Making Workflow Models Sound Using Petri Net Controller Synthesis

Juliane Dehnert[1] and Armin Zimmermann[2]

[1] Fraunhofer ISST, Mollstr. 1, 10178 Berlin, Germany
[2] Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany
juliane.dehnert@isst.fraunhofer.de, azi@cs.tu-berlin.de

Abstract. More and more companies use "process aware" information systems to make their business processes more efficient. To do this, workflow definitions must be formulated in a formal specification language, as they represent executable derivates of business process descriptions. Both for the less formal descriptions of business processes as well as the workflow definitions, Petri-net based approaches are used successfully. In the literature the business process descriptions are required to be well-structured, leading to a sound workflow definition. We argue that in many cases well-structuredness is too restrictive for practitioners. Relaxed soundness has been introduced previously as a more suitable requirement. The paper presents how methods from controller synthesis for Petri nets can be used to automatically make this type of models sound. For this reason we adopt the idea of controllability for Petri net workflow models.

1 Introduction

Over the last decade more and more companies work with "process aware" information systems. These systems are configured on the basis of explicit process descriptions. Examples are dedicated Workflow Management systems (WfMS), such as Staffware, but also normal ERP systems which became enhanced by a workflow module. Prerequisite for their use is the specification of workflow, the computer-supported parts of the company's business processes.

Both for the descriptions of business processes as well as the workflow definitions, Petri-net based approaches are used successfully. For the definition of workflow Petri nets are particularly suitable, as they have a formal syntax and an unambiguous, operational semantics. The operational semantics offers the possibility to use the process descriptions right away as input format for a WF-engine. Examples of WfMS using Petri net based process descriptions are COSA (Software Ley/COSA Solutions/Transflow [SL99]) and Income (Get Process AG). Moreover, their formal foundation allows to validate the derived process description prior to their use within a WfMS. This helps to avoid faulty situations at run-time and therefore saves costs and raises customer satisfaction. An important property that every workflow definition should satisfy is soundness [Aal98]. Soundness guarantees that there are no faulty executions at run-time.

A workflow definition describes a business process in a machine readable manner. As their modeling requires a deep inside into the application context, domain experts are often put in charge of the modeling, although they do not necessarily have high modeling expertise.

Well-structuredness has been proposed [Aal98,LSW98,MR00] as a property that assists non expert modelers in formalizing their business processes. It requires a strict block structuring of the process descriptions. The restriction to well-structuredness is also present in UML v1.4 activity diagrams [UML02]. Strict block structuring conditions are relaxed by allowing control-links (resp. synchronization edges) to synchronize tasks belonging to different parallel control flow paths in BPEL4WS [BEA03] and ADEPT [RD98].

The advantage of this structural property is purely technical and lies in its close relationship to soundness. It has been shown that well-structured process descriptions are sound, provided they are life.

Well-structuredness has its shortcomings. We will argue in the paper that modeling in a well-structured manner requires: 1) to have a comprehensive insight into the whole process, possibly spanning different organizational units, 2) to implement efficiency aspects via the ordering of tasks, and 3) to accept redundancy.

This paper uses relaxed soundness [DvdA04] as a different property which is better suited to assist the modeler. We show that relaxed soundness meets the intuition of the modelers, not requiring expertise beyond their own organizational unit. However, because relaxed soundness is weaker than soundness, an additional step is required to achieve a sound WF-net. One contribution of the paper is to show how methods from Petri net controller synthesis can be adopted to automatically make this type of models sound. For this reason we apply the idea of task controllability to Petri net workflow models.

The paper presents an algorithm for the generation of the robust subgraph, i.e. the part of the behavior of a workflow model that can be controlled to avoid faulty situations. This algorithm is a refined version of the one presented in [Deh02].

An advantage of the approach proposed here is that the result of the automatic transformation can be used to detect potential for a process optimization. The separation between business process modeling and soundness transformation enables the modeler to adapt the model easily if business process requirements change.

The remainder of the paper is organized as follows: In the next section an application example is used to introduce the chosen modeling technique, namely WF-nets. The suitability of possible properties is compared in addition. In Section 4 relevant methods from Petri net controller synthesis are briefly introduced and their application to the area of workflow modeling is described. In Section 4.2 we broaden the scope of the proposed methods to reactive workflow systems. This is done by representing the interaction with the environment within the process descriptions. Section 5 focuses on process optimization based on the prior computations. Finally, the results are summarized.

2 An application example

As modeling technique for the specification of workflow we use Petri nets. We refer to the class of Place/Transition nets and more in particular to Workflow nets (WF-nets). This net class was introduced in [Aal98,Aal00]. WF-nets were tuned to fit the requirements within the domain of workflow management. Petri net theory was exploited to develop adequate properties and efficient algorithms for that Petri net class [Aal00,VBA01].

