

# Advanced Reduction Rules for the Verification of EPC Business Process Models

Jan Mendling<sup>†</sup> and Wil van der Aalst<sup>‡</sup>

<sup>†</sup>Queensland University of Technology  
Level 5, 126 Margaret Street, Brisbane QLD 4000, Australia  
j.mendling@qut.edu.au

<sup>‡</sup>Eindhoven University of Technology  
P.O. Box 513, 5600 MB Eindhoven, The Netherlands  
w.m.p.v.d.aalst@tm.tue.nl

**Abstract:** Conceptual business process models such as Event-driven Process Chains (EPCs) play an important role in the business process management life cycle. The problem in this context is that most of today's commercial business process management tools provide only limited support for quality assurance beyond simple syntax checks. In this paper we focus on verification of behavioral correctness as one of the major quality issues for process models. To be more concise, we introduce advanced reduction rules for EPCs in order to address the requirements of performance and identification of errors in practice. We present the verification tool *xoEPC* that implements the reduction rules. It provides detailed feedback to the modeler where errors are located in the model, and generates a report on quality issues in a process model collection. We present the application of *xoEPC* to the analysis of the SAP reference model to illustrate the tool.

keywords: Business Process Modeling and ERP Systems, Conceptual Modelling, Evaluations of Systems Analysis and Design Modelling Methods and Techniques, Workflow Management

## 1 Introduction

Conceptual business process models such as Event-driven Process Chains (EPCs) play an important role in the business process management life cycle. Since they are often used as early formalizations of process requirements, they should be subject to quality assurance in order to avoid costly rework later in the implementation phase [Moo05, Ros06]. The problem in this context is that most of today's commercial business process management tools provide only limited support for quality assurance beyond simple syntax checks. For process models there is a need to verify behavioral properties like proper completion. Although EPCs are heavily used in practice, their behavioral semantics have been debated for a while (see [Kin06, MA07]) and only a weak notion of correctness called relaxed soundness has been defined [DA04]. Relaxed soundness checks whether there is an option

to proper completion for each task, but does not consider whether proper completion is always guaranteed. Therefore, it does not find all control flow errors. From a quality assurance perspective, there is a need for a comprehensive verification approach for EPCs building on a strict correctness criterion and a formal specification of EPC behavior.

The need for verification is supported by recent studies showing that errors can be found in a large number of process models from practice [MVD<sup>+</sup>08, GL07]. For a company being involved in business process modeling it is important that behavioral correctness of the models is not only verified, but that the modeler is also pointed to the part that causes an error. Furthermore, based on a complete inventory of errors in a business process model collection, a company is able to revise their modeling guidelines and educate their modelers to avoid error-prone patterns in the future. In this paper we address these challenges. We define a set of advanced reduction rules for EPC business process models that can be used to verify even large models. These reduction rules have been implemented in a new verification tool called *xoEPC* that provides detailed feedback about the causes of errors by projecting them onto the EPC.

The remainder of this paper is structured as follows. In Section 2 we introduce EPCs and their behavioral semantics in an informal way. We describe EPC soundness, the correctness criterion that we consider in this work. Section 3 describes a verification approach for EPCs based on reduction rules. The advantage of reduction is not only the performance, but also a precise identification of erroneous parts in the EPC. We describe how our verification tool *xoEPC* prioritizes the reduction rules and illustrate it by the help of an example. In Section 4, we discuss the application of *xoEPC* for the verification of the SAP Reference Model, an extensive model collection describing the business processes supported by the ERP system of SAP AG. Section 5 discusses related work, before Section 6 concludes the paper and gives an outlook on future research.

## 2 EPC Behavior and Soundness

The Event-driven Process Chain (EPC) is a business process modeling language for representing temporal and logical dependencies of activities in a business process [KNS92]. EPCs offer *function type* elements to capture activities of a process and *event type* elements describing pre- and post-conditions of functions. Furthermore, there are three kinds of *connector types* including AND (symbol  $\wedge$ ), OR (symbol  $\vee$ ), and XOR (symbol  $\times$ ) for the definition of complex routing rules. Connectors have either multiple incoming and one outgoing arc (join connectors) or one incoming and multiple outgoing arcs (split connectors). The informal (or intended) semantics of an EPC can be described as follows. The AND-split activates all subsequent branches in concurrency. The XOR-split represents a choice between one of alternative branches. The OR-split triggers one, two or up to all of multiple branches based on conditions. In both cases of the XOR- and OR-split, the activation conditions are given in events subsequent to the connector. The AND-join waits for all incoming branches to complete, then it propagates control to the subsequent EPC element. The XOR-join merges alternative branches. The OR-join synchronizes all active incoming branches. This feature is called non-locality since the state of all transitive

predecessor nodes has to be considered. In terms of connectors EPCs are quite similar to BPMN [OMG06] and YAWL [AH05].

