Relaxed Soundness of Business Processes

Juliane Dehnert[1,*] and Peter Rittgen[2]

[1] Institute of Computer Information Systems, Technical University Berlin, Germany
dehnert@cs.tu-berlin.de

[2] Institute of Business Informatics, University Koblenz-Landau, Germany
rittgen@uni-koblenz.de

Abstract. Business processes play a central role in the reorganization of a company and the (re)design of the respective information system(s). Typically the processes are described with the help of a semiformal, graphical language such as the Event-driven Process Chains (EPCs) by Scheer. This approach provides a suitable medium for the communication between the participants: the domain experts and the IT specialists. But these models leave room for interpretation and hence ambiguities which makes them less suitable as a basis for the *design* of information systems. To remedy this we suggest to transform the EPCs into a formal representation (Petri nets) *preserving the ambiguities*, i.e. all possibly intended behaviour. Now formal techniques can be used to find out whether the *possible* behaviours comprise *sensible* behaviour. If so, we call the net *relaxed sound*. By not limiting the modeler compared to previous ways (e.g. [8], [3]) we take a pragmatic approach to correctness which only requires that the net represents *some* valid behaviour. This allows us to draw conclusions on mistakes in the original EPC and to make suggestions for its improvement thereby enhancing both the model's quality and its suitability for software engineering.

1 Motivation

Business processes play a central role in the reorganization of a company and the (re)design of the respective information system(s). Typically the processes are described with the help of a semiformal, graphical language such as the Event-driven Process Chains (EPCs) by Scheer [15]. Approaches of this type are suitable for the analysis phase of an IT project where the focus is on communication: reaching an agreement on how the process should look like between participants with totally different backgrounds and "knowledge cultures": CEOs, heads of department, department staff, IT experts and so on. In this phase it is imperative that the language used represents the greatest common denominator of the people involved. And more than that it should leave room for interpretation: the more ways there are to interpret a certain construct the more likely it is that an agreement is reached. The participants might not (yet) be ready to specify the "final" behaviour in detail and decide for the "correct" interpretation. But although this feature is desirable in the analysis phase of IS development it constitutes a major problem in the design phase where we need an

* This work is supported by Deutsche Forschungsgemeinschaft (reference WE 1214-3-3a, research group Petri Net Technology)

K.R. Dittrich, A. Geppert, M.C. Norrie (Eds.): CAiSE 2001, LNCS 2068, pp. 157–170, 2001.

unambiguous description of the process. To remedy this problem we suggest to transform the EPCs into a formal representation (Petri nets) while *preserving the ambiguities*, i.e. all possibly intended behaviour. For this we use the workflow nets by van der Aalst based on which we can now employ established formal techniques to determine the correctness of the process in a pragmatic fashion: we analyse the net and try to find, among all the possible ways of behaviour, only the ones that are suitable, i.e. where the final state can be reached from the start state so that nothing blocks or remains undone. If enough of such execution paths exists we are satisfied and call the respective net *relaxed sound*. The execution paths leading to undesired behaviour can then be used to infer potential mistakes in the original EPC and to suggest possible improvements. An alternative approach might be to reduce the net to the well-behaved paths automatically and to use the resulting net directly as a basis for the design assuming that the EPC is of sufficient quality or a revision to costly. If the workflow net is not relaxed sound we can still do something: we can point out the parts of the EPC that lead to the net not being relaxed sound so that the modeler(s) know(s) where revision should take place. All in all the sketched approach allows us to enhance both the model's quality and its suitability for software engineering.

2 EPCs for Modeling Business Processes

Business processes have been at the heart of IS research for many years if the evidence of many publications concerned with this topic is anything to go . As a result, the amount of different approaches is equally high: IDEF, RAD, ARIS/EPC and Oracle Designer to name but a few. Despite this fact, one of these approaches plays a more predominant role, especially in practice, namely the Event-driven Process Chains (EPCs) of the Architecture of integrated Information Systems (ARIS) described in [15]. The reasons for this prevalence are manifold: on the pragmatic side, a commercial tool for EPCs (ARIS toolset) has been available for quite some time already. In addition, the great success of the company SAP suite of business applications tremendously promoted the use of this method. On the other side, EPCs have also been investigated quite thoroughly in research.

Nevertheless, there is still some argument concerning the suitability of EPCs for modeling business processes that are to be supported by an information system. Advantages such as being highly flexible and easy to learn and understand are compensated by significant disadvantages: first of all ambiguity and vagueness. It cannot be in the interest of the user if the processes described in the specification are interpreted differently by the designer. When this misunderstanding is discovered by the user it is often too late to correct the design accordingly. Where is the path that leads out of this dilemma and towards a better understanding between the participants in the software development process? We think the answer lies in a pragmatic interpretation of the correctness of an EPC as we will show in the following sections.

