is no verification technique available that allows business analysts to express forbidden behavior in an intuitive manner. In this paper, we discuss the specification of such forbidden behavior with the aid of EPCs and demonstrate the verification of this behavior against models of the desired behavior, also formulated as EPC diagrams. For this purpose, a novel approach to join EPC models and to interpret the result is discussed. It is based on a transformation of both EPC models to Module nets, a specific kind of Petri nets, and the application of verification methods already defined for this net class. The findings are illustrated with a running example that picks up an EPC process model from the SAP reference model.

1 Introduction

Event-driven Process Chains (EPCs) [KNS92] are frequently used as a process modeling language for specifying business requirements on a conceptual level. EPCs' popularity stems from a more descriptive representation of control flow in comparison to Petri nets [SL05]. Furthermore, extensive tool support as well as their integration with the conceptual enterprise modeling framework of ARIS [Sch00] has proliferated a wide-spread application in industry process modeling projects. EPCs can be regarded as a de facto standard for process reference modeling, at least in the German speaking countries. Some of these reference models such as an extensive model for the resale industry [BS96] and the SAP reference model for version 4.6 of R/3 [KM94, KT98] are documented in books and are accessible for academia and industry projects.

Beside their popularity, there has been an ongoing debate on formalization of EPCs, especially the non-locality of join connectors. Most authors define transformation semantics to Petri nets or EPC-specific transition systems. For an overview and a recent contribution see [Kin04, MNN05b]. Due to the ongoing debate on precise semantics, there have been only a few contributions on formal analysis of EPCs (see e.g. [DA04, DAW05]). Yet, such analysis is of paramount importance when certain business requirements for a specific EPC business process model have to be verified. We have observed that business analysts not only think in terms of normative behavior of a process, but frequently express forbidden behavior. In this context, we address two challenges in this paper: first, we define a

concept for business analysts to specify forbidden behavior by the help of EPCs, because EPCs are a business process modeling language with whom they are familiar. Second, we describe a methodology for the verification of forbidden behavior against a normative EPC business process model.

The paper is structured as follows. Section 2 presents a motivating example. We use a real-world EPC business process model taken from the SAP reference model to illustrate the verification challenge. Section 3 gives an overview on Module nets and how to use the join operator for Module nets for our verification goal. In Section 4, we specify a mapping from EPCs to Module nets, a specific class of Petri nets, and discuss how the merge operator of Module nets can be applied to verify forbidden behavior in EPCs. We then apply the merge operator to the example EPC and the forbidden behavior EPC to illustrate the verification approach. Section 5 relates our contribution to other work on process model verification with a focus on EPCs. Finally, Section 6 concludes the paper and gives an outlook on future research.

2   Motivating Example

EPCs are frequently used to specify business requirements on a conceptual level. That is also the case for the Period-End Closing Process for Periodic Product Cost Controlling given in Figure 1 (left). It is the second part of a real-life EPC process model taken from the SAP reference model [KT98]. The process is defined within the Period-End Closing (Controlling) directory of the Revenue and Cost Controlling part of the SAP reference model. It includes 15 events, 14 functions, and seven connectors. Each function included in the figure is further specified by linked EPC models. We use the example process to illustrate the problem of specifying and verifying forbidden behavior of EPC business process models.

Consider the Period-End Closing Process to be implemented within an enterprise called Exemplum AG. The process starts with the calculation of the actual prices based on historic sales data of the period. After that the periodic variance is calculated. This is used both for settlement and for cost analysis. Figure 1 (left) captures the implementation of the process with settlement and analysis being performed in concurrency in order to minimize cycle time. The process terminates when all final events occurred.[1]

Based on some unexpected periodic-end closing results in the last period, the management of Exemplum AG is wondering whether settlement activities possibly occur before the analysis has been performed and declare this behavior as forbidden. Furthermore, management charges the head of the controlling department to verify whether the forbidden behavior is allowed by the current implementation of the process. In the following, we will speak in particular of a specification when referring to the formal representation of the process requirements, and of implementation when referring to the behavior of the process that is actually in place and captured by the process model.