Figure 1 shows an EPC model for a loan request process as described in [NR02]. The start event *loan is requested* signals the start of the process and the precondition to execute the *record loan request* function. After the post-condition *request is recorded*, the process continues with the function *conduct risk assessment* after the XOR-join connector. The subsequent XOR-split connector indicates a decision. In case of a *negative risk assessment*, the function *check client assessment* is performed. The following second XOR-split marks another decision: in case of a *negative client assessment* the process ends with a rejection of the loan request; in case of a *positive client assessment*, the *conduct risk assessment* function is executed a second time under consideration of the positive client assessment. If the risk assessment is not negative, there is another decision point to distinguish new clients and existing clients. In case of an existing client, the *set up loan contract* function is conducted. After that, the AND-split indicates that two activities have to be executed: first, the *sign loan contract* function; and second, the *offer further products* subsequent process (represented by a process interface). If the client is new, the *analyze requirements* function has to be performed in addition to setting up the loan contract. The OR-join waits for both functions to be completed if necessary. If the *analyze requirements* function will not be executed in the process, it continues with the subprocess immediately. The *offer further products* process interface triggers a subsequent process for repeatedly offering products until the offering process is completed.

There are different structural patterns that lead to errors in business process models. They can be traced back to a lack of synchronization and to deadlocks. In the first case, there are too many tokens reaching a connector at the same time, while in the deadlock case a missing token cannot be provided to propagate. For finding errors, we consider EPC soundness as a correctness criterion. EPC soundness requires (1) that there is a set of initial markings such that every start event is included in at least one of these initial markings, (2) that for every marking, that is reachable from a marking in the set of initial markings, there exists a final marking that can be reached, and (3) that final markings are the only markings reachable from a marking in the set of initial markings such that no node can fire [Men07, p.106]. EPC soundness is more precise than relaxed soundness which is the only correctness criterion that has been applied to EPCs so far. EPC soundness builds on a formalization of an EPC as a transition system [MA07], and it can be verified by explicitly calculating the reachability graph. Yet, this approach suffers from the state-explosion problem [Val98]. In the following section we use an approach based on reduction and introduce new powerful reduction rules.

### 3 EPC Reduction Rules

In the section we consider reduction rules for the verification task. The advantage of reduction rules is that one can clearly identify the elements that cause a breach of EPC soundness. As a side effect reduction rules are much more efficient than the calculation of the reachability graph due to the inherent space explosion problem [Val98]. We take

