A Methodology for Workflow Modeling

From business process modeling towards
sound workflow specification

vorgelegt von
Diplom-Informatikerin
Juliane Dehnert

Eingereicht bei der Fakultät IV Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktorin der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Hartmut Ehrig
Berichter:	Prof. Dr. Herbert Weber
Berichter:	Prof. Dr. Wil van der Aalst

Tag der wissenschaftlichen Aussprache: 22. August 2003

Berlin 2003
D 83

Technische Universität Berlin
Institut für Softwaretechnik und Theoretische Informatik

A Methodology for Workflow Modeling

From business process modeling towards
sound workflow specification

Juliane Dehnert

Abstract

Supporting business processes with the help of workflow management systems is a necessary prerequisite for many companies to stay competitive. An important task is the specification of workflow, i.e. these parts of a business process that can be supported by a computer system. A workflow specification mainly refines a business process description, incorporating details of the implementation. Despite the close relation between the two process descriptions there is still no satisfactory link between their modeling. This fact mainly relies on the assignment to different peolpe (IT- vs. domain experts) having a different modeling culture.

The thesis provides a methodically well-founded approach for the specification of functional workflow requirements. It supports domain experts in their modeling of business processes in a semiformal manner and guides them stepwise towards a formal workflow specification, i.e. helping to bridge the gap between business process modeling and workflow specification.

The proposed approach acknowledges the need to describe business processes at different levels of abstraction and combines the advantages of different modeling languages that proved to fit the respective requirements. A semiformal modeling language is proposed to be used by the domain expert. As a prominent example, widely accepted in practice, are Event-driven Process Chains (EPCs). For the definition of the workflow specification we use a particular type of Petri nets. The strength of Petri-nets is their formally founded, operational semantics which enables their use as input format for workflow management systems.

The key concept for the proposed process model is the use of pragmatic correctness criteria, namely *relaxed soundness* and *robustness*. They fit the correctness requirements within this first abstraction level and make it possible to provide a feedback to the modeler.

To support the execution of the business process at run time, the resulting process description must be refined to fit the requirements of a workflow specification. The proposed process model supports this refinement step, applying methods from controller synthesis. A sound WF-system is automatically generated on the basis of a relaxed sound and robust process description. Only within this step do performance issues become relevant. Information that is incorporated relates to a certain scheduling strategy. The late determination of performance issues is especially desirable as corresponding information (the occurrence probability of a certain failure, costs of failure compensation, or priorities) will often only become available at run-time. Their incorporation towards the end of the proposed process model extends the possibility to reuse modeling results under changing priorities.

The resulting process description is sound. Using it as a basis for the execution support during run-time reliable processing can be guaranteed.

Zusammenfassung

Der Einsatz von Workflow Management Systemen (WFMS) in Unternehmen oder Verwaltungen mit einfach strukturierten und automatisierbaren Prozessen bietet ein hohes Potenzial für die Optimierung der Geschäftsprozesse. Für die Koordinierung von Geschäftsprozessen zur Laufzeit benötigen WFMS Workflow-Spezifikationen, die den automatisierbaren Anteil der Geschäftsprozesse in einer maschinenlesbaren Form beschreiben. In der Praxis werden Workflow-Spezifikationen bislang oft unabhängig von bereits existierenden Geschäftsprozessmodellen erstellt. Es existiert kein methodisch fundiertes Vorgehensmodell, dass die Modellierung von Gechäftsprozessen und die Weiterverwendung der erstellten Modelle für die Workflow-Spezifikation unterstützt [GHS95, AH02a].

Diese Arbeit schlägt ein durchgehendes Vorgehensmodell für die Spezifikation von Workflows in Form von Petrinetzen vor. In dem fünfstufigen Vorgehensmodell wird der Schwerpunkt auf die Modellierung der Kontrollflussaspekte gelegt. Im Rahmen der Modellierung werden die folgenden Schritte unterstützt: 1. Modellierung der Geschäftsprozesse 2. Formalisierung durch Petrinetze 3. Korrektheitstest und Fehlerkorrektur 4. Festlegung und Integration einer Ausführungsstrategie 5. Kontrollverfeinerung. Das Ergebnis ist ein Prozessmodell mit formal fundierter und operationaler Semantik, das zudem sound [Aal98] ist. Ein solches Modell entspricht den Anforderungen an eine Workflow-Spezifikation, deren Verwendung für ein WFMS eine zuverlässige Ausführung der Geschäftsprozesse zur Laufzeit garantiert. In dem ersten Schritt ”Modellierung der Geschäftsprozesse” wird die Verwendung semiformaler Modellierungstechniken unterstützt. Diese räumen dem Modellierer Spielraum in der Beschreibung der Prozesse ein. Im nächsten Schritt wird das erstellte Modell intern formalisiert. Die Formalisierung basiert auf einer Abbildung in Petrinetze. Dabei werden Mehrdeutigkeiten nicht eliminiert sondern explizit gemacht. Im dritten Schritt wird das Modell auf Korrektheit überprüft. Dafür werden neue, pragmatische Kriterien eingeführt. Es werden präzise Fehlermeldungen zurückgegeben, die ein iteratives Verbessern der Geschäftsprozessmodelle ermöglichen. In Schritt vier und fünf wird das erstellte Modell auf eine Workflow-Spezifikation abgebildet. Dazu wird auf die bereits erstellte Petrinetz-Formalisierung zurückgegriffen. Die Petrinetze werden zunächst so erweitert, dass eine Ausführungsstrategie festgelegt wird. Durch die Integration der Strategie werden alle vorher noch enthaltenen Mehrdeutigkeiten beseitigt. Abschließend werden Aktivitäten verfeinert.

Das vorgeschlagene Vorgehensmodell bindet in der Praxis bewährte Techniken ein und stellt angemessene Kriterien für die Fehlerkorrektur zur Verfügung. Das gesamte Vorgehensmodells ist methodisch unterlegt und greift auf Ergebnisse der Petrinetztheorie, der Spieltheorie und der Controller Synthesis zurück.

Publications based on this thesis

- W. Derks, **J. Dehnert**, P. Grefen, and W. Jonker. Customized atomicity specification for transactional workflow. *International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'01)*, pages 140–147. IEEE Computer Society, 2001.
- **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 Verlag, june 2001.
- **J. Dehnert**. Four Systematic Steps Towards Sound Business Process Models. In H. Weber, H. Ehrig, and W. Reisig, editors, *2nd Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems*, pages 55–64. Fraunhofer Gesellschaft ISST, September 2001.
- **J. Dehnert**. Expressing the Controllability of Business Processes. *Petri Net Newsletter*, 61, 2001.
- **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 Verlag, 2002.
- **J. Dehnert**. Four Steps Towards Sound Business Process Models. In G. Rozenberg, H. Ehrig, H. Weber, and W. Reisig, editors, *PNT-Volume, LNCS*, Advances in Petri Nets. Springer Verlag, 2003. to appear.
- **J. Dehnert**. Making EPCs fit for Workflow Management. *EMISA FORUM*, 23(1), 2003.
- R. Eshuis and **J. Dehnert**. Reactive Petri Nets for Workflow Modelling. In J. Esparza and C. Lakos, editors, *24th Int. Conf. on Application and Theory of Petri Nets*, volume 2679 of *LNCS*, pages 296–315. Springer Verlag, 2003.

Acknowledgments

There are many people I would like to thank helping me to go through writing a thesis. First of all I would like to thank my reviewer Wil van der Aalst of Eindhoven University of Technology. He spent a lot of time discussing my ideas with me, and made many useful suggestions. Big thanks also goes to Eric Verbeek, who read my statements very carefully providing me with a lot of valuable feedback.

Paul Grefen and Ralf Kutsche helped me with the application for a Leonardo da Vinci grant of the European Commission, which allowed me to go to the Netherlands for three months. Working there, together with Wijnand Derks at the KPN research, triggered the first ideas including the relaxed soundness criterion, which became an emphasis within my work.

Other people who influenced the contents of this thesis, are Axel Martens from the Humboldt University and Rik Eshuis from the TU Twente. The discussions with Axel Martens contributed to the elaboration of the robustness criteria. Working with Rik Eshuis, brought about a requirement specification for the semantics of process descriptions to be used in real workflow management systems.

Special thanks also goes to Philippe Darondeau and Benoît Caillaud who answered many questions about Petri net synthesis. I thank Eric Verbeek, Karsten Schmidt and Andreas Krüger, who implemented the proposed correctness criteria and therefore took part at the possibility to actually use the proposed process model.

A lot of helpful comments and motivation came from the members of my former research group at Humboldt University, in particular Jörn Freiheit, Hagen Völzer and Karsten Schmidt who provided me with various useful comments. I thank Herbert Weber for making available a good working environment at the Technical University Berlin, and the members of my working group, who provided a pleasant atmosphere to work in. Secretary Claudia Gantzer managed my travels and took off a lot of disagreeable organizational work. Last but not least I'd like to thank my friends and my family for their continuous motivation and support.

Contents

1 Introduction **6**
1.1 Background . . . 6
1.2 Problem statement . . . 9
1.3 Problem solving approach . . . 13
1.4 Thesis structure . . . 15

2 Preliminaries **17**
2.1 Place/Transition nets . . . 17
2.1.1 Structural properties of nets . . . 18
2.1.2 Execution of Petri nets . . . 20
2.1.3 Dynamic properties of net systems . . . 26
2.2 Workflow nets . . . 29
2.2.1 Structural and dynamic properties of WF-nets . . . 30
2.3 Related work . . . 32

3 Modeling business processes **34**
3.1 Introduction . . . 35
3.2 Modeling with EPCs . . . 36
3.2.1 EPC syntax . . . 36
3.2.2 EPC semantics . . . 38
3.3 Transformation into workflow nets . . . 39
3.3.1 Step 1: Mapping EPC elements to Petri net-modules . . . 40
3.3.2 Step 2: Module combination . . . 41
3.3.3 Step 3: Adding unique input/output places . . . 42
3.4 Properties of the derived WF-nets . . . 44
3.4.1 General properties . . . 45
3.4.2 Soundness . . . 46
3.4.3 Relaxed soundness . . . 49

3.5 Implementation 53
3.6 Case study 56
3.7 Application of concepts 57
3.7.1 Staffware 58
3.8 Related work 63
3.8.1 Main trends within business process modeling 64
3.8.2 Workflow-patterns 66
3.8.3 Resolving the ambiguity of the OR-connector 69

4 Embedding relaxed soundness into Petri net theory 71
4.1 Review of introduced correctness criteria 71
4.2 Relaxed soundness versus soundness 73
4.2.1 State machines 73
4.2.2 Marked graphs 73
4.2.3 Free-choice Workflow-nets 74
4.2.4 Well-structured Workflow-nets 75
4.3 Other relations 80
4.4 Related work 82

5 Non-controllable choice robustness 84
5.1 Workflow systems are reactive systems 85
5.1.1 Modeling requirements of reactive systems 85
5.1.2 Capabilities of Workflow-nets 85
5.1.3 Reflecting the interaction with the environment 87
5.2 Ability to control a process 89
5.3 Non-controllable choice robustness 91
5.3.1 Game, play, and winning strategy 92
5.3.2 Adaption for workflow modeling 93
5.4 An algorithm verifying robustness 98
5.4.1 Correctness of the algorithm 100
5.5 Embedding robustness into Petri net theory 102
5.6 Related work 105

6 Generating sound process specifications 110
6.1 Petri net synthesis 111
6.1.1 Preliminaries 112
6.1.2 Synthesizing safe Petri nets 117
6.1.3 Synthesizing general Petri nets 119
6.2 Applying Petri net synthesis to workflow modeling 120
6.3 Petri net controller synthesis 125

6.4 Applying controller synthesis to workflow modeling 130
6.5 Appraisal of results 132
6.6 Related work 134

7 Installing different scheduling strategies **137**
7.1 Optimization potential 137
7.1.1 Determination of system parameters 138
7.1.2 Scheduling strategies 138
7.2 Strategy determination and implementation 139
7.3 Installing an optimistic strategy 141
7.4 Appraisal of results 145
7.5 Related work 146

8 The process model **148**
8.1 Objective and Overview 148
8.2 Detailed description and illustration 152
8.2.1 1st step: Business process modeling (EPCs) 152
8.2.2 2nd step: The transformation of EPCs into WF-nets 155
8.2.3 3rd step: Correctness check & feedback 157
8.2.4 4th step: Strategy determination & implementation 163
8.2.5 5th step: Control refinement 167
8.3 Application of results 172
8.4 Related work 175

9 Conclusion and future work **180**
9.1 Conclusion 180
9.2 Summary of main contributions 181
9.3 Future work 183
9.4 Synthesis of a WF-net not covering all controllable transitions . . 184
9.5 Synthesis of a general Petri net: Example 185

Chapter 1

Introduction

In this thesis a methodically well-founded process model is presented for the specification of functional workflow requirements. It is intended to support domain experts modeling their business processes in a semiformal manner and guides them stepwise towards a formal workflow specification. The key of the process model is the introduction of a pragmatic correctness means which is gradually refined.

In Section 1.1 some terminology is introduced. Then, in Section 1.2 the problem is defined. In Section 1.3, the problem solving approach is explained. This is followed by an overview of the structure of the thesis.

1.1 Background

In every company there are procedures aimed at providing increased efficiency, consistency, and quality. Among them are concepts such as business process re-engineering and workflow management. Their objective is to clarify, improve, and coordinate complex activities and interactions.

Business process re-engineering projects are concerned with the design and re-design of business processes. A *business process* is an ordered set of *business activities.* The goal of a business process is to deliver specific products or services while ensuring the organization's overall interests. Examples for business processes are processing purchase orders over the phone, processing insurance claims, and processing a car registration. A business activity is an atomic amount of work that is performed by some processing entity or *actor*. Actors are individuals and/or software components. Examples of activities include updating a file or database, generating a bill, and assembling parts of a product.

The core of business process re-engineering projects are the *business process descriptions.* A business process description specifies in a semiformal manner which activities are executed in what order. It may be enriched by information about the associated actors, their organizational assignment, documents used and processed, and other related aspects. For an overview of aspects possibly relevant for the modeling of business processes, see [JB96].

There is a trade-off between the number of aspects that are incorporated into a business process description and the usability of the resulting model. Clearly, the more complex a description becomes the harder to make significant statements about its behavior. Depending on the goal of the modeling, e.g. pure communication and/or analysis demands, a suitable level of abstraction must be chosen. In this work we concentrated on the functional or control flow aspects, i.e. the activities and their ordering (sequential, conditional, parallel and iterative routing). These aspects are core of any business process description as they provide a conceptual basis for the integration of other aspects. This restriction allows for the use of analysis methods checking some desired behavioral properties a process description should satisfy. However, it is clear that other properties, e.g. describing dependencies between resources, may not be investigated.

Beside finding a suitable abstraction level, another issue in modeling business processes is the level of uncertainty reflected by the description. Business process descriptions often remain vague or cover only parts of the actual processes. This is due to various reasons. Often there is only partial knowledge about the processes on the side of the performing actors - the knowledge may either be distributed over several actors, or (always oftener) hidden in the applications used. Some processes are just difficult to describe as they are only loosely structured or subject of constant change, such as processes incorporating a lot of communication or negotiation (c.f. ad-hoc and/or collaborative processes [Aal98]). Some level of uncertainty may even be intended, leaving room for interpretation, which in turn facilitates the effort of different people to agree on a common modeling result.

There are modeling approaches accenting uncertainty. Ideas to express uncertainty range from gradual refinement concepts to the use of fragments, partial descriptions of the business processes. Within this work a certain degree of uncertainty is accepted and specific refinement concepts are introduced for their coverage.

The objective of *workflow management* is to support the execution of business processes. The parts of a business process that can be supported by a computer system are called *workflows.* In the glossary of the Workflow Management Coalition

[Coa00] a workflow is explained as the automation of a business process.

A *workflow specification*[1] is a representation which supports automated manipulation, such as modeling or enactment by a workflow management system. A workflow specification mainly contains the same information as a business process description but at a more elaborated level of abstraction.

It defines a collection of tasks[2] and the order of task invocation. Furthermore, it contains information relevant to controlling and coordination of the execution of its constituent tasks (e.g. required skills, possible actors, associated IT applications, processed data, and execution requirements).

An instance of a workflow specification is denoted as a *case* (e.g. [Aal98]). In a case, concrete documents, information and/or tasks are passed to processing entities for action, according to the procedural rules determined in the workflow specification. An example of a case is the process that handles an order from Marie K. The case may be distributed over several processing entities. Thus, the creditworthiness of Marie K. may be checked in the accountancy while the ordered item is already being assembled by employees from the production department. Cases are handled by a *workflow management system*.

A workflow management system (WFMS) is a computer system that implements workflow management functionality. This covers the definition of workflow specifications, their analysis, their simulation and the monitoring of the corresponding cases (cf.[Law97, Fis01]). Definition, analysis and simulation are done at design time. However, the core of the WFMS-functionality is the monitoring at run-time.

Monitoring of cases comprises their control and their coordination. For each monitored case, the WFMS ensures that the tasks of the case are performed in the right order, at the right time and by the right processing entities. This is done by activating tasks, assigning the embedded activities to processing entities and waiting for the activity to be completed. Which tasks are enabled at a certain point in time depends on the workflow specification. Enforcement of rules in a workflow specification by a WFMS is called enactment [Coa00]. It is done by the workflow engine, sometimes also called workflow controller.

A WFMS system that is instantiated with one or more workflow specifications is called a *workflow system*, just like a database management system instantiated with one or more database schemas is called a database system. Until now, when we used WFMS we sometimes actually meant a workflow system. From now on,

[1]In the vocabulary of the Workflow Management Coalition (WFMC) a workflow specification is a process definition [Coa00]

[2]Tasks stand for activities assigned to some actor.

we will use the term workflow system whenever we mean a WFMS instantiated with a workflow specification.

"Business process re-engineering and workflow management are natural partners. The rise of workflow management systems is an "essential enabler" for business process re-engineering efforts. Conversely, some business re-engineering efforts result in the purchase of a workflow management system" [AH02a].

1.2 Problem statement

A description of a business process and the corresponding workflow specification are very similar. They both refer to the same set of activities and their ordering, but at different levels of abstraction. The differences are due to different objectives and a diverse perspective.

A business process description is made by domain experts. It describes the processes from a user perspective. Activities are depicted as active bits that are executed by actors.

The objective of a business process description is to provide a basis for communication. The descriptions are used for various purposes. In the everyday life of a company they serve as manuals for process participants or as learning material for newcomers. In business process re-engineering projects they provide a basis for discussion in order to detect optimization potential. In preparation for the use of a WFMS they provide a basis for agreeing on the processes to be supported.

The business process description must be understandable for people from very different backgrounds and "knowledge cultures", e.g. heads of departments, department staff, and IT experts. A business process description should be intuitive and leave room for interpretation: the more ways there are to interpret a certain construct the more likely it is that agreement will be reached.

A workflow specification, in contrast, is made by IT-experts. It describes the process to be supported from a monitoring perspective. Activities are no longer the active bits but are embedded within tasks. A task comprises the initiation of an activity (e.g. someones inbasket is filled) and the waiting for its completion. A workflow specification is used as input for a WFMS and must therefore be machine readable. The description here refines representation of behavior for subsequent monitoring. A workflow specification must be unambiguous and may not contain any uncertainties. This is a necessary requirement in order to analyze and simulate the described processes and to monitor their execution at run-time. A workflow specification also contains details that are close to implementation. Whereas it is sufficient for a business process description to cover the set of desired process executions, a workflow specification also determines how these executions are achieved. Thus, the workflow specification incorporates a strategy fixing the efficiency of the executions supported at run-time.

Both process descriptions[3] cover the same matter of interest: the involved activities and their order. The close relation between the two descriptions suggests deriving one from the other by changing the level of abstraction. Modeling workflows, it would be good to enhance existing business process descriptions such that they can be used as inputs for a WFMS.

So far, there is no methodically well-founded process model that bridges the gap between business process and workflow modeling. One reason can be found in a badly organized communication between domain- and IT-experts. But even if those involved work closely together the continuous use of business process descriptions for the modeling of workflow is impossible as there is neither a standard modeling language supporting the different abstraction levels nor an exchange format in combination with transformation rules to transform business process descriptions into workflow specifications.

There are languages that have proved to be understandable for the average user. They provide a set of graphical modeling elements which are combined in a straightforward manner. Their semantics is “intuitive” but not formally founded.

On the other hand, there is a variety of languages that satisfy the requirements posed by a workflow specification. They have formal, operational semantics and provide concepts to specify aspects close to implementation. Still, where the one language supports understandable modeling it lacks formal semantics; the other has formal semantics but meets with rejection from modelers. There have been many attempts to bridge the gap between the two concerns: the need for a plain communication basis on the one side and an elaborate process description, covering details of the implementation, on the other side.

Existing WFMSs follow a pragmatic approach. They often use a proprietary modeling language with an intuitive graphical layout. The underlying semantics lacks a formal foundation. As a consequence, analysis issues, such as the warranty of a correct and reliable execution are not supported at design time. Failure detection is only possible while monitoring the process execution. It is clear that failures that are only detected at run-time may be very costly and may not please customers.

[3]The generic term *process description* will be used for both business process description and workflow specification.

Many research approaches address this drawback of practical systems. Considerable efforts went into the formalization of semiformal modeling languages, such as Activity diagrams [Esh02, EW00, EW01b], Statecharts [Har87, WW97], and Event-driven Process Chains [Aal99, CS94, LSW98, MR00, NR02, Rit00b, Rum99, WWKD$^+$97]. Still, these approaches do not provide a solution. The formalization removes ambiguities and restricts the expressiveness. This moves the derived description more towards a suitable input for workflow management systems, but does not meet with acceptance from modelers. Various interpretations which supported reaching an agreement between the different participants were discarded.

Other approaches try to adapt and/or facilitate the use of formal languages, such as Petri nets [Pet62] or CCS (Calculus of Communicating Systems) [Mil80]. Here the most common approach is the introduction of intuitive graphical patterns replacing constructs of the primary language [Aal98, AHKB03, Mil99]. The goal is to facilitate the understanding of the determined interpretation.

Coming from either direction, the main idea is to define a comprehensive modeling language which meets all requirements, i.e. provides concepts that support the different levels of abstraction. As much as such a general language would simplify life, so far none of the proposed languages provides concepts to cover the different levels of abstraction separately, e.g. through a suitable refinement relation. Instead, aspects of both abstraction levels get mingled.

The goal of this thesis is not to propose “yet another modeling language”, which could bridge the gap, but to take a different and more pragmatic approach.

The idea is to propose a cross-language process model which gradually guides the modeler towards a sound workflow specification. Such a process model would not be based upon one general modeling language but would support the combination of different, existing, and accepted techniques. The proposed procedure would start with the modeling of a business process using an “intuitive”, but semiformal modeling language. The procedure would finally guide the modeler towards a sound workflow specification, which is given in terms of a formal language.

In the following we will discuss the requirements that such a cross-language process model must satisfy.

Transformation rules The use of different techniques for different concerns provides a good basis to meet the suitable level of abstraction. However, it requires paying particular attention to a smooth transformation between the techniques used. Within the process model, rules should be provided to support the transformation of a semiformal description into a formal one. The transformation should maintain the various interpretations of the first process description but should made them explicit.

Suitable correctness criteria The gradual refinement of a process description should also cover a refinement of correctness criteria. It is clear that the final workflow specification must be sound. This is a necessary requirement in order to guarantee reliable process execution at run-time. Still, such a strong correctness criterion restricts the modeling capabilities at the level of business process descriptions. Therefore, alleviated correctness criteria must be introduced which fit the requirements posed within business process modeling. Here, in order to support the communication between participants, different interpretations should be allowed. Only wrong interpretations should be excluded.

Enhancement of implementation-close details The gradual refinement of a business process description towards a workflow specification should furthermore include the enhancement of details that are close to implementation, e.g. efficiency aspects. Whereas a business process description only determines the relevant activities and their order, a workflow specification should also contain information determining how the order is enforced.

Change of perspective Finally, the process model should cover the change of perspective. From a users perspective, where activities were depicted as the active bits, the process specification must be changed towards a monitoring perspective. Activities must be embedded into tasks. From the machine perspective, activities are seen as passive. The workflow engine only initiates their execution and awaits their completion.

Having sketched the background and the problem statement, we now formulate the goal of this thesis as follows.

> Based on existing modeling languages that each fulfill their specific purpose, we will provide a methodically well-founded process model guiding the modeler from a business process description towards a sound workflow specification.

1.3 Problem solving approach

As language for the workflow specification we chose Petri nets. Their suitability for this application domain has been examined and discussed extensively in the literature (e.g. [Aal98, AAH98]). They combine a graphical representation with a precise formal foundation. Their operational semantics allow the use of the derived process descriptions right away as input format of a WFMS. Example of existing WFMSs working on the basis of Petri net descriptions are COSA (Software Ley/COSA Solutions [SL99]) or Income (Get Process AG [Inc]).

As modeling language for the business process description we refer to Event-driven Process Chains (EPCs), which are fairly widespread. Reasons for their acceptance can be found in their use for the representation of the SAP reference models [KT97] and their tool-support through the ARIS tool set [SJ02].

But, EPCs are just one of a rich variety of accepted business process modeling languages. We emphasize that the proposed process model is not restricted to that choice, but may be adapted for other semiformal techniques.

Five steps have to be completed when guiding the modeler from a semiformal business process description (based on EPCs) towards a sound workflow specification based on Petri nets:

1. Business process modeling (EPCs),
2. Transformation into WF-nets,
3. Correctness check and feedback,
4. Strategy determination & implementation, and
5. Control refinement.

The whole process model is illustrated in Figure 1.1. It has been designed to support a modeler who is probably a domain expert but does not necessarily have high modeling expertise.

The first three steps cover the modeling and the revision of the business process. Only when the resulting process description is correct it is refined until it fits the requirements of a workflow specification. The intermediate step "Transformation into WF-nets" was introduced to provide a formal basis for the application of correctness criteria.

Step four covers the determination of efficiency aspects. Here, an execution strategy is determined and implemented. In step five, the description is refined towards a monitoring perspective. Tasks are refined by decomposing them into three phases, covering the initiation of the embedded activity, its processing, and the waiting for the activity to complete.

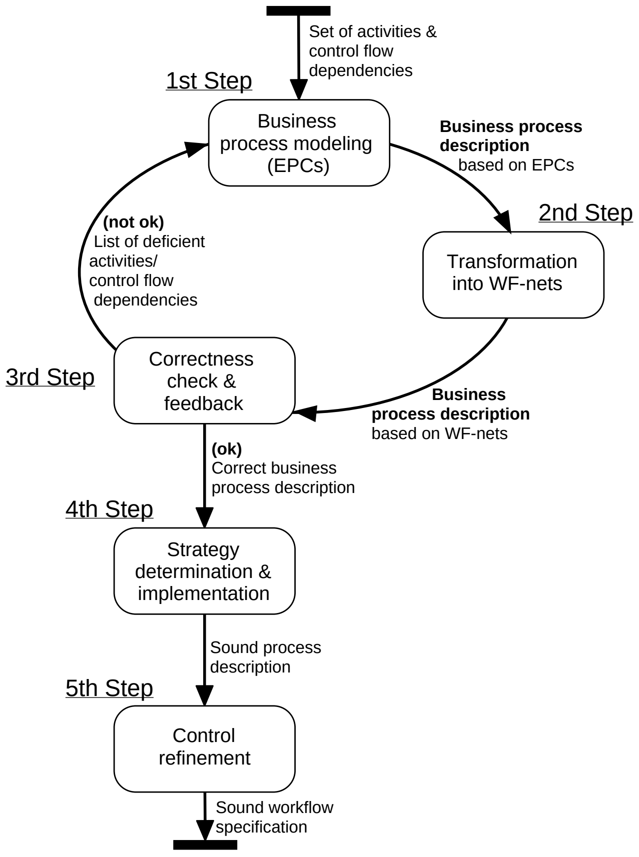

Set of activities & control flow dependencies
1st Step
Business process modeling (EPCs)
Business process description based on EPCs
2nd Step
Transformation into WF-nets
(not ok) List of deficient activities/ control flow dependencies
3rd Step
Correctness check & feedback
Business process description based on WF-nets
(ok) Correct business process description
4th Step
Strategy determination & implementation
Sound process description
5th Step
Control refinement
Sound workflow specification

Figure 1.1: A process model for workflow modeling

The key of the proposed process model lies in the correctness criteria for the business process descriptions. As mentioned earlier, the *soundness* criterion is only practical for workflow specifications. Applying soundness already on business process descriptions, their expressive power would be restricted. Instead of various possible interpretations only one would be considered correct. Therefore, less stringent correctness criteria were introduced, namely *relaxed soundness* and *robustness*. These two criteria, which describe a subset of soundness, provide an adequate correctness understanding for business process descriptions. Within the described process model, relaxed sound and robust process description are transformed into a sound specification. This is done in step four “Strategy Determination & Implementation”.