A WF-net is a Petri net which has a unique source place (i) and a unique sink place (o). This corresponds to the fact that any case handled by the process description is created if it enters the WfMS and is deleted once it is completely handled by the WfMS. In such a net, a task is modeled by a transition and intermediate states are modeled by places. A token in the source place i corresponds to a "fresh" case which needs to be handled, a token in the sink place o corresponds to a case that has been handled. The process state is defined by the marking. In addition, a WF-net requires all transitions and places to be on some path from i to o. This ensures that every task (transition) or condition (place) contributes to the processing of cases.

Figure 1 shows two WF-nets modeling the process "Handling of incoming order". Both process descriptions cover the ordering of a product which involves two departments: the accounting department handling the payment and the sales department handling the distribution.

In Figure 1a) the distributed organizational assignment is visible. The process starts by splitting the control-flow into two threads (`AND_process_order`), where the right one models the tasks of the accountancy and the left one models the tasks of the sales.

In accounting the customer's credit-worthiness is checked first (c.f. transition `check_credit`). The result of this task is either `ok` or `not_ok`. In case the result is positive the payment is arranged (`arrange_payment`), otherwise the instance is canceled and the customer is notified (`notify_customer`). On the sales side the order is recorded (`record_order`) and then either assembled (`pick`), wrapped (`wrap`), and delivered (`deliver`); or else canceled (`cancel`).

The threads of the two parallel departments are joined again in the transitions `AND_cancel` and `AND_accept`. The process "Handling of incoming order" is completed by archiving information on that instance (`archive`).

The WF-net in Figure 1b) describes the behavior of the same business process in a slightly different manner. The assignment of tasks to organizational units is neglected here. The tasks are ordered such that the net is well-structured instead (details see below). The two model variants are used in the following to show the differences and advantages between relaxed soundness and well-structuredness.

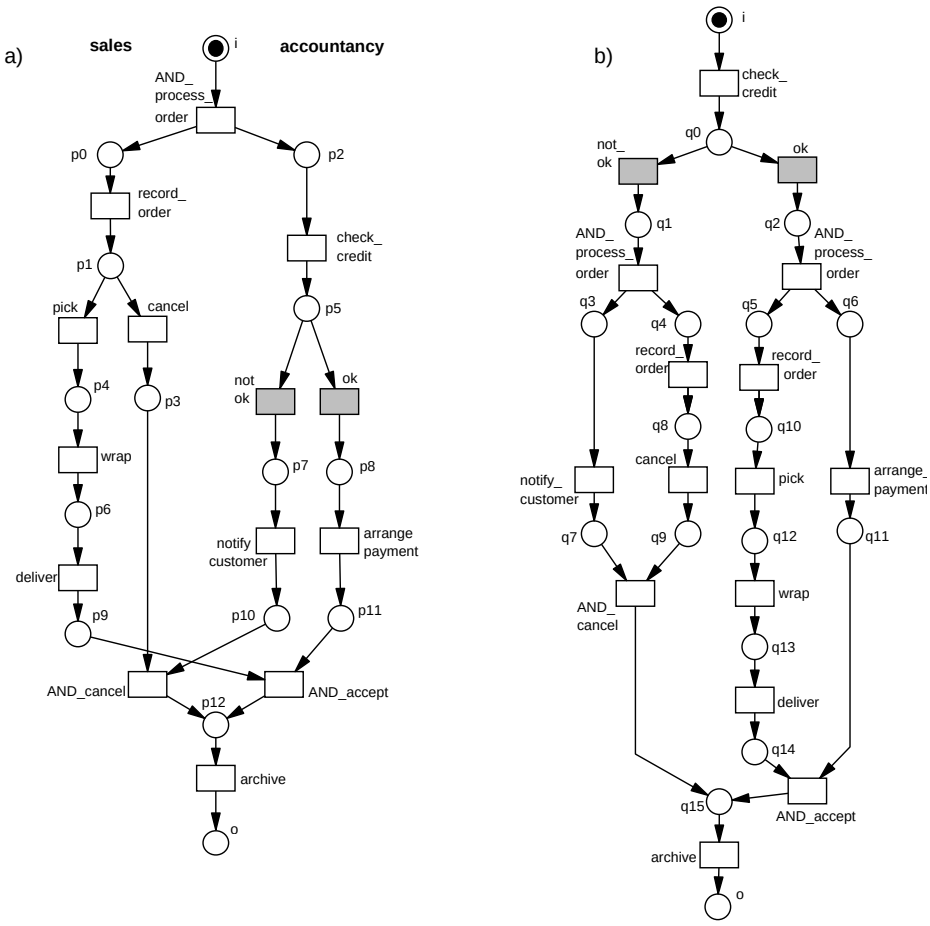

Fig. 1. WF-nets for process "Handling of incoming order"

3 Basic properties of workflow models

This section recalls and compares some properties of process descriptions.