The language of EPCs provides the user with a set of graphical notation elements for the representation of (business) functions, events and routing constructs to describe the control flow. Functions are used to model the dynamic part of the process. Typical functions are procurement, quality assurance or processing an invoice. They are decomposed hierarchically in a separate diagram, the so-called function diagram. The behaviour of each such function is modeled as a complete EPC. Another constructive element is the event. An event either triggers a function or

marks the termination of it. For example, the event *not_ok* triggers the function *complaint* whereas the event *data revised* marks the termination of *complaint*. Furthermore, to describe more complex behaviour such as sequential, conditional, parallel, and iterative routing, connectors are introduced. These fall into two categories: splits and joins. In both we have AND, XOR and OR connectors.

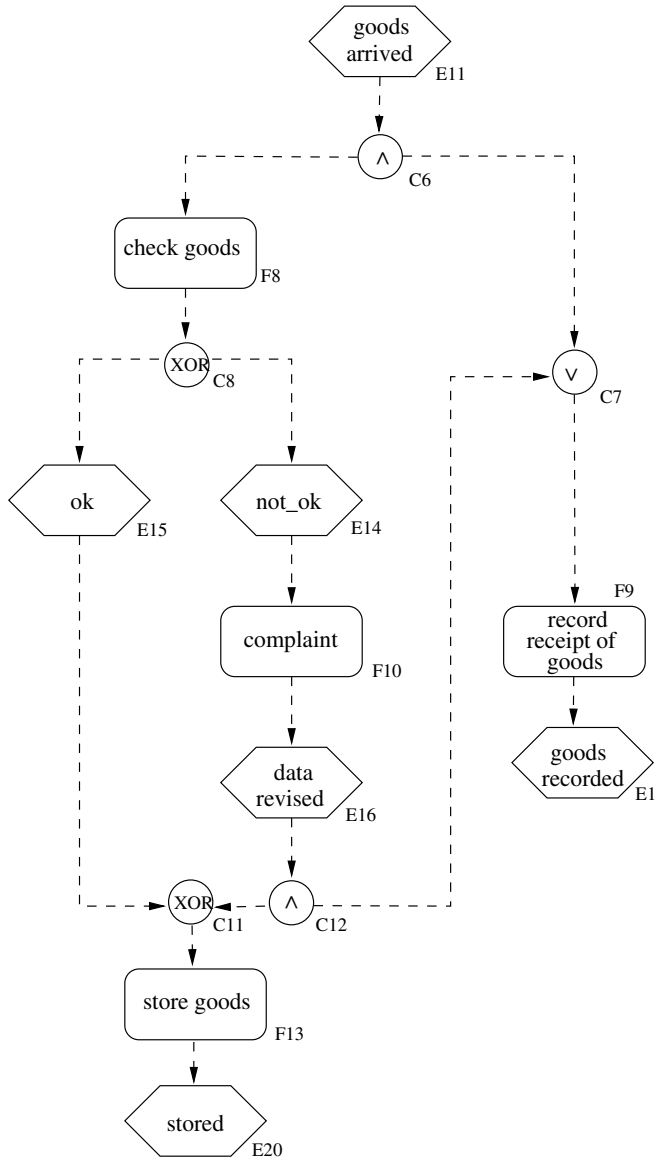

Fig. 1. Handling of incoming goods

Fig. 1 shows an EPC modeling the process "Handling of incoming goods" introduced by [8]. The process starts with the event *goods arrived*. After that the execution is split into two parallel paths (AND split *C6*), the left one checking the goods and performing the ensuing functions, the right one doing the accounting. The result of *check goods* is either *ok* or *not_ok*. In the latter case a complaint is compiled, in the former nothing happens. In either case (XOR join *C11*) the goods are stored afterwards (*store goods*). Connectors *C7* and *C12* make sure that in case of a complaint the corrected data is waited for before the receipt of goods is recorded. Otherwise the receipt can be recorded straight away.

EPCs are a semiformal method for business process modeling. Although they have been utilized quite successfully there authors did neither define a comprehensive and consistent syntax nor a corresponding semantics. A first formal description of EPCs was given in [6] (syntax) and [4] (semantics) but only for a restricted subset of EPCs. Other approaches to formalization have been developed by [14], [3], [13] and [9]. In this paper we refer mainly to the approach of [3]. Here syntax and semantics of an EPC have been described formally. The definition of an EPC includes:

- Events and functions have exactly one incoming and one outgoing arc (except start and end events).
- There are no isolated nodes.
- Connectors are either splits or joins.
- An event is always followed by a function and vice versa (modulo connectors).