1.4 Thesis structure

The objective of this thesis is to provide a process model for workflow modeling. Starting with an informal description of the activities involved and their functional relation, the modeler will be guided towards a sound process description. The result will be used as a basis for the execution support of the modeled process. The structure of the thesis mainly follows the order of the five proposed steps (cf. Figure 1.1).

In Chapter 2 we review relevant concepts and results from Petri net theory. Workflow nets are introduced, a subclass of Petri nets suitable for workflow modeling. Existing correctness criteria are proposed and compared.

Chapter 3 focuses on the modeling, analysis, and revision of business processes. We present a new method appraising the quality of the derived process descriptions and providing corresponding feedback for the modeler. The method incorporates mapping EPCs to Petri nets and checking the correctness of derived WF-nets using a new correctness criterion, namely relaxed soundness. At the end of the chapter it is shown that the proposed method also applies to other modeling techniques, especially those used within real workflow applications.

In Chapter 4 the new criterion *relaxed soundness* is embedded into Petri net theory. Relations between existing properties and relaxed soundness are considered.

Chapter 5 describes a further part of the process model. Looking at a workflow system as a reactive system it is argued for another correctness criterion indicating that the process can react robustly to possible events coming from the environment. A corresponding property is introduced and an algorithm is provided for its verification. The algorithm is proven to be correct and complete. Finally, relations between robustness and other properties are discussed.

Chapter 6 focuses on the generation of sound process descriptions. It starts with an introduction of the theoretical background related to the theory of regions. Applying existing results, various ways are considered to generate a sound process description on the basis of only relaxed sound and robust specifications.

Chapter 7 focuses on the implementation of different scheduling strategies. Transforming a relaxed sound and robust process description into a sound process description, one scheduling strategy becomes fixed. Possible strategies and their implementation are discussed.

In Chapter 8 the whole process model is described in detail. The newly introduced concepts are embedded. Throughout the chapter the proposed procedure is applied to a running example. At the end of this chapter, we discuss the application of the derived process description as input for real WFMSs. This is illustrated by an example using Staffware, one of the leading workflow management systems.

The conclusions are presented in Chapter 9. The proposed process model bridges the gap between business process and workflow modeling. It enables the use of process descriptions made in business re-engineering projects for the modeling of workflow specifications. Passing through five steps, the domain expert is guided through refining a business process description until it can be used as input for a WFMS. There are various benefits that come with the proposed procedure. These, and major results of the research outcomes are summarized in the last chapter. Finally topics for future research are identified.

Chapter 2

Preliminaries

Within this work Petri nets are used to model workflows. Petri nets are one of the most popular formal models of concurrent systems. Their introduction goes back to the early 1960s ([Pet62]). Since then, there have been tremendous developments in both theory and applications. The latest compilation (March 2003) of the scientific literature related to Petri nets (http://www.daimi.au.dk/PetriNets/bibliographies/pnbibl.html) contains more than 8000 entries from hundreds of authors all around the world. Well-known introductions to the application and theory of Petri nets are [Mur89, Rei85] and [Jen92, Jen95]. A more recent survey can be found in [RR98a, RR98b].

Petri nets have been a popular choice for many application domains. This development is based on their graphical representation, their excellent formal foundation, and the rich variety of existing analysis techniques and tools.

In this chapter, concepts and results from Petri net theory are reviewed which will be used later in the thesis. In the first section, Place/Transition nets are defined and their structural and dynamical properties are outlined. Some results relating the properties are recalled from Petri net theory. In Section 2.2, WF-nets, a subclass of Place/Transition nets, are introduced and relevant properties are discussed. The chapter finishes with a section on related work.

2.1 Place/Transition nets

A Petri net is a directed bipartite graph. The two sorts of nodes are called places and transitions. Places are represented by circles, and transitions by boxes.

Definition 2.1 (Place/Transition net).
A Place/Transition net (P/T net) is a triple (P, T, F): P is a finite set of places, T is a finite set of transitions $(P \cap T = \emptyset)$, $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation)[1].

Given a node x of $P \cup T$, the set ${}^\bullet x = \{y | (y, x) \in F\}$ is a *preset* of x. The set $x^\bullet = \{y | (x, y) \in F\}$ is a *postset* of x. The elements in the preset (postset) of a place are called its input (output) transitions. Similarly, the elements in the preset (postset) of a transition are its input (output) places.

Given a set X of nodes, $X \subseteq P \cup T$, we define ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$. Given two sets of nodes X and Y, $X \setminus Y$ denotes the set of nodes of X that do not belong to Y.

2.1.1 Structural properties of nets

In this section, concepts describing structural properties of Petri nets are introduced. Structural properties are properties that depend only on the network structure of the net. Since, Petri nets can be viewed as special graphs, graph terminology also applies to nets.

Definition 2.2 (Path in PN).
In the Petri net $PN = (P, T, F)$, a path from a node x_0 to a node x_n is a non-empty sequence $(x_0, \ldots, x_n)$ such that $(x_i, x_{i+1}) \in F$ for $0 \leq i \leq n - 1$. A path $(x_0, \ldots, x_n)$ is said to lead from x_0 to x_n. Note that a sequence containing one element is an empty path.

Elementary paths are of special interest.

Definition 2.3 (Elementary).
A path $(x_0, \ldots, x_n)$ in the Petri net $PN = (P, T, F)$ is elementary iff $x_i \neq x_j$ for any two nodes $x_i, x_j, 0 \leq i < j \leq n$.

Definition 2.4 (Strongly connected).
A Petri net is strongly connected iff for every pair of nodes x and y, there is a path leading from x to y.

Special combinations of elementary paths are called handles. The concept of handles has been introduced first in [ES90].

[1]Unless stated otherwise we always refer to ordinary Place/Transition nets with arc weights equal to one.

Definition 2.5 (PT-handle, TP-handle).
In the Petri net $PN = (P, T, F)$, a place-transition pair $(p, t) \in P \times T$ is a PT-handle if there are two elementary paths from p to t sharing only the two nodes p and t; a transition-place pair $(t, p) \in T \times P$ is a TP-handle if there are two elementary paths from p to t sharing only the two nodes p and t.

As handles often denote potential problems in the design flow, a concept for their non-existence is needed.

Definition 2.6 (Well-handled).
A Petri net is well-handled if it has no PT-handles and no TP-handles.

Another important graph property concerns the existence of cycles.

Definition 2.7 (Cycle-free, pure).
A Petri net (P, T, F) is cycle-free, if for any node x there is no non-empty path leading from x to x. A Petri net (P, T, F) is pure iff $F \cap F^{-1} = \emptyset$[2].

In a pure net, for each transition t holds: ${}^\bullet t \cap t^\bullet = \emptyset$ (no self loops). This property is important for the analysis of nets, because it means that there is a one-to-one correspondence between the net and its incidence matrix (see Def. 2.24).

The next definition provides a way to express the absence of redundancy.

Definition 2.8 (t-simple).
A Petri net is t-simple when no two transitions t and t' have the same sets of input and output places, i.e. $\forall t, t'$ ${}^\bullet t \neq {}^\bullet t'$ or $t^\bullet \neq t'^\bullet$

Important subclasses of Petri nets are state machines, marked graphs, and free-choice nets, each with specific constraints on the graphical structure.

In a state machine all transitions have exactly one input and one output place.

Definition 2.9 (State machine).
A Petri net $PN = (P, T, F)$ is a state machine iff $\forall t \in T : |{}^\bullet t| = |t^\bullet| = 1$.

A Petri net is called a marked graph if all places have exactly one input and one output transition.

Definition 2.10 (Marked graph).
A Petri net $PN = (P, T, F)$ is a marked graph iff $\forall p \in P : |{}^\bullet p| = |p^\bullet| = 1$.

[2] $F^{-1} = \{(y, x)|(x, y) \in F\}$ is the inverse of F

In free-choice nets [DE95], both synchronization and conflict are allowed, but may not interfere, i.e. either a choice is preceded by a synchronization or synchronization is preceded by a choice. This way the result of a choice between two transitions is not influenced by the rest of the net.

Definition 2.11 (Free-choice nets).
A Petri net $PN = (P, T, F)$ is a free-choice net (basically extended free-choice) iff $\forall t, t' \in T : {}^\bullet t \cap {}^\bullet t' = \emptyset \vee {}^\bullet t = {}^\bullet t'$.

The next definition introduces the subnet concept. It is used to define *S- and T-components* and related Petri net properties.

Definition 2.12 (Subnet).
The Petri net $PN' = (P', T', F')$ is a subnet of net $PN = (P, T, F)$ iff $P' \subseteq P, T' \subseteq T$, and $F' = F \cap ((P' \times T') \cup (T' \times P'))$.

Definition 2.13 (S-component).
Subnet $PN' = (P', T', F')$ is an S-component of the Petri net $PN = (P, T, F)$ iff PN' is a strongly connected state machine such that $\forall p \in P' : {}^\bullet p \cup p^\bullet \subseteq T'$.

Definition 2.14 (S-coverability).
A Petri net $PN = (P, T, F)$ is S-coverable iff for each place $p \in P$ there is an S-component $PN' = (P', T', F')$ of PN such that $p \in P'$.

Definition 2.15 (T-component).
Subnet $PN' = (P', T', F')$ is a T-component of the Petri net $PN = (P, T, F)$ iff PN' is a strongly connected marked graph such that $\forall t \in T' : {}^\bullet t \cup t^\bullet \subseteq P'$.

Definition 2.16 (T-coverability).
A Petri net (P, T, F) is T-coverable iff for each transition $t \in T$ there is a T-component (P', T', F') of PN such that $t \in T'$.

2.1.2 Execution of Petri nets

In this section markings are introduced and the firing rule, by means of which a net is transformed into a dynamic system. A place can contain zero or more *tokens*. Tokens are represented by black dots. The global *state* of a Petri net is called the *marking*.

Definition 2.17 (Markings).
The marking of a Petri net (P, T, F) is the distribution of tokens over places $M : P \longrightarrow \mathbb{N}$ that assigns to every place the number of tokens $M(p)$ that reside in p. A place p is marked at a marking M iff $M(p) > 0$.

A marking M changes by firing a transition t. A transition t may fire only if it is *enabled.* A transition t is *enabled* in marking M, written $M \xrightarrow{t}$ iff every input place of t contains at least one token. If a transition t is enabled in marking M, it may fire: one token is removed from every input place and one token is added to every output place.

Definition 2.18 (Firing rule).
A transition $t \in T$ is said to be enabled by the marking M iff $\forall p \in {}^\bullet t : M(p) \geq 1$. In this case t can fire. Firing of transition t leads from marking M to marking M', where

$$M'(p) = \begin{cases} M(p) - 1 & \textit{if } p \in {}^\bullet t \setminus t^\bullet, \\ M(p) + 1 & \textit{if } p \in t^\bullet \setminus {}^\bullet t, \\ M(p) & \textit{otherwise} \end{cases}$$

written as $M \xrightarrow{t} M'$.

We write $M \longrightarrow M'$ to indicate that by firing one transition in M marking M' can be reached. Next, we define the concepts firing sequences and reachable markings.

Definition 2.19 ((Finite) Firing sequence).
A sequence of transitions $\sigma = t_1 t_2 t_3 ...$ is a firing sequence, enabled at M_1 *if there are markings $M_2, M_3 ...$, such that $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} M_3 \xrightarrow{t_3}$ ... We write $M_1 \xrightarrow{\sigma}$. A finite sequence of transitions $\sigma = t_1 t_2 ... t_{n-1}$ gives a* finite firing sequence, enabled at M_1. *Here, we write $M_1 \xrightarrow{\sigma} M_n$.*

Note, the empty sequence ϵ is enabled at any marking M and satisfies $M \xrightarrow{\epsilon} M$.

Definition 2.20 (Reachable marking).
A marking M_n is reachable *from M_1 (notation $M_1 \xrightarrow{*} M_n$) if there is a finite firing sequence σ such that $M_1 \xrightarrow{\sigma} M_n$.*

The set of markings reachable from a marking M is denoted as $R_{PN}(M)$:
$R_{PN}(M) = \{M' | M \xrightarrow{*} M'\}$.

Net system

To emphasize the difference between the static and the dynamic level of a net, the concept of a net system is introduced.

Definition 2.21 (Net system).
A net system S (or just a system) is a pair (PN, M_i), obtained by associating an initial marking M_i to the Petri net PN.

Note, that in this thesis properties are sometimes transfered from nets to systems, e.g. saying that a system is well-handled also implies that the underlying Petri net is well-handled.

Reachability graph

The behavior of a Petri net can be described via a labeled transition system. A labeled transition system is a directed graph with nodes representing states and edges representing state transitions. The edges of the graph are labeled. The label denotes what happens when the action represented by the edge is taken. Most transition systems have a distinguished node, indicating the initial state.

Definition 2.22 (Transition system).
A Transition System (TS) is a quadruple $TS = (V, L, E, v_{in})$, where V is a nonempty set of states, L is a set of labels, $E \subseteq V \times L \times V$ is a transition relation, and $v_{in} \in V$ is an initial state. A transition system is finite if V and L are finite.

The reachability graph of a system $S = (PN, M_i)$ is the transition system where the states are the reachable markings, the labels are the transitions, the distinguished initial state is the initial marking and the labeled edges are all triples (M, t, M') such that M, M' are reachable markings satisfying $M \xrightarrow{t} M'$.

Definition 2.23 (Reachability graph).
For the system $S = (PN, M_i)$ with $PN = (P, T, F)$, the reachability graph $RG_S = (V, L, E, v_{in})$ is a transitions system with:

$V = R_{PN}(M_i)$ as set of states, $L = T$ and $E = \{(M, t, M') | M, M' \in R_{PN}(M_i) \wedge t \in T \wedge M \xrightarrow{t} M'\}$ as set of labeled edges, and $v_{in} = M_i$ as initial state.

As the set of labels and the initial state are implicitly specified by the system $S = (PN, M_i)$ the reachability graph RG_S is determined by the set of reachable markings V together with the set of edges E. We therefore use the shortcut notation: $RG_S = (V, E)$.

There is a direct correspondence between a path in the reachability graph and a firing sequence in the Petri net. A path π of the reachability graph is a sequence (finite or infinite) of labeled edges:

$$\pi = (M_1, t_1, M_2)(M_2, t_2, M_3)(M_3, t_3, M_4)...$$

As the path corresponds to the firing sequence $t_1 t_2 t_3...$ enabled in M_1, we also write $M_1 \xrightarrow{\pi}$. Correspondingly, the elements of E will often be denoted by $M \xrightarrow{t} M'$ instead of (M, t, M').

Through the edges of the reachability graph RG a predecessor/successor relationship is defined on the elements of V. M_1 is *predecessor* of M_n if there is a path leading from M_1 to M_n ($M_1 \xrightarrow{*} M_n$). We call M the *immediate predecessor* of M' if there is an edge leading from M to M' ($M \longrightarrow M'$).

The functions $pred_{RG}, Pred_{RG} : V \longrightarrow 2^V$ denote the set of (immediate) predecessors of a state $M \in V$.

$$\begin{aligned} &pred_{RG}(M) := \{M' | M' \in V \wedge M' \longrightarrow M\} \\ &Pred_{RG}(M) := \{M' | M' \in V \wedge M' \xrightarrow{*} M\} \end{aligned}$$

By analogy, the functions $succ_{RG}$ and $Succ_{RG}$ are defined, denoting the sets of immediate successors and successors for a state $M \in V$.

$$\begin{aligned} &succ_{RG}(M) := \{M' | M' \in V \wedge M \longrightarrow M'\} \\ &Succ(M)_{RG} := \{M' | M' \in V \wedge M \xrightarrow{*} M'\} \end{aligned}$$

The set of successors of a marking M in a system $S = (PN, M_i)$ is of course equivalent to the set of reachable markings from M, $Succ_{RG}(M) = R_{PN}(M)$.

$Pred_{RG}(o)$ denotes the set of markings $M \in R_{PN}(M_i)$ such that there is a path from M to o, $M \xrightarrow{*} o$. Clearly, $Pred_{RG}(o)$ is only defined if $o \in R_{PN}(M_i)$.

The construction of the reachability graph RG is straightforward, although termination cannot be guaranteed, because it might be infinite. This is the case if the corresponding Petri net is unbounded (cf. Def. 2.33), i.e. contains places which have no limit to the number of tokens. As a consequence the number of markings in the reachability graph is infinite.

The incidence matrix of a net

A more efficient way to analyze the behavior of a Petri net is based on the introduction of the *incidence matrix*. The sets of transitions and places within a Petri net are finite. The definition of markings therefore suggests using a vector notation. The marking $M : P \longrightarrow I\!N$ can be represented as a vector of length $|P|$.

Definition 2.24 (Incidence matrix of a net).
For the Petri net $PN = (P, T, F)$, the incidence matrix $\mathbf{C} : (P \times T) \longrightarrow \{-1, 0, 1\}$ *is defined by*

$$\mathbf{C}(p,t) = \begin{cases} -1 & \textit{if } p \in {}^{\bullet}t \setminus t^{\bullet}, \\ +1 & \textit{if } p \in t^{\bullet} \setminus {}^{\bullet}t, \\ 0 & \textit{otherwise} \end{cases}$$

In an incidence matrix, the rows represent the places of a Petri net and the columns represent the transitions. The column vector $P \longrightarrow \{-1, 0, 1\}$ of $\mathbf{C}$ associated to a transition t is denoted by $\mathbf{t}$. Similarly, the row vector $T \longrightarrow \{-1, 0, 1\}$ associated to a place p is denoted by $\mathbf{p}$.

The entry $\mathbf{C}(p, t)$ corresponds to the change of the marking of the place p caused by the occurrence of the transition t. Hence if t is enabled at a marking M and $M \xrightarrow{t} M'$ then $M' = M + \mathbf{t}$. For a generalization of this equation to sequences of transitions, we need the following definition:

Definition 2.25 (Parikh vectors of transition sequences).
For the Petri net $PN = (P, T, F)$ and a finite sequence of transitions σ, the Parikh vector $\vec{\sigma} : T \rightarrow I\!N$ of σ maps every transition t of T to the number of occurrences of t in σ.

With this definition, the Marking Equation Lemma can be formulated.

Lemma 2.26 (Marking equation lemma).
For every finite firing sequence $M \xrightarrow{\sigma} M'$ of a Petri net PN the following marking equation holds:

$$M' = M + \mathbf{C} \cdot \vec{\sigma}$$

One of the structural properties of Petri nets are the net invariants, namely *transition invariants* (Def. 2.27) and *place invariants* (Def. 2.30).

Transition invariants are related to firing sequences which reproduce a marking. A transition invariant is a mapping that assigns an integer to each transition denoting how many times the corresponding transition must fire in order for a marking to be repeated. Note that the order of the transitions in the firing sequence is lost if vector notation is used.

Definition 2.27 (Transition invariant).
A transition invariant (T-invariant) of a Petri net $PN = (P, T, F)$ is a vector $Y \neq \vec{0}$ that satisfies the following equation:

$$\mathbf{C} \cdot Y = \overbrace{(0, 0, \ldots 0)}^{|P|}$$

A T-invariant Y of a Petri net is called *semi-positive* if $Y \geq \vec{0}$ and $Y \neq \vec{0}$ [3]. A T-invariant is called *positive* if $Y > \vec{0}$, i.e. $Y(t) > 0$ for every transition t. The support of a semi-positive T-invariant Y, denoted by $\langle Y \rangle$, is the set of transitions t satisfying $Y(t) > 0$.

A semi-positive T-invariant Y is minimum if no semi-positive invariant X satisfies $\langle X \rangle \subset \langle Y \rangle$. T-invariants have the following fundamental property. Let M and M' be markings of a Petri net PN, and let σ be a sequence of transitions such that $M \xrightarrow{\sigma} M'$. We have $M = M'$ iff the Parikh vector $\vec{\sigma}$ is a T-invariant of PN.

The other way around does not hold. Not every T-invariant corresponds to a firing sequence that reproduces a specific marking. The T-invariant must furthermore be *realizable* [Lau02] with respect to a reachable marking of the net system.

Definition 2.28 (Realizable T-invariant).
A T-invariant $Y \geq \vec{0}$ of a system $S = (PN, M_i)$ with $PN = (P, T, F)$ is realizable in S iff there is a marking M reachable from M_i and a firing sequence σ with $M \xrightarrow{\sigma}$ and $\vec{\sigma} = Y$.

A Petri net $PN = (P, T, F)$ is said to be covered by T-invariants if for any $t \in T$ there is a semi-positive T-invariant Y which supports t, $t \in \langle Y \rangle$.

Proposition 2.29 (T-components induce minimum T-invariants).
Let $N' = (P', T', F')$ be a T-component of $PN = (P, T, F)$. Then Y, with

$$Y(t_i) = \begin{cases} 1 & : \quad t_i \in T' \\ 0 & : \quad \textit{otherwise} \end{cases}$$

is a minimum T-invariant of PN.

Place invariants are related to sets of places whose weighted token sum always remains constant. Place invariants are represented by an n-column vector X, where n is the number of places of the net whose non-zero entries correspond to the places that belong to the particular place invariant, with zeros everywhere else.

[3]We write $X \geq Y$ ($X > Y$) if $X(a) \geq Y(a)$ ($X(a) > Y(a)$) for every element a of a finite set A, where X and Y are mappings from A to $\mathbf{Q}$, with $\mathbf{Q}$ denoting the set of rational numbers.

Definition 2.30 (Place invariant).
A place invariant (P-invariant) of a Petri net is a vector X that satisfies the following equation:

$$X^T \cdot \mathbf{C} = 0, \text{ for all } M \in R_{PN}(M_i)$$

where X^T is the transpose of vector X.

A place invariant has the fundamental property that the weighted token sum in the places of the invariant remains constant at all markings and this sum is invariant given the initial marking M_i of the net. Therefore place invariants satisfy the following equation:

$$M \cdot X = M_i \cdot X$$

The concepts semi-positive and positive P-invariant are defined as for T-invariants.

There are two sets of places that are interesting for specific analysis issues: siphons and traps.

Definition 2.31 (Siphon).
A set R of places, $R \subseteq P$, of a Petri net $PN = (P, T, F)$ is a siphon if ${}^{\bullet}R \subseteq R^{\bullet}$. A siphon is called proper if it is not the empty set. A siphon R is called minimum if there is no siphon R' with $R' \subset R$.

The characteristic property of a siphon is that once unmarked it remains unmarked.

Definition 2.32 (Trap).
A set R of places, $R \subseteq P$, of a Petri net $PN = (P, T, F)$ is a trap if $R^{\bullet} \subseteq {}^{\bullet}R$. A trap is called proper if it is not the empty set. A trap R is called minimum if there is no trap R' with $R' \subset R$.

Once marked a trap remains marked.

2.1.3 Dynamic properties of net systems

In this section, some dynamic properties of net systems (boundedness, liveness, and related properties) are reviewed, followed by selected relations between structural and dynamic properties of net systems.

Definition 2.33 (Bounded, safe).
A system (PN, M_i) is bounded iff for each place p there is a natural number b such that, for every reachable marking, the number of tokens in p is less than or equal to b (b-bounded). The system is safe iff for each place the maximum number of tokens does not exceed 1 (1-bounded).

Definition 2.34 (Structurally bounded).
A Petri net PN is structurally bounded if it is bounded for any initial marking.

Definition 2.35 (Dead, live transition).
Let (PN, M_i) be a system. A transition $t \in T$ is

- *dead if there is no marking M reachable from M_i which enables t.*
- *live if t can always fire again. From every reachable marking M a marking M' is reachable ($M \xrightarrow{*} M'$) which enables t.*

A system (PN, M_i) is live if every transition is live. A system (PN, M_i) is deadlock-free if every reachable marking enables at least one transition.

Definition 2.36 (Structurally live).
A Petri net PN is structurally live if there is a marking M_i of PN such that (PN, M_i) is a live system.

Definition 2.37 (Well-formed).
A Petri net PN is well-formed if it is structurally bounded and structurally live.

The results presented in the following describe relations between structural and dynamic properties of net systems. The first and the second result stem from [ES90] and pertain to the absence of handles.

Theorem 2.38 ([ES90] Theorem 3.1).
Let $PN = (P, T, F)$ be a strongly connected net. If PN has no TP-handle, PN is structurally bounded.

Theorem 2.39 ([ES90] Theorem 3.2).
Let $PN = (P, T, F)$ be a strongly connected Petri net without TP- and PT-handles.

a) PN is structurally live.

b) PN is covered by S-components

c) PN is covered by T-components

The next results were presented in [DE95]. They refer to the dynamic behavior that can be deduced assuming restricted net-structures, such as state machines (cf. Def. 2.9) and marked graphs (cf. Def. 2.10).

Theorem 2.40 (Liveness theorem, [DE95] Theorem 3.3).
Let $PN = (P, T, F)$ be a state machine. The system (PN, M_i) is live iff PN is strongly connected and M_i marks at least one place.

Theorem 2.41 (Boundedness theorem, [DE95] Theorem 3.5).
Let $PN = (P, T, F)$ be a state machine and let the system $S = (PN, M_i)$ be live. (PN, M_i) is b*-bounded iff $\sum_{p \in P} M_i(p) \leq b$.*

From this theorem the following corollary was derived (cf.[DE95] Corollary 3.19).

Corollary 2.42 (Place bounds in live T-systems).
Let (PN, M_i) be a live marked graph. (PN, M_i) is bounded iff PN is strongly connected[4].

Theorem 2.43 (Liveness in strongly connected T-systems, [DE95] Theorem 3.17).
Let $PN = (P, T, F)$ be a strongly connected marked graph and (PN, M_i) be a system. The following statements are equivalent:

(a) (PN, M_i) is live.

(b) (PN, M_i) is deadlock-free.

(c) (PN, M_i) has an infinite firing sequence.

The final theorem refers to free-choice nets.

Theorem 2.44 ([DE95] Theorem 5.8).
Let PN be a well-formed, and free-choice net.

(a) PN has a positive P-invariant.

(b) Every system (PN, M) is bounded.

(c) A system (PN, M) is live iff every S-component of PN is marked at M.

[4]Note that a T-system [DE95] is a strongly connected marked graph with an initial marking.

2.2 Workflow nets

In [Aal97] Petri net theory is applied to process modeling and workflow nets are introduced (WF-net). 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 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 nodes (i.e. 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.

Definition 2.45 (WF-net).
A Petri net $PN = (P, T, F)$ is a WF-net, if:
(i) PN has two special places, i and o. Place i is the only source ($^\bullet i = \emptyset$) and place o is the only sink ($o^\bullet = \emptyset$).
(ii) Let $t^* \notin T$. The short-circuited net $\overline{PN} = (P, T \cup \{t^*\}, F \cup \{(o, t^*), (t^*, i)\})$ is strongly connected.

Figure 2.1 gives an example of a WF-net. It describes the processing of complaints[5].

Considering the behavior of a WF-net, we will always investigate the life-cycle of a single case, thus consider systems where initially only the source place i is marked ($M_i(i) = 1$ and for all $p \in P \setminus \{i\} : M_i(p) = 0$). So if $PN = (P, T, F)$ is a WF-net, we will investigate the behavior of the corresponding WF-system (PN, i)[6]. The relation between a WF-net PN and a WF-system (PN, i) is used to transfer properties from nets to systems, e.g. saying that a WF-net is sound implies that the corresponding WF-system is sound.