---

[1]This matches implicit termination semantics of EPC end events [MNN05a].
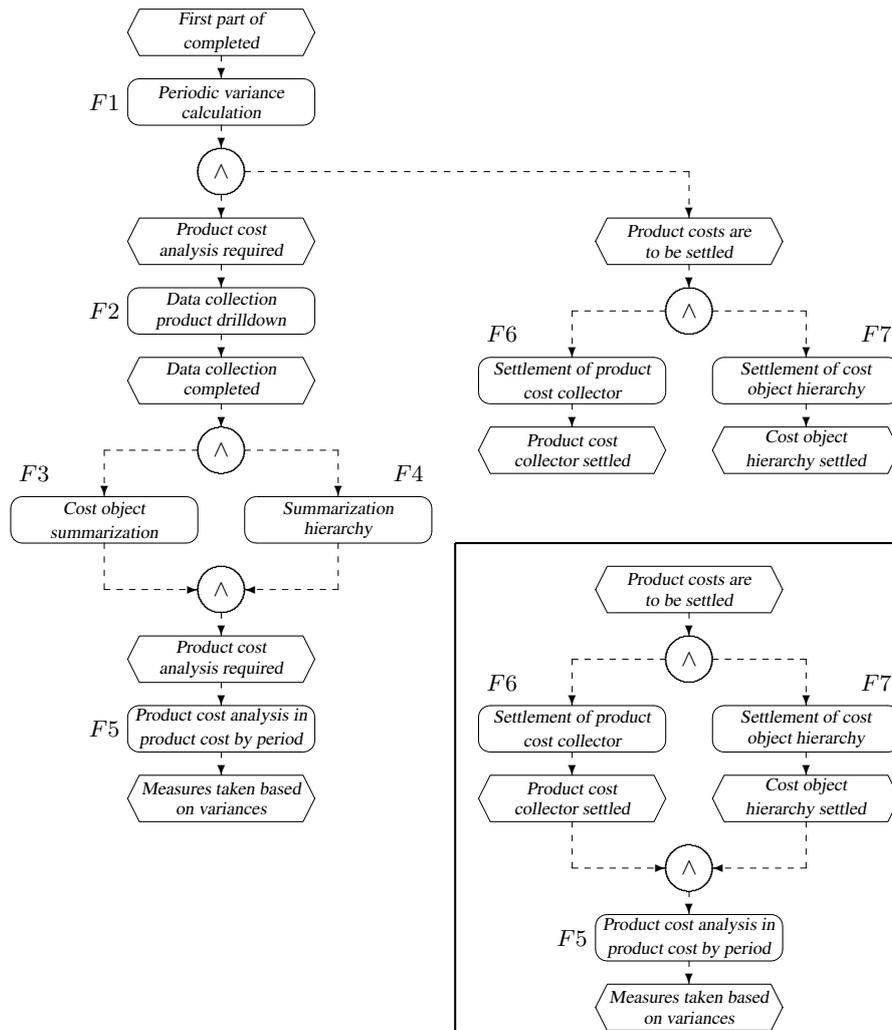
Figure 1: (Left) EPC for Period-End Closing: Periodic Product Cost Controlling taken from the SAP Reference Model [KT98]   (Lower right) Specification of forbidden behavior as EPC

For such a business scenario as outlined above, it is important that business analysts are allowed to specify forbidden behavior in a representation with whom they are familiar. The head of controlling defines a process fragment that captures the forbidden behavior as an EPC: after the start event, the settlement activities are executed followed by the analysis activities. Figure 1 (lower right) gives this forbidden behavior as an EPC. Please note that the specification of such forbidden behavior does not require any modification or extension neither to EPC syntax nor semantics. The aim of the subsequent sections is to consider the implemented EPC process and the forbidden behavior both specified as an EPC as input to an analysis technique that allows a respective verification. The Semantic Process Language (SPL) and Module nets build the foundation of our proposed technique.