As in most of the approaches mentioned the semantics of an EPC is defined by a mapping onto Petri nets. Petri nets are used because they have a clear and precise definition [10] and a similar graphical notation to that of EPCs. In addition they give us access to many existing analysis techniques and tools. The approach presented in [3] is based on the classical Petri net whereas [8], [13] and [9] use high-level variants. Alternatively we could define a formal semantics for EPCs and then define the criteria of section 4 directly on EPCs. Work to that effect is in progress.

In this paper EPCs are transformed into workflow nets but contrary to [3] we allow for ambiguity as e.g. introduced by the incorporation of the OR connector. We do this to increase the flexibility of the modeling method in the early phase of software engineering, i.e. to provide room for interpretation. This is necessary to foster the integration of the incompatible views on the common domain held by the heterogeneous parties involved in this process. Moreover it requires less modeling expertise and a less precise knowledge of the domain. Hence contrary to all existing approaches we do not resolve ambiguity. Instead, we ensure that the model is reasonable in spite of ambiguity, i.e. that it covers some reasonable behaviour that can be used as a basis for either the revision of the EPC or the design of the information system. In the following section we show how EPCs are transformed into workflow nets which are then used to assert certain properties of the EPC, most notably soundness and relaxed soundness.

3 Transformation into Workflow Nets

Workflow nets (WF nets) have been introduced by van der Aalst [2] applying Petri-Net theory to the specification of workflow processes. A WF net is a Petri net which

has a unique source place (*i*) and a unique sink place (*o*). In addition a WF net requires all nodes (i.e. transitions and places) to be on a path from i to o. This ensures that every task (transition) and every condition (place) contribute to the business process.

In this paper we use the definitions of a Petri net and WF net from [2], namely:

Definition (Petri net). A Petri net is a triple (P, T, F) where:

- P is a finite set of places,
- T is a finite set of transitions,
- $F \subset (P \times T) \cup (T \times P)$ is a set of arcs (flow relation).

The function M: $P \rightarrow N$ is called marking of PN. M(p) is the number of tokens contained by the place p for the marking M. A transition t is said to be enabled by a marking M iff each input place p of t contains at least one token. An enabled transition may fire. If transition t fires then t consumes one token from each input place and produces one token in each output place. Note that the term $M_1 \xrightarrow{t} M_2$ means that the firing of transition t takes the process from state M_1 to state M_2. $M_1 \xrightarrow{*} M_2$ indicates that there is some firing sequence of transitions that leads from state M_1 to state M_2. We use (PN, M_0) to denote a Petri net with an initial state M_0. A state M' is reachable in (PN, M_0) iff $M_0 \xrightarrow{*} M'$.

Definition (strongly connected). A Petri net is strongly connected if and only if for every pair of nodes (i.e. places and transitions) x and y there is a path leading from x to y.

A WF net is a special Petri net defined as follows.

Definition (WF net): A Petri net PN = (P, T, F) is a WF net if and only if:

i. PN has two special places: *i* and *o*. Place *i* is a source place and place *o* is a sink place.
ii. If we add a transition *t** to PN which connects place *o* with *i* then the resulting Petri net is strongly connected.

The transformation of EPCs into Petri nets takes place in two steps. First we map elements of the EPC onto Petri net-modules. In the second step we provide rules to combine the different modules to form a complex process model. Our set of transformation rules is shown in Fig. 2.

Events and functions are transformed into places and transitions respectively including in- and outgoing arcs. Routing constructs such as AND split, AND join, XOR split, XOR join, OR split and OR join are mapped to small Petri net-modules. The Petri net-modules describe the behaviour of the routing constructs explicitly. This is primarily relevant for the OR, because its semantic has not been described consistently.

In Fig. 3 we see an EPC with an OR join on the left and its Petri net translation on the right side. The EPC as well as the Petri net have the semantics that E can be reached if either F_1 or F_2 or both occur. In the EPC all these different cases are described through one connector whereas in the Petri net-module all possibilities are modeled explicitly via the transitions t_a, t_b and t_c. The behaviour of the EPC and the

Petri net are equivalent, because both accept the same executions. Note that the case that E is reached twice if F_1 and F_2 occur sequentially has not been excluded.