In the following, existing correctness criteria for WF-nets are presented. We will look at the structural property *well-structuredness* and the dynamic property *soundness.*

[5] A different version of this example was presented in [Aal98].

[6] Note that there is an overloading of notation: the symbols i and o are used both as identifiers for the start and end places as well as to depict the markings where these places contain one token. The relevant meaning can be derived from the context.

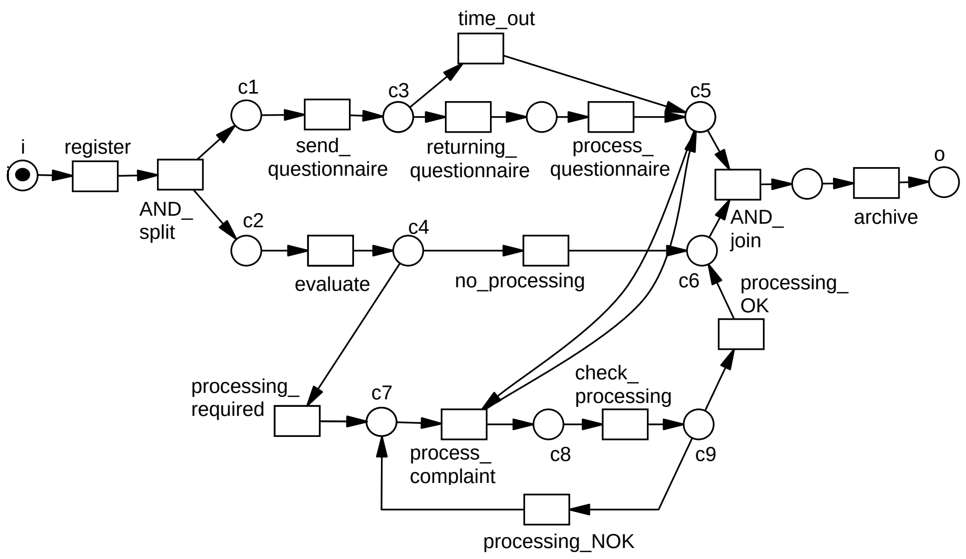


Figure 2.1: A WF-net modeling the processing of complaints

2.2.1 Structural and dynamic properties of WF-nets

Many of the Petri net properties introduced in Sections 2.1.1 and 2.1.3 hold for strongly connected nets, e.g. T- and S-coverability and liveness. WF-nets are not strongly connected but have one input and one output place.

A possibility to apply existing properties to WF-nets consists in short-circuiting the net by adding a single transition t^* ($t^* \notin T$), which connects place o and place i. The short-circuited net $\overline{PN}$ is used to infer to the behavior of the primary WF-net, e.g. saying that a WF-net is live implies that the short-circuited net is live. In this way, most of the properties can be applied to WF-nets.

The case is different for properties that do not refer to strongly connected nets. Here, it is not possible in all cases to infer from the validity of a property in the short-circuited net to its validity in the primary WF-net, but for most of the properties this relation is easy to prove or to disprove. For example, it is obvious that the WF-net is free-choice or t-simple if the short-circuited WF-net is. The additional transition t^* establishes a new connection between two places. Its introduction neither introduces a non-free-choice conflict nor violates the *t-simple*-condition (see Def. 2.8). But the relation does not hold for the absence of cycles (*cycle-free*) or for *boundedness.* This can be confirmed by a simple counter-example like the one shown in Figure 2.2. The primary WF-net PN is bounded and cycle-free, whereas

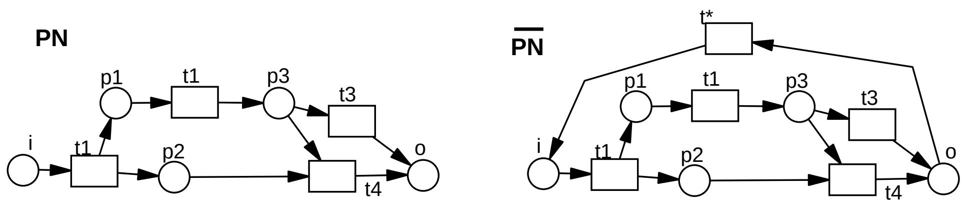


Figure 2.2: A simple counter-example

the short-circuited net $\overline{PN}$ satisfies neither of these properties.

Since, there is no proof (or disproof) that the short-circuited net $\overline{PN}$ is *well-handled* if the primary WF-net PN is, in [Aal97] a new concept, namely *well-structuredness* was introduced.

Definition 2.46 (Well-structured).
A WF-net PN is well-structured if the short-circuited net $\overline{PN}$ is well-handled.

The WF-net of Figure 2.1 is not well-structured. There are both PT- and TP-handles. Examples are the transition-place pair *AND_split/c5* and the place-transition pair *c5/AND_join.*

An important correctness criteria for WF-nets is *soundness* as introduced in [Aal98]. Soundness ensures that the process can *always* terminate properly, i.e with a single token in place *o* and all the other places empty, in addition, it requires that there are no dead tasks, i.e. each existing task can be executed.

Next, we show soundness as defined in [Aal98].

Definition 2.47 (Soundness).
A WF-system $S = (PN, i)$ is sound iff:

(i) *For every state M reachable from state i, there is a firing sequence leading from state M to state o (option to complete).*
Formally: $\forall M : (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o)$.

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

(iii) *There are no dead transitions in S.*
Formally: $\forall t \in T \exists M, M' : (i \xrightarrow{*} M \xrightarrow{t} M')$

The WF-system from Figure 2.1 is sound. In [Aal97] a necessary and sufficient condition for soundness is given:

Theorem 2.48.
Let PN be an arbitrary WF-net and $\overline{PN}$ the corresponding short-circuited net. (PN, i) is sound iff $(\overline{PN}, i)$ is live and bounded.

2.3 Related work

The use of Petri nets for the application domain workflow management started in the late 1970s ([Ell79, EN93]). Their suitability for this application domain has been examined and discussed extensively in the literature (e.g. [Aal98, AH02b]), and the interested reader is referred to [Obe96, Aal97, AH02b] for further reading.

For the analysis of the modeled processes a wide variety of results from Petri net theory are available. In general it is checked whether a given process description meets a specification e.g. given by a temporal formula. As correctness notion for WF-nets we referred to *soundness.* In [Aal98] it was argued that this property covers a minimum set of requirements every process description should satisfy.

Soundness [Aal98] is a combination of three conditions, stating that:

(i) option to complete it should always be possible to complete a case that is handled according to the process,

(ii) proper termination it should not be possible that the workflow process signals completion of a case while there is still work in progress for this case, and

(iii) no dead tasks for every task, there should be an execution (firing sequence) of the process that executes it.

In most approaches using WF-nets these three conditions have been regarded as pivotal with respect to a correctness statement. However, in [Kin98] it was argued that in some applications a stronger version of soundness is required. A fourth condition was proposed, namely safeness. The enhanced version was called *strong soundness.* The concept of strong soundness was applied for example in [AB02] and [Aal02].

In the context of inter-organizational workflow, the soundness criterion was carried over to compound workflow nets [Aal02] and called *global soundness.* This concept was adopted and changed slightly in [KMR00]. Here, a compound WF-net is called globally sound if all the workflow modules *which were invoked* (i.e. where the token was removed from its start place) will eventually terminate properly.

A less stringent interpretation of soundness can be find in [HSV03]. Here soundness only refers to the first two requirements (*option to complete* and *proper termination*). The third requirement (*absence of dead task*) is omitted. Based on this interpretation, soundness is generalized to the notion of *k-soundness*, allowing the observation of more than one instance (namely k) at the same time. A WF-net is called k-sound if any marking reached from k tokens in the initial place can reach k tokens in the final place. Regarding the original soundness as 1-soundness, a WF-net is called sound iff it is k-sound for each $k > 0$.

For further correctness notions applied to process descriptions (possibly not specified in terms of WF-nets) the reader is referred to the related work section of Chapter 3.

Chapter 3

Modeling business processes

In this chapter, the basis for the process model (cf. Figure 1.1) is established. In the first step the business process is modeled from a user perspective. We assume the person carrying out the modeling to be a domain expert but perhaps without modeling expertise. A graphical but only semiformal modeling language is used.

As a typical representative of such a language we will refer to Event-driven Process Chains (EPCs). To appraise the correctness of the modeled processes, EPCs are provided with formal semantics. This is done by defining a mapping from EPCs onto Petri nets. Throughout the transformation, the ambiguities contained in the primary description are maintained but made explicit. The corresponding Petri net will be checked for correctness, and feedback will be given to revise the primary EPC. As a measure of correctness, a pragmatic correctness criteria is proposed which reflects the remit for the initial modeling.

The chapter is structured accordingly. After a short introduction (Section 3.1) the syntax of EPCs is introduced in Section 3.2. Subsequently, the current debate about their semantics is reviewed and a proposal is made. In Section 3.3, rules are provided for the transformation of EPCs into Petri nets. The derived Petri net is checked for correctness in Section 3.4. The suitability of existing Petri net properties is discussed and an adapted correctness criteria is proposed. In the following sections, the state of implementation is described (Section 3.5) and a case study is evaluated (Section 3.6). In Section 3.7 the proposed method is summarized and its application to other modeling techniques is outlined. Finally related work is discussed.

3.1 Introduction

Business processes play a central role in the reorganization of a company. Their modeling has been at the heart of IS development for many years. The number of different languages that have been proposed is correspondingly large. Many of these modeling techniques have been investigated quite thoroughly. Examples are Activity Diagrams and Statecharts (both from the UML), Petri nets, or EPCs. Besides these, there are numerous languages that have been developed independently, often in an ad hoc style.

The general objective of all these languages was to find a suitable level of abstraction which meets the intuition of the domain experts. The meaning of a given description should be evident. This supports the handling of the business process descriptions. They shall be made and read by an average user without high modeling expertise. A common understanding of business process descriptions is important, as they are primarily used as basis for communication between the participants. The communication should also be possible between people with totally different backgrounds and knowledge cultures, such as CEOs, heads of department, department staff, IT experts, and so on.

The modeling languages are often only semiformal. This way, there is room for interpretation: the more ways there are to interpret a certain construct the more likely it is that an agreement will be reached. The participants might not (yet) be ready to specify the final behavior in detail and decide on the correct interpretation.

A semiformal language can introduce ambiguity and vagueness which constitute a major problem if the processes described are to be supported by a workflow management system. What has been an advantage for the purpose of communication then becomes a drawback in the design phase of a workflow system. Here, an unambiguous and machine-readable description of the process is needed.

As a way out of this dilemma, we propose the use of a pragmatic criterion to determine the correctness of the business process description. The proposed property describes the capability of a process description to cover sufficient useful interpretations. If it comes to system development, the process description is enhanced by further details, determining the final interpretation.

To apply the proposed correctness criterion, the primary language has to be provided with formal semantics. This is done by mapping it onto Petri nets.

As typical representative for a business process modeling language, we refer to EPCs of the Architecture of Integrated Information Systems (ARIS) [Sch94]. EPCs are fairly widespread. One reason for their acceptance lies in their use for the representation of the SAP reference models [KT97]. EPCs are a graphical and semiformal modeling language. They are easy to learn and to understand but involve ambiguity and vagueness.

3.2 Modeling with EPCs

In the following, EPCs are described in detail. First their syntax is presented. After that, various perceptions of their semantics are reviewed and a proposal is made.

3.2.1 EPC syntax

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. Figure 3.1 depicts the EPC elements that are used to describe the control-flow of a business process. *Functions* are used to model the dynamic part of the process. Typical functions are procurement, quality assurance, or processing an invoice. 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 behavior, such as sequential, conditional, parallel, and iterative routing, *connectors* are introduced. These fall into two categories: splits and joins. In both categories there are AND ($\wedge$), XOR ($\times$), and OR ($\vee$) connectors.

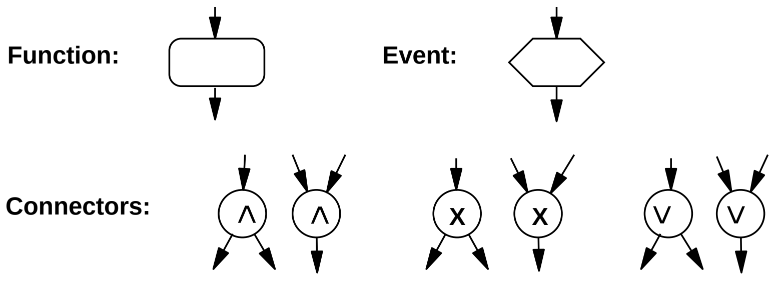


Figure 3.1: EPC elements

The elements are connected to form a complex process model. The composition is restricted by some syntactical rules, such as:

- There is at least one start and one end event.
- Events and functions have exactly one incoming and one outgoing arc (except start and end events).
- For every two elements there is a path between the two (ignoring the direction of arcs)
- An event is always followed by a function and vice versa (ignoring connectors).

Running example We illustrate the modeling with the help of an example. Figure 3.2 shows an EPC modeling the process “Handling of incoming goods” introduced in [LSW98].

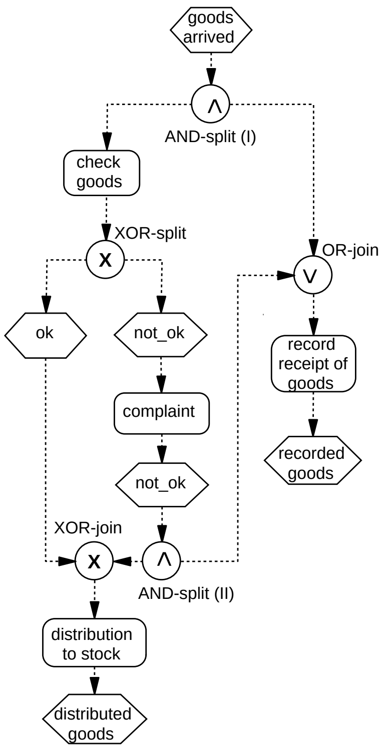


Figure 3.2: EPC: “Handling of goods”

The process starts with the event `goods arrived`. After that the execution is split into two parallel paths (`AND-split (I)`), modeling the work of two departments. The left one checking the goods and performing the ensuing functions, the right one doing the accounting. The checked goods are either `ok` or `not_ok`. In the latter case, a complaint is compiled, in the former nothing happens. In either case (`XOR-join`), the goods are distributed afterwards (`distribution to stock`). Connectors `OR-join` and `AND-split (II)` 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.

3.2.2 EPC semantics

EPCs are a semiformal method of business process modeling. Although, they have been applied quite successfully, their authors defined neither a comprehensive and consistent syntax nor the corresponding semantics [KNS92]. With their widespread use, the need for a formal foundation increased. Several approaches propose a formalization. Most of them suggest a mapping on existing techniques, such as Petri nets or Statecharts. This way it becomes possible to use existing analysis and verification techniques. Examples are [CS94, vU97, LSW98, Aal99, MR00, Rod99] for Petri nets and [WWKD$^+$97] for Statecharts. Other approaches to formalization have been developed by [NR02, Rum99, ADK02]. Here, semantics are defined by means of a transition system.

All the approaches have one thing in common: they assign operational semantics (execution semantics) to the EPCs .

This is the main difference between existing approaches and the one proposed here, where non-operational semantics is assigned with EPCs. Supporting non-operational semantics we take the line of reasoning followed in the first publications about EPCs[1] ([KNS92, Sch94]).

We are confident that the non-operational interpretation of EPCs fits an intuitive modeling understanding and is one reason why EPCs are said to be easy to learn and to understand. Modeling business processes, the modeler normally starts by describing "what an execution (should) look like", hence describes a set of accepted/good executions. This procedure fits the non-operational semantics. Here, an EPC is interpreted as a pattern that describes accepted executions. Deficient executions are only described implicitly, as these executions which do not fit the described pattern.

[1]In later publications the syntax of EPCs was enhanced by what is called a *process folder*. Process folders resemble tokens in a Petri net indicating the current state.

In contrast to EPCs, Petri nets do have operational semantics. A Petri net specification describes more than the set of accepted executions, and also covers “how an execution is reached”. This difference has been neglected in previous attempts of mapping EPCs to Petri nets. EPCs have been considered deficient if the corresponding Petri nets were deficient, cf. [Aal99, LSW98]. As a consequence, the modeling facilities of EPCs were restricted so that the Petri net correctness criteria could be transferred to EPCs.

Starting with non-operational semantics also has another advantage. The modeler does not have to think about the “how” of the execution. Determining the “how” of an execution is related to fixing the efficiency of the execution. The problem is that related information may not yet be clear in an early design phase. It is often only available later on, or it may change during the lifetime of a workflow system.

Nevertheless, we also formalize EPCs by a mapping onto Petri nets. In contrast to previous approaches, the ambiguities are deliberately maintained. Consequently, the resulting Petri net will not satisfy traditional correctness measures. There may be faults such as deadlocks and/or residual tokens. We therefore provide an adapted correctness criterion. The new criterion provides a pragmatic measure of the correctness of the primary EPC.

Here, Petri nets are used for the same reasons as in other approaches. They have a clear and precise definition [Mur89] and a graphical notation similar to that of EPCs. In addition, they provide many existing analysis techniques and tools. In the following, we will refer to the class of Workflow nets (WF-nets). WF-nets were introduced in Chapter 2 and proposed in [Aal97, Aal98, Aal00b]. WF-nets were tuned to fit the requirements within the domain of workflow management. Petri net theory was exploited to assemble adequate properties and efficient algorithms for that Petri net class [Aal97].

3.3 Transformation into workflow nets

The transformation of EPCs into Petri nets uses three steps. First, the elements of the EPC are mapped onto Petri net-modules. In the second step, rules are provided to combine the different modules to form a complex process model. A third step becomes necessary if the primary EPC had more than one start and/or end event. In that case, the derived WF-net must be supplemented by some extra in- and output places to satisfy the WF-net syntax.

3.3.1 Step 1: Mapping EPC elements to Petri net-modules

During the first step every EPC element is mapped onto corresponding elements on the Petri net-side. The mapping was illustrated in Figure 3.3. Events and

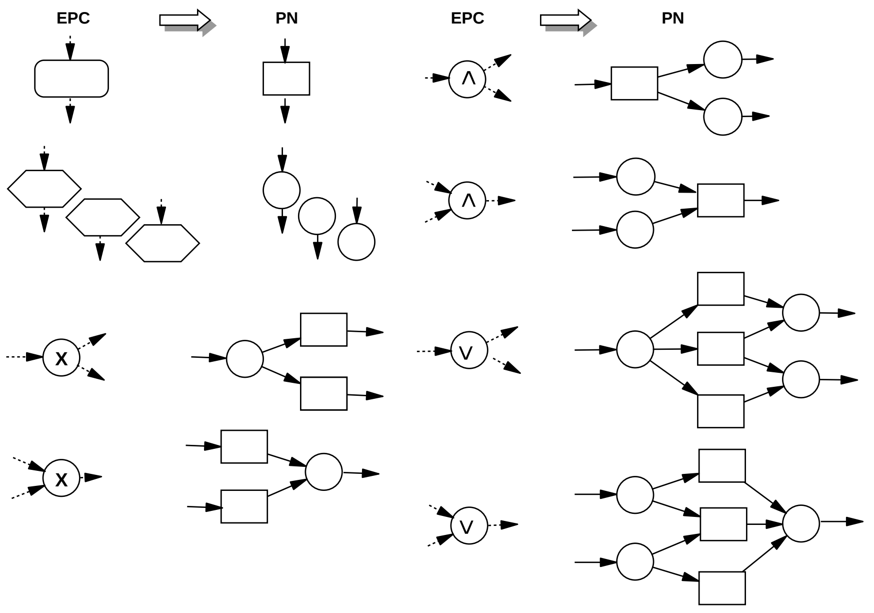


Figure 3.3: Step 1: Mapping EPC elements onto Petri net-modules

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 onto small Petri net-modules. The Petri net-modules describe the behavior of the routing constructs explicitly. This is particularly relevant for the OR, because its semantics has not been described consistently.

Figure 3.4 shows 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: C can be reached if either A or B or both occur[2]. In the EPC all these different cases are described through one connector. In the Petri net-module all possibilities are modeled explicitly via the transitions t_A, t_{AB} and t_B. The behavior of the EPC and the Petri net are equivalent because both accept the same executions.

[2]If A and B occur one after the other C can also be reached twice.

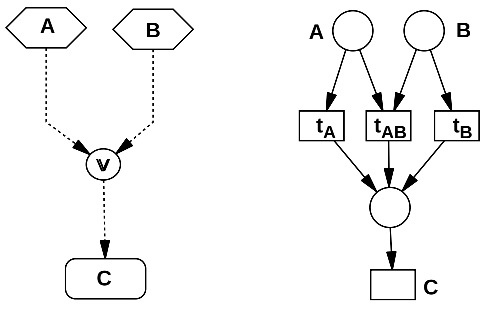


Figure 3.4: Transformation of the OR-connector

3.3.2 Step 2: Module combination

In the second step, the single Petri net-modules are joined to form a connected Petri net. Depending on the interface of the adjacent modules, one of the following combination rules is applied:

Case1: If input and output elements are of the same kind (e.g. both places) then the elements are unified.

Case2: If input and output elements are different (place and transition) then the arcs are fused.

Figure 3.5 illustrates the first and second step of the transformation of EPCs into Petri nets, showing unification of elements and fusing of arcs.

The proposed transformation approach is slightly more general than the transformations described in [Rod99] and [Aal99]. The rules presented here can also be applied to transform EPCs where connectors follow each other immediately, e.g. the XOR-join and the AND-split (II) in Figure 3.2. Another advantage of this approach is that the resulting Petri net does not contain any places or transitions not corresponding to elements of the EPC. The transformation rules by [Rod99], [LSW98] and [Aal99] all contain rules which explicitly introduce new pseudo places and transitions to meet the Petri net syntax; the resulting Petri net may contain elements which have no counterpart in the application domain. In contrast to any other approach, a simple mechanism was provided to aggregate several start and end events which may be connected over different paths.

3.3.3 Step 3: Adding unique input/output places

Applying Step 1 and Step 2, an EPC is translated into a Petri net but not necessarily into a WF-net. If the EPC contained more than one start, and/or end event, the resulting net may have more than one start and/or sink place. There are no EPC syntax-rules that restrict the number of start and end events. Moreover, if there are several start events (or end events), it is not clear whether they are mutually exclusive.

Then, a new start place and/or a new sink place is added. These new places are connected to the Petri net so that the places representing the primary start events (or end events) of the EPC are initialized (cleaned up). The connection of the new places with the primary places is not trivial but depends on the relation of the corresponding events in the EPC.

One way to determine the relation would be to track the paths, starting from the different start events (end events), until they join[3].

The connection of the new place with the primary places would than be a Petri net module that corresponds to the connector complementing the one that was found.

Consider two start events that are connected via an XOR-join. They are treated as mutually exclusive. The two corresponding start places in the Petri net will be linked to the new inserted place by a complementing XOR-split. This was illustrated in the upper row of Figure 3.6. The lower row of Figure 3.6 gives an example connecting a new end place to existing end places. Here, an AND-join was inserted complementing the AND-split.

The general case may be more difficult. There could be more than only two different start events (end events) with paths possibly meeting in various connectors of different type. To avoid a lengthy procedure we propose to link different start places by a Petri net module corresponding to an OR-split and different end places by a Petri net module corresponding to an OR-join. This way all possible dependencies are covered. Note that some of them may not reflect actual dependencies.

Applying the Steps 1 to 3, an EPC is transformed into a WF-net. The transformation is unique, in the sense that each EPC refers to only one WF-net.

Running example The transformation was applied to the example from Figure 3.2. The derived WF-net is shown in Figure 3.7. For convenience, the Petri

[3]The paths finally join. The EPC syntax rules state that: For every two elements there is a path between the two (ignoring the direction of arcs)

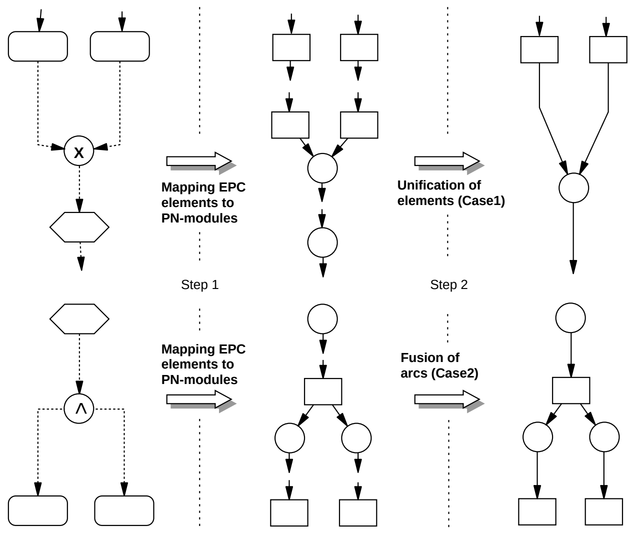

X
Mapping EPC elements to PN-modules
Unification of elements (Case1)
Step 1
Step 2
Mapping EPC elements to PN-modules
Fusion of arcs (Case2)
Λ

Figure 3.5: Connecting Petri net-modules

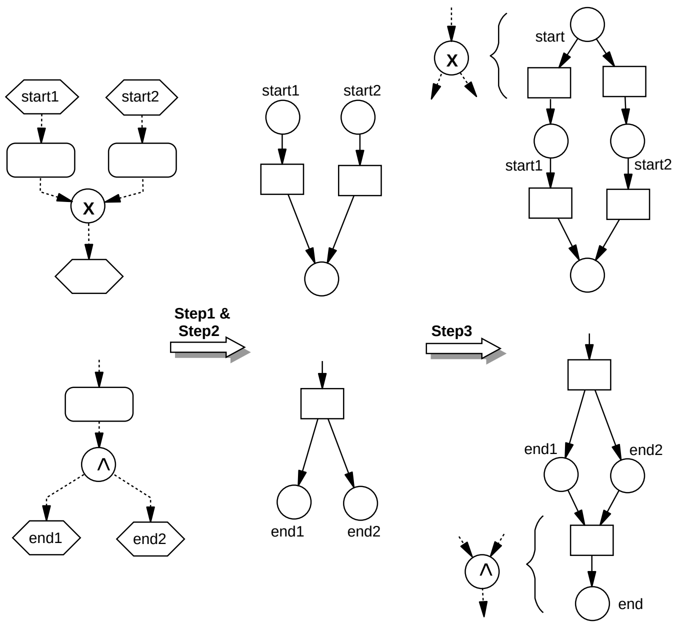


Figure 3.6: Step 3: Adding new start and sink places

net-modules which correspond to the routing constructs of the EPC have been highlighted with dotted rectangles. The sink place *o* and the transitions `t10,t11,t12`, corresponding to an OR-join, have been added within Step 3.

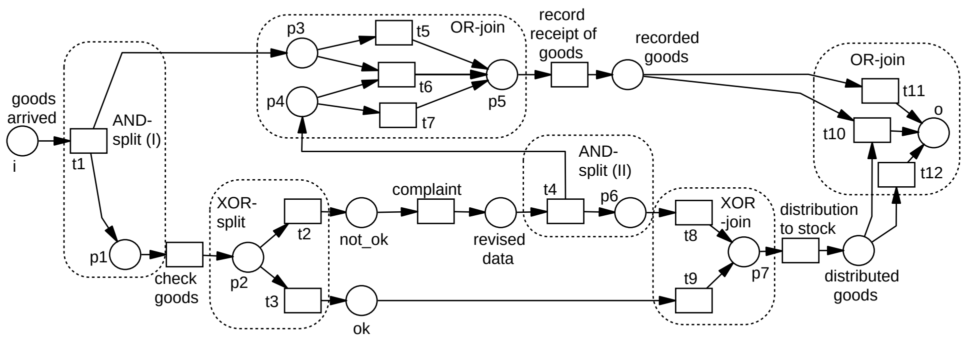


Figure 3.7: WF-net: "Handling of goods"

The Petri net-module which replaces the `OR-join` explicitly describes the behavior of this routing construct. Transition `t5` models the direct recording, and transition `t6` models waiting for the revision to be completed. The alternative `t7` has been introduced as part of the corresponding Petri net-module but has no expression in the original EPC. The EPC does not describe any accepted execution where the task `record receipt of goods` is triggered only through a negative result of the check. The accounting department is always involved. This is ensured by the first connector, the `AND-split (I)`. Thus, the alternative described by the transition `t7` does not belong to any of the accepted executions of the EPC. Transforming the OR-connector, the ambiguity of the OR is carried to the WF-net. Here, the decision whether to execute transition `t5`, `t6` or transition `t7` can not be resolved locally anymore.