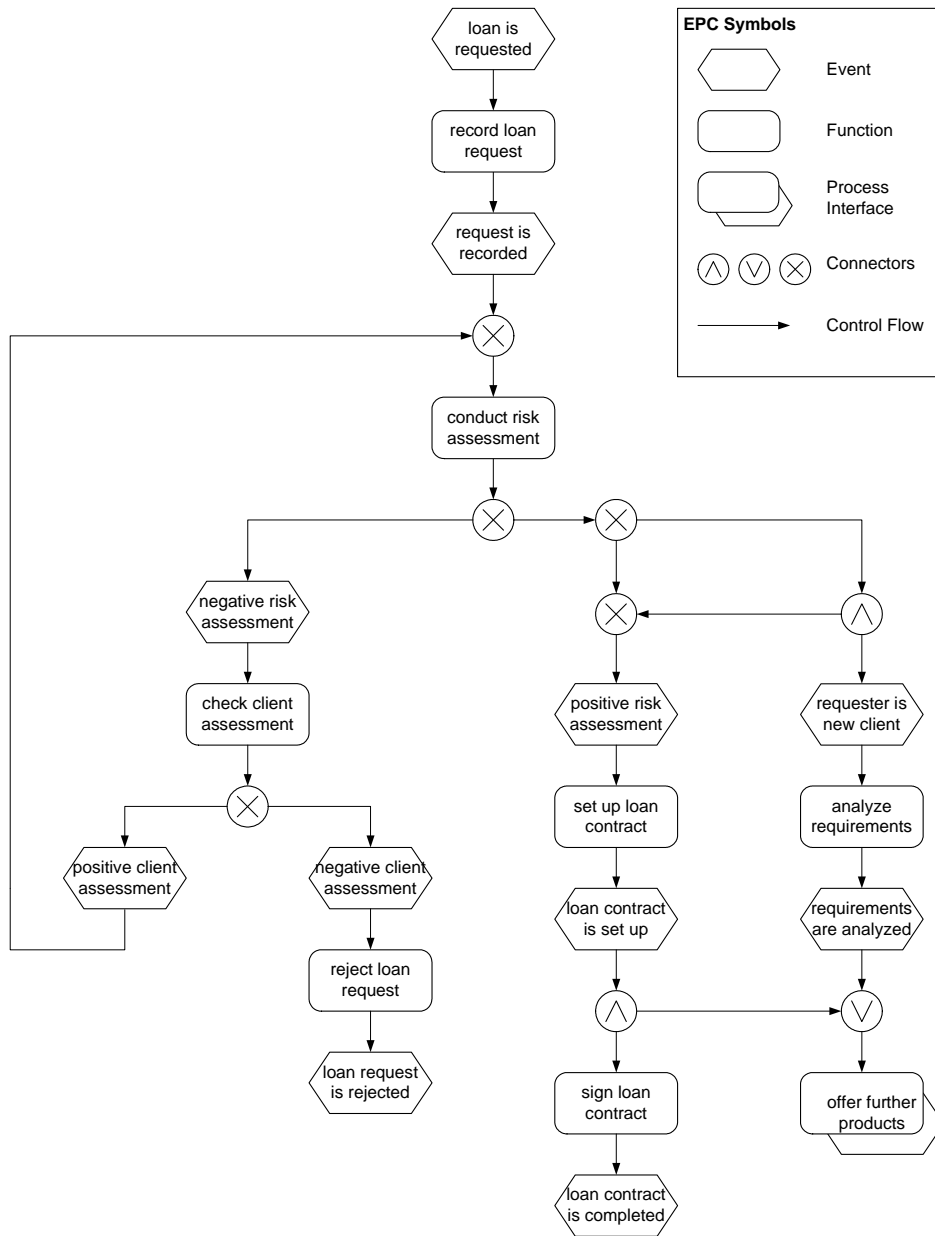


Figure 1: EPC for a loan request process [NR02]

previous work by [DVJVA07] as a starting point for our reduction rule set (Section 3.1) and extend it with new rules (Section 3.2). In this context, a *reduction rule*  $T$  is a binary relation that transforms a source  $EPC_1$  to a simpler target  $EPC_2$  that has less nodes and/or arcs (cf. [Esp94]). A reduction rule is bound to a *condition* that defines for which arcs and nodes it is applicable. For space limitations we refer to [Men07] where we formalize the *construction* of the reduced EPC and *error cases* for several of the rules. Our strategy is to record errors in the reduction process, eliminate the error structure, and continue with further reduction rules. This way, we might be able to find further errors in the remaining model. Since the reduction rules are not complete, we have to further inspect the model using the reachability graph if it is not reduced to the empty model. For details on the preservation of syntactical properties and EPC soundness by the rules we refer again to [Men07].

### 3.1 EPC Reduction Rules by [DVJVA07]

Figure 2 summarizes the reduction rules of [DVJVA07] in four classes.

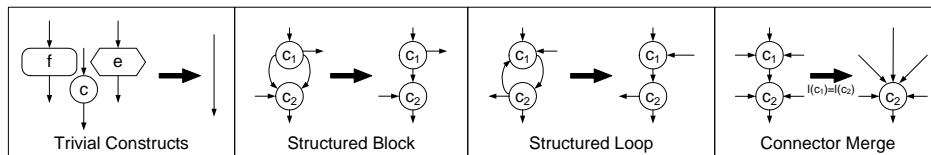


Figure 2: Overview of EPC reduction rules by [DVJVA07]

**Trivial Constructs** The reduction of trivial constructs allows for eliminating sequential nodes from EPC which have one input and one output arc. These nodes do not cause deadlock problems and neither does their removal.

**Structured Blocks** Structured blocks include a split and a join connector with multiple arcs from the split to the join. If the type of both connectors is equivalent, the rule matches the structured component rule by [DVJVA07]. In these cases, it is safe to fuse the parallel arcs. Still, there are four error cases: if the split is an XOR or an OR and the join is an AND, and if the split is an AND or an OR and the join an XOR. In these cases, we record the error and continue searching for further errors in the reduced model.

**Structured Loops** Structured loops include a join as an entry to the loop and a split as exit, with one arc from the join to the split, and one in the opposite direction. If the type of both connectors is XOR, the rule is similar to the XOR-loop rule by [DVJVA07]. In these cases, it is safe to delete the back arc. Still, there are error cases: if the entry is not an XOR, the loop cannot be entered because the entry-join deadlocks. Furthermore, if the exit is not an XOR, the continuously produces tokens

on the same out arc with a lack of synchronization. In both cases, the EPC is not sound and an error is recorded before applying the reduction rule.