3 Theoretical Foundation

The Semantic Process Language (SPL) introduced in [Sim05] is a formal language to specify wanted and forbidden behavior. Such specifications can then be verified against a respective implementation with the aid of (binary) operands provided by SPL. Since the SPL semantics are defined via Petri nets, respective verification techniques can also be applied to other process modeling approaches with comparable concepts. We apply them to the EPC example of Section 2.

Modules are the words or formulas of SPL. They define sets of processes. In a process, actions occur or are prohibited in a sequential order. Modules, moreover, support the definition of alternative, concurrent and iterated behavior. The semantics of modules are defined via their Petri net implementations. For this, canonical building rules take modules as input and generate Module nets as output. Specific firing sequences of these nets (their processes) are then interpreted as processes of the implemented modules. Figure 2 shows the relationship between the concepts.
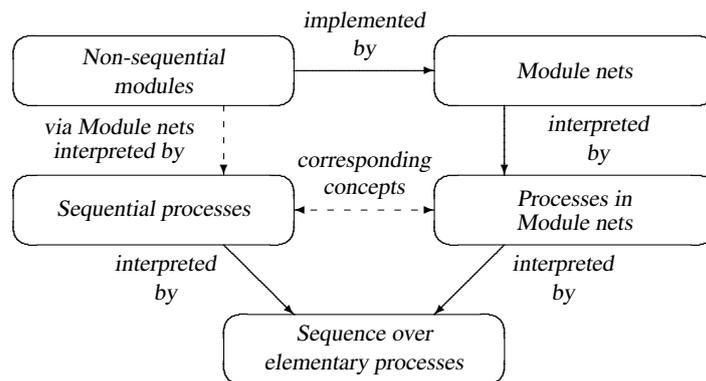

Figure 2: Relationship between the concepts of the paper

Within Module nets, the firing of an explicit *start* transition with empty preset marks the

236

beginning of each process. Its termination is indicated by the firing of an explicit *goal* transition with empty postset which reproduces the empty initial marking. In a process, *start* and *goal* transition are only allowed to occur exactly once. All other transitions are associated to sets of actions. The firing of these transitions is interpreted as the occurrence or prohibition of these related actions.

Module nets are closely related to Workflow nets where an initial start marking is transformed into a specific goal marking which implicitly means the reproduction of the empty initial marking in the remaining of the net. Also here, the firing of the remaining transitions is interpreted as the occurrence of actions within a workflow. Prohibition of actions is no intended operation of Workflow nets.

For proving process properties within modules, the and-operator is of central importance which is implemented by joining the participating Module nets. Hereby, equally named transitions are merged into a single one. Moreover, the concurrent occurrence and prohibition of actions symbolized by two transitions is prevented with the aid of conflict places between them. For technical details refer to [Sim06]. The resulting net, in principle, specifies the intersection of the process sets of the participating modules (or Module nets, respectively). It can therefore be used for proving wanted and forbidden behavior. We will concentrate on forbidden behavior in the following.

Assume, M_I is a module describing an implemented business process and M_F is a module specifying forbidden behavior, i.e. processes that should not occur in the implementation M_I. Now module $M_I \oslash M_F$ is defined (in principle) as the intersection of the processes of M_I and M_F and is calculated by joining the module net representations of these two modules. If the result does not contain any process (i.e. no firing sequence exists which reproduces the empty initial marking by firing *start* and *goal* transition exactly once), then the implementation M_I fulfills the specification M_F. Otherwise, if there exist processes in $M_I \oslash M_F$, then M_I must include processes which are specified by M_F as unwanted. In this case, the implemented business process described by M_I must be changed.