3.4 Properties of the derived WF-nets

Applying the proposed rules, an EPC is transformed into a WF-net. But which structural and behavioral properties hold for the resulting WF-nets? After a short general discussion, this section examines soundness. We will argue that soundness does not always provide an adequate means for the correctness check of translated EPCs, and propose the use of a less stringent correctness measure.

3.4.1 General properties

First, some general structural and behavioral properties of Petri nets are considered.

- The resulting WF-net is *pure*, i.e. does not contain any self-loops. This is a result of the EPC syntax-rule stating that events and functions have at most one incoming and one outgoing arc. Actually, there are three syntactical correct combinations which would be transformed into non-pure Petri net modules (cf. Figure 3.8). Still the occurrence of these bizarre combinations can be excluded as there are no meaningful interpretation.

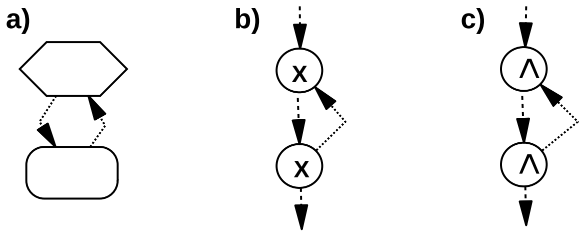


Figure 3.8: EPCs that are transformed into non pure Petri nets

- The syntactical rules for the modeling with EPC do not preclude the introduction of *cycles*. If the structure of an EPC is cyclic, the resulting WF-net will have cycles as well. Therefore, an acyclic structure cannot be assumed.
- The resulting WF-net may not be *bounded*. Unboundedness is a consequence of the introduction of badly matched cycles, i.e. cycles that contain an AND-split followed by an XOR- or OR-join.
- If the EPC contains only connectors of type AND and XOR the resulting WF-net is *free-choice*. The subnets translating connectors of these types are free-choice. Furthermore, any combination of XOR-splits and AND-joins again result in free-choice nets. The input and output elements of the corresponding subnets are different. Therefore Case 2 of the second transformation step is satisfied. The subnets are always joined by fusion of arcs. The argumentation is illustrated in Figure 3.9.

  If the EPC contains an OR-join, the resulting WF-net does not satisfy the free-choice property. The subnet introduced through the translation of an OR-join introduces non-free-choice behavior to the WF-net.

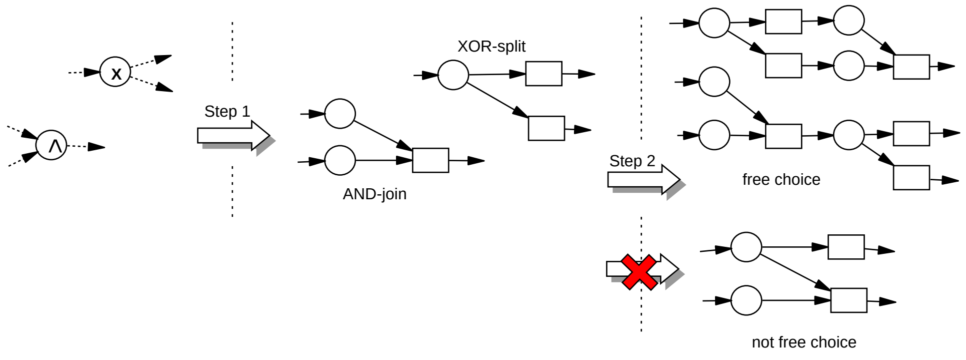


Figure 3.9: The combination of XOR- and AND- connectors does not lead to non free-choice subnets

- Using EPCs, the modeler is not obliged to model in a well-structured way, i.e. not every split has to be complemented by a corresponding join. Therefore, the derived WF-net may not be *well-handled.* Possible consequences are dead transitions and/or tokens that remain in the net although it already terminated. A consequence of residual tokens is that even if the WF-net itself is bounded or safe, this is not carried forward to the short-circuited WF-net.
- The derived WF-system may not be *live*[4], as there may be deadlocks and/or dead transitions. Non-liveness may arise if an AND-split is complemented by an XOR- or OR-join.
- The resulting WF-system may not be sound. A WF-system is sound if the short-circuited WF-system is live and bounded (cf. Theorem 2.48). As discussed above, neither of these properties does necessarily hold.

The next section investigates whether soundness of the resulting WF-net provides an adequate means to conclude the quality of the underlying EPC and to help with the revision of the process specification if necessary. It will be seen that the restrictions imposed on the WF-net and therefore on the primary EPC are too strong.

3.4.2 Soundness

In [Aal97] soundness was introduced as a correctness criterion for WF-nets. It was argued that this criterion covers a minimum set of requirements which a process

[4]Remember that a WF-system is live if the short-circuited WF-system is live.

description should satisfy. A WF-net is sound if termination is guaranteed and there are no residual tokens and neither deadlocks nor livelocks. Furthermore, there are no dead transitions. Residual tokens denote work that remained pending although the execution of the case had already terminated. Deadlocks indicate situations where the execution got stuck, and livelocks are when the execution makes no real progress any more. Dead transitions stand for tasks that do not contribute to the processing of workflow instances, as they are never executed.

We consider an EPC to be sound if the corresponding WF-net is sound. The check for soundness was implemented within the Petri net-tool Woflan [VA00]. Woflan not only states whether the process description is sound or not, but also provides the modeler with further information in order to support the location of deficient parts of the WF-net. Still, if the WF-net is not sound, it may not be possible to provide precise feedback supporting the redesign of the EPC. We will illustrate this by means of the running example.

Running example

The WF-net that resulted from the translation of the EPC "Handling of goods" was shown in Figure 3.7. The WF-net is not sound. There are firing sequences that do not terminate properly, e.g. the sequence:

- `t1, check goods, t2, complaint, t4, t5, record receipt of goods, t8, distribution to stock, t10.`

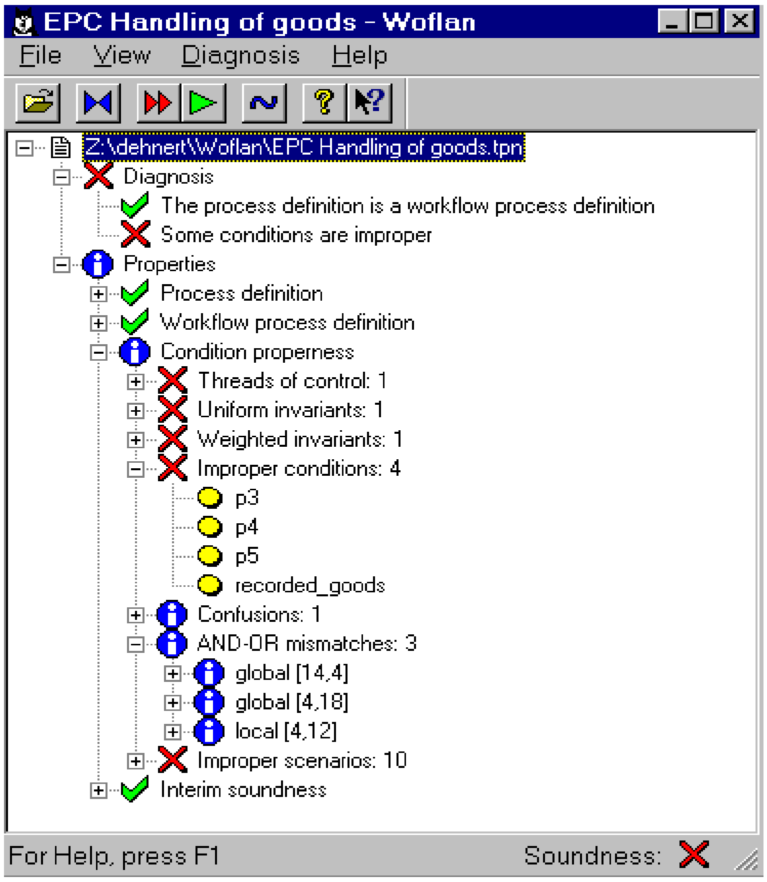
Here, the case terminated but there is a residual token in place `p4`.

Figure 3.10 shows the interface of Woflan reflecting the diagnosis for the running process specification. The modeler is provided with various kinds of feedback. Here, the modeler is pointed at four improper conditions namely, `p3,p4,p5`, and `recorded goods`. Furthermore, there are deficient firing sequences, called improper scenarios, such as the one mentioned above.

Still, the feedback does not point precisely to the deficient elements within the process specification. The OR-connector in the primary EPC was used to assure that in case of a complaint, the corrected data is waited for. If the goods were accepted (test result is `ok`) the receipt was recorded straight away. These two interpretations are reflected by transitions `t5` and `t6` in the resulting Petri net. Transition `t7` does not have any expression in the process. Still, as it is not dead this is not detected in the test for soundness.

In order to receive a sound WF-net, the EPC specification has to be changed in such a way that all executions of the corresponding WF-net terminate properly, i.e. residual tokens are avoided, as well as livelocks and deadlocks.

To change the EPC "Handling of goods" accordingly, the EPC has to be re-arranged in a well-structured[5] way, avoiding the use of OR-connectors. This change is not trivial and the resulting EPC looks completely different.



```
Z:\dehnert\Woflan\EPC Handling of goods.tpn
  Diagnosis
    The process definition is a workflow process definition
    Some conditions are improper
  Properties
    Process definition
    Workflow process definition
    Condition properness
      Threads of control: 1
      Uniform invariants: 1
      Weighted invariants: 1
      Improper conditions: 4
        p3
        p4
        p5
        recorded_goods
      Confusions: 1
      AND-OR mismatches: 3
        global [14,4]
        global [4,18]
        local [4,12]
      Improper scenarios: 10
    Interim soundness
For Help, press F1                                Soundness:
```

Figure 3.10: Woflan diagnosis for process: "Handling of goods"

[5]every split is complemented by a corresponding join

Impact of the soundness requirement

The impact of soundness as the correctness measure for EPCs is substantial. Soundness imposes operational semantics. As a consequence, the modeler is required to think about the “how” of the execution, which involves the consideration of efficiency aspects. In the introduction (cf. Section 3.1) it was argued that the specification of business processes should be as abstract as possible. The consideration of efficiency aspects requires detailed information about the process, e.g. duration and costs of tasks as well as the availability of resources. This information requires a much deeper insight and a higher level of detail than is available or desired for the modeling of business processes. Here, the focus is on communication. The objective is to come to a common process understanding between all participants. All further information unnecessarily complicates the description.

Soundness can only be achieved through a restriction of the EPC modeling facilities. The requirement for soundness demands the modelers to restrict themselves to well-structured EPCs and avoiding the use of the OR-connector. These restrictions reduce the expressiveness of EPCs and furthermore impose higher requirements on the modeling knowledge of the domain experts.

For these reasons, an adjusted correctness criterion would be desirable. The new correctness measure would support non-operational semantics. It would allow the modeler to postpone decisions considering the efficiency of the process execution as long as possible, i.e. close to implementation. Furthermore, it would not restrict the EPC modeling facilities and hence reflect the assumed modeling knowledge adequately. Such a new less stringent criterion is proposed in the next section.

3.4.3 Relaxed soundness

Based on the observations in the previous section, we propose a new criterion: *relaxed soundness*. Initially, relaxed soundness was introduced in [DDGJ01]. The new criterion is intended to represent a more pragmatic view of correctness. It is weaker (in a formal sense) and therefore easier to accomplish. Relaxed soundness does not impose the need to avoid situations with residual tokens or livelocks/deadlocks. Therefore, it is 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 is a sound firing sequence, i.e. a sequence that can be carried forward such that it terminates properly.

We will define the term *sound firing sequence* to explain the differences between the criteria soundness and relaxed soundness in formal terms.

Definition 3.1 (Sound firing sequence).
Let $S = (PN, i)$ be a WF-system. A firing sequence σ is sound if $i \xrightarrow{\sigma} M$ and $\exists \sigma', M \xrightarrow{\sigma'} o$.

A sound firing sequence can be extended, such that marking o is reached. Correspondingly, an *unsound* firing sequence is a firing sequence which ends in a marking from which marking o is not reachable.

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. Note, that soundness subsumes relaxed soundness.

Definition 3.2 (Relaxed soundness).
A workflow system $S = (PN, i)$ is relaxed sound iff each transition of PN is an element of some sound firing sequence. $\forall t \in T \quad \exists M, M' : (i \xrightarrow{*} M \xrightarrow{t} M' \xrightarrow{*} o)$.

Intuitively, relaxed soundness means that there are enough executions which terminate properly (i.e. state o was reached and there are no residual tokens) so that every transition is covered. A relaxed sound WF-net may have other firing sequences which do not terminate properly, but deadlock before termination or leave tokens in the net. In spite of that, relaxed soundness is still reasonable, because it requires that all relevant behavior is described correctly.

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. Using relaxed soundness as a correctness measure for EPCs, non-operational semantics are supported. The corresponding WF-net is checked only to allow for reasonable behavior. Interpreting relaxed soundness of a WF-net within the terms of the primary EPC, every function can be executed reaching a desired set of end events.

The results from the correctness check of the WF-net can be transferred directly to the primary model. If the result of the relaxed soundness check is positive, we can conclude that the EPC represents reasonable behavior.

If the result is negative, the modeler gets a list of transitions which are not contained in any sound firing sequence. According to the proposed transformation, every deficient transition either corresponds to a task or to a connector within the EPC. This means that either the task or one of the possible choices described by a connector are not included in an execution that terminates properly. It can be concluded that the corresponding part in the EPC needs improvement. 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.

This way, precise feedback is provided which will help the modeler to improve the process description until the corresponding WF-net fits the property.

Note. The feedback may become less precise if a transition at the beginning of the WF-net is not contained in any sound firing sequence. Then, all following transitions would be denoted as deficient as well. This could happen, if e.g. a split at the beginning is never followed by a corresponding join. In such a case the modeler should start to check the failure prone EPC-elements in the order of occurrence within the failure message, focusing on the interplay of the connectors.

Running example

The process specification from Figure 3.7 is not relaxed sound. There are no sound firing sequences containing transitions `t7,t11`, and `t12`.

Transitions `t11` and `t12` are part of the OR-join subnet that was introduced within Step 3 of the transformation rules. The OR-join was introduced to cover all possible dependencies among the end events, but possibly also reflects dependencies that actually do not exist. The test for relaxed soundness detects the transition which do not match any existing dependency. Corresponding transitions (here transition `t11` and `t12`) need not be communicated to the modeler, but can be omitted.

Transition `t7` belongs to the subnet of the OR-connector. Accordingly, the feedback for the modeler on the EPC side would be that the OR-connector incorporates behavior (namely the XOR-branch for the accounting) which never leads to a set of desirable end-events. The feedback could propose replacing the OR-join by an AND-join and a XOR-join. The revised EPC is shown in Figure 3.11.

After that replacement, the translation of the revised EPC corresponds to the WF-net shown in Figure 3.12, which is relaxed sound. The following two sound firing sequences contain all transitions:

- `t1, check goods, t2, complaint, t4, t13, t6, record receipt of goods, t8, distribution to stock, t10`
- `t1, check goods, t5, t3, t9, record receipt of goods, distribution to stock, t10`

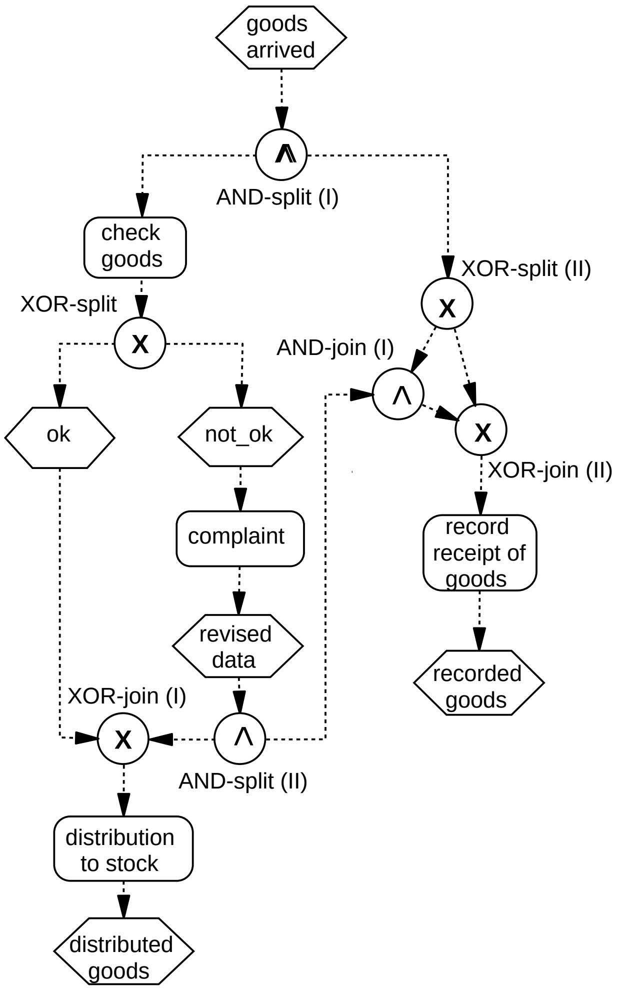


Figure 3.11: Revised EPC

Although the revised WF-net is still not sound, we can conclude that the revised EPC represents reasonable behavior and can be used as basis for refinement.

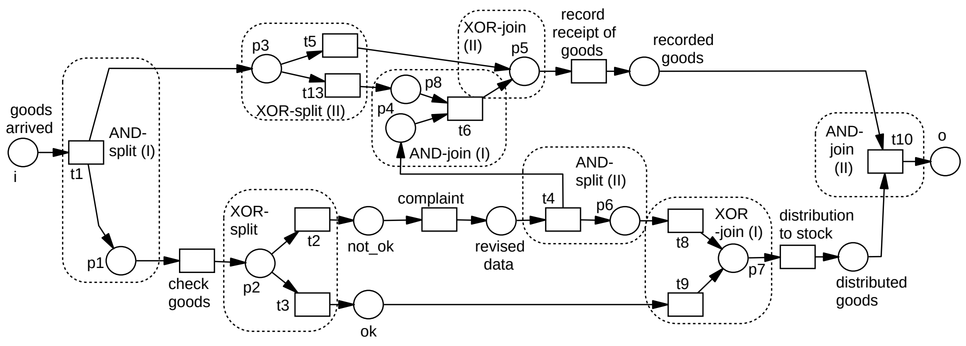


Figure 3.12: Revised WF-net

3.5 Implementation

For the transformation of EPCs into WF-nets, a prototypical framework has been implemented. As input format, an EPC description based on XML is used, which makes it possible to exchange process description with other tools, such as the ARIS tool set [SJ02].

The EPC description is transformed into PNML [WK02], an XML input format which is already accepted by various Petri net tools. In a first step the derived Petri net is tested for boundedness. After that, the process description is checked for relaxed soundness. The relaxed soundness test was implemented within Petri net tools such as LoLA (Low Level Petri Net Analyzer) [Sch99] and Woflan [VA00].

LoLA includes features such as: analysis of reachability of a given state and finding dead transitions. It covers the use of extended computation tree logic formulae (eCTL) [Roc00]. Within eCTL it is possible to quantify not only over states but also over state transitions. The combination of Petri nets and eCTL allows checking for relaxed soundness: for each transition t the reachability of the end state o is verified while including transition t in the path from i to o. So relaxed soundness is proved by enumeration of the required number of sound firing sequences. Worst case complexity occurs if the result is negative, then the whole reachability graph is scanned. In the positive case, the performance will be much better.

Unlike LoLA, the algorithm implemented in Woflan determines which transitions are covered by the subgraph of sound firing sequences. The subgraph itself is fairly simple: it consists of all backwards paths from o to i. When all transitions are covered, the algorithm stops. In the worst case, this would require visiting every node and every arc in the subgraph once. The algorithm's complexity is computed through the costs of computing the subgraph and the computing time needed for the search on the subgraph. The general complexity is again in the size of the reachability graph, but unlike in LoLA, the negative case may have a better performance than the positive one. The algorithm will be available in Woflan 3.0.

The feedback to the modeler from the relaxed soundness test is precise and understandable. It not only indicates the existence of deficiencies, but also shows why something is wrong and how it can be repaired.

So far, both implementations only work for bounded WF-nets. If the WF-net is unbounded, the reachability graph and possibly also the sound subgraph become infinite. The traditional way to cope with unbounded nets was the notion of what was called the coverability graph of a system. This is a variant of the reachability graph where infinite paths are represented by finite paths of infinite markings. Although the coverability graph can be used to decide boundedness of places and deadness of transitions [DR98] it does not provide enough information to test for relaxed soundness. The problem is that the coverability graph may not contain enough arcs. By definition, a coverability graph does not allow paths between infinite and finite markings.

Figure 3.13 shows a small net modeling the interaction of a producer and a consumer. The WF-net is unbounded, as the place `buffer` is unbounded. It is easy to see that this WF-system is relaxed sound. An example of a sound firing sequence containing all transitions is:

- `initiate, produce, consume, stop_producing, finish, clean_up`.

A part of the corresponding reachability graph and its coverability graph are shown in Figure 3.14.

The coverability graph cannot be used to prove relaxed soundness. The graph in Figure 3.14(b) does not contain arc $((p_1 b^* p_3), consume, (p_1 p_3))$ or arc $((p_2 b^* p_3), consume, (p_2 p_3))$ although these state transition are possible in the corresponding WF-net.

A pragmatic solution to testing unbounded WF-nets for relaxed soundness is using only a finite part of the reachability graph. If the chosen finite part of the reachability graph contains enough sound firing sequences, relaxed soundness of the corresponding WF-net is proven. Of course a given WF-net may still be relaxed sound even if a suitable finite part of the reachability graph has not been found.

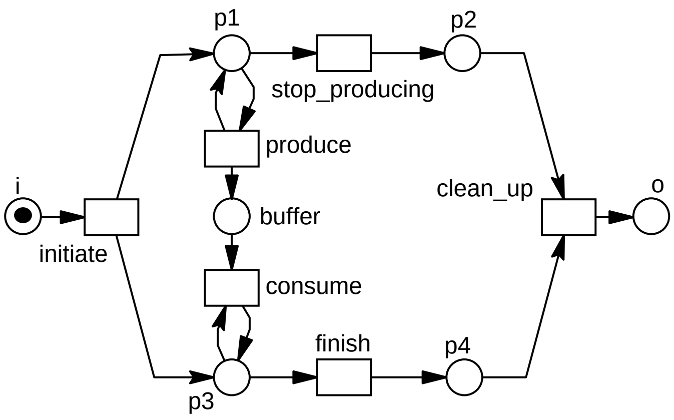


Figure 3.13: WF-net modeling the interaction between a producer and a consumer

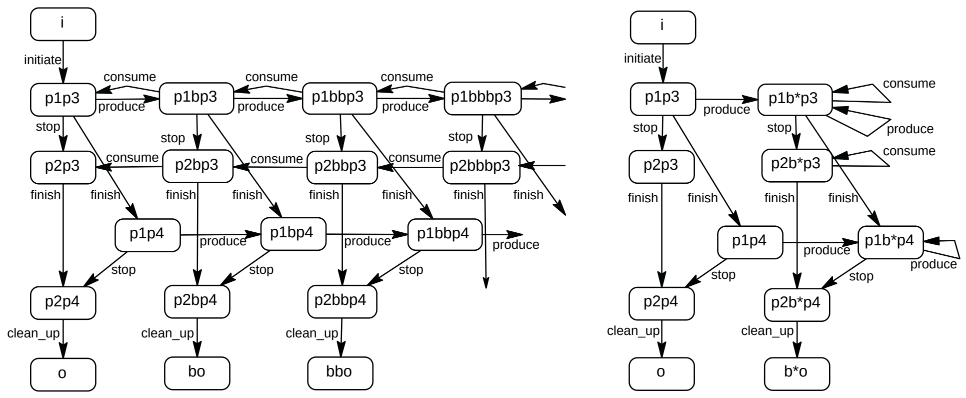


Figure 3.14: (a) Part of the reachability graph (b) Coverability graph

Complexity	Number of EPCs	Sound (S)	Relaxed sound (RS)	RS but not S
0	22	22	22	0
1	25	22	25	3
2	29	22	29	7
3	13	7	12	5
4	13	4	11	7
5	16	10	15	5
6	4	1	3	2
7	3	1	3	2
8	1	0	0	0
9	3	2	2	0
$\geq$10	1	0	0	0
All	130(100%)	91(70%)	122(94%)	31(24%)

Table 3.1: Properties at different complexities

3.6 Case study

The objective of the investigation was a comparison of the practical applicability of relaxed soundness versus soundness. In [Pie02] a number of EPCs were checked for soundness and relaxed soundness. 130 EPCs from an SAP R/3 implementation project served as case study for our investigation. By following the proposed transformation rules all EPCs were transformed into WF-nets. The resulting nets have been tested for correctness using the model checker LoLA.

Of the process descriptions examined, 94 percent were relaxed sound and 70 percent were sound. Table 3.1 presents the results. The first column of the table contains the number of connectors contained in the EPC description, used to express the complexity of the EPC specification. The second column contains the number of EPCs assigned to that class of complexity. More than half of the EPCs contained no more than two connectors. Thus, even if these process descriptions included many tasks, their structure was mainly sequential. The remaining columns show the number of EPCs that satisfy soundness, relaxed soundness, and relaxed soundness only.

The more complex a process specification becomes, the more likely that the specification does not satisfy soundness, although the main impediment to soundness was based on the use of the OR-connector. Only 13 percent (4 process descriptions) of the only relaxed sound processes impede soundness based on a bad combination of

AND- and XOR-connectors. It is clear that one case study does not provide enough data to derive representative results. Also note that the EPCs are mainly from one author. Still, we think the high percentage of relaxed sound process descriptions indicates that the property is appropriate for the intuition of the modeler.

3.7 Application of concepts

In this chapter we have proposed using a semiformal, graphical language for the modeling of business processes and introduced an adjusted correctness criterion.

A semiformal language is more likely to meet the intuition of the domain experts. As a consequence, the derived process descriptions may provide a suitable base for communication. Modeling with a semiformal language usually involves ambiguities which, seen as room for interpretation, are necessary in the early stage of analyzing a business process. In preparation for later stages of software development, the *useful interpretations* must be identified. This notion was formalized in terms of the relaxed soundness criterion.

Here, we used Event-driven Process Chains for the initial modeling of business processes. EPCs are widely accepted due to their tool support through the ARIS tool set [SJ02] and their use in the SAP reference models [KT97]. However, there are many more alternatives for the modeling of business processes. Other techniques that are used to specify the control-flow aspects are e.g. Statecharts and Activity Diagrams from the UML [Mar00], various kinds of Petri nets [Mur89], Task Structures [HN93], and workflow graphs [SO99]. These techniques are mainly used in a similar setting as EPCs. The focus is on modeling and communication issues. A further, but minor role is played by the formal analysis of the modeled process descriptions. The above mentioned modeling techniques were investigated quite thoroughly in research with respect to their use for workflow modeling. Still, these languages are only used sporadically in real systems.

Most workflow management systems use a proprietary workflow language. These languages combine an intuitive, graphical appearance with concepts that are close to implementation. A major drawback is the lack of formal semantics. As a consequence, analysis issues, such as the warranty of a correct and reliable execution are not supported at design time. Often, the appraisal of the process description is checked only by inspecting the behavior of the WFMS at run-time. It is clear that failures that are detected only then may be very costly and may not please customers.

We will conclude this part by showing that the concepts developed in this chapter are also applicable for modeling languages used in practical systems. Using Staffware as an example of a real WFMS, we will show the relevance of more pragmatic correctness criterion, such as relaxed soundness.