**Connector Merge** Consecutive connectors of the same type and the same cardinality (split or join) can be merged. Figure 2 depicts the join-connector case.

### 3.2 Advanced Reduction Rules

Figure 2 summarizes the advanced reduction rules that we introduce in this paper. For a formalization of each rule, refer to [Men07].

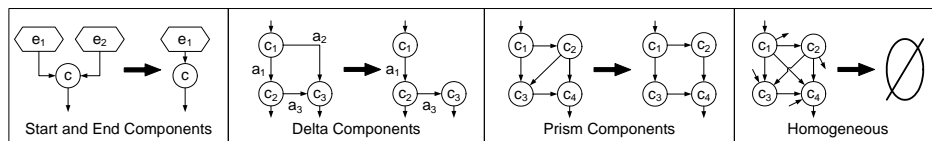


Figure 3: Overview of advanced EPC reduction rules

**Start and End Components** There are three different types of start and end components: structured ones, unstructured ones out of acyclic control flow, and unstructured ones out of cyclic control flow.

- A structured start component contains two start events and one join connector, while a structured end component has one split connector and two end events. In both cases, the second event and the respective arc can be eliminated without affecting the overall soundness.
- Unstructured start and end components model jumps into and out of an acyclic control structure. AND-splits to end events can be eliminated since they cannot cause any errors. (X)OR-splits to end events before AND-joins can result in a deadlock, and it is by no means guaranteed that this deadlock can be avoided. Therefore, we record an error and eliminate the end-event branch responsible for the error.
- Unstructured start and end components on a loop require some specific concepts. The idea here is to consider consecutive pairs of connectors from starts into a loop and pointing out of the loop to ends. The first case causes an error if the connectors are AND-splits or OR-splits, because a repetition of the loop can cause multiple tokens on this loop exit. In the second case an error is reported if the connector is an AND-entry into the loop. A repetition of the loop would require further tokens on the same entry which cannot be provided.

**Delta Components** A delta component contains one input arc to a first split, which is followed by another split and a join in the postset. Furthermore, the preset of the join connector only includes the two splits. There are two output arcs from a delta

component: one from the second split and one from the join. Essentially, there are  $3^3$  versions of delta components for each combination of three connector labels. Each of them belongs to one of three types: split-one reducible, split-two reducible, and non-reducible.

- The split-one reducible deltas can be reduced by deleting the arc between the first split and the join without losing a combination of tokens on the two output arcs. Deltas with both splits being the same connector type and the first being an XOR and the second an OR belong to this group. There are potential deadlock cases when the first split is an (X)OR and the join an AND, and when the first split is an AND or OR and the join an XOR.
- The split-two reducible deltas can be reduced by deleting the arc between the second split and the join without losing a combination of tokens on the two output arcs. This rule is applicable for OR-split as first split and XOR-split as a second. There is an error if the join is not an OR.
- The non-reducible are the rest group, i.e. XOR-split followed by AND-split, AND-split followed by non-AND-split, and OR-split followed by AND-split. There are errors with XOR-split, AND-split, and AND-join; AND-split and XOR-join; AND-split, non-AND-split, and AND-join; and OR-split and non-OR-join.

**Prism Components** A prism component extends a delta component with an additional connector that joins the two output arcs. If this fourth connector is an OR, the arc between the second split and the first join can be deleted without error. As a result the whole prism is collapsed by the trivial construct rule afterwards. If the last join is not an OR, an error is recorded.

**Homogeneous** In some cases, the EPC is trivially correct, for example, if there are only XOR connectors in the model. This rule marks the desirable last reduction step.

### 3.3 *xoEPC*: An Implementation of Reduction Rules

We have implemented the reduction rules of Sections 3.1 and 3.2 as a verification tool called *xoEPC*. This tool prioritizes the rules depending on how easy they can be applied and according to how much information about errors is lost. Accordingly, the trivial construct rule is tried first, and the merge rule last. This is because after a connector merge it is not directly clear whether a detected error is caused by one or the other connector that were merged.