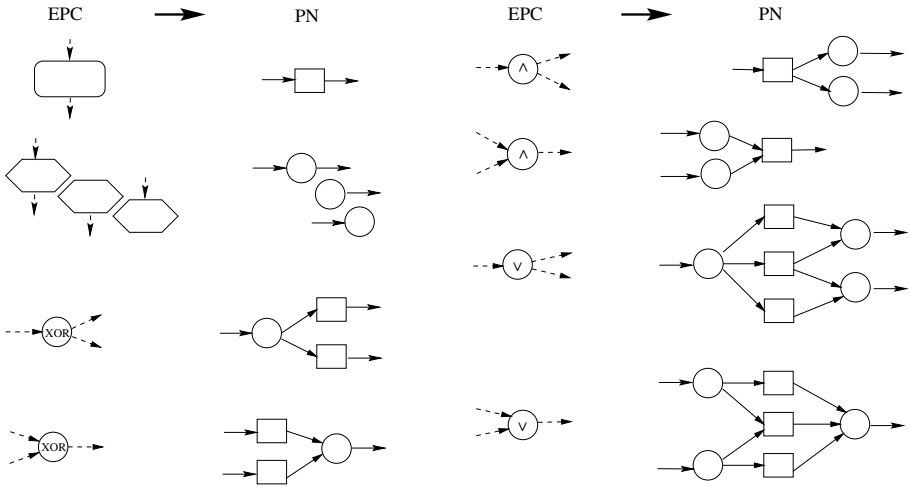

Fig. 2. Transformation rules for an EPC into a place/transition net (rule 1)

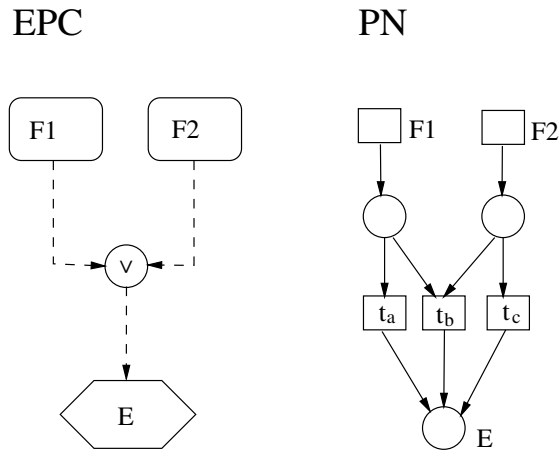

Fig. 3. Transformation of the OR-Connector

To form a coherent Petri net the single modules are (automatically) connected as follows (rule 2):

a) if input and output elements are different (place and transition) then the arcs are fused

b) if input and output elements are of the same kind (e.g. both places) then the different nodes are unified.

Fig. 4 illustrates the transformation of EPCs into place/transition nets.

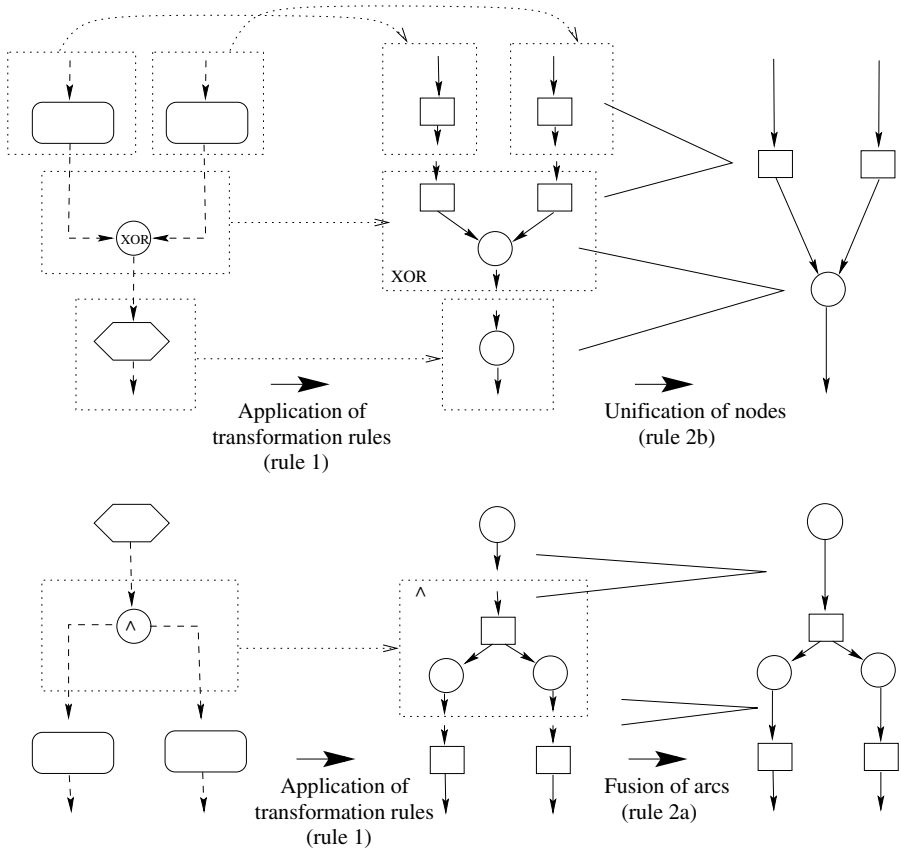

Fig. 4. Illustrating the transformation

The proposed transformation approach is slightly more general than the transformations described in [13] and [3]. The rules presented here can also be applied to transform EPCs where connectors follow each other immediately as e.g. *C6*/*C7* in Fig. 1.