In order to apply this idea also to EPCs, we firstly must transform a given EPC into a Module net. Although this is partially possible as demonstrated in [SD04], such a transformation has some restrictions:

- Due to the non-local interpretation of some EPC elements (especially of the or-connector), the elements of an EPC cannot simply be mapped into Petri net elements in general.

- While places in Petri nets only have an intrinsic meaning for the model, events within EPCs have a meaning which is related to the modeled domain. These concepts are therefore in principle hardly transformed into each other. For details see e.g. [DA04, MNN05b].

Independently from these general considerations, we can transform structural aspects of EPCs into Module net concepts, i.e. we do not resolve the semantics of event inscriptions in the Petri net transformation. Nonetheless, a meaningful interpretation of the transformation result is possible as the example shows.

4   Verification Process and its Application to the Example

In order to verify the EPC example of Figure 1 (left) against the specification of the forbidden behavior depicted as an EPC in Figure 1 (lower right), we must transform both EPCs into Module nets first. As mentioned in the previous section, our transformation only takes the EPC structure as input and does not interpret the meaning of the events. Moreover, our transformation benefits from the absence of an or-connector in the considered example which allows us to do the transformation element by element.
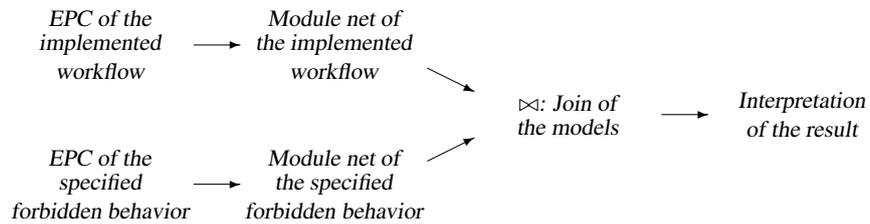

Figure 3: Phases of the verification approach

After this transformation, we take the Module net representations of the EPCs and join them. The resulting net, then, must not allow the execution of any process without contradicting the specification of the forbidden behavior. Depending on the result, we draw our conclusions concerning our considered EPC models. Figure 3 shows the phases of our approach in a diagram.
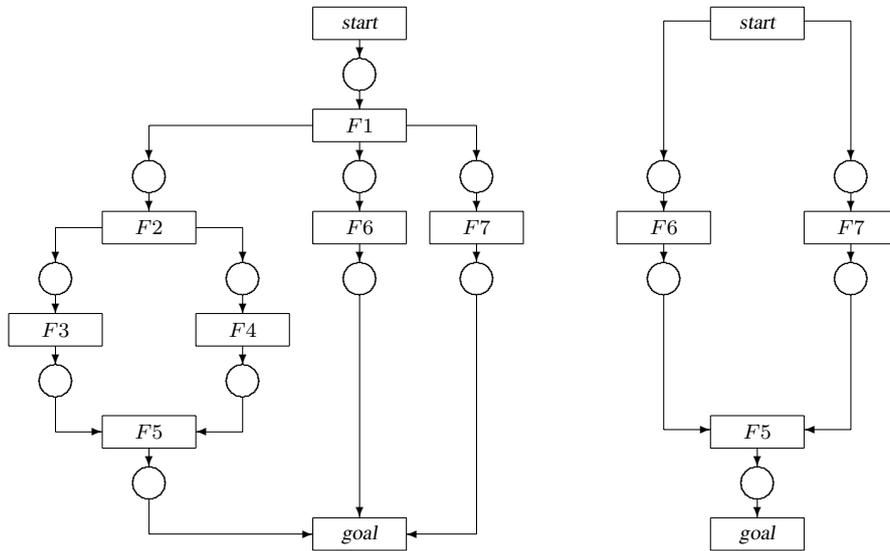

Figure 4: Module net transformation of the implemented workflow and the specification of the forbidden behavior

Figure 4 (on the previous page) shows the Module net transformation of the EPCs of Figure 1 in a single diagram. While in the net of the implemented workflow (left) functions $F6$ and $F7$ occur concurrently to the remaining functions except $F1$, in the specification (right) function $F5$ must occur afterwards.