Figure 4 illustrates how *xoEPC* works on the Loan Request EPC from [NR02]. Deleting the *trivial constructs* yields the EPC that is shown in (b). It consists of eight connectors, one start event, two end events, and an end process interface. Applying the *unstructured end component reduction* with the AND-split and the trivial construct reduction results in the EPC depicted in (c). On the right-hand side, there is a delta component that cannot

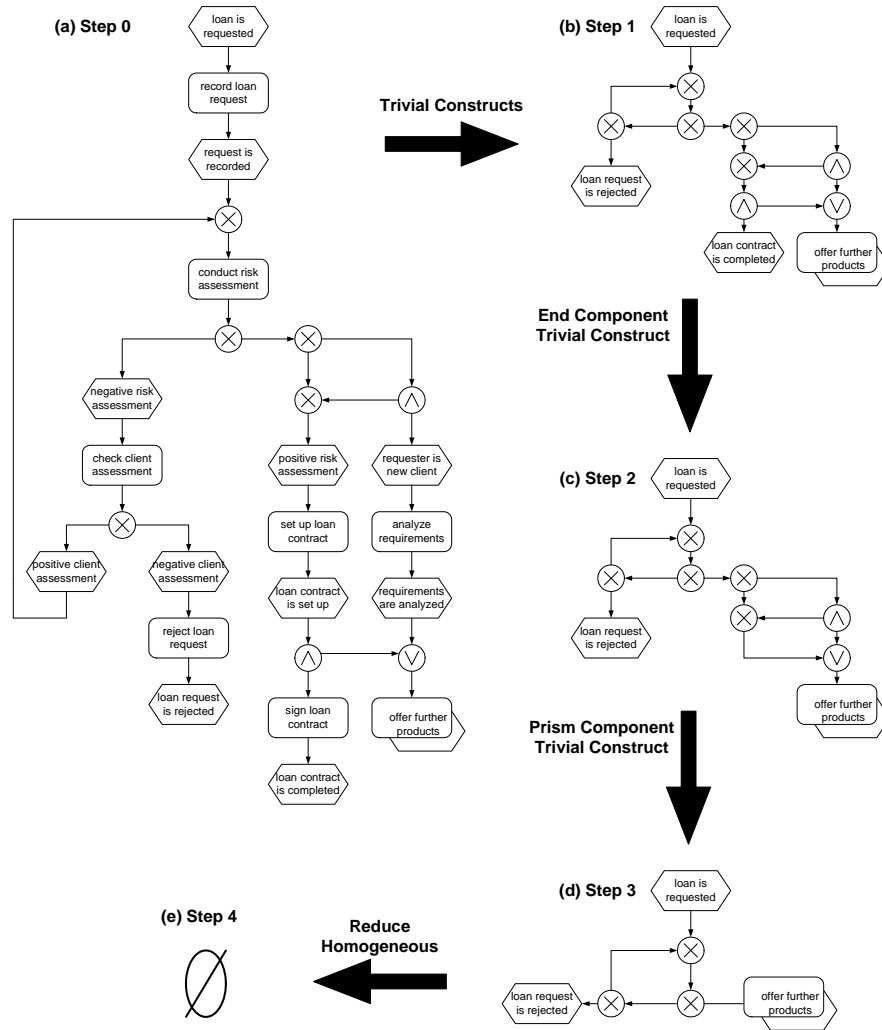


Figure 4: Stepwise reduction of the Loan Request EPC

be reduced, but which, together with the OR-join as a fourth connector, yields a well-behaving prism component. The *prism component reduction* then results in the EPC given in (d). Since there are only XOR-connectors left, the Homogeneous Reduction can be applied to reduce the EPC to the trivial EPC.

## 4 Application to the SAP Reference Model

In order to test the performance of *xoEPC*, we turn to the SAP Reference Model [KT98]. This extensive model collection includes almost 10,000 individual models and describes the business processes supported by the ERP system of SAP AG. 604 of them are EPCs that have at least one event and one function and 600 of them are interpretable, i.e. they do not include functions and events with more than one input or output arc. On the following pages, we discuss the reduction performance from four perspectives: (1) Processing time of reduction, (2) Extent of reduction, (3) Applicability of reduction rules, and (4) Number of errors found. We rely on information that is recorded in the *errorresults.xml* file during the reduction.

**Processing Time of Reduction** The processing of the whole SAP Reference Model took about 18 minutes on a desktop computer with a 3.2 GHz processor. Figure 5 shows how the processing time is distributed over different models. Each EPC model is represented by a number on the horizontal axis ordered by the processing time, i.e. the EPC that was processed the fastest is on the very left position, and the most time-consuming EPC is found on the very right-hand side of the spectrum. Each EPC is assigned its processing time as the ordinate. Please note that the ordinate is given in a logarithmic scale. In Figure 5, it can be seen that about 320 models are processed in less than 1,000 milliseconds, i.e. one second. Furthermore, we see that only a few (16 EPCs) take more than 10 seconds. It is interesting to note that the 13 largest models with more than 80 nodes are also the 13 models that require the most processing time. The number of nodes and the processing time are correlated with a Pearson's coefficient of correlation of 0.592, showing that the performance very much depends on the size of the EPC. The maximum processing time was two minutes and 22 seconds for an EPC with 111 nodes, 136 arcs, and 32 connectors. The performance of the analysis is much better than the performance of the relaxed soundness analysis reported in [MVD<sup>+</sup>08], which took about seven hours and 45 minutes on the same computer, in contrast to less than 30 minutes with reduction rules.