3.1 Soundness

In [Aal98] *soundness* was introduced as a correctness criterion for WF-nets. A WF-net is sound if all its firing sequences are sound. A firing sequence is sound if it can terminate properly, which means that eventually there is a token in place o and at that moment there are no other tokens left in the net. Soundness of a WF-net excludes dead transitions, deadlocks and livelocks.

The WF-net in Figure 1b) is sound, while the one in Figure 1a) is not. This is caused by firing sequences that do not terminate properly in the left model, e.g.

- `AND_process_order, record_order, pick, wrap, deliver, check_credit, not_ok, notify_customer`.

In this firing sequence the case deadlocked having tokens in place `p9` and `p10`.

It is clear that a WF-net which shall be used as input for a WfMS must be sound. Serving as a scheduling basis, soundness of the workflow definition is necessary to guarantee a smooth processing of the supported business process at runtime. Things are different for the modeling phase of a workflow, because it is not obvious for a modeler to see whether a complex workflow model is sound or not. To support the domain experts in formalizing their business processes, different properties are therefore required. In the literature *well-structuredness* [Aal98,LSW98,MR00] and *relaxed soundness* [DR01] have been considered helpful.

Well-structuredness

A WF-net is well-structured[3], if every split is complemented by a corresponding join. In terms of Petri-net theory this property is characterized by the absence of *handles*[4] [ES90]. Note that the WF-net in Figure 1b) is well-structured, whereas the other WF-net is not. An example for a handle is the place-transition pair (AND_process_order, p12).

Well-structuredness is a structural property, whose validity can be easily reviewed. This and the close relationship to soundness[5] motivated its use as a requirement during workflow modeling. There are however sound WF-nets which are not well-structured. These WF-nets would be disregarded although suitable for the use within WfMSs. This shortcoming of well-structuredness was also addressed in [CWBH+03]. Providing refinement rules for the generation of sound WF-nets the authors propose some conditions under which well-structuredness can be relaxed while keeping soundness.

There are other disadvantages imposed by well-structuredness. Modeling in a well-structured manner requires a deep insight into the whole process. The tasks of the process must all be organized in well-structured blocks which may be combined again only in a well-structured manner. Such a hierarchical design ignores the organizational assignment of tasks and therefore requires overview of the whole process. This can hardly be assumed if the process to be described is spanning different organizational units of the company, involving various modelers. A further disadvantage is that the modeler might be forced to implement

[3] In the context of Event-driven Process Chains the terms hierarchical modeling and well-formedness [LSW98,MR00] were used synonymously.

[4] A handle is a pair of two different nodes (a place and a transition) that are connected via two elementary paths sharing only these two nodes.

[5] A well-structured net is structurally bound and structurally life [ES90]. Liveness and boundedness of a WF-net imply soundness.

efficiency aspects at an early design state. The modeler might be restricted by imposing well-structuredness in a way that induces him/her into coming up with process descriptions such as the WF-net from Figure 1b). Determined through the ordering, the tasks of the sales can only start after the customer check of the accountancy was performed. Parallel execution of sales and accountancy is then restricted. Last but not least, redundancy was introduced. Some tasks (`AND_process_order` and `record_order`) had to be represented by multiple transitions.

Relaxed soundness

An alternative property was introduced with relaxed soundness [DR01]. This property has been adapted from soundness with the intention to represent a more pragmatic view of correctness. It is weaker (in a formal sense) than soundness and therefore easier to accomplish.

Modeling business processes domain experts record the tasks and their order as they observe them to happen (or as they wish them to happen). This means they gather the desired behavior. Domain experts are no Petri net specialists. It may therefore happen that they overlook side effects of their model, i.e. firing sequences that do not express desired behavior. Relaxed soundness reflects this process understanding as it requires only that all relevant behavior is described correctly. It does neither forbid situations with residual tokens nor livelocks/deadlocks. A relaxed sound WF-net should be interpreted as follows: it specifies all business processes in terms of sequences of tasks for which a firing sequence from the initial state i to the final state o exists such that the transitions for these tasks occur in the order of a sound firing sequence.

Whereas in a sound WF-net *all* firing sequences are sound, relaxed soundness only requires that there are so many sound firing sequences that each transition is contained in one of them. A relaxed sound WF-net may have other firing sequences which do not terminate properly, e.g. by a deadlock or with tokens left in the net.

The process specification shown in Figure 1a) is relaxed sound. The following sound firing sequences contain all transitions:

- `AND_process_order, record_order, pick, wrap, check_credit, ok, deliver, arrange_payment, AND_accept, archive`,
- `AND_process_order, check_credit, not_ok, notify_customer, record_order, cancel, AND_cancel, archive`,

This definition still leaves room for ambiguities since it does not demand the precision of workflow definitions as they are required for their execution within a WfMS. Compared to well-structuredness, relaxed soundness does not make any assumptions on the structure of the WF-net. A relaxed sound WF-net may contain cycles and/or choices that do not satisfy the free-choice property. In contrast to soundness, it does not require all firing sequences to be sound, but only requires all tasks to be covered by at least one sound firing sequence.