Another advantage of this approach is that the resulting Petri net is minimal in the sense that it does not contain places or transitions not corresponding to elements of the EPC. The transformation rules by [13], [8] and [3] all contain rules which explicitly introduce new pseudo places and transitions to meet the Petri net syntax. The resulting Petri net may contain many elements which have no counterpart in the application domain.

To transform an EPC into a WF net something more has to be done. A WF net has exactly one start and one sink place. One problem when modeling with EPC is caused by an unclear concept of start and end events. There is no rule that restricts the

amount of start and end events. A start (end) event is defined as an event without an incoming (outgoing) edge. Furthermore it is not clear whether the start (end) events are mutually exclusive. So translating the EPC into a Petri net does not necessarily lead to a Petri net with exactly one start and one sink place. In this case one further transformation step is required to yield a WF net. We add a new start place and a new sink place and connect them to Petri net-modules which initialize (clean up) the places representing the start and end events of the EPC in the right manner. The module introduced complements the first (last) connector on the paths from the start (end) events. For further particulars we refer the reader to [13] where this rule (rule 3) has been introduced and to the example below.

Applying the proposed rules 1 to 3, an EPC is transformed into a WF net. This transformation is unique, in the sense that to each EPC belongs exactly one WF net. An example for such a transformation is shown in Fig. 5. Here the EPC from Fig. 1 has been transformed into a WF net. For convenience we surrounded the Petri net-modules which correspond to the routing constructs of the EPC with dotted rectangles.

Transition $t10\_{}_{AND\text{-}Join}$ and the sink place *o* have been added due to rule 3. Transition $t10\_{}_{AND\text{-}Join}$ corresponds to an AND connector which complements the last connector on the paths from the end events *E12* and *E20*, namely connector *C12*. Transition $t10\_{}_{AND\text{-}Join}$ bundles the different path and leads to the sink place *o*.

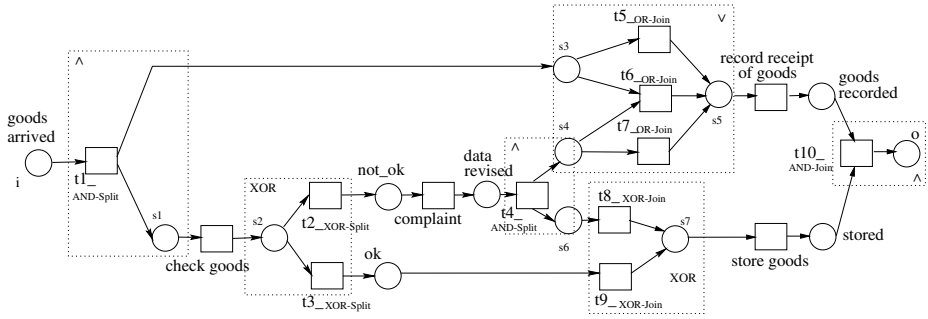

Fig. 5. WF net "handling of incoming goods"

Let us have a closer look at the Petri net-module which replaces the OR join *C7*. The Petri net-module makes the behaviour of this routing construct explicit. Transition $t5\_{}_{OR\text{-}Join}$ models the "straight away recording" and transition $t6\_{}_{OR\text{-}Join}$ models the waiting for the revision to be completed. The alternative $t7\_{}_{OR\text{-}Join}$ has been introduced as part of the corresponding Petri net-module, but has no expression in the original EPC. This alternative can not be chosen in the EPC, because of the AND-connector *C6* before.

By transforming the OR connector we carry the ambiguity of the OR to the WF net. The decision whether to execute transition $t5\_{}_{OR\text{-}Join}$, $t6\_{}_{OR\text{-}Join}$ or transition $t7\_{}_{OR\text{-}Join}$ can not be resolved locally anymore.

WF nets are a class of Petri nets for which theoretical results and efficient analysis techniques exist (cf. [1]). In the following section we will apply different criteria to check the workflow net and therefore the corresponding EPC for correctness.

4 Soundness and Relaxed Soundness

In this section we will introduce the properties of soundness and relaxed soundness as a means both to check the quality of the underlying EPC and to help with the revision of the business process if necessary. We subsequently discuss their applicability.

4.1 Soundness

Van der Aalst [1] introduced soundness as a correctness criterion for workflow nets. He argues that this criterion covers a minimal set of requirements a process definition should satisfy. Soundness ensures that the process can **always** terminate with a single token in place *o* and all the other places are empty. In addition, it requires that there is no dead task, i.e. any task can be executed. The following definition of soundness is taken from [1].