3.7.1 Staffware

Staffware is one of the most widespread workflow management systems with more than 550,000 users worldwide [Cas98, Sta99, Sta00]. It does not support any of the modeling languages discussed but uses a proprietary workflow language. In the following, the language will be sketched.

Modeling within Staffware

The tasks in Staffware are called steps. In contrary to Petri nets and EPCs, the Staffware modeling language does not support a notion of states, i.e. a concept similar to places in a Petri net or events in an EPC.

Staffware processes always start with a start step denoted by a symbol representing a traffic light. An end is denoted by a stop sign. A Staffware process must have only one start step, but may have several stop steps. Normal tasks are denoted by icons that resemble a sheet of paper. The semantics of such steps are rather OR-join/AND-split than XOR-join/AND-split; i.e a step becomes enabled if *one* of the preceding steps is completed and the completion of the step will trigger all subsequent steps. Since the OR-join/AND-split semantics is fixed, two additional building blocks are provided: wait and condition. *Wait* is used to synchronize flows and has AND-join/AND-split semantics. It is denoted by an hourglass-icon. To model choices, the condition building block is used. Note that Staffware only allows for binary choices of type XOR. The basic semantics of a step[6], a condition, and a wait are shown in Figure 3.15.

The process specification in Staffware is supported by a component called *Staffware Process Definer*. A screen-shot is shown in Figure 3.16.

[6]Note that the subnet for a normal step does not meet precisely the Staffware semantics. In Staffware, a step that gets enabled a second time (before it was executed) would still only be executed once. The second enabling is ignored. In the Petri net this situation is reflected by two input tokens. If every input place contains a token the required behavior could be matched prioritizing the middle transition over the others. Still if the two token are contained in one of the input places, there is no suitable Petri net interpretation, as merging of tokens is not covered by any standard Petri net semantics.

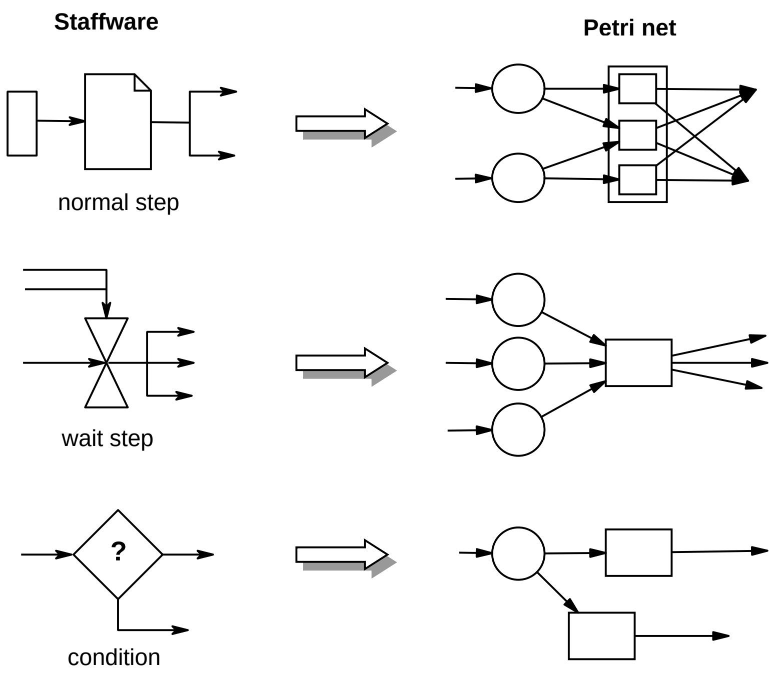


Figure 3.15: Staffware elements and their semantics

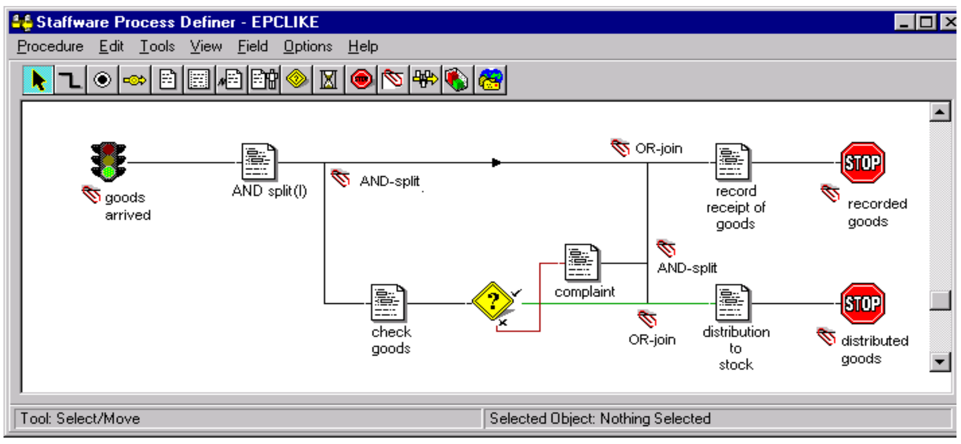


Figure 3.16: The modeling tool of Staffware

Applying relaxed soundness

The objective is again to support the modeler with precise feedback for the revision of the process specification. Applying the same procedure as for EPCs we will investigate whether the pragmatic correctness criteria *relaxed soundness* is also beneficial when modeling with other languages.

We again propose to transform the primary process description into a WF-net, to check the resulting WF-net for correctness, and to use the analysis results for revision of the primary specification.

We will not repeat the whole procedure but just illustrate the application of the proposed concepts using again the running example.

Running example

Figure 3.16 shows the Staffware description "similar" to the primary process "Handling of goods" (cf. Figures 3.2, and 3.7). Some of the connectors from the primary description do not have to be modeled explicitly in Staffware. This concerns e.g. the OR-connector, whose semantics is covered implicitly by the step `record receipt of goods`. The computation of the results of task `check goods` is modeled via a condition.

The process specifications from Figure 3.2 or Figure 3.7 did not satisfy relaxed soundness. It is now considered whether this correctness statement can be carried forward to the Staffware description.

The Process Definer does not support any form of analysis. Therefore, another component of Staffware, the *Audit Trail (AT)*, is used to monitor the executions of individual cases. Using the AT for our purpose we will focus on sound and unsound executions and compare the result in the AT.

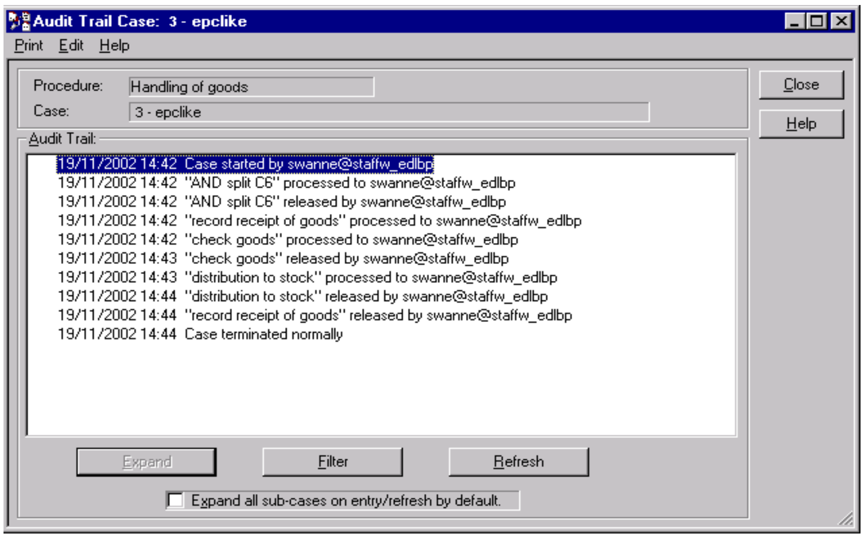
Figure 3.17 shows the user interface of the AT. In the upper part of the window the recorded process is denoted and the case is identified. In the lower part the audit trail is given, showing when which tasks were assigned and completed by which actors. The first is reflected with a line: "\$date\$ \$time\$ \$task\$ *processed to* \$actor\$". The completion is denoted by "\$date\$ \$time\$ \$task\$ *released by* \$actor\$".

Note, the actual time, at which the actor starts processing the assigned task is not recorded. To determine the ordering of tasks within a case the release time was used.

The audit trail of Figure 3.17 reflects an execution where the result of the task `check goods` was `ok`. The displayed audit trail corresponds[7] to the following firing sequence of the WF-net from Figure 3.7:

- `t1, check goods, t3, t9, distribution to stock, t5, record receipt of goods, t10`.

In the primary WF-net this firing sequence is sound. As expected, the result of the audit trail says "Case terminated normally".


Audit Trail Case: 3 - epclike
Print
Edit
Help
Close
Help
Expand
Filter
Refresh
Expand all sub-cases on entry/refresh by default.

Procedure: Handling of goods

Case: 3 - epclike

Audit Trail:

```
19/11/2002 14:42  Case started by swanne@staffw_edlbp
19/11/2002 14:42  "AND split C6" processed to swanne@staffw_edlbp
19/11/2002 14:42  "AND split C6" released by swanne@staffw_edlbp
19/11/2002 14:42  "record receipt of goods" processed to swanne@staffw_edlbp
19/11/2002 14:42  "check goods" processed to swanne@staffw_edlbp
19/11/2002 14:43  "check goods" released by swanne@staffw_edlbp
19/11/2002 14:43  "distribution to stock" processed to swanne@staffw_edlbp
19/11/2002 14:44  "distribution to stock" released by swanne@staffw_edlbp
19/11/2002 14:44  "record receipt of goods" released by swanne@staffw_edlbp
19/11/2002 14:44  Case terminated normally
```

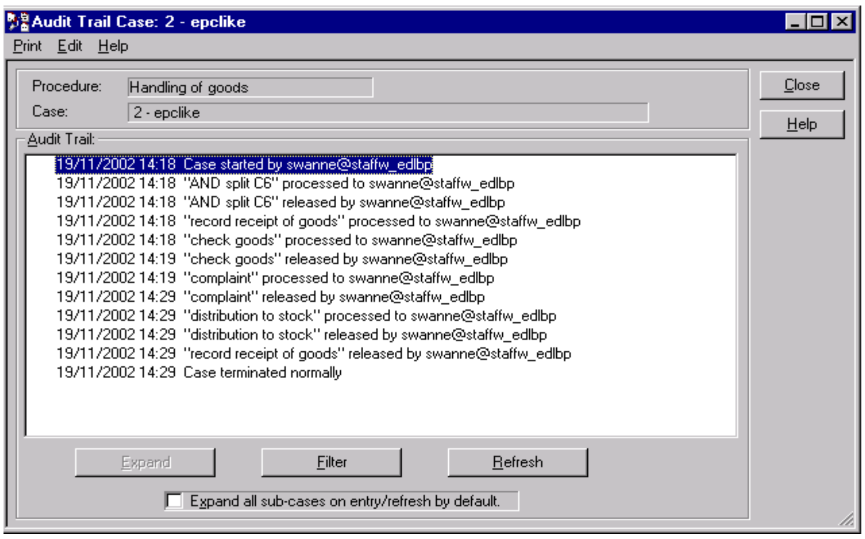
Figure 3.17: Staffware Audit Trail (I) showing sound execution

A further audit trail is given in Figure 3.18. It shows an execution where the result of task `check goods` was `not_ok`. Note that the task `record receipt of goods` is completed last. The displayed audit trail corresponds to the following firing sequence of the WF-net from Figure 3.7:

- `t1, check goods, t2, complaint, t4, t8, distribution to stock, t6, record receipt of goods, t10`.

In the WF-net this firing sequence is sound. As expected, the result of the audit trail again says "Case terminated normally".

[7]Correspondence is meant in terms of the notion of *observation equivalence* (also *weak bisimulation equivalence*) [Mil80, GW96]



Procedure: Handling of goods
Case: 2 - epclike
Audit Trail:

```
19/11/2002 14:18  Case started by swanne@staffw_edlbp
19/11/2002 14:18  "AND split C6" processed to swanne@staffw_edlbp
19/11/2002 14:18  "AND split C6" released by swanne@staffw_edlbp
19/11/2002 14:18  "record receipt of goods" processed to swanne@staffw_edlbp
19/11/2002 14:18  "check goods" processed to swanne@staffw_edlbp
19/11/2002 14:19  "check goods" released by swanne@staffw_edlbp
19/11/2002 14:19  "complaint" processed to swanne@staffw_edlbp
19/11/2002 14:29  "complaint" released by swanne@staffw_edlbp
19/11/2002 14:29  "distribution to stock" processed to swanne@staffw_edlbp
19/11/2002 14:29  "distribution to stock" released by swanne@staffw_edlbp
19/11/2002 14:29  "record receipt of goods" released by swanne@staffw_edlbp
19/11/2002 14:29  Case terminated normally
```

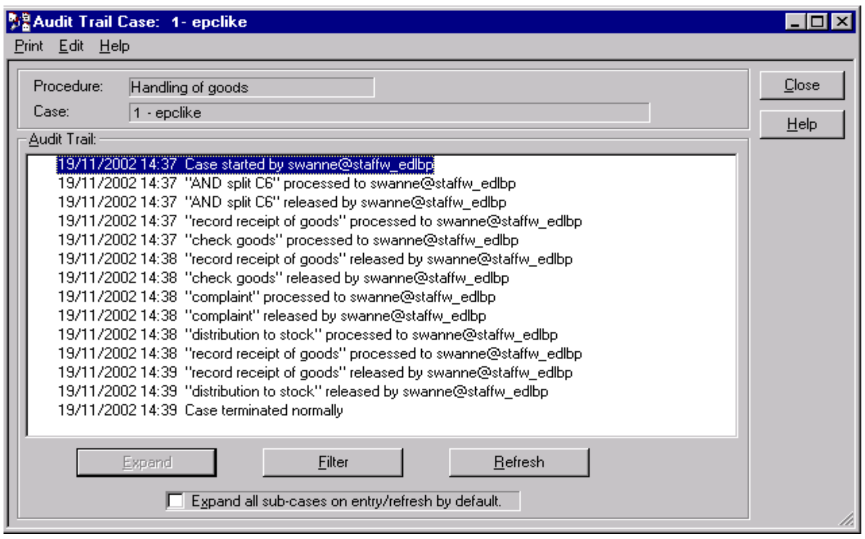
☐ Expand all sub-cases on entry/refresh by default.

Figure 3.18: Staffware Audit Trail (II) showing sound execution

Probably the most interesting audit trail is shown in Figure 3.19. Within this execution, the task `record receipt of goods` is completed before the result of task `check goods` was computed. The result turns out to be `not_ok` which entails a second `record receipt of goods`. The audit trail corresponds to the following firing sequence of the WF-net from Figure 3.7:

- `t1, t5, record receipt of goods, check goods, t2, complaint, t4, t8, t7, record receipt of goods, distribution to stock, t10, t11`

This firing sequence is not sound. It does not satisfy the property of proper termination. But the result of the audit trail again says “Case terminated normally”. The result indicates that relaxed soundness does not match exactly with the correctness understanding of Staffware. Here, an even more alleviated correctness measure is required to assess modeling results. It would require every transition to be part of a terminating firing sequence which need not necessarily be sound. The requirement for proper termination is abandoned. The result shows again that the trend towards less stringent correctness measures is reasonable.



Audit Trail Case: 1- epclike

Procedure: Handling of goods

Case: 1 - epclike

Audit Trail:

```
19/11/2002 14:37  Case started by swanne@staffw_edlbp
19/11/2002 14:37  "AND split C6" processed to swanne@staffw_edlbp
19/11/2002 14:37  "AND split C6" released by swanne@staffw_edlbp
19/11/2002 14:37  "record receipt of goods" processed to swanne@staffw_edlbp
19/11/2002 14:37  "check goods" processed to swanne@staffw_edlbp
19/11/2002 14:38  "record receipt of goods" released by swanne@staffw_edlbp
19/11/2002 14:38  "check goods" released by swanne@staffw_edlbp
19/11/2002 14:38  "complaint" processed to swanne@staffw_edlbp
19/11/2002 14:38  "complaint" released by swanne@staffw_edlbp
19/11/2002 14:38  "distribution to stock" processed to swanne@staffw_edlbp
19/11/2002 14:38  "record receipt of goods" processed to swanne@staffw_edlbp
19/11/2002 14:39  "record receipt of goods" released by swanne@staffw_edlbp
19/11/2002 14:39  "distribution to stock" released by swanne@staffw_edlbp
19/11/2002 14:39  Case terminated normally
```

☐ Expand all sub-cases on entry/refresh by default.

Figure 3.19: Audit result of an unsound execution

3.8 Related work

The modeling of business processes emerged as a research area well over a decade ago. Since then a wide variety of different modeling languages has been developed. Most of them focus on the control flow aspect, i.e. the tasks and their ordering. This is natural as their specification provides a conceptual basis for the integration of other aspects. Apart from this no consensus has been reached as to what should be essential ingredients of a business process modeling language. Only recently, there are efforts collecting all possibly relevant concepts compiling a pattern-library for workflow modeling and to establish a formal foundation for control-flow aspects of process specification languages, see e.g. [AHKB03, AH02b].

The following section is subdivided into three parts. First we provide an overview of selected approaches to modeling business processes. We focussed on languages that provide a graphical representation in combination with formally founded semantics allowing for verification of specific properties, such as termination, absence of deadlocks and dead tasks. In the second part we will examine the advanced control-flow patterns in the provided pattern-library and classify those that are covered by the relaxed soundness property. In the third part, we single out one specific pattern, namely the OR-join and investigate how different approaches resolve its inherent ambiguity.

3.8.1 Main trends within business process modeling

UML techniques There are some approaches, such as [Esh02] and [WWKD$^+$97], using UML techniques for the modeling of business processes. These approaches focus on the description of control-flow aspects. In order to provide additional support for the modeler, formal semantics is introduced for Activity Diagrams [EW00, EW01b] and Statecharts [WW97]. The drawback of the approaches lie in the disagreement within the UML community. There is not a common understanding, but every new application creates its new tailored semantics. [Esh02] lists four general requirements which a process specification should satisfy. The combination of the four resembles an even stricter version of soundness (than the one used for WF-nets [Aal98]) requiring additionally the absence of divergence. [WW97] does not focus on a special correctness criterion for business processes but considers general safety and liveness properties, such as the absence of deadlocks and the reachability of certain states.

Petri nets Numerous approaches use Petri nets for the modeling of business processes. Examples are [Aal98, AAH98, BP98, DE00, DFZ02, DGS95, KG98, LO02, Obe96, Sim02]. As well as the control-flow aspects they also concern resource aspects ([BP98, DFZ02, KG98]), data flow ([LO02]) and time aspects ([AAH98, DE00, DFZ02, Obe96, Sim02]).

Still, the use of Petri nets for business process modeling often meets with the refusal of modelers. The process descriptions require a level of detail which is not always necessary and which poses difficulties communicating the process specification among people of different knowledge cultures. Furthermore, the mapping of domain artefacts onto Petri net elements needs some modeling expertise. A basic example is illustrated in Figure 3.20. Application experts find it more intuitive to use EPC-like elements (depicted in the second row of Figure 3.20). Here, the semantics of the connectors seem to be clear. The corresponding Petri net notation (depicted in the first row) often causes confusion. Some approaches try to facilitate the use of Petri nets by introducing abbreviations [Aal98], or separate models for different concerns [DFZ02].

For the analysis of the modeled processes a wide variety of results from Petri net theory are available. The soundness criterion [Aal98] was introduced as useful conglomerate appraising the control-flow of business process descriptions.

Workflow Graphs In [SO99] *workflow graphs* are introduced to model the control-flow of business processes. A workflow graph is a simple, directed, acyclic

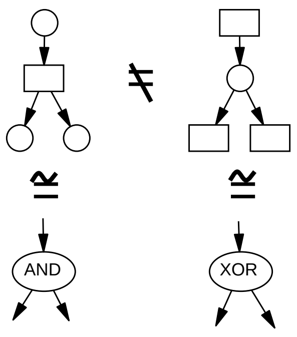


Figure 3.20: Simple pattern in Petri nets (up) and EPCs (down)

graph consisting of a finite set of nodes (tasks and conditions) and a finite set of control flows representing directed arcs between two nodes. Workflow graphs were introduced as a more direct way of modeling workflow processes. Still the limited number of elements, together with the assembly rules restrict the modeling power of workflow graphs (cf. [AHV02] corresponding WF-nets are free-choice). Structural conflicts in workflow graphs, such as the absence of deadlocks, are detected by applying special graph reduction techniques. Note the results presented in [SO99] are incorrect as the reduction rules presented are not complete. In [AHV02] an alternative algorithm was presented and it was shown that the absence of structural conflicts coincides with soundness of the corresponding WF-net.

Task Structures Another representative of a process control specification language are Task Structures (see, e.g. [HN93]). In Task Structure diagrams sequential composition, choice, iteration, parallel execution and synchronization can be expressed.

A formal semantics for Task Structures has been provided in terms of Process Algebra [HN93] and Petri nets [AH98]. The mapping of Task Structures onto Petri nets (namely WF-nets extended with arc weights) allows for the verification of Task Structures using Petri net based analysis techniques. It was shown that soundness of Task Structures corresponds to liveness of the corresponding WF-net.

Others Approaches, using again different modeling techniques, are [CCPP95] and [RD98]. In [CCPP95] a workflow description language is proposed where a process model comprises tasks and routing tasks. The latter concept can carry different semantics depending on the type of the routing task, of which particularly interesting are partial and iterative joins. The semantics has been illustrated by state diagrams. However, verification issues are neglected.

In [RD98] fundamentals of the ADEPT Workflow Model are presented. This model is based on the concept of symmetrical control structures, where various structures such as splits, joins and loops are specified as symmetrical blocks with explicit start and end points. These blocks may be arbitrarily nested, but they are not allowed to overlap, i.e. the nesting must be regular. The ADEPT language is formally founded supporting the analysis of the control-flow. Beside the reachability of all nodes, every process model should satisfy that from every reachable state of the process termination can be guaranteed.

Most workflow management systems use a proprietary workflow language. Examples are the *Flow Definition Language* of the IBM approach (FlowMark/MQ Series Workflow), the *Web Service Flow Language* [Ley01] a further development designed to model business processes based on Web Services or *XLANG* [Tha01] a similar approach enforced by Microsoft. A common standard has been found with the *Business Process Execution Language for Web Services* (BPEL4WS) [BEA03]. There are also the internal modeling languages of other workflow management systems, e.g. *Business Process Maps* used within Action Workflow, the specific modeling language used with Staffware, Verve, or InConcert, and many others. These languages combine an intuitive, graphical modeling (based on the assembly of predefined workflow patterns) with concepts that are close to implementation. Analysis issues are mostly neglected.

The modeling capabilities of several workflow management systems were investigated in [Kie02]. The different modeling languages supported have been classified in terms of four evaluation strategies used. Furthermore a mapping of workflow pattern to Petri nets is provided, capturing their interpretation formally. Applying standard Petri net analysis techniques the process descriptions can be checked for (strict) termination, and the absence of deadlocks and livelocks. A process description is considered to be *well-behaved* if it is safe and (strictly) terminating.

3.8.2 Workflow-patterns

Investigating contemporary workflow products a wide range of workflow patterns has been identified [AHKB03]. There are basic control patterns, such as sequence, parallel split (AND-split), synchronization (AND-join), exclusive choice (XOR-split), and simple merge (XOR-join) that should be supported by any conceptual process language. Beside them some advanced branching and synchronization patterns have been identified: multiple choice (OR-split), multiple merge, discrimina-

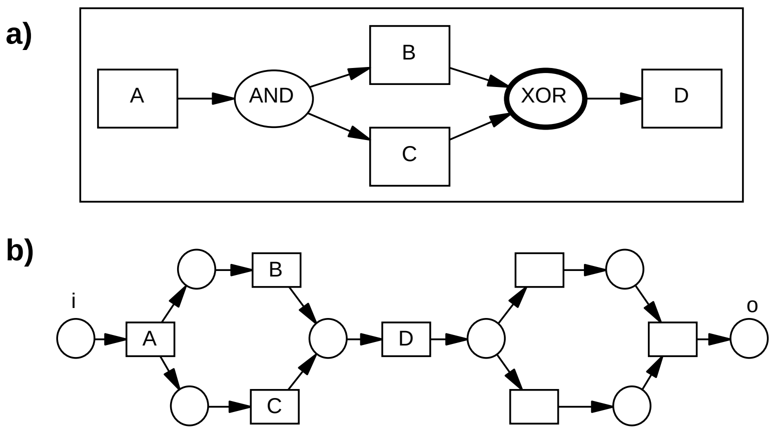


Figure 3.21: a) Pattern diagram “Multiple Merge”; b) Relaxed sound WF-net

tor, N-out-of-M join, and synchronizing merge (OR-join). The basic control patterns as well as the multiple choice and the synchronizing merge have already been used within EPCs. In Figure 3.3 a translation into Petri net modules was shown which could be integrated into WF-nets showing relaxed soundness.

We will now show that also for the remaining advanced control-flow patterns there is an explicit modeling in terms of Petri nets which could be integrated into a relaxed sound WF-net.

Multiple merge is a point in a workflow process where two or more branches reconverge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started once for every incoming branch that gets activated [AHKB03] (i.e. in the pattern diagram shown in Figure 3.21 a, D will be instantiated twice). The Petri net module describing this behavior coincides with the translation of the normal XOR-join shown in Figure 3.3. The respective behavior depends on the context. If the merge was preceded by a parallel split (AND-split) it must be followed by a synchronization (AND-join). Only then relaxed soundness of the corresponding WF-net may be satisfied. Figure 3.21 b) shows a relaxed sound WF-net incorporating a multiple merge. Note that there is no sound WF-net incorporating a translation of this pattern, as it always introduces the possibility to deadlock.

Discriminator The discriminator is a point in a workflow process that waits for one of the incoming branches to complete before activating the subsequent

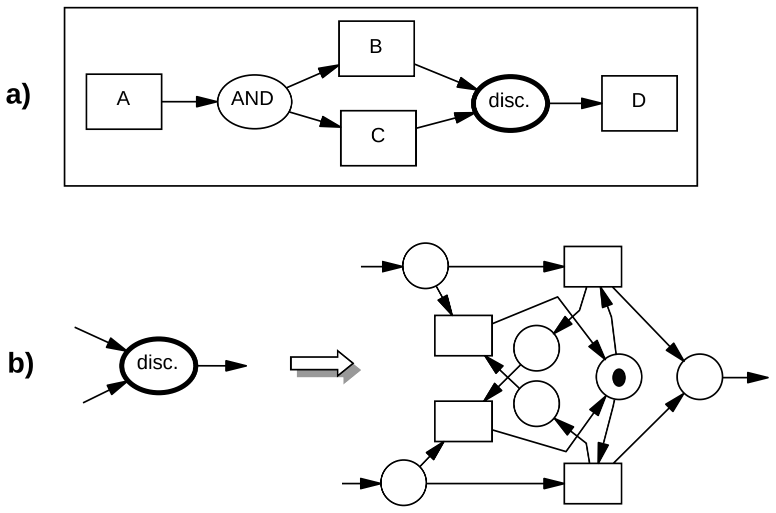


Figure 3.22: a) Pattern diagram "Discriminator"; b) Corresponding net module

activity. From that moment on it waits for all remaining branches to complete and "ignores" them. Once all incoming branches have been triggered, it resets itself so that it can be triggered again [AHKB03]. The pattern diagram is given in Figure 3.22 a). A Petri net-module describing this behavior explicitly is shown in Figure 3.22 b). Clearly, incorporating this module into a WF-net relaxed soundness (as well as soundness) can be satisfied.

N-out-of-M join is a point in a workflow process where M parallel paths converge into one. The subsequent task should be activated once N paths have completed. Completion of all remaining paths should be ignored. Similarly to the discriminator, once all incoming branches have "fired", the join resets itself so that it can fire again. This control pattern can be depicted as combination of synchronization and discriminator [AHKB03]. Figure 3.23 shows the original and the refined pattern diagram. There are Petri net modules describing the behavior of synchronization and discriminator. Both could be incorporated in a WF-net showing relaxed sound (or even sound) behavior. We conclude that also a combination of both patterns, i.e. the "N-out-of-M join", can be part of a (relaxed) sound WF-net.