Tests checking relaxed soundness have been implemented within Petri net tools such as LoLA [Sch99] (Low Level Petri Net Analyzer) and Woflan [VBA01]. Both algorithms parse the reachability graph, to decide whether a given WF-system is relaxed sound or not. To guarantee termination, the WF-systems must have been checked for boundedness before. This is a drawback of the proposed approach, as this requires the construction of the coverability graph, with a theoretical worst-case complexity of non-primitive recursive space [EN94].

4 Synthesis of sound WF-nets

We already stated that a process description which will be used as input for a WfMS must be sound. This corresponds to the requirement that supporting a business process at run-time, any faulty execution should be precluded. We will now describe how a relaxed sound WF-net can be made sound. The proposed transformation is automated.

Making a relaxed sound WF-net sound means to restrict the set of all possible firing sequences to a subset of sound ones. Looking at the reachability graph RG of the relaxed sound WF-net, this comes down to finding a WF-net with a behavior equal to a sound subgraph of RG. Naturally it would be nice not to generate a new net, but to change the primary WF-net such that it implements the restricted behavior. Both the generation of a new WF-net as well as the change of the primary WF-net are feasible methods.

The first possible approach uses methods from Petri net synthesis [CKLY98]. Based on a subgraph of the reachability graph containing only sound firing sequences, a WF-net is synthesized. The behavior of the synthesized net is isomorphic to the sound subgraph. A disadvantage of this method is that the derived WF-net does not necessarily look like the primary WF-net. As the net is generated on the basis of the reachability graph, information such as place names, layout, and ordering of transitions are ignored. The new net therefore only coincides with the primary WF-net in the names of the transitions.

We therefore favor a different method, which applies methods from Petri net controller synthesis. The idea is to compute and introduce places that supervise or control the behavior of the Petri net. These places, called *controller places* [YMLA96] or *monitors* [GDS92], avoid entering a set of *forbidden states*[6]. The information needed for their computation can be gained in various ways, e.g. from place invariants [YMLA96], general mutual exclusion conditions (GMECs) [GDS92], or sets of forbidden markings [GRX03]. Because the original net is kept and enhanced with additional elements, the resulting net will be easily recognized by the modelers.

[6] An additional place can only restrict the behavior because the place can block transitions but it cannot enable transitions which are not enabled in the net without the place.

4.1 Applying Petri net controller synthesis for workflow modeling

We favor the computation of the controller places based on a set of forbidden markings [GRX03], because the prerequisites (set of forbidden markings, state transitions to be prevented) can be directly mapped to our approach. Starting from a sound subgraph, the forbidden markings correspond to all states that are beyond the sound subgraph. State transitions to be prevented correspond to state transitions leaving the sound subgraph. For every one of these instances an equation system is established which is used to compute a controller place inhibiting this forbidden state transition. The equation system consists of three equations: 1) the event separation condition — an equation which in terms of the incidence matrix describes the interdiction of the corresponding state transition — 2) the *marking equation lemma*, and 3) the general property of T-invariants[7]. All three equations should hold in the resulting net. The first represents the new requirements: state transitions leaving the subgraph become disabled. The latter two represent the behavior described by the sound subgraph, which should be maintained independently from the introduction of new places.

The equation systems of different instances may have common solutions. As a result, the number of needed controller places is generally much smaller than the number of forbidden state transitions. The set of controller places together with the associated arcs determine, what was called the *synchronization pattern*. Adding the pattern to the primary WF-net a new WF-net is generated, that supports a subset of the primary behavior.

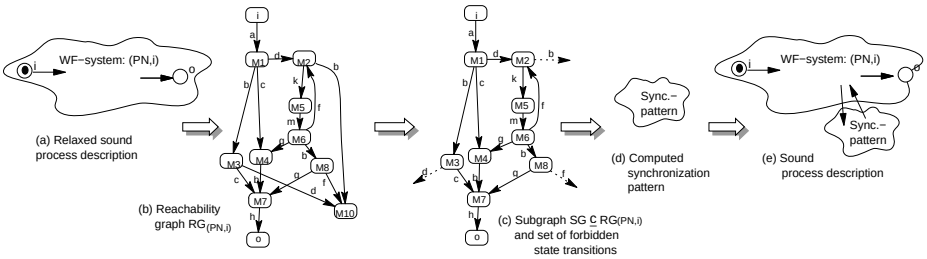

Fig. 2. Applying controller synthesis for workflow modeling.

Figure 2 illustrates the application of controller synthesis to workflow modeling.

Applying either one of the synthesis methods, all firing sequences supported by the resulting net are sound, as they are covered by a sound subgraph. Moreover, the calculated net again satisfies the properties of a WF-net: from the construction it can be concluded that it is strongly connected, having one source and one sink place [DvdA04].

[7] A (short-circuited) relaxed sound WF-net is covered with T-invariants [Deh03].