Definition (Soundness). A process modeled by a WF-net PN = (P,T,F) is sound if and only if:

i. For every state *M* reachable from the initial state *i* (one token in place *i*) there exists a firing sequence leading from state *M* to state *o*. Formally: $\forall M : \left(i \xrightarrow{*} M\right) \Rightarrow \left(M \xrightarrow{*} o\right)$

ii. State *o* is the only state reachable from state *i* with at least one token in place *o*. Formally: $\forall M : \left(i \xrightarrow{*} M \wedge M \geq o\right) \Rightarrow (M = o)$ (proper termination)

iii. There are no dead transitions in PN with initial marking *i*. Formally: $\forall t \in T : \exists M, M' : \left(i \xrightarrow{*} M \xrightarrow{t} M'\right)$

A WF net is sound if the process terminates properly in any case, i.e. termination is guaranteed and there are no spare tokens and neither deadlock nor livelock. Spare token signalize that some information was not used during execution, whereas dead- and livelock indicate situations where the execution got stuck respectively no real progress could be reached anymore.

We consider an EPC to be sound if its corresponding WF net is sound.

The WF net in Fig. 5 is not sound. There are firing sequences that do not terminate properly, e.g. the sequence: t1 $_{\text{AND-Splt}}$, check goods, t2 $_{\text{XOR-Split}}$, complaint, t4 $_{\text{AND-Split}}$, t5 $_{\text{OR-Join}}$, record receipt of goods, t8 $_{\text{XOR-Join}}$, store goods, t10 $_{\text{AND-Join}}$.

In order to make the process sound the model has to be changed in such a way that the execution paths are restricted to sound firing sequences only where a sound firing sequence is one that terminates properly. To achieve this we have to avoid spare tokens as well as livelock and deadlock situations. These problems are, among others, the result of the incorporation of the OR connector.

Resolving the Ambiguity of the OR Connector

The ambiguous meaning of the OR connector has been discussed extensively in most formalization approaches. There are almost as many solutions as approaches.

In [11] the ambiguity of the OR connector is handled through a syntax extension on the side of the EPC. The connectors are extended by comment flags which describe the desired behaviour explicitly (wait-for-all, first-come, every-time). Wait-for-all means that the OR join waits for all paths that have been activated by the complementing opening split which the modeller has to identify as such. In the first-come (every-time) case the OR join triggers on the first path (on every path) that is completed. The first-come ignores the termination of the remaining paths. Hence the latter two do not require a complementing split. Note that this approach forces the modeler to resolve the ambiguity which he might not want to do as we already pointed out earlier.

[13] and [9] resolve the ambiguity adding places (communication channels) to the Petri net. Their task is to keep the information about the choice made by the OR split. This information is used to synchronize the corresponding OR join accordingly. [4] and [8] introduce different tokens for the same reason. All approaches mentioned impose the requirement to model in a well-structured way, i.e. every split has to be complemented by a corresponding join. Modeling with well-structured EPCs restricts the modeler considerably in his/her expressiveness and it also poses substantial requirements on the modeling expertise. Modeling with well-structured EPCs is based on a strict top-down design process which can hardly be enforced in practice.

Apart from this elements are introduced to synchronize parallel threads. Synchronization always serializes the execution. Suppose the probability that the system ends in an inconsistent state is very small. It may then be more efficient to recover (seldom) than to wait (every time). The introduction of synchronization forces the designer to think about efficiency aspects of the execution already during the modeling. Moreover it is generally problematic to introduce additional elements to the net because it thereby potentially diverges from the semantics of the original EPC. Often this leads to models which are quite different from the primary specification which in turn requires revision. This revision requires communication with the users which is complicated by the fact that the changes are often motivated by technical requirements only and have no counterpart in the application domain. Let us consider again the example “Handling of incoming goods”.

We will change the WF net of Fig. 5 in such a way that a sound WF net is obtained (see Fig. 6). Firstly, transition $t7_{\_OR\text{-}Join}$ is removed. As discussed earlier it has no meaning in the original EPC and can therefore be removed without any consequences for the EPC. Then we introduce a place S_1 which takes care that transition $t5_{\_OR\text{-}Join}$ only executes if the *credit check* was *ok*. Through this construct the different threads (recording and check) have been synchronized and therefore serialized. This behaviour is not required in the original EPC. To change the EPC accordingly it has to be decomposed and composed again in a well-structured way. This change is not trivial and results in a completely different looking and behaving EPC.

Hence (strict) soundness demands that the modeler either restricts himself/herself to well-structured EPCs right from the start or he/she has to hazard the consequences of a substantial and costly revision. EPCs are a graphical modeling method which is used in the early analysis phase of an IT project where the focus is on communication. The goal of its use is reaching an agreement on how the process should look like

between participants with totally different background and "knowledge cultures". The resulting models are only the base for further investigation. The applied correctness criteria should reflect the modeling knowledge and should therefore not be too strict. It should allow the modeler to postpone the more precise specification as long as possible i.e. to shift to later phases such as design and implementation.