**Extent of Reduction** All original EPCs together include 12,529 nodes, while the reduced EPCs have only 1,118 altogether. The average per model is about 21 nodes before and 1.85 nodes after the reduction. This means that 91% of the nodes are deleted in the various reduction steps. Figure 6 shows the EPCs ordered by size of the reduced model related to the reduced size. Two things can be seen from this figure. 103 of 604 EPCs could not be reduced completely. Furthermore, these 103 EPCs have less than 29 nodes, which is a bit more than what is the average for the unreduced models. Only 15 of them

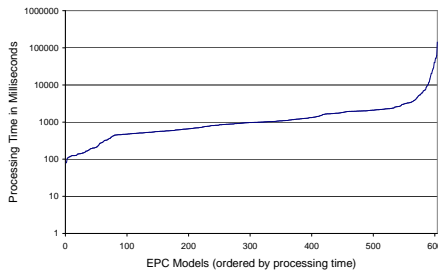


Figure 5: Processing time for reducing an EPC

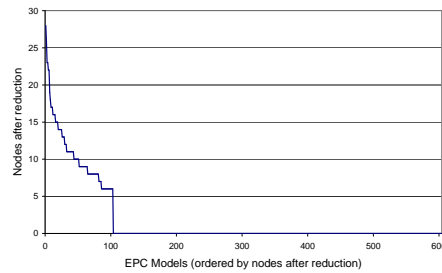


Figure 6: Size of reduced EPCs

have more than 15 nodes.

**Applicability of Reduction Rules** Figure 7 shows how often the eight reduction rules can be applied for the verification for the SAP Reference Model. Not surprisingly, the reduction of trivial constructs is used most often (about 6,600 times). It is followed by the start and end component reduction with about 2,400 applications. This is a good result, considering that large sets of start and end events have been a major verification problem in previous studies [VA06, MVD<sup>+</sup>08]. The structured block rule is the fourth best on terms of the frequency of application with 345 reductions. The homogeneous rule is still applied quite often (460 times), since this rule can only be applied once for an EPC. The remaining four rules are applied less frequently. They still play an important role for the reduction approach as a whole. Running the reduction without the structured loop, delta, prism, and merge rule causes 13 additional models to not be reduced completely, and 53 errors would be missed. Some of the reduced EPCs yield a specific pattern (see [Men07]). These rest patterns could be used for designing additional reduction rules in future research.

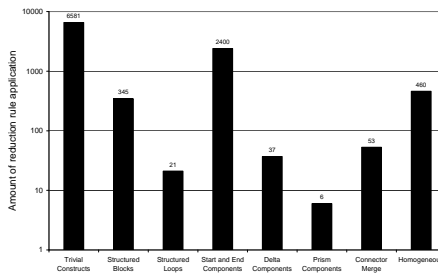


Figure 7: Number of Rule Applications

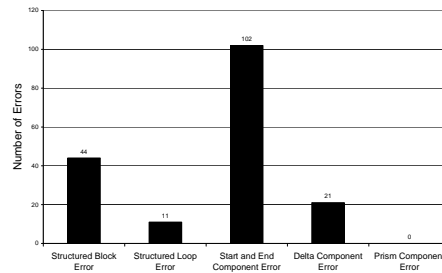


Figure 8: Number of Errors per Type

**Number of Errors found** Prior research on the verification of the EPCs of the SAP Reference Model has shown that there are several formal errors in the models (cf. [ZR96, DVJVA07, MVD<sup>+</sup>08]). In [MMN<sup>+</sup>06], the authors identify a lower bound for the number of errors of 34 (5.6%) using the relaxed soundness criterion. The reduction based on

*xoEPC* identifies 90 models for which altogether 178 errors were found (see Figure 8). This is almost three times as many models as in an earlier study which used the relaxed soundness criterion (see [MMN<sup>+</sup>06]). Furthermore, there are 103 models that are not completely reduced, and for 57 of them no error was found in the reduction phase. We analyzed these models using the reachability graph approach reported in [MA07, Men07]. 68 of the 103 unreduced EPCs were not sound, and 36 unsound EPCs were not detected by the reduction rules. This yields 126 EPCs with errors in total for the SAP Reference Model.