Figure 5 shows the join of both Module nets where redundant and implicit places have been left away. Since there exist firing sequences which reproduce the empty initial marking by firing *start* and *goal* once (for example $\langle start, F1, F2, F3, F6, F4, F7, F5, goal \rangle$), we can conclude on the following: the (structure of the) EPC model of the implemented workflow does not rule out that the settlement occurs before the final product cost analysis has been conducted. Consequently, the implemented workflow must be changed in order to fulfill the new management policy.
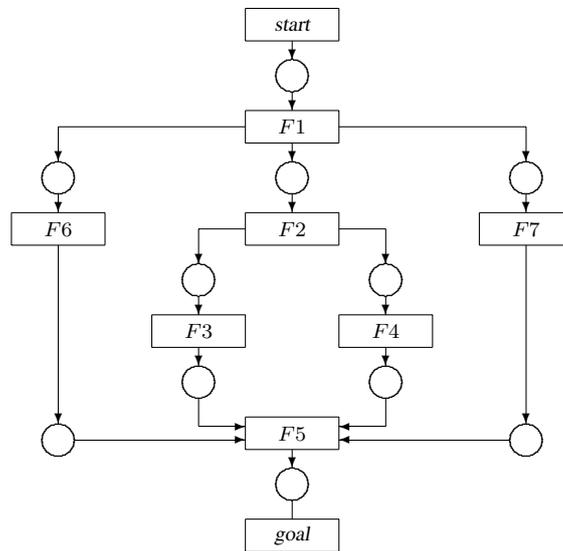

Figure 5: Join of implemented workflow and specified forbidden behavior

5   Related Work

Mappings to Petri nets (like those discussed in [Kin04]) play an important role for the application of formal analysis techniques for EPCs. Some work is especially dedicated to verification of EPCs. Dehnert presents an approach for the derivation of sound Workflow nets from EPCs. The original EPC specification is refined in a stepwise manner based on the relax-soundness and robustness criteria of the underlying Workflow net [DA04]. A Workflow net is relaxed-sound if every transition participates in a proper case. It is weaker than soundness [Aal98] that requires each reachable state to always lead to a proper terminatation of the process. With techniques known from controller synthesis it is possible to restrict relaxed sound Workflow nets to sound behavior [Deh03]. Van Dongen et al. define an approach for the interactive verification of EPCs based on reduction rules and

a mapping to Petri Nets [DAW05]. The idea is to cut off structured blocks of EPCs and map the remaining model to Petri nets. The user is then asked to interactively eliminate undesirable execution paths. This technique is applied for the analysis of parts of the SAP reference model. Mendling et al. present a transformation of yEPCs to YAWL which is also applicable for standard EPCs [MNN05a, MMN06]. This transformation allows formal analysis with analysis tools such as WofYAWL [VAH05]. All these three approaches have in common that they consider only the EPC implementation as input for the analysis. Our work presented in this paper takes an EPC implementation and an EPC specification of forbidden behavior as input for a verification approach based on Module nets. This approach is unique as it covers verification of forbidden behavior in EPCs.

A more general line of related research is dedicated to the verification of dynamic systems in automatic control. It is, however, only partially applicable to the EPC models. In dynamic control, research aims to precisely specify systems with the aid of Petri nets [ZV99]. Hereby, controller and the system to be controlled are either defined separately from each other [Sim01], or the controller is synthesized from the system's specification and a given set of forbidden state which the system may never reach [RW87b, RW87a]. Synthesis and verification can then be generalized to the question whether specific states or state sequences (in terms of markings and follower markings) may be reached. Model checking techniques based on temporal logic help to solve this question [Bra91, Sch02]. However, they all demand expertise on formal specification languages, how to use formal model checking algorithms, and how to interpret the results.