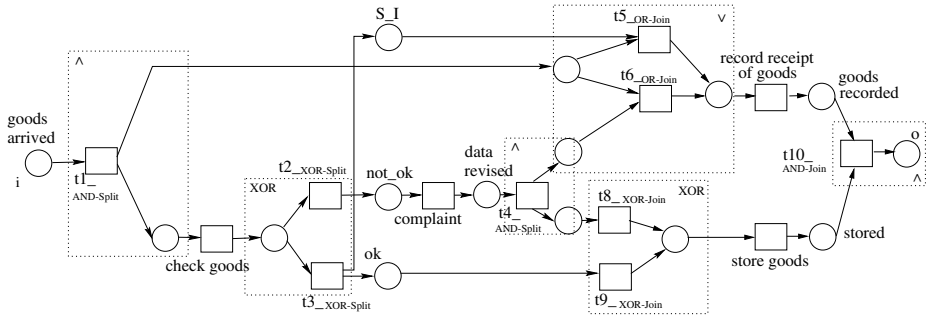

Fig. 6. Extended WF net

Hence we suggest to introduce a new relaxed soundness criterion replacing the strict version.

4.2 Relaxed Soundness

We propose to relax the soundness criterion to the new criterion *relaxed soundness* which has been introduced in [5]. Relaxed soundness is intended to represent a more pragmatic view on correctness which is weaker (in a formal sense) but more easily applicable to application-oriented modeling. It does not require to avoid situations with spare tokens or livelocks/deadlocks. It is therefore suitable to check WF nets which have been derived through the transformation of (not necessarily well-structured) EPCs containing OR connectors.

The idea behind relaxed soundness is that for each transition there exists a sound firing sequence, i.e. a sequence that takes the initial state *i* to the final state *o*. No spare tokens should be left in the Petri net in this case.

Definition (Relaxed sound).

A process specified by a WF net PN = (P, T, F) is relaxed sound if and only if every transition is in a firing sequence that starts in state *i* and ends in state *o*. Formally: $\forall t \in T : \exists M, M' : \left(i \xrightarrow{*} M \xrightarrow{t} M' \xrightarrow{*} o\right)$

Intuitively relaxed soundness means that there exist enough executions which terminate properly (i.e. without spare tokens). Enough means at least so many that every transition is covered. We argue that this criterion is closer to the intuition of the modeler. It does not force the modeler to think about all possible executions and then to care for proper termination in all cases. In spite of that relaxed soundness is still reasonable because it requires that at least all intended behaviour has been described correctly. Note that (strict) soundness implies relaxed soundness.

In terms of the EPC this means that every function can be executed reaching a desired set of end events. If the WF net is not relaxed sound then we have transitions that are not contained in any sound firing sequence. Hence the corresponding part in the EPC needs improvement. Put in other words: as a general rule we have to consider transitions that are not contained in some sound firing sequence when we are looking for parts of the process that need revision.

Let us now check the net in Fig. 5 for relaxed soundness. For this purpose we have to find a sound firing sequence for every transition. The check for relaxed soundness has been automated. The criterion can be checked with the help of the Petri net tool LoLA (**Lo**w **L**evel Petri Net **A**nalyzer) that has been implemented at the Humboldt University of Berlin [7]. It includes features such as: analysis of reachability of a given state and finding dead transitions. Recently LoLA has been extended to prove for extended computation tree logic formulae (eCTL) [12]. Within eCTL it is possible to quantify not only over states but also over state transitions. The combination of Petri nets and eCTL allows to check for relaxed soundness: for each transition *t* the reachability of the end state *o* is verified while it is required to include transition *t* in the path from *i* to *o*.

So far relaxed soundness can be proven only by enumeration of enough sound firing sequences. As far as the authors can judge there are no structural properties such as lifeness and boundness for soundness from which the relaxed soundness property can be derived.

The net in Fig. 5 is not relaxed sound because there is no sound firing sequence containing transition $t7_{\text{OR-Join}}$. As we pointed out earlier transition $t7_{\text{OR-Join}}$ can be left out of the WF net without loss of semantics regarding the EPC. Then the net is relaxed sound. The following two sound firing sequences contain all remaining transitions:

- $t1_{\text{AND-Split}}$, check goods, $t2_{\text{XOR-Split}}$, complaint, $t4_{\text{AND-Splt}}$, $t6_{\text{OR-Join}}$, record receipt of goods, $t8_{\text{XOR-Join}}$, store goods, $t10_{\text{AND-Join}}$
- $t1_{\text{AND-Split}}$, check goods, $t5_{\text{OR-Join}}$, $t3_{\text{XOR-Split}}$, $t9_{\text{XOR-Join}}$, record receipt of goods, store goods, $t10_{\text{AND-Join}}$

From this we can conclude that the EPC represents reasonable behaviour and can hence be used as a basis for software design.