The possibility to cover the above translations of advanced workflow patterns (especially the *multiple merge*) within relaxed sound WF-nets provide a further evidence for the reasonability of the new criterion.

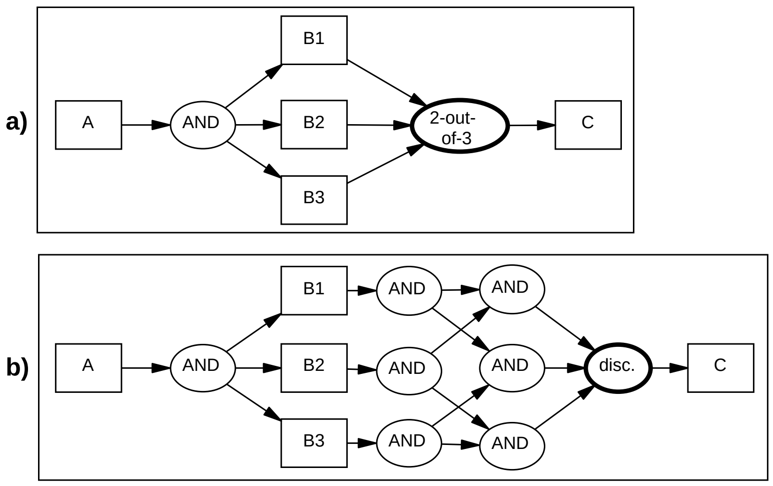


Figure 3.23: a) Pattern diagram “N-out-of-M join”; b) Refined pattern diagram using a combination of synchronization and discriminator

The solutions for the mentioned workflow patterns are rather straightforward. Patterns that pose more difficulties are *multiple-choice* and *synchronizing merge*. These patterns are already known from the modeling with EPCs, referred to as OR-split and OR-join connectors. Translations, possibly covered by relaxed sound WF-nets, have been shown in Figure 3.3. In the next paragraph we will refer to other approaches providing formal semantics for these patterns.

3.8.3 Resolving the ambiguity of the OR-connector

Many modeling techniques provide constructs such as *multiple choice* and *synchronizing merge* [AHKB03]. The multiple-choice pattern (OR-split) can choose multiple alternatives from a given set of alternatives. The synchronizing merge (OR-join) should have the capability to synchronize parallel threads and to merge alternative threads. The difficulty here is to decide when to synchronize and when to merge. This decision cannot be made locally. Though the non-locality of the OR-join has its complications, it is a pattern frequently used in a wide variety of workflow processes.

Many workflow management systems support pragmatic solutions resolving the ambiguity of the OR-connector. For the individual processing of the OR-join within systems such as InConcert, Eastman, MQSeries Workflow, eProcess, and

Domino Workflow, we refer to [ADK02, AHKB03, Kie02].

We will look more closely at the formalization approaches within EPCs. Regarding the OR-connector, there are almost as many solutions as approaches. Still, in contrast to our formalization they all tend to resolve the inherent ambiguity. In [Rit00a] 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 behavior explicitly (wait-for-all, first-come, every-time).

[Rod99] and [MR00] resolve the ambiguity by 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. [CS94] and [LSW98] introduce different tokens for the same reason. All these approaches require well-structured modeling, i.e. every split has to be complemented by a corresponding join. This restricts the modeler considerably and also places substantial demands on the modeling expertise. Modeling with a well-structured EPC is based on a strict top-down design process which can hardly be enforced in practice.

New ideas to handle the ambiguous meaning of the OR-connectors were sketched in [ADK02]. The approach reveals difficulties with the formalization provided in [NR02]. In [NR02] EPC semantics were defined based on a transition system but contain a cyclic definition of the transition relation. Therefore the semantics are subject to multiple interpretations. [ADK02] points at two solutions to the problem. One refers to a fixed-point interpretation of the transition relation. Another way was the definition of two transition relations; the first would be completely local, the second would be non-local, but would only use the first for checking whether the OR-join should wait for another thread to be synchronized.

Chapter 4

Embedding relaxed soundness into Petri net theory

Having defined the new criterion of *relaxed soundness*, the purpose of this chapter is to raise the understanding of the new criterion by investigating relations between existing Petri net properties and relaxed soundness.

In Section 4.1 the classical soundness notion is reviewed and interrelations between soundness and relaxed soundness are discussed. In Section 4.2 the proposed relations will be proven correct. Section 4.3 focuses on relations to other Petri net-criteria. In the last section related work is discussed.

4.1 Review of introduced correctness criteria

We start reviewing the introduced correctness criteria for WF-nets. At first we come back to soundness as introduced in [Aal97], cf. Def. 2.47.

Definition 4.1 (Soundness).
A process specified by a system $S = (PN, i)$ is sound iff:

(i) For every state M reachable from state i, there is a firing sequence leading from state M to state o (option to complete).
Formally: $\forall M : (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o)$.

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

(iii) There are no dead transitions in S.
Formally: $\forall t \in T \exists M, M' : (i \xrightarrow{*} M \xrightarrow{t} M')$

Soundness ensures that the process can always terminate with a single token in place *o* and with all the other places empty. In addition, it requires that there are no dead tasks, i.e. each task can be executed.

Soundness requires all firing sequences of the corresponding WF-net to be sound. Remember that a sound firing sequence (cf.D̃ef. 3.1) is a sequence that can become extended such that it terminates properly.

Definition 4.2 (Sound firing sequence).
Let $S = (PN, i)$ be a system. A firing sequence σ leading to a marking $M : i \xrightarrow{\sigma} M$ is sound if $\exists \sigma', M \xrightarrow{\sigma'} o$.

In Chapter 3, Def. 3.2 relaxed soundness was introduced as a new correctness criterion for WF-nets. Relaxed soundness only requires that there are enough sound firing sequences so that each transition is covered.

Definition 4.3 (Relaxed soundness).
A process specified by a system $S = (PN, i)$ is relaxed sound iff every transition of PN is element of a sound firing sequence. $\forall t \in T \quad \exists M, M' : (i \xrightarrow{*} M \xrightarrow{t} M' \xrightarrow{*} o)$.

In contrast to a sound WF-net, a relaxed sound WF-net may have firing sequences which do not terminate properly, but deadlock before or leave tokens in the net.

Relaxed soundness was derived from soundness. It poses weaker requirements to a process description than soundness. From the given definitions it can easily be seen that a *sound* WF-system will also necessarily be *relaxed sound.*

Lemma 4.4 (Soundness implies relaxed soundness).
Let $S = (PN, i)$ be a WF-system. If S is sound, then S is relaxed sound.

It is equally clear that relaxed soundness does not necessarily imply soundness. For a better understanding of the new criterion, we will consider whether there is a property *X*, such that relaxed soundness and *X* imply soundness.

We will prove that a "missing piece" between relaxed soundness and soundness is well-handledness or well-structuredness. Remember (cf. Section 2.6) that the structural property well-handledness (well-structuredness) states that there are no PT- and no TP-handles in PN ($\overline{PN}$).

The claim is comprised in the following proposition:

Proposition 4.5.
Let $S = (PN, i)$ be a WF-system. Let S be relaxed sound and PN be well-structured. S is sound.

The proof of this claim will be provided in several steps. We start by proving the proposition for special WF-nets such as state machines or marked graphs. The result is then applied to the class of free-choice WF-nets. In the last step the result is proved for general WF-nets.

4.2 Relaxed soundness versus soundness

To prove the given proposition, we first look at simple WF-nets, such as state machines and marked graphs.

4.2.1 State machines

State machines are always well-handled. This can be easily concluded from the definition. State machines only contain branching places but no branching transitions. Both would be necessary to build a PT- or TP-handle.

Theorem 4.6. *Let PN be a WF-net with input place i. If PN is a state machine, then PN is sound.*

Proof The short-circuited net $\overline{PN}$ is a state machine. The transition t^* being added to short-circuit the WF-net again satisfies $|^\bullet t| = |t^\bullet| = 1$. Taking into account Theorem 2.40 we know that the initially marked and strongly connected state machine $(\overline{PN}, i)$ is live and bounded (Theorem 2.41). With Theorem 2.48 we can conclude that PN is sound.

□

4.2.2 Marked graphs

Marked graphs are always well-handled. In contrast to state machines they only have branching transitions but no branching places. Again, both would be necessary to build a PT- or TP-handle.

Theorem 4.7. *Let PN be a WF-net with input place i. Let PN be relaxed sound and the short-circuited net $\overline{PN}$ be a marked graph. PN is sound.*

Proof $\overline{PN}$ is a marked graph. As PN with input place i is relaxed sound, we can conclude the existence of an infinite firing sequence in $(\overline{PN}, i)$. For its construction, we take one of the sound firing sequences that exist in PN, subsequently fire t^* and repeat this infinitely often. Taking into account Theorem 2.43, we know that $(\overline{PN}, i)$ is live. With Corollary 2.42 we furthermore know that $(\overline{PN}, i)$ is bounded. With Theorem 2.48, we can conclude that PN is sound.

□

We will now broaden the scope of the proposition and prove its validity for the class of free-choice nets.

4.2.3 Free-choice Workflow-nets

Free-choice nets are not per se well-structured. Both branching transitions and branching places exist and may be combined to TP- or PT-handles. We therefore extend the proposition and additionally require the free-choice WF-net to be well-structured.

Theorem 4.8. *Let PN be a free-choice and well-structured WF-net with input place i. Let PN be relaxed sound. Then PN is sound.*

Proof PN is well-structured. Therefore, the short-circuited net $\overline{PN}$ is well-handled and strongly-connected. Using Theorem 2.38/2.39 it can be concluded that $\overline{PN}$ is well-formed (structurally bounded and structurally live). Soundness of PN coincides with liveness and boundedness of $(\overline{PN}, i)$, Theorem 2.48. Therefore, $\overline{PN}$ needs to be live and bounded in the initial marking i.

Boundedness of $(\overline{PN}, i)$ follows from the fact that $\overline{PN}$ is structurally bounded. It remains to prove that $(\overline{PN}, i)$ is live. Figure 4.1 illustrates the relations described.

PN with input place i is relaxed sound. Thus, it can be concluded that there is an infinite firing sequence σ of $(\overline{PN}, i)$ which supports each transition.

Let $\{\sigma_1, \sigma_2, ..., \sigma_n\}$ be a set of sound firing sequences in (PN, i) containing each transition at least once. For the construction of an infinite firing sequence σ they are linked through firing of transition t^*. This again is done infinitely often. The constructed infinite firing sequence is:

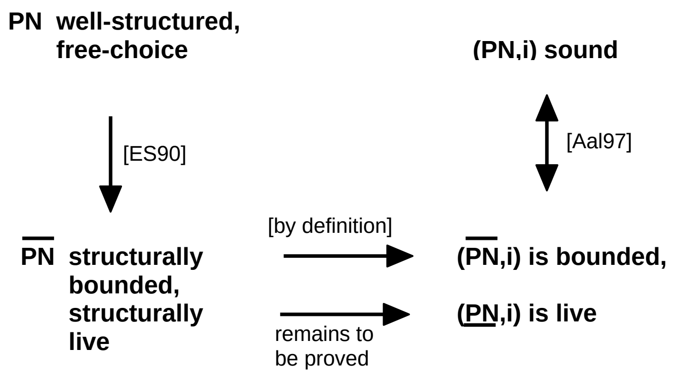

PN well-structured, free-choice
[ES90]
$\overline{PN}$ structurally bounded, structurally live
[by definition]
remains to be proved
($\overline{PN}$,i) is bounded,
($\overline{PN}$,i) is live
[Aal97]
(PN,i) sound

Figure 4.1: Proof relations: A free-choice net showing relaxed soundness is sound

$$\sigma = \sigma_1 t^* \sigma_2 t^* ... t^* \sigma_n t^* \sigma_1 t^* \sigma_2 t^* ... t^* \sigma_n ...$$

With Theorem 2.39 we know that $\overline{PN}$ is covered by S-components.

Every S-component is a minimum siphon and a minimum trap of $\overline{PN}$. As $\overline{PN}$ is covered by S-components, place i is part of a minimum trap. The initial marking i therefore marks a minimum trap.

The infinite firing sequence σ is enabled at i and contains all transitions. Since every transition has an input place and an output place (strongly connected), every place and therefore every trap is marked during the occurrence of the sequence. Since marked traps remain marked, every trap and therefore every S-component is marked in i. With Theorem 2.44(c), it can be concluded that $(\overline{PN}, i)$ is live and PN is therefore sound.

□

4.2.4 Well-structured Workflow-nets

In this section we will show that the validity of the theorem can also be carried forward to non-free-choice WF-nets. We first establish some prerequisites. We recall a transformation rule ([DE95, Hac72]) that transforms a non-free choice net PN into a free-choice net PN'. We therefore replace every arc $(p, t) \in F$ in PN by a sequence $(p, t')(t', p')(p', t) \in F'$ and extend the sets P and T accordingly. With the introduction of these additional sequences, the decision points become localized, which characterizes free-choice nets. Figure 4.2 illustrates the rule which is applied to transfer non-free-choice nets into free-choice nets.

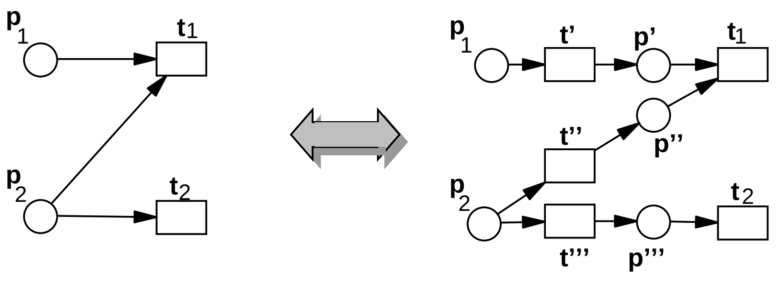
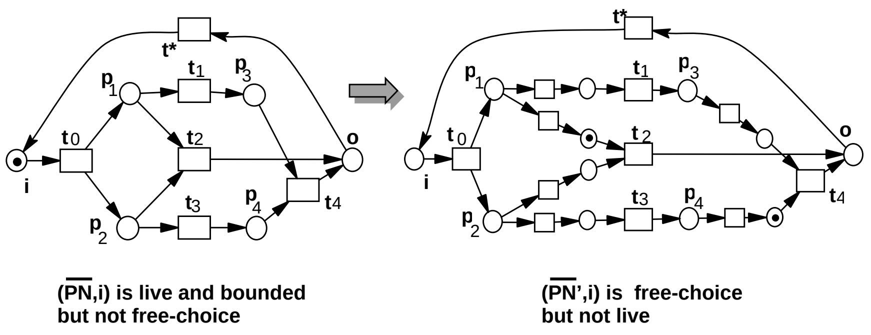


Figure 4.2: Free-choice transformation rule

Although one would think that the introduction of these simple sequences preserves properties such as boundedness and liveness, this does not hold in every case. Figure 4.3 shows an example illustrating that the forward direction of the proposed transformation does not maintain liveness. The example is a counter-example to the reduction rule 9 in [Sta90] where the contrary is claimed.



t*
p1
t1
p3
t0
t2
o
i
t3
p4
t4
p2
t*
p1
t1
p3
t0
t2
o
i
t3
p4
t4
p2
(PN,i) is live and bounded but not free-choice
(PN',i) is free-choice but not live

Figure 4.3: A counter-example vitiating the consistence claim of rule 9 in [Sta90]

However, every live and bounded free-choice net remains live and bounded if it is transformed into a non-free-choice net (retracing the free-choice transformation).

Lemma 4.9. *Let $PN = (P, T, F)$ be a Petri net and $p \in P, t \in T$, and $(p, t) \in F$. Let $t', p' \notin T \cup P$ and $PN' = (P \cup \{p'\}, T \cup \{t'\}$, $(F \backslash \{(p, t)\}) \cup (\{(p, t'), (t', p'), (p', t)\})$, and M be a marking of PN. If the system (PN', M) is live and bounded, then (PN, M) is live and bounded.*

Proof In PN' a transition t' and a place p' have been added. Their introduction enhances the behavior of the primary system. Still, any behavior of the primary system (PN, i) can be simulated by the enhanced system (PN', i). Simulation also works in the opposite direction. Regarding, the firing of transition t' as a silent move (cf. e.g.[GW96]), any execution of the system (PN', i) can be mapped onto an execution in (PN, i).

bounded: We know the enhanced system (PN', i) is bounded. As it is possible to simulate all behavior of the primary system (PN, i), it must be bounded as well.

live: The enhanced system (PN', i) is live. As it is possible to simulate all behavior of the enhanced system by the primary system (PN, i), regarding the firing of transition t' as a silent move, (PN, i) must be live as well.

We can conclude that (PN, i) is live and bounded if (PN', i) is.

□

Furthermore, it is apparent that properties such as well-handledness and relaxed soundness are preserved throughout the free-choice transformation.

Lemma 4.10. *Let $PN = (P, T, F)$ be a WF-net and $p \in P, t \in T$, and $(p, t) \in F$. Let $t', p' \notin T \cup P$ and $PN' = (P \cup \{p'\}, T \cup \{t'\}, (F \backslash \{(p, t)\}) \cup (\{(p, t'), (t', p'), (p', t)\})$, and M be a marking of PN. If the system PN is well-structured, then PN' is well-structured.*

Proof PN is well-structured, i.e. there are no PT- or TP- handles in the short-circuited net $\overline{PN}$. PN and PN' only differ with respect to the insertion of what was called a SISO-sequence [Fre02] (Single Input Single Output), between place p and transition t. This insertion does not introduce a new path to PN but just extends an existing one. Therefore, it can be concluded that $\overline{PN'}$ does not contain any PT- or TP- handles i.e. PN' is well-structured.

□

Lemma 4.11. *Let $PN = (P, T, F)$ be a WF-net and $p \in P, t \in T$, and $(p, t) \in F$. Let $t', p' \notin T \cup P$ and $PN' = (P \cup \{p'\}, T \cup \{t'\}, (F \backslash \{(p, t)\}) \cup (\{(p, t'), (t', p'), (p', t)\})$, and M be a marking of PN. If the system (PN, M) is relaxed sound, then (PN', M) is relaxed sound.*

Proof (PN, M) is relaxed sound, i.e. every transition is covered by a sound firing sequence. Let σ be a sound firing sequence containing transition t: $\sigma = \sigma_1\, t\, \sigma_2$.

The difference between PN and PN' only relates to the insertion of a SISO-sequence between place p and transition t. To show relaxed soundness of the system (PN', M) it must therefore only be shown that the extended firing sequences $\sigma' = \sigma_1\, t'\, t\, \sigma_2$, is a sound firing sequence of (PN', M). This follows directly from the construction of PN'.

□

We are now ready to prove the primary proposition.

Theorem 4.12. *Let PN be a well-structured WF-net with input place i. Let (PN, i) be relaxed sound. (PN, i) is sound.*

Proof Let PN be a non-free-choice, but well-structured WF-net with input place i. Let the WF-system (PN, i) be relaxed sound. We apply the free-choice transformation rule and with Lemmata 4.10 and 4.11 obtain a relaxed sound and well-structured WF-system (PN', i) which is additionally free-choice. We short-circuit PN' and obtain the strongly connected, well-handled and free-choice net $\overline{PN'}$. Using Theorem 2.38/2.39, we can conclude that $\overline{PN'}$ is well-formed (structurally bounded and structurally live). With Theorem 4.8, we can infer that $(\overline{PN'}, i)$ is live and bounded. As the reverse direction of the free-choice transformation preserves these properties, we can conclude that $(\overline{PN}, i)$ is also live and bounded. Therefore, (PN, i) is sound.

□

Figure 4.4 illustrates the relations used in the proof.

As a consequence of this result, a WF-system can be proved sound if it satisfies some structural properties (namely having no handles) and relaxed soundness. Furthermore, we can conclude that if the WF-net is relaxed sound but not sound, there must be PT- or TP-handles in $\overline{PN}$. As soundness of a net PN corresponds to liveness and boundedness of $\overline{PN}$, it can be concluded that if $\overline{PN}$ is bounded, it is not live, and vice versa. To illustrate the level of understanding now attained, the relations between the Petri net classes investigated are shown in Figure 4.5.

Let (PN, i) be a relaxed sound WF-system. With Theorem 4.12 we know that well-structuredness of PN is sufficient to conclude soundness of (PN, i). Still, it is not a necessary condition. An example for a sound WF-system which is not well-structured was given in Figure 2.1.

So far, the relation between relaxed soundness and soundness has been investigated. In the rest of this chapter, relations to other Petri net properties (T-invariants, T-coverability) are considered.

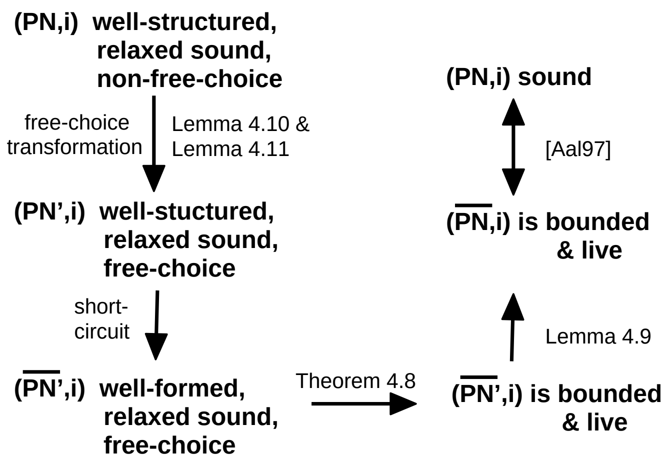

(PN,i) well-structured, relaxed sound, non-free-choice
free-choice transformation
Lemma 4.10 & Lemma 4.11
(PN',i) well-stuctured, relaxed sound, free-choice
short-circuit
($\overline{PN'}$,i) well-formed, relaxed sound, free-choice
Theorem 4.8
($\overline{PN'}$,i) is bounded & live
Lemma 4.9
($\overline{PN}$,i) is bounded & live
[Aal97]
(PN,i) sound

Figure 4.4: A well-structured net showing relaxed soundness is sound

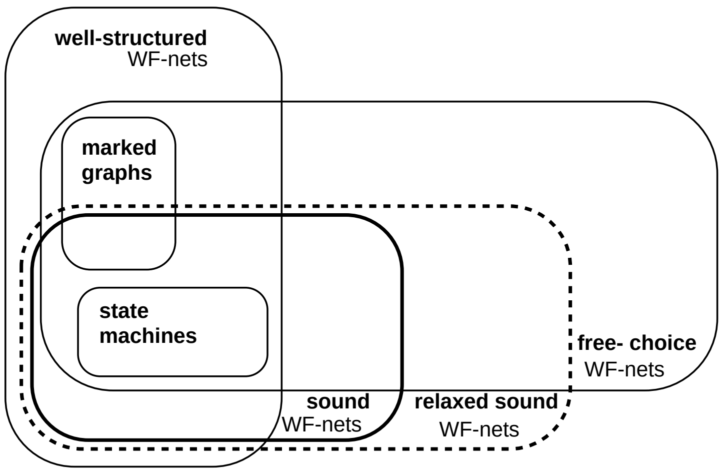


Figure 4.5: Relations between different Petri net-properties

4.3 Other relations

Relaxed soundness states that every transition within a WF-net contributes to the desired behavior of the corresponding WF-system, i.e. is covered by one of the sound firing sequences. It is obvious that there is a relation between relaxed soundness and the number of T-invariants.

We will now prove the following theorem:

Theorem 4.13. *Let $PN = (P, T, F)$ be a WF-net with input place i. Let the system (PN, i) be relaxed sound. The short-circuited net $\overline{PN}$ is covered by T-invariants.*

Proof Relaxed soundness states that each transition $t \in T$ is contained in a sound firing sequence. Let $\sigma = t_1...t...t_n$ be a firing sequence leading from state i to state o: $i \xrightarrow{\sigma} o$. In $\overline{PN}$ the additional transition t^* returns the token from place o to place i, resetting the system to its initial state. $\overline{PN}$ is covered by transition invariants because for each transition t the vector $(t_1...t...t_n, t^*)$ denotes a T-invariant covering t.

□

The existence of enough T-invariants is therefore a necessary condition for relaxed soundness. The test for a cover of T-invariants was implemented in the verification tool Woflan [VA00]. Woflan can compute whether a net is covered by semi-positive T-invariants. Worst-case complexity of the computation is exponential in the number of transitions. Typically, the complexity will be much better. Once the set of semi-positive T-invariants has been computed it is also necessary to check whether all transitions are covered. From Theorem 4.13, it follows that transitions that are not covered by any invariant are not contained in any sound firing sequence. They are therefore candidates for revision.

The converse does not hold. The coverability of $\overline{PN}$ by T-invariants is a necessary but not a sufficient condition for relaxed soundness of (PN, i). Even if we assume that PN has no dead transitions it is not possible to conclude relaxed soundness. A counter-example is shown in Figure 4.6. The short-circuited net is covered by T-invariants, e.g. the positive T-invariant $Y = (2, 1, 1, 1, 2, 2)$. Furthermore, there are no dead transitions. But the underlying WF-system is not relaxed sound. There is no sound firing sequence.

There is a relation between T-components and T-invariants (cf.Proposition 2.29). Exploiting this relation one could assume that the T-coverability of the WF-net

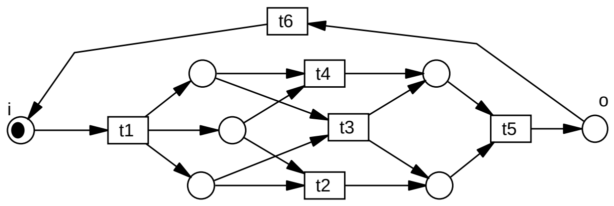


Figure 4.6: Counter-example: The short-circuited WF-net is covered by T-invariants and there are no dead transitions. Nevertheless the underlying WF-system is not relaxed sound.

together with the assumption of no dead transitions in PN is sufficient to conclude relaxed soundness.

By providing counter-examples we will show that the T-coverability of a WF-net is neither sufficient nor necessary to conclude relaxed soundness. Figure 4.7 shows two WF-nets. The first WF-net (a) is covered by T-components and has no dead transitions. Still it is not relaxed sound, as there are no sound firing sequences covering transition t and t'. The problem is that the induced T-invariants are not realizable. The second WF-net (b) is relaxed sound but not T-coverable.

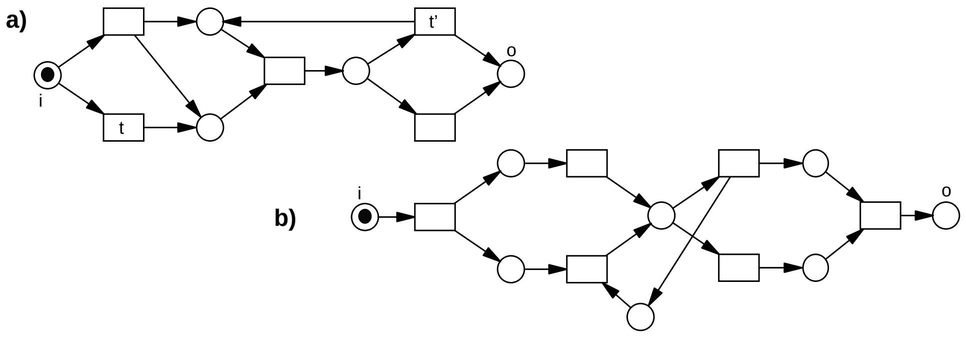


Figure 4.7: Examples disproving a relationship between relaxed soundness and the T-coverability of a WF-net

4.4 Related work

The section on related work is subdivided into two parts. First we search the literature on properties related to relaxed soundness. Then one specific approach is sketched, where further correctness notions relevant for process descriptions have been identified.

Relaxed soundness In the available literature there are no criteria within Petri net theory that resemble relaxed soundness. It does not represent a classic Petri net property, as relaxed soundness is formulated with respect to the underlying transition system. Our investigations suggest that relaxed soundness cannot be restated using a combination of classical Petri net properties, analogous to the result provided in [Aal97] stating that a WF-net is sound if and only if the short-circuited net is live and bounded.