Comparing the application of a rule to the error cases, as depicted in Figure 8, offers some first insight on the question why errors are introduced in models. While 44 errors in structured blocks is often in absolute terms, it is little compared to the 345 times a structured block was reduced. This is different for structured loops and delta components: the relation of errors to no-error cases is about 1:2, i.e. 11 to 21 for loops and 21 to 37 for deltas. It seems as if modelers have more problems with these sophisticated building blocks than with the structured blocks. The start and end components may be regarded as the extreme opposite, with 102 error cases compared to 2,400 applications. Still, this is the highest value in absolute terms and points to a problem with using unstructured start and end events. Surprisingly, the prism is applied six times. Even though it is the reduction pattern with the most nodes, it causes no errors. One explanation could be that modelers are aware that non-trivial components are best joined with an OR.

## 5 Related Work

The research in this paper is motivated by general considerations on quality of conceptual process modeling. As such it relates to work on quality frameworks by [KSJ06, Moo05, BRU00] as it captures a specific correctness criterion, i.e., EPC soundness. The contribution of the paper is related to work on process model verification and reduction rules.

The state explosion problem is one of the motivations for considering reduction rules for Petri nets. A set of six reduction rules that preserve liveness, safeness, and boundedness of a Petri net is introduced in [Ber86, Ber87] and summarized in [Mur89, p.553]. Yet, this set of rules is not complete, i.e., there are live, safe, and bounded Petri nets that cannot be reduced to the trivial model by these rules. For the Petri net class of free choice nets, [Esp94] show that there exists a complete reduction kit including rules for fusion of places and transitions similar to (a) and (b) of [Mur89] and two linear dependency rules to eliminate nonnegative linearly dependent places and transitions. By showing that soundness corresponds to liveness and boundedness of the short-circuited net, [Aal97] makes the reduction kit of [Mur89] applicable to the analysis of workflow nets.

In a different stream of research, [SO00] discuss the applicability of reduction rules for business process models that are defined in a language called workflow graphs. They provide a kit including five rules. [LZLC02] show that this reduction kit is not complete by giving a counter example. They extend the reduction kit to seven rules. In [AHV02] the authors show that the original reduction kit is not complete and use a different approach

building on well-known Petri net results. By providing a mapping to Petri nets, they show that the resulting net is free choice. On the one hand, this makes the complete reduction kit of [Esp94] applicable. On the other hand, basic Petri net analysis techniques and tools can be applied and soundness can be checked in polynomial time.

A set of reduction rules for flat EPCs was first mentioned in [DVJVA07]. The idea is to eliminate those structures of an EPC that are trivially correct for any semantics formalization. [WVA<sup>+</sup>06a] discuss reduction rules for Reset nets, a Petri net class that offers so-called reset arcs which clean tokens from the net. Their reduction kit includes seven rules mainly inspired by [Mur89, Esp94]. Based on a Reset net formalization [WVA<sup>+</sup>06b] define a reduction kit for YAWL. The reduction rules of *xoEPC* are specifically tailored for EPCs. In particular, we introduce novel rules to deal with multiple start and end events as well as the delta and prism component rules. These rules significantly contribute to the reduction of EPCs beyond the rules presented in the above mentioned papers.

## 6 Conclusion and Future work

In this paper we presented *xoEPC*, a tool for verification of EPC business process models and its application to a process model collection from practice, namely the SAP Reference Model. The development of *xoEPC* is motivated by the fact that quality assurance for process models needs a formal foundation and must provide appropriate information for the people involved in process modeling. Our tool *xoEPC* addressed these challenges by (1) building on a comprehensive formalization of EPC behavior and EPC soundness as a notion of correctness, (2) by providing detailed feedback about the causes of errors by projecting them onto the EPC, and (3) by generating an error inventory report that can be used for identifying a necessity for training or for revision of modeling guidelines. Based on the EPC soundness criterion it finds 126 error models in the SAP Reference Model compared to 34 using relaxed soundness.

Currently, our research covers only one single aspect related to the quality of process models. In future research we aim to broaden our perspective. By utilize *xoEPC* in the analysis of other business process model collections from practice we want to study how information generated by *xoEPC* can be used to improve the process of process modeling in the respective organization, which might eventually contribute to establishing process modeling as an engineering discipline beyond current practice where it is often conducted as an art.

## References

- [Aal97] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, LNCS 1248, pages 407–426. 1997.
- [AH05] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.