Still, the subgraph given by assembling all sound sequences does not necessarily provide a reasonable base for the computation of the sound WF-net. Remember that the resulting WF-net does not support state transitions leaving the sound subgraph. Corresponding transitions of the resulting WF-net become disabled in markings, where they could fire in the primary net. In the following we will argue that prevention from firing is only reasonable if the task, modeled by the affected transition, represents *controllable* behavior.

4.2 WF-systems are reactive systems

We consider a WF-system to be a reactive system [Deh02]. They run in parallel with their environment, respond to inputs from the environment and produce output events which in turn influence the environment.

The interaction with the environment takes place via incoming external events or via the evaluation of external information. The reactive system has to respond to external events and to incorporate the possible outcomes of the information evaluation.

An external event could be an incoming query, an acknowledgment from a customer, a message from another company, information from a business partner, or just a timeout. Examples for the evaluation of external information are questions about available capacities, the check for credit-worthiness of a customer, and the identity check of a co-operating partner.

Reflecting the interaction with the environment, we distinguish controllable and non-controllable tasks. In the process description this is reflected in a corresponding classification of the transitions. *Controllable transitions* model internal tasks, i.e. tasks whose execution is covered by the local workflow control. In contrast to that, *non-controllable transitions* represent the behavior of the environment. Their firing cannot be forced by the local workflow control but depend either on the evaluation of external data or on an incoming external event.

Throughout this paper we represent non-controllable transitions by gray boxes. We assume that non-controllable transitions are free-choice and do not conflict with controllable transitions. This reflects the fact that the behavior of the environment cannot become disabled through the local control. In the remainder we will consider only WF-nets which satisfy these restrictions.

4.3 Impact of controllability upon the generation of sound WF-nets

Applying methods from Petri net (Controller) Synthesis, the resulting WF-net does not support state transitions leaving the sound subgraph. Corresponding transitions of the resulting WF-net become disabled in markings, where they could fire in the original net. It is obvious that the state transitions to be prevented must not reflect uncontrollable behavior, as this would exceed the capabilities of the local workflow control.

Consider the sound subgraph in Figure 3a). It contains all sound firing sequences of the WF-net “Handling an incoming order”, which are highlighted

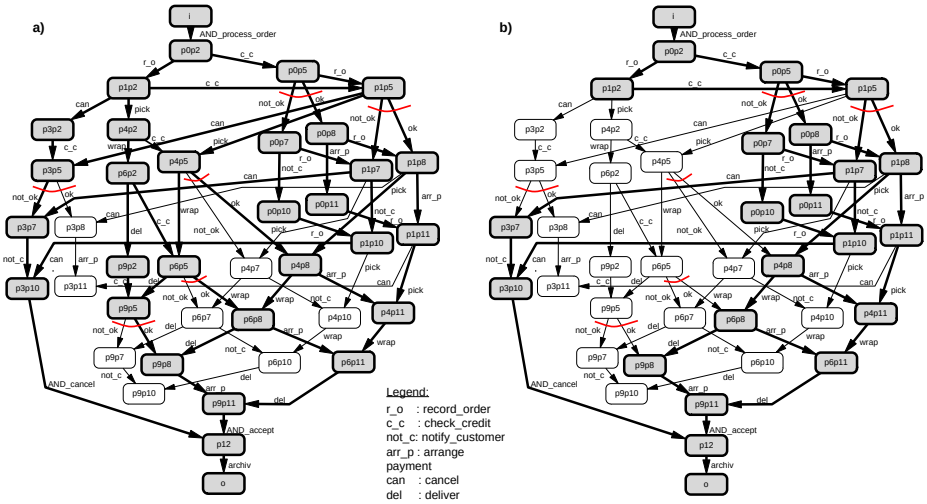

Fig. 3. Reachability graph with highlighted sound subgraph a) robust subgraph b)

in the figure. Enforcement of this desired behavior is not possible, as there are non-controllable state transitions (depicted by a bow) leaving the subgraph. The corresponding transitions (`ok` and `not_ok`) reflect the outcome of a decision based on an evaluation of external data, and is not left to the discretion of a local workflow control.

Consequently, the subgraph must be restricted furthermore until all state transitions leaving the subgraph reflect controllable, and therefore preventable, behavior.

Such a subgraph exists if the WF-net is not only relaxed sound but also *non-controllable choice robust* (short: robust). This criterion provides a means to describe robustness of a WF-system against all possible requests from the environment. A WF-system is robust if 1) there is a sound subgraph of the reachability graph which starting in i ends in o, 2) contains at least one t-labeled state transition for any non-controllable transitions, and 3) has only controllable state transitions leading out of the subgraph.

Assuming progress for non-controllable transitions, the existence of such a subgraph guarantees that it is possible to terminate properly independent from the influence of the environment. While all non-controllable transitions are covered by the subgraph, there is always a way to react and to terminate properly. Hence, if a WF-system is robust, the workflow controller can guarantee proper termination independently from all possible influences of the environment.