Process algebras are used little for business processes modeling since they do not properly support visualization [Bas03, p. 382] typically required by business analysts. Moreover, it is not clear whether they are expressive enough to describe all kinds of business process problems [Aal05] - although workflow patterns could be defined in [PW05].

The findings on soundness of Workflow nets (already mentioned above) are only partially transferrable to EPCs. Since they are not suggested as an execution language for workflow management systems (due to a non-local interpretation of the models [Kin04]), other more process oriented properties like the once discussed in Section 2 are of major importance.

6   Conclusion

This paper presents a novel approach for the verification of forbidden behavior in EPCs. Based on a running example of a process model from the SAP reference model, we presented a mapping to Module nets for this purpose. Furthermore, we demonstrated how the join operator of Module nets can be used to verify forbidden behavior of an EPC. The presented approach supports the modeling of both: normative business processes as they are defined in an EPC business process model and specifications of undesired behavior. In opposite to model checking or process algebra, a single representation (in this case EPCs) is used. The presented transformation into Module nets could be conducted by a tool automatically and without confronting a user with these intermediary results.

A weakness of the presented approach is the transformation of the EPC models into Module nets since it does not interpret the intention of event inscriptions used within the EPC

model - experience shows that this typically requires dedicated domain knowledge. Consequently, forbidden behavior might be prohibited by these events although the EPC structure would allow it. Since our approach recognizes only the structural possibility of forbidden behavior, it is more restrictive than necessary. We, however, assume that controlling the control flow in EPCs over events instead of structural operands is an indicator that restructuring the process is useful. Our future work on specification and verification of forbidden behavior with EPCs will be on immediately applying our method to the original EPC models. A particular challenge will be the definition of a join for or-connectors.

References

[Aal98]   W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 1998.

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

[Bas03]   T. Basten. Verifying Petri net Models using Process algebra. In C. Girault and R. Valk, editors, *Petri nets for Systems Engineering*, chapter 16.5. Springer, Berlin, 2003.

[Bra91]   J. C. Bradfield. Proving Temporal Properties of Petri Nets. In *Advances in Petri Nets*, LNCS 524, 1991.

[BS96]   J. Becker and R. Schütte, editors. *Handelsinformationssysteme*. Verlag Moderne Industrie, Landsberg/Lech, 1996.

[DA04]   J. Dehnert and W. M. P. van der Aalst. Bridging The Gap Between Business Models And Workflow Specifications. *Int. J. Cooperative Inf. Syst.*, 13(3):289–332, 2004.

[DAW05]   B. F. van Dongen, W. M. P. van der Aalst and H. M. W.Verbeek. Verification of EPCs: Using Reduction Rules and Petri Nets. In O. Pastor and J. F. Cunha, editors, *Proc. Advanced Information Systems Engineering (CAiSE)*, LNCS 3520, pages 372–386, Porto, Portugal, June 13-17, 2005. Springer.

[Deh03]   J. Dehnert. *A Methodology for Workflow Modeling - From business process modeling towards sound workflow specification*. PhD thesis, TU Berlin, 2003.

[Kin04]   E. Kindler. On the semantics of EPCs: A Framework for Resolving the vicious circle. In J. Desel, B. Pernici and M. Weske, editors, *Proc. Business Process Management (BPM 2004)*, LNCS 3080, pages 82–97, Potsdam, Germany, 2004. Springer.

[KM94]   G. Keller and S. Meinhardt. *SAP R/3 Analyzer*. Walldorf, 1994. cited after [Sch96].

[KNS92]   G. Keller, M. Nüttgens and A.-W. Scheer. Semantische Prozeßmodellierung auf der Basis Ereignisgesteuerter Prozeßketten (EPK). Technical Report 89, Universität des Saarlandes, Institut für Wirtschaftsinformatik, Saarbrücken, 1992.

[KT98]   G. Keller and T. Teufel. *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping*. Addison-Wesley, 1998.