- [AHV02] W.M.P. van der Aalst, A. Hirschsall and H.M.W. Verbeek. An Alternative Way to Analyze Workflow Graphs. In A. Banks-Pidduck, J. Mylopoulos, C.C. Woo and M.T. Ozsu, editors, *CAiSE'02, Proceedings*, LNCS 2348, pages 535–552. 2002.
- [Ber86] G. Berthelot. Checking Properties of Nets Using Transformations. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, LNCS 222, pages 19–40. 1986.
- [Ber87] G. Berthelot. Transformations and Decompositions of Nets. In W. Brauer, W. Reisig and G. Rozenberg, editors, *Advances in Petri Nets 1986*, LNCS 254, pages 360–376. 1987.
- [BRU00] J. Becker, M. Rosemann and C. von Uthmann. Guidelines of Business Process Modeling. In W.M.P. van der Aalst, J. Desel and A. Oberweis, editors, *Business Process Management*, pages 30–49. 2000.
- [DA04] J. Dehnert and W.M.P. van der Aalst. Bridging The Gap Between Business Models And Workflow Specifications. *International J. Cooperative Inf. Syst.*, 13(3):289–332, 2004.
- [DVJVA07] B.F. van Dongen, M.H. Vullers-Jansen, H.M.W. Verbeek and W.M.P. van der Aalst. Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. *Computers in Industry*, 58(6):578–601, 2007.
- [Esp94] J. Esparza. Reduction and Synthesis of Live and Bounded Free Choice Petri Nets. *Information and Computation*, 114(1):50–87, 1994.
- [GL07] V. Gruhn and R. Laue. What business process modelers can learn from programmers. *Science of Computer Programming*, 65(1):4–13, 2007.
- [Kin06] E. Kindler. On the semantics of EPCs: Resolving the vicious circle. *Data & Knowledge Engineering*, 56(1):23–40, 2006.
- [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.
- [KSJ06] J. Krogstie, G. Sindre and H.D. Jørgensen. Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, 15(1):91–102, 2006.
- [KT98] G. Keller and T. Teufel. *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping*. 1998.
- [LZLC02] H. Lin, Z. Zhao, H. Li and Z. Chen. A Novel Graph Reduction Algorithm to Identify Structural Conflicts. In *HICSS-35 2002, Track 9*. 2002.
- [MA07] J. Mendling and W.M.P. van der Aalst. Formalization and Verification of EPCs with OR-Joins Based on State and Context. In J. Krogstie, A.L. Opdahl and G. Sindre, editors, *CAiSE 2007, Proceedings*, LNCS 4495, pages 439–453. 2007
- [Men07] J. Mendling. *Detection and Prediction of Errors in EPC Business Process Models*. PhD thesis, Vienna University of Economics and Business Administration, 2007.
- [MMN<sup>+</sup>06] J. Mendling, M. Moser, G. Neumann, H.M.W. Verbeek, B.F. van Dongen and W.M.P. van der Aalst. Faulty EPCs in the SAP Reference Model. In J.L. Fiadeiro S. Dustdar and A. Sheth, editors, *BPM 2006, Proceedings*, LNCS 4102, page 451457. 2006.

- [Moo05] D.L. Moody. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering*, 55(3):243–276, 2005.
- [Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
- [MVD<sup>+</sup>08] J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst and G. Neumann. Detection and Prediction of Errors in EPCs of the SAP Reference Model. *Data & Knowledge Engineering*, 64(1):312–329, 2008.
- [NR02] M. Nüttgens and F.J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In J. Desel and M. Weske, editor, *Promise 2002, Proceedings*, LNI 21, pages 64–77. 2002.
- [OMG06] OMG, ed. Business Process Modeling Notation (BPMN) Specification. Object Management Group, February 2006.
- [Ros06] M. Rosemann. Potential pitfalls of process modeling: part A. *Business Process Management Journal*, 12(2):249–254, 2006.
- [SO00] W. Sadiq and M.E. Orłowska. Analyzing Process Models using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.
- [VA06] H. M. W. Verbeek and W.M.P. van der Aalst. On the verification of EPCs using T-invariants. BPMCenter Report BPM-06-05, 2006.
- [Val98] A. Valmari. The State Explosion Problem. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, LNCS 1491, pages 429–528. 1998.
- [WVA<sup>+</sup>06a] M.T. Wynn, H.M.W. Verbeek, W.M.P. van der Aalst, A.H.M. ter Hofstede and D. Edmond. Reduction Rules for Reset Workflow Nets. BPMCenter Report BPM-06-25, 2006.
- [WVA<sup>+</sup>06b] M.T. Wynn, H.M.W. Verbeek, W.M.P. van der Aalst, A.H.M. ter Hofstede and D. Edmond. Reduction Rules for YAWL Workflow Nets with Cancellation Regions and OR-Joins. BPMCenter Report BPM-06-24, 2006.
- [ZR96] O. Zukunft and F.J. Rump. *Business Process Modelling*, chapter From Business Process Modelling to Workflow Management: An Integrated Approach, pages 3–22. 1996.