The robustness criterion together with an algorithm constructing the maximal robust fragment were introduced in [Deh02]. The algorithm decides whether a given bounded WF-system is robust, and if so, returns the maximum robust subgraph $SG = (SG\_Nodes, SG\_Edges)$ of the system's reachability graph RG.

Initialization:
SG_Nodes := Pred(o, RG);
SG_Edges := all edges of RG that connect nodes in SG_nodes;
Illegal_states:= nodes in SG_Nodes from where
non-controllable state transitions leave the subgraph SG;

Body:
while Illegal_states $\neq \emptyset$ **do**
SG_Nodes:= SG_Nodes \ Illegal_states;
SG_Edges:= edges of RG that connect nodes in SG_nodes;
(cut illegal states and state transitions *)*
SG_Nodes:= Succ(i, SG) $\cap$ Pred(o, SG);
SG_Edges:= edges of RG that connect nodes in SG_nodes;
(recompute strongly connected component *)*
Illegal_states:= nodes in SG_Nodes from where
non-controllable state transitions leave the subgraph SG;
(recompute current set of illegal states *)*
od

Test and output
if all non-controllable transitions are represented in the robust subgraph
then print (The WF-system is robust); return SG:=(SG_Nodes,SG_Edges);
else print (The WF-system is not robust);
return not covered non-controllable transitions;
fi

Fig. 4. Robustness algorithm

The algorithm otherwise aborts with the result "not robust", displaying the set of non-controllable transitions which may inhibit proper termination. Figure 4 shows an improved variant of the algorithm using an informal notation. Sets frequently used in the algorithm are the sets of all direct and indirect predecessors Pred(n, G) (successors Succ(n, G)) of a node n within a graph G. These sets contain all nodes that lie on any path that lead to (start at) this node.

The algorithm mainly works as follows. It initially marks all states that potentially belong to the desired fragment and then progressively removes mistaken candidates. Potential states are all lying on a path from state i to state o. Illegal states are states from where non-controllable state transitions leave the fragment. The algorithm stops if the iteration of this procedure does not identify any more illegal states.

An algorithm similar to the presented one has been introduced in the context of manufacturing systems recently [GRX03]. This algorithm computes a maximally permissive behavior, starting from a reachability graph and avoiding a set of forbidden states. Our algorithm differs in the computation of the strongly connected component, because the existence of i and o states in a WF-net can be

exploited. In our algorithm an additional robustness check is performed on the resulting subnet, requiring that all non-controllable transitions are covered. This guarantees that none of the possible behavior of the environment is neglected. In [GRX03] it is proved that the algorithm is of polynomial complexity in the number of states of the reachability graph. The complexity of our algorithm is the same because the additional robustness check is only polynomial in the number of transitions.

The application of the algorithm shows that the example WF-net "Handling an incoming order" is robust. The resulting subgraph is shown in Figure 3b). Thereby the WF-net reflects a set of accepted (sound) executions which can be enforced independently from the moves of the environment. Applying the Petri net controller synthesis algorithm to the robust subgraph, two controller places $Pc1$ and $Pc2$ are computed. Adding the places and corresponding arcs to the original WF-net, the process description shown in Figure 5 is derived.

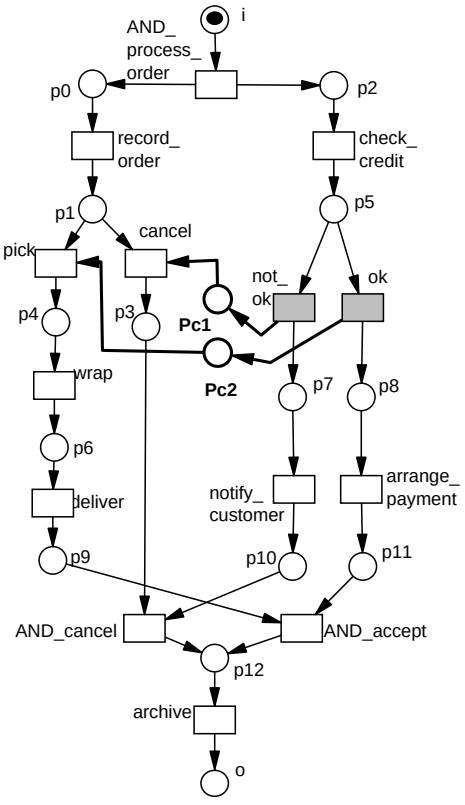

Fig. 5. Sound WF-net "Handling an incoming order" with controller places

The resulting WF-net is per construction sound. Using the enhanced process description as a workflow specification, i.e. as input for a WfMS, it can now be guaranteed that only sound executions will occur.

5 Interactive process improvement

Implementing the robust subgraph, the set of sound firing sequences has been restricted. This is done to avoid executions which are not sound, but could otherwise not be prevented due to the behavior of the environment.