In an approach described recently in [Sim02], a similar notion to the term *sound firing sequence* is provided. The approach deals with a special class of Petri nets with one *start transition s* (the only transition with empty preset), one *goal transition g* (the only transition with empty postset), and initially unmarked. In [HSV03] a strongly connected net of this kind has been called tWF-net. Here the concept of a WF-net has been generalized to cover tWF-nets and sWF-nets, where tWF-nets start and end each with one transition and sWF-nets start and end each with one place.

In the approach described in [Sim02], a firing sequence reproducing the empty marking has been called a *process*, whereby the start transition s and the goal transition g occur exactly once. The latter requirement could easily be implemented by adding one input place i to the start transition s and one output place o to the goal transition g. This way the net is transformed into a WF-net (or sWF-net). What was called a *process* in the primary net would then correspond to a sound firing sequence (reaching state o) in the extended net, initially marked in i. Correspondingly, in this specific net-class, relaxed soundness could be reformulated w.r.t. what was called a *process*.

Serializability and separability Only recently, another attempt was made to find further correctness notions relevant for process descriptions [HSV03]. The classical formulation of soundness (cf. [Aal98]) does not combine with refinement. Problems may occur if new firing sequences arise. The latter may happen if the inserted WF-net is triggered more than once.

Therefore, in [HSV03] soundness was generalized to the notion of *k-soundness*, allowing the observation of more than one instance (namely k) at the same time, cf. Section 2.3. Here, a WF-net is called k-sound if any marking reached from k tokens in the initial place can reach k tokens in the final place. Regarding the original soundness as 1-soundness, a WF-net is called sound iff it is k-sound for each $k > 0$.

Extending the applicability of WF-nets for the investigation of compositional processes, further correctness concepts get necessary. First, the notion of *serializability* is introduced, a behavioral property stating that the behavior of a WF-net with k initial tokens can be seen in some sense as a combination of the behavior of k copies of the net each of which has one initial token [HSV03]. Only looking at the markings of the net this notion is softened to the notion of *weak separability*. It is proven that a weak separability is sufficient to reduce the problem of soundness to 1-soundness. Finally, *separability* a notion stronger than weak separability but not as restrictive as serializability is introduced. For a subclass of WF-net, it is proven that this notion combines w.r.t. to refinement.

In contrast to soundness, every relaxed sound WF-net gained by refinement will satisfy relaxed soundness again, as soon as the primary WF-nets were relaxed sound. This follows immediately as new and possibly problematic firing sequences that arise through the replacement of a transition or place by a tWF-net or a sWF-net respectively, may be ignored.

Chapter 5

Non-controllable choice robustness

So far, the modeler was guided towards a process description that is relaxed sound. It is clear that relaxed soundness is too weak for the corresponding specification to be used as basis for execution support. The criterion does not exclude bad executions but just makes a statement about the existence of good ones. The next question is whether relaxed soundness provides a sufficient prerequisite for the generation of a sound specification.

Until now, a workflow system has been regarded as a stand-alone application, but workflow systems are reactive systems. They run in parallel with their environment, respond to inputs from the environment and produce output events which take effect back in the environment. It is assumed that the knowledge about the behavior of the environment is low. At best, possible actions of the environment are known but not their order.

Starting at these assumptions, it will turn out that relaxed soundness does not suffice as the basis to generate a sound specification. A further correctness criterion is needed, indicating that the process can act robustly to (all) possible events coming from the environment.

In this chapter, the quality measure for process descriptions is extended. The property introduced is called *non-controllable choice robustness*. It provides a means to describe robustness of a system against all possible requests from the environment.

The chapter is organized as follows: First, the requirements for the new perspective are described and our modeling paradigm is introduced. In Section 5.2 we show by example deficiencies of the previous correctness criteria. The new property is introduced in Section 5.3. In Section 5.4, an algorithm is proposed verifying the property, and its correctness is proven. The new criteria is embedded into Petri net theory in Section 5.5. Finally, related work is discussed.

5.1 Workflow systems are reactive systems

In this section modeling requirements of reactive systems are investigated, the capabilities of WF-nets are considered, and WF-nets are refined in order to describe the behavior of reactive systems in an adequate manner.

5.1.1 Modeling requirements of reactive systems

Workflow systems are reactive systems. They run in parallel with their environment, respond to inputs from the environment and produce output events that have impact on 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 the question about available capacities, the check for creditworthiness of a customer and the identity check of a co-operating partner.

Possible results of an information evaluation as well as the occurrence of external events should be reflected within the process specification. This is essential as the further execution differs with respect to various results or different external events.

5.1.2 Capabilities of Workflow-nets

In a WF-net the interaction with the environment is not reflected explicitly. As stated in Section 2.2, tasks are modeled by transitions and intermediate states are modeled via places. Looking at WF-nets and incorporating the discussion in [Aal98], it can be seen that transitions are also used for other purposes than the modeling of tasks.

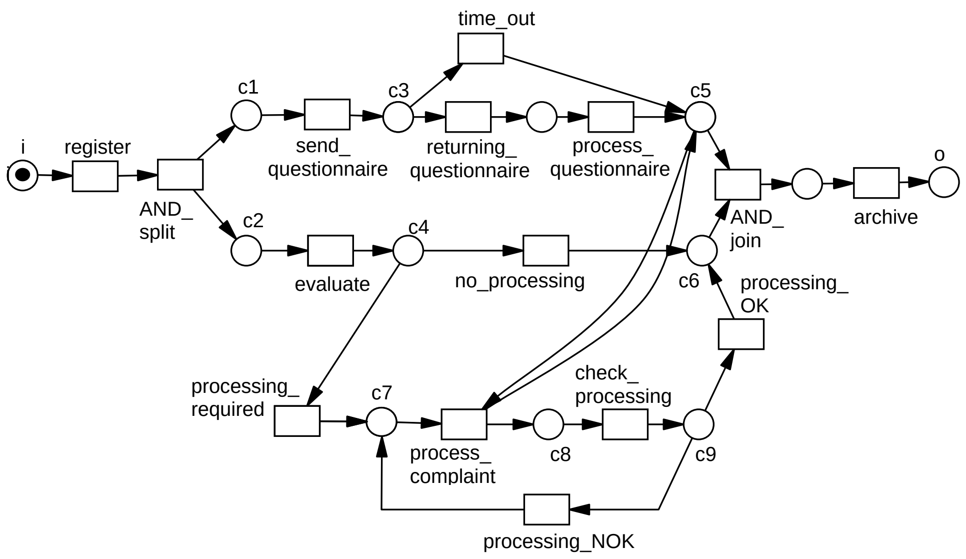


Figure 5.1: A WF-net modeling the processing of complaints

Transitions are used in four different ways:

Tasks Mostly, transitions are used to model tasks. Tasks stand for activities that are executed by some actor (a human or a machine). Tasks in the process of Figure 5.1 are `register`, `send_questionnaire`, `evaluate`, `process_questionnaire`, `process_complaint`, `check_processing` and `archive`.

Decisions Transitions are also used to depict the outcome of decision-making processes. Alternative decisions are assumed to be mutually exclusive. Examples in Figure 5.1 are `processing_OK`, and `processing_NOK`, as well as `no_processing`, and `processing_required`. Transitions reflecting a decision always occur immediately after a task transition, e.g. here `evaluate` and `check_processing`.

External events Sometimes transitions represent external events. Examples of such transitions in the process shown in Figure 5.1 are `time_out`, modeling a time event, or `returning_questionnaire` (modeling the income of an answered questionnaire). It cannot always be assumed that external events are mutually exclusive. Consider the example in Figure 5.2 modeling a part of a library process. Possible external events are `reader returns books` (e.g. in person), `reader asks for extension` (e.g. using a web interface) and `time elapsed`. Although, it is unlikely that more than one event arises, this cannot be excluded. However, a reasonable assumption is that conflicting external events do not occur at exactly the same point in time. This way a system waiting for an external event is able to determine which task is scheduled next.

Routing Finally, transitions are used for the sole purpose of *routing* a case. This usually occurs when a case needs to be split into parallel parts or parallel parts of a case need to be merged. Examples in Figure 5.1 are `AND_split` and `AND_join`.

To differentiate the different purposes for which a transition is used it can be assigned to one of the following types:

$$type : T \rightarrow \{task, event, decision, routing\}.$$

5.1.3 Reflecting the interaction with the environment

The process description will be used to support the execution of a case at run-time. Then a workflow engine, also referred to as workflow controller, is used to schedule the case according to the rules specified in the process description.

Operating on the process description, the workflow controller decides which enabled transition is executed and when. Still, the workflow controller may not enforce the firing of any type of transitions. Transitions that are beyond the control of a workflow engine are transitions of the type *event* and *decision.*

It is clear that a workflow controller cannot force an external event to occur[1]. The kind of external event, as well as its occurrence time, are beyond control. For example, in the library it is clear that the workflow controller is not able to force a reader to return books to the library.

Furthermore, the workflow controller cannot force the evaluation of external data to end with a specific result. It is clear that whether e.g. a customer is creditworthy or not, is beyond the control of the workflow engine.

Transitions of type *event* and *decision* are called non-controllable. Their firing can not be forced by the local workflow control but depend either on the evaluation of external data or on the kind of incoming event.

[1]Note that transitions modeling a time-out are of type *event*, expressing the inability of the WF-controller to influence the elapse of time.

To refer to these two types, the set of transitions T of a WF-net $PN = (P, T, F)$ is split into disjoint sets of *controllable* and *non-controllable* transitions: $T = T_{NC} \cup T_{CON}$ and $T_{NC} \cap T_{CON} = \emptyset$. *Controllable* transitions (T_{CON}) model transitions whose executions, in contrast to the execution of *non-controllable* transitions (T_{NC}), are covered by the local workflow control. *Controllable* transitions will be denoted by white boxes, and *non-controllable* transitions will be represented by gray boxes.

In the preceding discussion it was assumed that alternative outcomes of one decision-making process are mutually exclusive. Furthermore, it was stated that conflicting external events do not occur at the same time. Following these assumptions, non-controllable transitions are modeled as part of a choice which satisfies the free-choice property and does not contain any controllable transitions. This restricted modeling reflects the fact that the behavior of the environment cannot become disabled through the local control. We refer to these choices as *non-controllable choices*. Conversely, we speak about *controllable choices* if the choice only contains controllable transitions, i.e. transitions of type *routing* and type *task*.

Further assumptions for a non-controllable choice are *progress* and *strong fairness*.

Progress: We assume that if a system waits for the outcome of a non-controllable choice, one of the non-controllable transitions will eventually fire.

Strong fairness: We assume that if the same non-controllable transition is enabled infinitely often it will fire infinitely often.

An example of a non-controllable choice is shown in Figure 5.2.

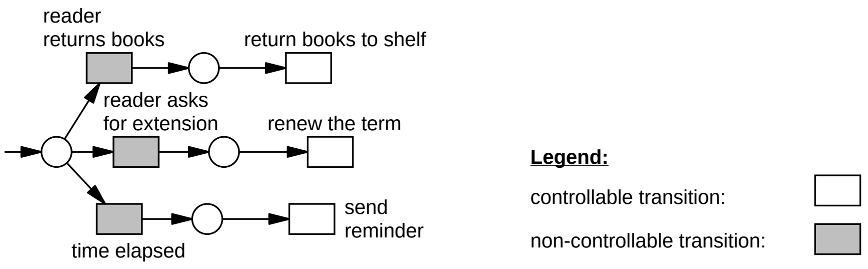


Figure 5.2: Part of a library process

In the following we will only consider WF-nets whose choices are controllable or non-controllable. These WF-nets are called *choice-consistent*. They are defined as follows:

Definition 5.1 (Choice consistent WF-net).
A WF-net $PN = (P, T, F)$, with $T = T_{CON} \cup T_{NC}$ and $T_{CON} \cap T_{NC} = \emptyset$ is choice consistent iff:
(i) Controllable transitions do not conflict with non-controllable transitions:
$^\bullet T_{CON} \cap {^\bullet T_{NC}} = \emptyset$.
(ii) Non-controllable transitions are free-choice:
$\forall t, t' \in T_{NC} : {^\bullet t} \cap {^\bullet t'} = \emptyset \vee {^\bullet t} = {^\bullet t'}$.

Using the term WF-net in the following, we always refer to the refined version of a *choice consistent WF-net.*

5.2 Ability to control a process

The general objective was to provide a criterion describing the possibility to control a process independently from the behavior of the environment. It is clear that non-controllable choices have to be considered when defining such a criterion, because non-controllable transitions cannot be influenced by a controller. A process can be controlled if there is a way to terminate properly, independently from the outcome of the non-controllable choices (but assuming progress and strong fairness). A bad combination of non-controllable, or controllable and non-controllable choices may inhibit proper termination. This could happen, even when the process description satisfies relaxed soundness, which only makes a proposition about the existence of proper executions, but does not necessarily cover all possible combinations of choices.

In the following, two examples are discussed for which proper termination cannot be guaranteed. The process description in Figure 5.3 models the planning of a trip. It consists of parallel booking queries for flights and hotels. One possibility of termination is modeled via the transition `finish`. Here, both booking tasks succeeded (`f:ok, h:ok`). The trip can be started. The other possibility is modeled via the transition `cancel_trip`. The trip is canceled if neither a hotel (`h:not_ok`) nor a flight (`f:not_ok`) can be found. The results of the booking queries are modeled as non-controllable choices because the decision computation relies on external data.

The process description in Figure 5.3 is relaxed sound. There are sound firing sequences containing all transitions:

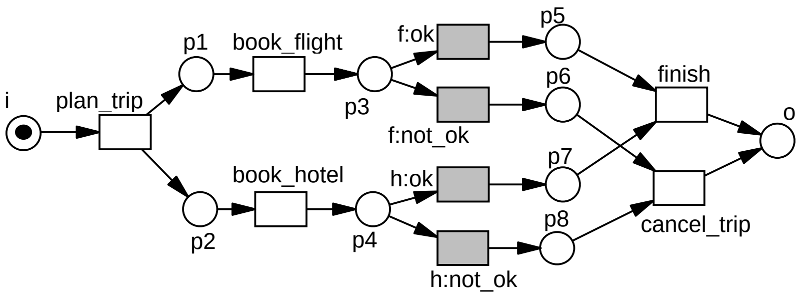


Figure 5.3: Example: "Planning trip"

Example "Planning trip"

- `plan_trip, book_flight, book_hotel, h:ok,f:ok, finish`
- `plan_trip, book_flight, book_hotel, f:not_ok, h:not_ok, cancel_trip`

Although the process is relaxed sound, it cannot always be controlled properly. The results of the bookings cannot be influenced by a controller. The system deadlocks if one of the bookings succeeds but the other one fails.

Another example is shown in Figure 5.4. It describes the work of a cashier[2]. Predicting the behavior of the customer, the cashier already holds an amount of change waiting for the customer to pay. This is modeled via the controllable choice `hold_changeA` or `hold_changeB`. The payment of the customer is modeled via activities `payA` and `payB`. The cashier had guessed correctly if the amount of change held corresponds to the sum provided by the customer. Only then is the money exchanged. This is modeled via transitions `collectA` and `collectB`.

The choice of the customer `payA` and `payB` is non-controllable. The amount the customer tenders cannot be influenced but only depends on the mood and money of the customer.

The process description in Figure 5.4 is relaxed sound. There are sound firing sequences containing all transitions:

[2]Note that the example does not match a realistic business process but only illustrates the concept

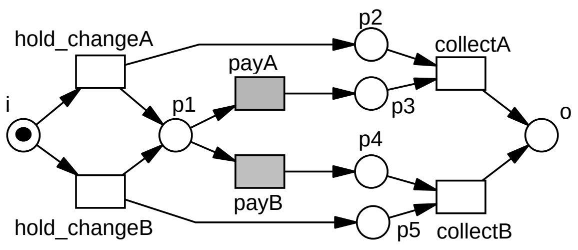


Figure 5.4: Example: "Provide change"

Example "Provide change"

- `hold_changeA, payA, collectA`
- `hold_changeB, payB, collectB`

Nevertheless it cannot be scheduled properly for any case. This is because the controllable choice `hold_changeA` or `hold_changeB` takes place before the non-controllable choice. Hence, the cashier cannot react if a different amount was chosen by the customer. The modeled process does not terminate properly but deadlocks.

These examples show combinations of controllable and non-controllable choices which prevent the possibility of guaranteeing proper termination. It is not possible to force the process to choose between alternative transitions correctly because the non-controllable choices can not be influenced. Therefore, relaxed soundness does not cover controllability. It is not possible to guarantee proper termination as soon as events from outside the system are considered, or a data dependent decision influences the behavior of the system.

5.3 Non-controllable choice robustness

In this section, a further correctness criterion for process descriptions is introduced, denoted as *non-controllable choice robustness.* It provides a further means to describe the correctness of a system. If a system is non-controllable choice robust (short: robust), it is possible to control its execution so that it terminates properly. This is possible despite all interaction with the environment.

To define the criterion, we will look at our problem as a game between the workflow controller and the environment as an opponent who is trying to interfere with the process execution, so that an unsound firing sequence is generated. The question is whether the workflow controller can win the game, i.e. react to the moves of the adversary and thus terminate properly.

Looking at our setting from this perspective we can refer to numerous results of controller synthesis using terms and notions from game theory [PR89, McN93, AMP94, Tho95]. In the next sub-section we will refer to these notions and deduce results for workflow modeling.

5.3.1 Game, play, and winning strategy

To start with, terms *game, play, strategy* and *winning strategy* are reviewed. The definitions have been adapted from the concepts used in [Tho95].

A game is defined as a tuple $(\mathcal{G}, \phi)$ consisting of a game graph $\mathcal{G}$ and a temporal formula ϕ (winning condition).

A *game graph* is of the form $\mathcal{G} = (Q, Q_0, Q_1, E, q_i, F)$, where Q is a finite set of states, Q_0, Q_1 defines a partition of Q (depicting the states where it is the turn of player $j, j = 0, 1$ to perform an action), and $E \subseteq (Q_0 \times Q_1) \cup (Q_1 \times Q_0)$ is a set of edges. The underlying graph is required to be bipartite with respect to the state transitions. $q_i \in Q$ is the initial state of the game and $F \subseteq Q$ is a set of accepting states.

A play on a game graph $\mathcal{G}$ corresponds to a path ρ in $\mathcal{G}$ starting in q_i. The winner of a play is fixed by the winning condition ϕ. The first player wins a play ρ if ρ satisfies ϕ. We only focus on the most basic condition on *reachability*, which is satisfied if the first player can force a visit of some state in F. For a survey of possible winning conditions the reader is referred to [Tho95].

A strategy for a given game is a rule that tells a player how to choose between several possible actions in any game position. A strategy is a winning strategy if the player[3] always wins no matter what the environment does [PR89].

A strategy[4] for player 0 in the game $(\mathcal{G}, \phi)$ can be depicted as fragment[5] ($SG \subseteq \mathcal{G}$) of the game graph.

A strategy SG is a winning strategy for player 0 if all the plays on SG win, i.e. all paths ρ in SG satisfy ϕ.

[3]All terms are defined from the perspective of player 0 - the workflow controller.

[4]We only consider no-memory strategies, where the next move only depends on the current state cf. [Tho95].

[5]$SG = (Q', Q'_0, Q'_1, E', q_i, F)$ is a fragment of a game graph $\mathcal{G} = (Q, Q_0, Q_1, E, q_i, F)$ if $Q' \subseteq Q, Q'_0 \subseteq Q'_0, Q'_1 \subseteq Q'_1$, and $E' = E \cap ((Q'_0 \times Q'_1) \cup (Q'_1 \times Q'_0))$.

A winning strategy SG is called *maximum* iff there is no winning strategy SG' such that $SG \subset SG'$.

5.3.2 Adaption for workflow modeling

In the following, we will adapt the new concepts for application in the domain of workflow modeling. It would be natural to define the game graph on the basis of the reachability graph RG of a system $S = (PN, i)$, where q_i corresponds to i and the set of accepting states F to $\{o\}$. As the definition of a game graph requires a finite set of states, we only consider bounded systems, i.e. systems having a finite reachability graph. Boundedness of WF-systems can be checked by standard Petri net tools. The game graph is then of the form $\mathcal{G}_{RG} = (V_{RG}, E_{RG}, i, o)$.

This adaption is not straightforward, as the set of states in V_{RG} is not bipartite. This is introduced through the possible concurrent behavior described in the Petri net. The players (workflow controller and environment) do not have to move in turn and in some states either could make the next move. But the moves of the different players are reflected through the labels at the state transitions. If the label t of a state transition is a controllable transition $t \in T_{CON}$ then a controller move is represented. If the state transition is labeled with a non-controllable transition $t \in T_{NC}$ it corresponds to a move by the environment. With regard to the labels, the state transitions are referred to as controllable or non-controllable state transitions.

Using this distinction, a strategy for the workflow controller can then again be defined as a fragment of the game graph, (V_{RG}, E_{RG}, i, o) where E_{RG} can be decomposed with respect to the labeling.

Definition 5.2 (Strategy).
Let $(\mathcal{G}_{RG}, \phi)$ be a game. Let $\mathcal{G}_{\mathcal{RG}} = (V_{RG}, E_{RG}, i, o)$ be a game graph, where $RG = (V_{RG}, E_{RG})$ is the reachability graph of a WF-system $S = (PN, i)$.
$SG \subseteq RG$, with $SG = (V_{SG}, E_{SG})$, $V_{SG} \subseteq V_{RG}$, and $E_{SG} \subseteq E_{RG}$, is a strategy if:

1. *$i \in V_{SG}$*
2. *For each $M \in V_{SG}$ there is a directed path from i to M.*
3. *SG is self-contained with respect to the possible moves of the adversary: for all $M \in V_{SG}, t \in T_{NC}, M' \in V_{RG}$: if $(M, t, M') \in E_{RG}$ then $(M, t, M') \in E_{SG}$.*

Using the winning condition of “proper termination” ($\phi = \Box\Diamond o$ - in LTL parlance [MP92]) a winning strategy for the workflow controller is defined as follows:

Definition 5.3 (Winning strategy).
Let $(\mathcal{G}_{RG}, \phi)$ be a game defined on the reachability graph RG of a system $S = (PN, i)$ with $\phi = \Box\Diamond o$. Let SG be a strategy for the WF-controller. SG is a winning strategy if it only contains paths that satisfy ϕ. Therefore, SG additionally meets the following requirements:

1. *$o \in V_{SG}$*
2. *For each $M \in V_{SG}$ there is a directed path from M to o in SG.*

This definition of the term strategy, which is usual in controller synthesis (e.g. [Tho95]) is not very strong. It means that certain choices may never be presented to the environment. For our setting, a stronger understanding is necessary. A strategy should incorporate the requirement that all possible moves by the environment have to be covered at least once. This corresponds to the requirement that it should be possible to react to *all* possible moves of the environment.

Definition 5.4 (Complete strategy).
Let $(\mathcal{G}_{RG}, \phi)$ be a game defined on the reachability graph RG of a WF-system $S = (PN, i)$. Let SG be a strategy for the WF-controller. The strategy SG is called complete *if all possible moves of the adversary are covered.*
Formally: $\forall t \in T_{NC} : \exists M, M' \in V_{SG}, (M, t, M') \in E_{SG}$.

Based on these notions, it is now possible to express non-controllable choice robustness of a WF-system.

Definition 5.5 (Non-controllable choice robustness).
Let $(\mathcal{G}_{RG}, \phi)$ be a game defined on the reachability graph RG of a WF-system $S = (PN, i)$ with $\phi = \Box\Diamond o$. The system S is non-controllable choice robust (short: robust) iff there exists a complete winning strategy SG for the workflow controller.

A WF-system is robust if there is a fragment of the reachability graph which starts in i, ends in o, contains at least one t-labeled state transition for any non-controllable transitions $t \in T_{NC}$, and has only controllable state transition leading out of the fragment[6]. Assuming progress for non-controllable choices, the existence of such a fragment guarantees that it is possible to reach state o (terminate

[6]Sometimes, we will refer to this fragment as “robust fragment”

properly) independent of the outcome of non-controllable choices. While all non-controllable transitions (all possible moves of the adversary) are covered by the fragment, 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 moves by the adversary.

The process descriptions of Figure 5.3 and 5.4 are not robust because there is no robust fragment. Figure 5.5 shows the reachability graphs of both processes. Non-controllable choices have been depicted as curves around the corresponding state transitions.

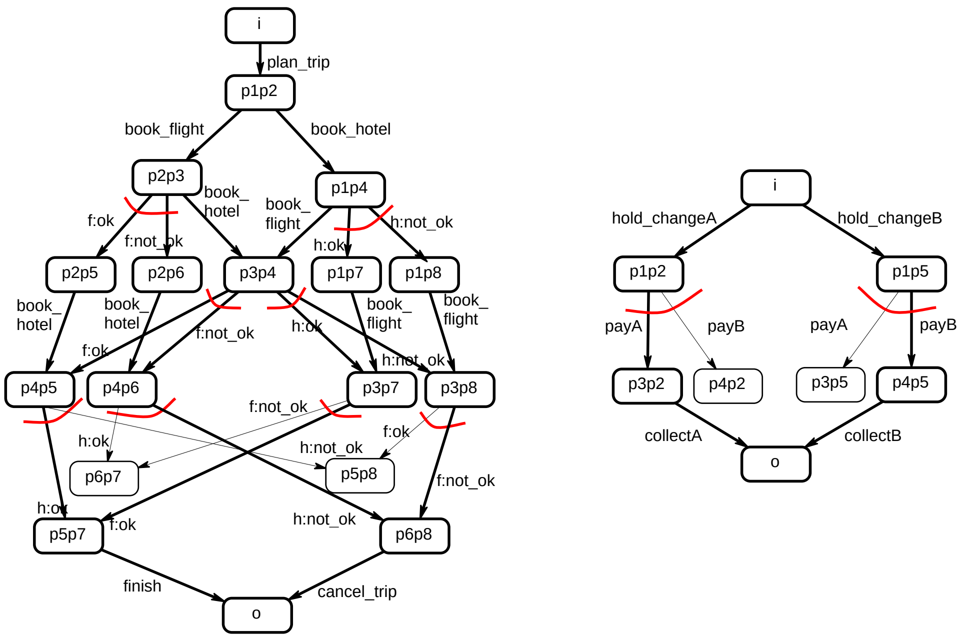


Figure 5.5: RGs for examples: “Planning trip” and “Provide change”

The fragments depicted by showing associated state transitions in bold satisfy all requirements except Requirement 3 in Definition 5.2. There are non-controllable state transitions leaving the fragment.

In the following, we will add some behavior to the process descriptions to make them robust. Figures 5.6 and 5.7 show the modified WF-nets. In the example “Planning trip”, one further controllable task is introduced. The new task `abort_hotel_booking` is executed if the flight booking did not succeed.

In the process description of the second example (Figure 5.7) we added two fur-

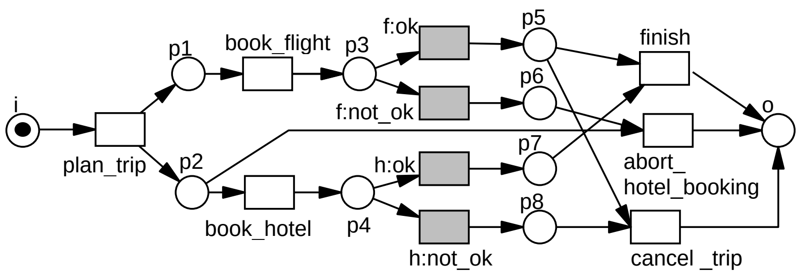


Figure 5.6: Revised example: “Planning trip”

ther tasks to cope with the possible deadlocks. They are executed if the predicted change does not fit the sum provided by the customer. The transitions `changeBA` and `changeAB` model the behavior of the cashier.