5 Conclusion

We started with the assumption that business processes play a central role in reorganizing a company and (re)designing its information system(s). In doing so we typically describe the processes with the help of some semiformal, graphical language such as the Event-driven Process Chains. Modeling with EPCs usually involves ambiguities which, seen as “room for interpretation”, are necessary in the early stage of analyzing a business. But for the later stages of software development we must identify the useful interpretations. This notion is formalized in our paper in terms of the relaxed soundness criterion. To make use of this criterion we first have to transform the EPC into a workflow net. This is done applying a fixed set of rules. The transformation does not resolve ambiguities but makes them explicit. The resulting WF net is used as a basis to check properties the process should satisfy. Relaxed

soundness ensures that the modeled process meets some reasonable requirements. It enable us to check EPCs which contain OR connectors which typically presents a problem in other approaches.

Main aspects of future work include:

- finding a way to transform a relaxed sound net into a sound net (automatically) by reducing the net to the well-behaved paths; if this can be done the resulting net can be used directly as a basis for the design assuming that the EPC is of sufficient quality or a revision to costly,
- defining relaxed soundness directly for EPCs without requiring the intermediate step to Petri nets,
- integrating the approach into an analysis and design tool
- testing the approach in practical situations with large-scale models

In this paper we propose a new correctness criterion which is suitable to check the process model in an early phase of software engineering. Our approach allows us to draw conclusions on mistakes in the original EPC and to make suggestions for its improvement thereby enhancing both the model's quality and its suitability for software engineering.

Bibliography

1. Aalst, W.M.P. van der: Verification of Workflow Nets. In: P. Azema and G. Balbo: Application and Theory of Petri Nets 1997, Lecture Notes in Computer Science, vol. 1248, Springer, Berlin, 1997, pp. 407-426
2. Aalst, W.M.P. van der: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, **8** (1) 1998, pp. 21-66
3. Aalst, W.M.P. van der: Formalization and Verification of Event-driven Process Chains. Computing Science Reports 98/01, Eindhoven University of Technology, Eindhoven, 1998.
4. Chen, R., Scheer, A.-W.: Modellierung von Prozessketten mittels Petri-Netz-Theorie. Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 107 (in German), University of Saarland, Saarbrücken, 1994
5. Derks, W., Dehnert, J., Grefen, P. and Jonker, W.: Customized atomicity specification for transactional workflow. In: Cooperative Database Systems for Advanced Applications (CODAS'01), 2001, To appear
6. Keller, G. and Teufel, T.: *SAP R/3 prozeßorientiert anwenden: iteratives Prozeß-Prototyping zur Bildung von Wertschöpfungsketten*. Addison-Wesley, Bonn, 1997.
7. Schmidt, K.: LoLA, a Low Level Petri Net Analyzer. Humboldt-Universität, Berlin. http://www.informatik.hu-berlin.de/~kschmidt/lola.htm
8. Langner, P., Schneider, C, Wehler, J.: Ereignisgesteuerte Prozessketten und Petrinetze. Report No. 196, Computer Science Department, University of Hamburg, FBI-HH-B-196/97, March 1997.
9. Moldt, D., Rodenhagen, J.: Ereignisgesteuerte Prozessketten und Petrinetze zur Modellierung von Workflows. In: Visuelle Verhaltensmodellierung verteilter und nebenläufiger Software-Systeme, vol. 24/00-I, Münster, 2000, pp. 57-63.
10. Murata, T.: Petri Nets: Properties, Analysis, and Applications. Proc. of the IEEE, **77** (4) 1989, pp. 541-580
11. Rittgen, P.: EMC - A Modeling Method for Developing Web-based Applications. International Conference of the International Resources Management Association (IRMA) 2000, Anchorage, Alaska, USA, May 21 - 24, 2000

12. Roch, S.: Extended Computation Tree Logic. In: H.D. Burkhard, L. Czaja, A. Skowron and P. Starke: Workshop Concurrency, Specification & Programming, Informatik-Bericht 140, Humboldt-Universität, Berlin, Oct. 2000, pp. 225-234.
13. Rodenhagen, J.: Darstellung ereignisgesteuerter Prozessketten (EPK) mit Hilfe von Petrinetzen. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 1996.
14. Rump, F.: Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten. Formalisierung, Analyse und Ausführung von EPKs. Teubner, Stuttgart, 1999
15. Scheer, A.-W.: Business Process Engineering, Reference Models for Industrial Enterprises. Springer, Berlin, 1994