Consider again the relaxed sound process description of the example "Handling of incoming order" (Figure 1a)). The firing sequence

- `AND_process_order, record_order, pick, wrap, check_credit, ok, deliver, arrange_payment, AND_accept, archive`.

is sound but became forbidden in the enhanced process description. The reason can be found in the non-controllable outcome of the check for credit-worthiness, which represents a choice of the environment.

Before using the enhanced process description as input for a WfMS, the restrictions with respect to the original specification should be communicated to the modeler. As the whole set of sound firing sequences were specified, she should approve the reduced set of accepted executions. The evaluation could either be done based on the revised, sound WF-net or on the reachability graph.

Approval based on the revised WF-net This method could be used if the sound WF-net was computed applying the Petri net controller synthesis method. Only then it can be assumed that there is a high similarity between the primary and the resulting process description. Looking at the introduced places the modeler has to evaluate whether the thereby introduced synchronization is acceptable to be supported at run-time.

Approval based on the reachability graph Looking at the difference between the relaxed sound subgraph and the robust subgraph, all those firing sequences are described which have been specified in the primary relaxed sound WF-net, but will not be supported in the resulting sound WF-net. The domain expert should decide whether it is acceptable to disregard these executions at run-time.

The idea to use the reachability graph as supplementary interface to the domain experts was introduced in [AdM00] first. The authors propose to use both the Petri net and the corresponding reachability graph as interface to the modeler and to use the basic synthesis algorithm [NRT92] to transfer between both descriptions. Adequate for their modeling approach is the Petri net class of *Elementary Net Systems*. In contrast to our approach all process models are assumed to be acyclic, free-choice and sound. Interaction with the environment is not considered.

Both methods point at executions which might have been considered useful originally, but were eliminated to make the model sound. However, these disregarded executions might express desirable behavior. If so, the process description

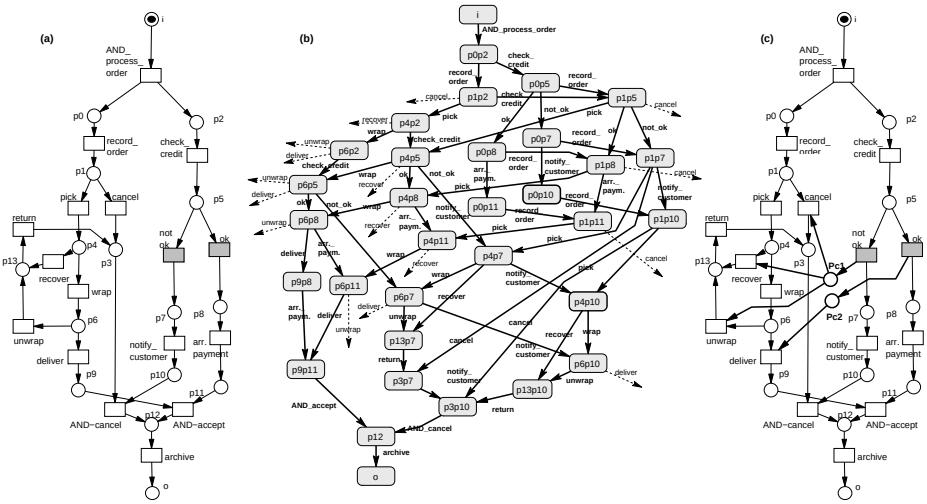

Fig. 6. Optimized sound WF-net "Handling of incoming order" with controller places

must be revised. The sound but prevented firing sequences can be enabled, if some recovery behavior is added. This is necessary if the behavior of the environment would lead to deadlocks otherwise. Recovery behavior can be added by integrating new transitions into the WF-net. By firing of one of these transitions it is possible to escape a former deadlock state – leading to a state where proper termination is again possible. Clearly, these transitions should only then be implemented if the corresponding tasks are reversible in reality.

We will consider again our running example. Applying the synthesis method to the robust fragment from Figure 3b), a pessimistic strategy was implemented. In favor of avoiding deadlocks only sequentialized executions are supported. In the derived WF-net (c.f. Figure 5) the customer check is always executed before the sales department may start the delivery process. All sound executions covering parallel execution of sales and accountancy have been precluded.

The domain experts may reject this process description. They know that the customer check and the delivery process both take a long time. Furthermore, the probability that the customer check is `not ok`, is very small. Therefore they want to support the parallel execution of sales department and accountancy. A more optimistic approach should thus be implemented. The delivery of the order to the customer should be started already, hoping that the customer check will be `ok`. Only in the rare case that the decision `not_ok` was taken, the order should be returned to stock and canceled finally.

For the specification of the necessary recovery behavior, we assume that all tasks within the sales department that occur before the delivery can be reset without extraordinary charges. This affects tasks `pick` and `wrap`. Corresponding recovery tasks are `return` and `unwrap`. After the item has been returned to stock,

the instance should be canceled. Task `deliver` is considered to be non-reversible. The revised WF-net is shown in Figure 6 a). Notice that the integrated tasks only show one possible way of modeling the recovery behavior.