[MMN06]   J. Mendling, M. Moser and G. Neumann. Transformation of yEPC Business Process Models to YAWL. In *Proc. ACM Symp. on Applied Computing (SAC)*. ACM, 2006.

[MNN05a] J. Mendling, G. Neumann and M. Nüttgens. Yet Another Event-Driven Process Chain. In W. M. P. van der Aalst, B. Benatallah, F. Casati and F. Curbera, editors, *Proc. Business Process Management (BPM 2005)*, LNCS 3649, pages 428–433, Nancy, France, Sep. 5-8, 2005. Springer.

[MNN05b] J. Mendling, G. Neumann and M. Nüttgens. Yet Another Event-driven Process Chain - Modeling Workflow Patterns with yEPCs. *Enterprise Modelling and Information Systems Architectures - an International Journal*, 1(1):3–13, October 2005.

[PW05] F. Puhlmann and M. Weske. Using the π-Calculus for Formalizing Workflow Patterns. In W. M. P. van der Aalst, B. Benatallah, F. Casati and F. Curbera, editors, *Business Process Management (BPM 2005)*, LNCS 3649, 153-168, Nancy, France, 2005. Springer.

[RW87a] P. J. Ramadge and W. M. Wonham. Modular Feedback Logic for Discrete Event Systems. *SIAM Journal of Control and Optimization*, 25(5):1202–1218, 1987.

[RW87b] P. J. Ramadge and W. M. Wonham. Supervisory Control of a Class of Discrete Event Processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[Sch96] A.-W. Scheer. Modellunterstützung für das kosten-orientierte Geschäfsprozessmanagement. In C. Berkau and P. Hirschmann, editors, *Kostenorientiertes Geschäftsprozessmanagement*, pages 3–25. Vahlen, München, 1996.

[Sch00] A.-W. Scheer. *ARIS - Business Process Frameworks*. Springer, Berlin3rd ed. 2000.

[Sch02] K. Schmidt. *Explicit State Space Verification*. Habilitationsschrift, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Dec 2002.

[SD04] C. Simon and J. Dehnert. From Business Process Fragments to Workflow Definitions. In F. Feltz, A. Oberweis and B. Otjacques, editors, *EMISA 2004 - Informationssysteme im E-Business und E-Government*, LNI P-56, pages 95–106, Luxemburg, 2004.

[Sim01] C. Simon. Developing Software Controllers with Petri Nets and a Logic of Actions. In *IEEE Int. Conference on Robotics and Automation, ICRA 2001*, Seoul, Korea, 2001.

[Sim05] C. Simon. Incremental Development of Business Process Models. In *EMISA 2005, Development Methods for Information Systems and their Application*, Klagenfurt, 2005. 222-235.

[Sim06] C. Simon. Integration of Planning and Production Processes. In *Mathmod 2006, Special Session: Petrinets*, Wien, Österreich, 2006.

[SL05] K. Sarshar and P. Loos. Comparing the Control-Flow of EPC and Petri Net from tha End-User Perspective. In W. M. P. Aalst, van der, B. Benatallah, F. Casati and F. Curbera, editors, *Proc. Business Process Management (BPM 2005)*, LNCS 3649, pages 434–439, Nancy, France, Sep. 5-8, 2005. Springer.

[VAH05] H. M. W. Verbeek, W. M. P. van der Aalst and A. H. M. ter Hofstede. Verifying Workflows with Cancellation Regions and OR-joins: An Approach Based on Invariants. Beta Working Paper Series 156, Eindhoven University of Technology, http://fp.tm.tue.nl/beta/publications/working

[ZV99] M. Zhou and K. Venkatesh. *Modeling, Simulation, and Control of Flexible Manufacturing Systems - A Petri net Approach*, volume 6 of *Intelligent Control and Intelligent Automation*. World Scientific, Singapore, New Jersey, 1999.

Praxisbeiträge