The resulting WF-net is again relaxed sound and robust. The robust subgraph is shown in Figure 6b). All sound firing sequences of the initial, relaxed sound WF-system (c.f. Figure 1a)) are maintained. Some additional, but less efficient executions are accepted too. Implementing the computed synchronization pattern results in the sound WF-system shown in Figure 6c).

6 Summary

This paper showed that relaxed soundness as a property for workflow modeling is better suited than well-structuredness. The gap between the resulting process description and a sound workflow definition is bridged by an automatic transformation. Methods from Petri net controller synthesis are adopted for this task. Thereby, a synchronization pattern is added to the original WF-net, installing a certain task ordering. Thus only in this second step efficiency aspects become determined. We showed that the results of the computation point out optimization potential. The advantages of the proposed approach are obvious. Modelers, normally domain experts, are not required to possess highly developed modeling skills and are relieved of thinking about efficiency aspects during the modeling. Moreover, the concept of task controllability is transferred to the domain of workflow modeling. This is a necessary prerequisite for the application of controller synthesis, and enables the description and analysis of workflow systems as reactive systems.

References

[Aal98] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[Aal00] W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net based Techniques. In *Business Process Mangement: Models, techniques, and Empirical Studie*, volume 1806 of *LNCS*, pages 161–183. Springer Verlag, 2000.

[AdM00] A. Agostini and G. de Michelis. A Light Workflow Management System Using Simple Process Models. *Computer Supported Cooperative Work*, 9(3/4):335–363, 2000.

[BEA03] BEA Systems, IBM Corporation, Microsoft Corporation, SAP AG, Siebel Systems. *Business Process Execution Language for Web Services (Version 1.1*, 2003.

[CKLY98] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE Transactions on Computers*, 47(8):859–882, 1998.

[CWBH+03] P. Chrzastowski-Wachtel, B. Benatallah, R. Hamadi, M. O'Dell, and A. Susanto. A top-down petri net-based approach for dynamic workflow

modeling. In W. van der Aalst, A. ter Hofstede, and M. Weske, editors, *1st International Conference on Business Process Management (BPM)*, volume 2678 of *LNCS*, pages 336–353. Springer, 2003.

[Deh02] J. Dehnert. Non-controllable choice robustness: Expressing the controllability of workflow processes. In J. Esparza and C. Lakos, editors, *23rd Int. Conf. on Application and Theory of Petri Nets*, volume 2360 of *LNCS*, pages 121–141. Springer, 2002.

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

[DR01] J. Dehnert and P. Rittgen. Relaxed Soundness of Business Processes. In K.L. Dittrich, A. Geppert, and M.C. Norrie, editors, *Advanced Information System Engineering, CAISE 2001*, volume 2068 of *LNCS*, pages 157–170. Springer, 2001.

[DvdA04] J. Dehnert and W. van der Aalst. Bridging the Gap Between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems (IJCIS)*, 3(3), 2004. to appear.

[EN94] J. Esparza and M. Nielsen. Decidability Issues for Petri Nets: A Survey. *Journal of Information Processing and Cybernetics*, 30:143–160, 1994.

[ES90] J. Esparza and M. Silva. Circuits, Handles, Bridges and Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *LNCS*, pages 210–242. Springer, 1990.

[GDS92] A. Giua, F. DiCesare, and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 974–979, Chicago, IL, 1992.

[GRX03] A. Ghaffari, N. Rezg, and X. Xie. Design of a live and maximally permissive petri net controller using the theory of regions. *IEEE Transactions on Robotics and Automation*, 19(1):137–142, 2003.

[LSW98] P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event-driven Process Chains. In J. Desel and M. Silva, editors, *Application and Theory of Petri nets*, volume 1420 of *LNCS*, pages 286–305. Springer, Berlin, 1998.

[MR00] D. Moldt and J. Rodenhagen. Ereignisgesteuerte Prozessketten und Petrinetze zur Modellierung von Workflows. In *Visuelle Verhaltensmodellierung verteilter und nebenläufiger Software-Systeme*, volume 24/00-I of *Fachberichte Informatik*, pages 57–63, 2000.

[NRT92] M. Nielsen, G. Rozenberg, and P. S. Thiagarajan. Elementary transition systems. *Theoretical Computer Science*, 96(1):3–33, April 1992.

[RD98] M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[Sch99] K. Schmidt. LoLA: A Low Level Analyser. In *Proc. Int. Conf. Application and Theory of Petri net*, volume 1825 of *LNCS*, pages 465–474, 1999.

[SL99] Software-Ley. *COSA 3.0 User Manual.* Software-Ley GmbH, Pullheim, Germany, 1999.

[UML02] Unified Modeling Language: version 1.4.2, ISO, 2002.

[VBA01] H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001.

[YMLA96] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsakli. Feedback control of Petri nets based on place invariants. *Automatica*, 32(1):15–28, 1996.