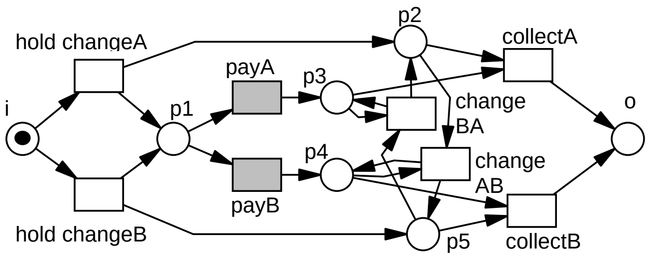


Figure 5.7: Revised example: “Provide change”

Both specifications are now robust. The graphs are shown in Figure 5.8. The robust fragments have been depicted by showing the associated state transitions in bold. Note, the strategy for the example “Planning trip” (cf. Figure 5.8) restricts the possible behavior. It suggests delaying the booking of the hotel until the flight booking succeeded. If the result of the flight booking was negative (`f:not_ok`), the hotel booking and thus the trip are canceled directly (`abort_hotel_booking`).

The strategy for the example “Provide change” (cf. Figure 5.7) does not formulate any restrictions as it coincides with the reachability graph.

In the following section, a constructive algorithm is provided. The algorithm checks whether a bounded WF-system is robust. In the positive case, the maximum and complete winning strategy for the workflow controller is returned.

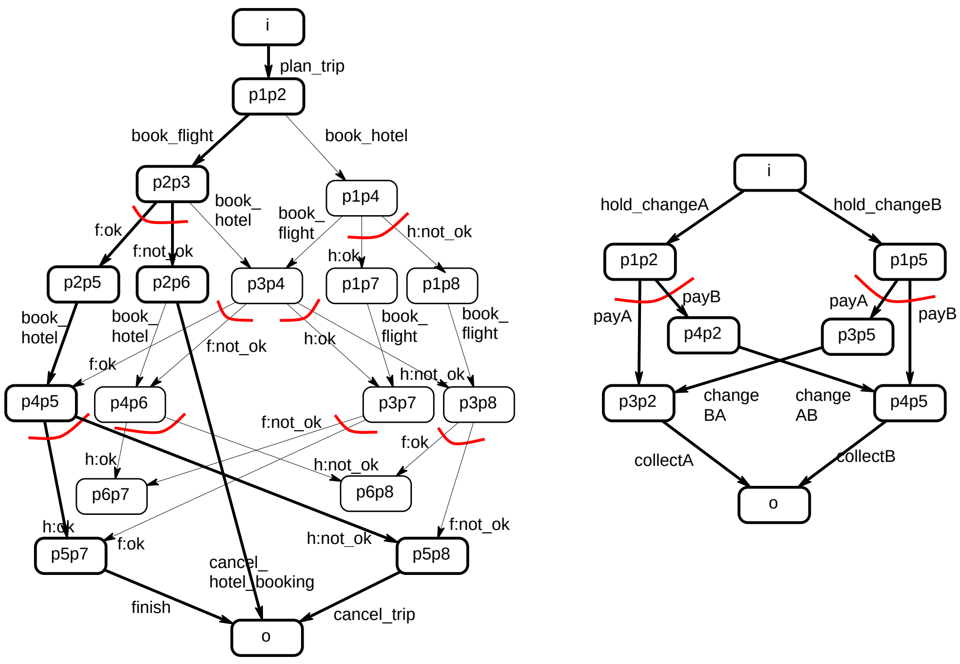


Figure 5.8: RGs for revised examples: “Planning trip” and “Provide change”

5.4 An algorithm verifying robustness

There are various algorithms in the literature that determine the existence of a strategy in an effective manner, e.g. [NYY92, McN93, AMP94, Tho95]. Still, these algorithms work on a proper game graph (bipartite graph) and do not necessarily compute a complete strategy. For the domain of workflow modeling we adopted the algorithmical idea and provide an algorithm that operates on a finite reachability graph.

The algorithm is given in Figure 5.9. It decides whether a given WF-system $S = (PN, i)$ is robust and in the positive case returns the maximum and complete winning strategy $SG = (V_{SG}, E_{SG})$. Let $PN = (P, T_{CON} \cup T_{NC}, F)$ be a choice-consistent WF-net and (PN, i) be bounded. Let $RG_S = (V_{RG}, E_{RG})$ be the (finite) reachability graph of the WF-system $S = (PN, i)$. As prerequisite for the computation of $SG = (V_{SG}, E_{SG})$, furthermore, the set $Pred_{RG}(o) \subseteq V_{RG}$ of predecessors of state o (see Section 2.1.2 for definition) is computed beforehand.

The algorithm 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. Consequently, the algorithm starts by marking all elements of the predecessor set of o: $V_{SG} := Pred_{RG}(o)$[7]. The corresponding state transitions are determined thereafter. Furthermore, the set of illegal states is computed (denoted by the auxiliary variable X). Illegal states are states from where non-controllable state transitions leave the fragment.

The core of the algorithm is Step (ii). By passing the **while**-loop the current fragment is diminished until it either satisfies the properties of a winning strategy or coincides with the empty fragment. In the entry-condition of the **while**-loop, it is tested whether there are illegal states (**while** $X \neq \emptyset$) indicating that the current fragment does not satisfy Requirement 3 of a strategy. Within the loop, the set of states of the fragment is diminished by the illegal states and corresponding state transitions are cut (a1/a2). Subsequently, the remaining fragment is computed (b1/b2). Finally the set X of illegal states is recomputed. The loop will eventually terminate, as the empty fragment does not satisfy the entry-condition of the **while**-loop.

In the next step (Step (iii) in Figure 5.9) it is checked whether the resulting fragment is empty, which is the case if the intersection of $Succ_{SG}(i)$ and $Pred_{SG}(o)$ was

[7]The set $Pred_{RG}(o)$ is finite as the reachability graph is finite.

```
(i)    $V_{SG} := Pred_{RG}(o)$;
       $E_{SG} := E_{RG} \cap (V_{SG} \times T \times V_{SG})$;
         (* Initially, fragment $SG$ is set to the sound subgraph. *)
       $X := \{M \in V_{SG} | (M, t, M') \in E_{RG} \wedge t \in T_{NC} \wedge M' \notin V_{SG}\}$;
         (* Illegal states are computed. *)
(ii)   while $X \neq \emptyset$;
         do
(a1)       $V_{SG} := V_{SG} \setminus X$;
(a2)       $E_{SG} := E_{RG} \cap (V_{SG} \times T \times V_{SG})$;
           (* Illegal states and corresponding state transitions are cut. *)
(b1)       $V_{SG} := \{M \in V_{SG} | M \in Succ_{SG}(i) \cap Pred_{SG}(o)\}$;
(b2)       $E_{SG} := E_{RG} \cap (V_{SG} \times T \times V_{SG})$;
            (* Coherent fragment is recomputed. *)
           $X := \{M \in V_{SG} | (M, t, M') \in E_{RG} \wedge t \in T_{NC} \wedge M' \notin V_{SG}\}$;
            (* Current set of illegal states is computed *)
         od
(iii)  if $V_{SG} = \emptyset$;
       then print ($S$ is not robust, there is no winning strategy); abort;
       fi
(iv)   $Y := T_{NC} \setminus \{t | (M, t, M') \in E_{SG} \wedge t \in T_{NC}\}$;
       if $Y > \emptyset$;
         (* The strategy is tested to be complete. *)
       then print ($S$ is not robust, there is no complete strategy); return $Y$;
       else print ($S$ is robust); return $(V_{SG}, E_{SG})$;
       fi
```

Figure 5.9: Robustness algorithm

empty. In this case the algorithm aborts with the information “S is not robust, there is no winning strategy”.

Finally, Step (iv) checks whether the derived winning strategy is complete, i.e. contains at least one t-labeled state transitions for any non-controllable transition $t \in T_{NC}$. The auxiliary variable Y determines the set of non-controllable transitions not covered by the fragment. If this set is non-empty, the algorithm terminates, returning Y together with the information “S is not robust, there is no complete strategy”. If Y is empty, the algorithm terminates with the statement “S is robust” and returns the derived fragment.

5.4.1 Correctness of the algorithm

We will now investigate the correctness and completeness of the proposed algorithm. We will first prove the following:

Theorem 5.6. *Let $S = (PN, i)$ be a bounded and robust WF-system. The winning strategy computed within Steps (i) and (ii) of the robustness algorithm is maximum.*

Proof Let SG_W be the maximum and complete winning strategy for the workflow controller. As the WF-system is robust we know that SG_W is a fragment of the reachability graph RG_S satisfying the following properties:

1. It contains state i: $i \in V_{SG}$ (cf. Requirement 1 of Def. 5.2),
2. For each $M \in V_{SG}$ there is a directed path from i to M (cf. Requirement 2 of Def. 5.2),
3. $\forall M \in V_{SG}, t \in T_{NC}, M' \in V_{RG}$: if $(M, t, M') \in E_{RG}$ then $(M, t, M') \in E_{SG}$ (cf. Requirement 3 of Def. 5.2)
4. $o \in V_{SG}$ (cf. Requirement 1 of Def. 5.3)
5. For each $M \in V_{SG}$ there is a directed path from M to o in SG (cf. Requirement 2 of Def. 5.3)
6. $\forall\, t \in T_{NC} : \exists M, M' \in V_{SG}, (M, t, M') \in E_{SG}$ (cf. Def. 5.4)

The algorithm operates on the reachability graph RG_S and computes a fragment SG_A. We will prove that the fragment computed as a result of Steps (i) and (ii) coincides with the maximum winning strategy: $SG_A = SG_W$.

$SG_A = SG_W$ holds if the following two propositions hold:

I) SG_A only contains elements that belong to SG_W.

II) SG_W only contains elements that belong to SG_A.

This is shown in the following.

Proposition I: SG_A only contains elements that belong to SG_W

Initially, set V_{SG_A} is set to $Pred_{RG}(o)$. As there exist a complete winning strategy for the WF-controller we know that $Pred_{RG}(o)$ is not empty. Furthermore it can be concluded that the set V_{SG_A} satisfies Properties 1,2,4, and 5. In case there are non-controllable state transitions that leave the fragment (Property 3 is violated) the **while**-loop is entered.

Within the **while**-loop the current fragment is diminished by the illegal states (see line (a1)) and corresponding state transitions (see line (a2)). After that the coherent fragment is recomputed (cf. Figure 5.9(b1) and (b2)). The resulting fragment satisfies again Properties 1,2,4, and 5. The **while**-loop is reentered as long as Property 3 is not satisfied. The **while**-loop will eventually terminate as the empty fragment does not satisfy the entry-condition. The resulting fragment SG_A satisfies the Properties 1-5 of a winning strategy.

Because SG_W is, by definition, the maximum winning strategy, SG_W necessarily embeds SG_A.

Proposition II: SG_W only contains elements that belong to SG_A

Initially, set V_{SG_A} is set to $Pred_{RG}(o)$ (Step(i)). It holds that $V_{SG_W} \subseteq Pred_{RG}(o)$ (follows from Property 5). Therefore, the proposition holds initially. During the **while**-loop (Step (ii)) the set of V_{SG_A} is possibly diminished.

We have to show that no elements are removed from V_{SG_A} that belong to V_{SG_W}. There are only two places were elements are removed from the set V_{SG_A}, namely in the lines indicated by (a1) and (b1). The elements removed in Line (a1) do not belong to V_{SG_W} as they violate Property 3. The elements removed in Line (b1) do not belong to V_{SG_W} as they violate Property 2 and/or Property 5. Thus both removals are safe. As furthermore no new elements are added, it can be concluded that the proposition also holds after the **while**-loop.

$\square$

Theorem 5.7. *The WF-system $S = (PN, i)$ is robust iff the algorithm terminates returning "S is robust".*

We first prove the direction: If the WF-system $S = (PN, i)$ is robust, the algorithm terminates with the result "S is robust".

Proof The WF-system $S = (PN, i)$ is robust, i.e. there is a complete winning strategy for the workflow-controller. The winning strategy is not empty (includes at least state i and state o) and contains a t-labeled state transition for any non-controllable transition $t \in T_{NC}$ (satisfies Property 6).

The fragment computed within Steps (i) and (ii) is maximum. It especially embeds the aforementioned winning strategy. Therefore the conditions in Step (iii) and (iv) are met and the algorithm returns "S is robust".

We will now prove the other direction: If the algorithm terminates returning "S is robust", the WF-system $S = (PN, i)$ is robust.

Proof The maximum winning strategy computed in Step (i) and (ii) is not empty (cf. Step(iii)) and complete (cf. Step(iv)), i.e. satisfies all properties of a robust fragment (see Def. 5.5).

$\square$

Applying this algorithm, a choice-consistent WF-system can be revised until it satisfies the property non-controllable choice robustness.

5.5 Embedding robustness into Petri net theory

In this section we will investigate the relationship between the new criterion robustness and other properties from Petri net theory.

The three criteria *soundness* and *relaxed soundness* and *robustness* are closely related. Soundness implies non-controllable choice robustness as well as relaxed soundness. The subset diagram in Figure 5.10 illustrates the relation between the three different criteria. Note that non-controllable choice robustness implies soundness if the WF-net only contains non-controllable choices. In contrast, if the WF-system contains no non-controllable choices, relaxed soundness implies robustness.

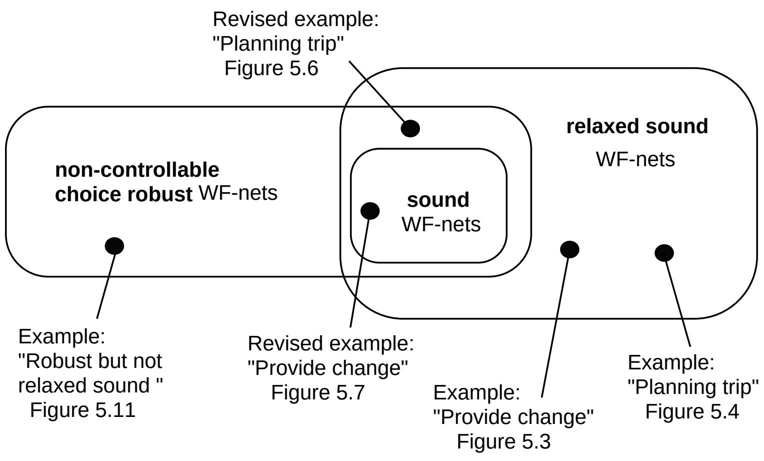


Figure 5.10: Relation of the three criteria

‘ In the previous sub-sections, some examples were presented that illustrate the different subsets depicted in Figure 5.10. The process shown in Figure 5.7 (Page 96) is sound and therefore also relaxed sound and robust. An example of a process which is relaxed sound and robust but not sound is shown in Figure 5.6 (Page 96). Furthermore, there are processes that are relaxed sound but not robust and vice versa. Examples for the first case have been discussed at the beginning of this chapter, cf. Figure 5.4 (Page 91) and Figure 5.3 (Page 90). A process which is robust but not relaxed sound contains internal transitions that are not part of a sound firing sequence. This implies that the process can be controlled but some executions determined by internal choices are not chosen because they do not terminate properly. An example is given in Figure 5.11.

The depicted WF-system is robust as there is a complete winning strategy for the workflow controller, but the WF-net is not relaxed sound, as there is no sound firing sequence containing task e.

There is one phenomena considering WF-systems which are relaxed sound and robust. One would expect that the robust fragment computed by the proposed algorithm contains a t-labeled state transition for *every* transition $t \in T$, i.e. especially also t-labeled state transitions for all controllable transition $t \in T_{CON}$. Still, there are counter-examples were controllable transitions have been removed from the fragment as a consequence of a dependence to non-controllable transitions. Consider the example shown in the Figure 5.12.

Here, a choice which was considered to be controllable ($t1$ and $t2$), is followed

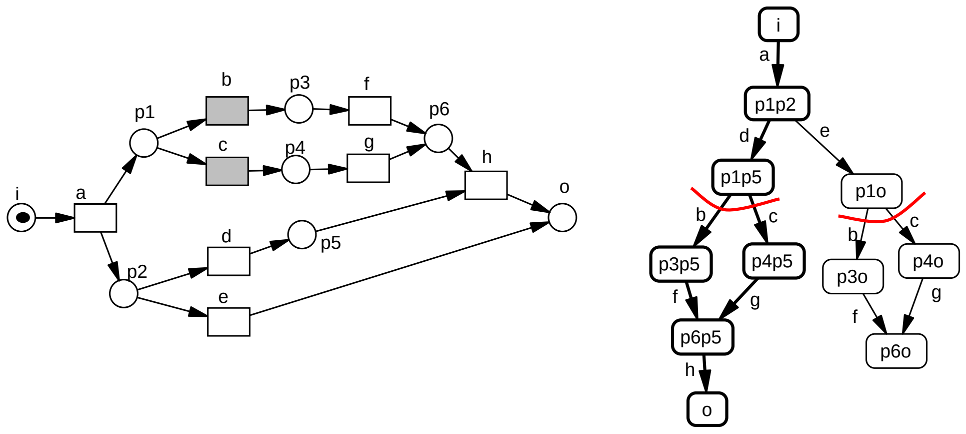


Figure 5.11: A robust WF-system which is not relaxed sound

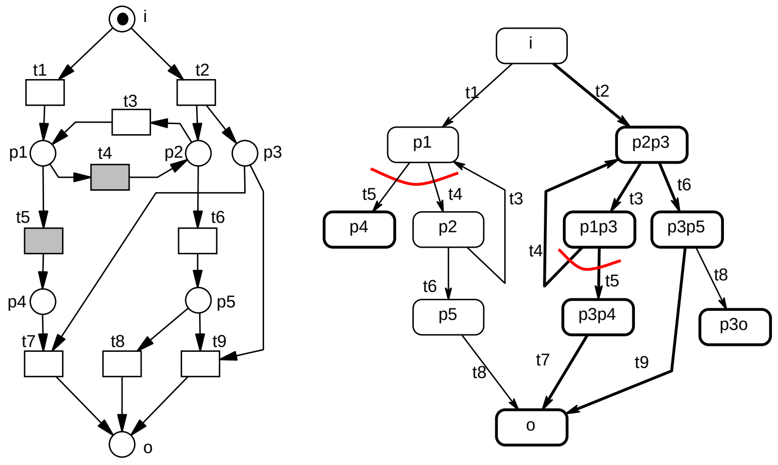


Figure 5.12: A relaxed sound and robust WF-system

by a non-controllable choice ($t4$ and $t5$). In order to terminate properly the non-controllable choice *must* (in case $t1$ was chosen before) revoke the first choice in order to terminate properly. Clearly, such a dependency between controllable and non-controllable choices denotes a modeling deficiency. To provide the modeler with feedback the robustness algorithm could be enlarged with a test constructed in analogy to Step (iv) reporting the controllable transitions not covered by the robust fragment.

In general we will assume that the robust fragment computed for a relaxed sound and robust WF-system covers all transitions. Still the particular case will not be ignored. Consequences that arise for the proposed procedure model, cf. Figure 1.1 will be discussed in Chapters 6, 7 and 8.

5.6 Related work

The related work section is subdivided into three parts. First some general references are given on controller synthesis. In the second part we investigate the taken assumptions regarding the environment and compare them with other approaches. Finally, the classification of choices into controllable and non-controllable choices is compared with another possible classification from the literature.

Theory on games - Controller synthesis In this chapter we looked at a workflow system as a reactive system, whose behavior depends on the interaction with the environment. Studying reactive systems, controller synthesis problems arise naturally. In the literature the reactive system is often called a *plant*. It is viewed as an existing program which specifies the ways in which the system can react to its inputs. Given a specification, e.g. a temporal logic formular, the goal is now to come with a strategy to interact with the environment (in a way that is allowed by the plant), such that the behavior satisfies the specification. In other words, the strategy acts as a controller for the plant, restricting its behavior so that the specification is met.

For reactive systems the synthesis problem has been posed as early as 1957 in the context of digital circuits [Chu63]. The problem was solved by [BL69]. Here an algorithm was presented which decides the realizability of a given specification and in this case synthesizes a circuit (or finite-state reactive program) from the specification.

A lot of progress has been made since then. This is evidenced by a wealth of literature on controller synthesis problems for reactive systems, such as [PR89, ALW89, McN93, AMP94, Tho95, ES98]. Beside finding (efficient) algorithms for the computation of a winning strategy on a finite game graph ([PR89, ALW89, McN93], the scope of the research broadened towards other directions. Examples are the shift of results to games played on infinite graphs [AMP94, ES98], the consideration of reactive environments (which are able to disable some of their responses) [KMTV00], and the synthesis of distributed controllers [PR90] and [MT01].

For a tutorial review on common concepts we refer to [NYY92] and [Tho02].

Assumptions regarding the environment Most approaches in modeling workflow processes assume reasonable behavior of the environment. Hence, they do not provide the modeler with a means to check whether their processes react robustly to any possible request from the environment. These approaches disregard malicious requests trying to misuse the services.

An issue that has been approached by several researchers concerns co-operation between companies. The corresponding keywords are cross- and inter-organizational workflow [Aal00a, DDGJ01, Mar01, EW01a, GA01]. In this setting the environment is represented by the co-operating partners. An important question tackled in most of these approaches is whether two or more processes (may) interact in a sound manner. In contrast to our starting point, they originate from the assumptions that the co-operating partners share a common interest, and at least parts of their processes are revealed to the public. [Aal00a, DDGJ01] and [GA01] consider different views of the process descriptions of the parties involved. They distinguished between a public view [Aal00a] (external level [GA01]) and a private view (conceptual level [GA01]). The public view contains a generalized process description, with communication activities, as well as activities that might be of value for the other parties. The private view contains a refined process description which is used for the intra-enterprise communication. Specific correctness measures are provided for the different views. They guarantee consistent cooperation between the individual processes as well as sound intra-enterprise processing.

The approaches are based on co-operation and the (partial) disclosure of processes. These assumptions are used for the validation of the interoperability where the external views are assembled and the compound specification is checked for some desired properties.

The approach proposed here is based on more general assumptions. The knowledge about the behavior of the environment is assumed to be only partial. At best, possible actions of the environment are known but not their order. Furthermore, the system and the environment are not co-operating partners revealing to each other their processes, but separate players or even opponents. Starting with these assumptions, a correctness criterion was sought indicating that the process on its own is robust to (all) possible interaction from outside.

Approaches based on a similar setting are [Mar01, AHT02] and [EW01a]. In [Mar01], the focus is on the collaboration of different web services in order to generate a new compound service. The question is whether it can be decided locally that the interface of one single service fits to possible interfaces of other web services. A web service is modeled as a Petri net-module consisting of a WF-net and a set of interface places. The environment of a web service is modeled as a WF-net too. Both nets can be connected via the interface places. In [Mar01] a criterion is defined to state the *usability* of such a Petri net-module. Generally speaking, a module is usable if there is at least one environment that forms a sound closed system together with the module. The approach does not assume any possible environment but checks whether there is one, such that the service can be executed properly.

In [AHT02] the inter-operability of different components is investigated aiming at the identification of certain rules for the construction of component based architectures. Components are considered as independent parts of an architecture, each with their own thread of control and collaborating (by message exchange) to form a working system. Components are described with the help of *component-nets* (C-nets), labeled P/T-nets having the same structure as WF-nets.

The inter-operability of two components is organized via an interface, a set of places that is connected to transitions in both components. The construction rules propose to combine components following a *client server* approach. This implies that in the relation-ship between connected components there is always one component which has the role of *control component* and one component that has the role of *server component.* Applying further rules, these elementary client-server compositions (which are again C-nets) are combined to form the final architecture, which corresponds to a tree of connected components.

The approach poses considerable restrictions to the structure of components and their combination. This is justified as applying the proposed rules consistency[8] of the derived architecture can be concluded by construction.

Applying this approach to model the interaction of a system and its environment, clearly the environment would be considered as a server providing some required information, or interacting via external events. Consequently, the behavior of the

[8]The visible behavior of the compound architecture corresponds to the behavior of its specifying root component.

environment must be described following the proposed limitations. Further investigations are necessary in order to prove whether this is possible matching realistic scenarios.

In [EW01a, Esh02] a different modeling technique is used to model workflow processes, namely *activity diagrams* from the UML. The authors argue that their semantics provide a more adequate way to model reactive behavior. They tailor a formal semantics for activity diagrams faithfully fitting the requirements of the application domain. However, they have so far not provided a specific way of determining robustness against interactions from outside.

A further classification of choices In this chapter we classified choices w.r.t. to the controllability of the *outcome*. Non-controllable choices depict choices whose outcome depends on interaction with the environment, whereas the outcome of controllable choices can be decided locally. An orthogonal criterion to distinguish choices, is the *moment* of choice. The moment of choice coincides with the moment one of the alternative transitions is executed. The moment of choice depends on where the *initiative* to execute the contained transitions lies.

Event transitions depict behavior of the environment, hence the *initiative* of a choice consisting of *event*-transitions lies with the environment. Such a choice was called implicit choice (also referred to as the deferred choice pattern) [Aal98, AHKB03]. Here the moment of choice is deferred until the external event occurs. An example is the choice within the library process of Figure 5.2.

In contrast, an explicit choice [Aal98, AHKB03], is made the moment all previous tasks are completed. The initiative for an explicit choice lies with the workflow control. All choices that consist of transitions of type *decision, task*[9], and/or *routing* are explicit choices. They are made the moment all previous tasks are completed.

Figure 5.13 gives examples for explicit choices; they have been highlighted by showing corresponding transitions in bold. The process parts of Figure 5.13 (a) and (c) depict clippings of the complaints processing from Figure 5.1. Figure 5.13 (b) depicts a clipping of the process "Handling of incoming order" from Figure 8.9(b).

The choice in Figure 5.13 (a) is a choice between transitions of type *decision.* The transitions `processing_OK` and `processing_NOK` model the possible outcomes of the evaluation of task `check_processing`. The upper notation in

[9]Note that a task is initiated by the workflow control but executed by some external actor. This distinction will be addressed in the last step of the proposed procedure ("Control Refinement"), cf. Figure 1.1

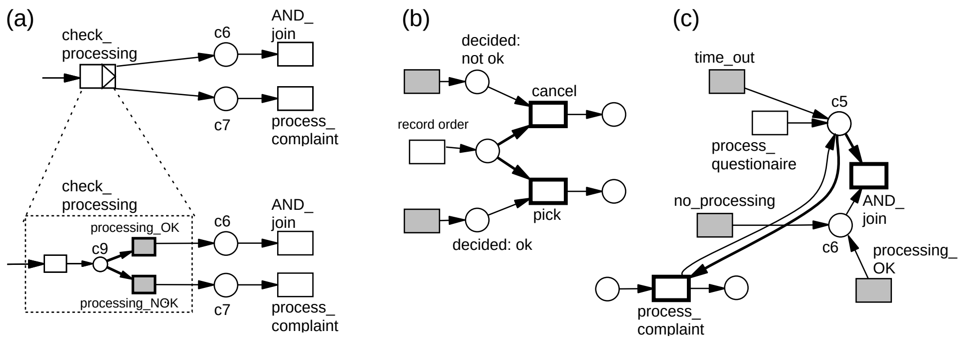


Figure 5.13: Examples of explicit choices

Figure 5.13 (a) is a shortcut for explicit choices. It was introduced in [Aal98]. It can be used only if the moment of choice depends on just one previous task. The choice in Figure 5.13 (b) depicts a choice between transitions of type task. After the completion of task `record_order` and depending on the result of some evaluation (`decided: not ok` or `decided: ok`) either task `pick` or task `cancel` is initiated.

The last choice, depicted in Figure 5.13 (c), models the choice between transitions of type *routing* and *task*. Here the case is either routed via an `AND-join` or the task `process_complaint` is initiated.

The possible influences of workflow control and environment on different choices are summarized in Table 5.1.

	Initiative	**Outcome**
Choice of *event* transitions (free-choice)	environment	environment
Choice of *decision* transitions (free-choice)	workflow control	environment
Choice of *task* and/or *routing transitions*	workflow control	workflow control

Table 5.1: Choice classification

We will come back to this classification in Chapter 8. Transitions with the initiative on the side of the workflow control have to be considered within the “Control refinement”, the last step of the proposed procedure model (cf. Figure 1.1).

Chapter 6

Generating sound process specifications

The modeler has so far been guided towards a relaxed sound and robust process specification. Relaxed soundness indicates the existence of enough sound firing sequences. Robustness indicates the existence of a strategy which guarantees sound execution independently from the moves of the environment. Still, the modeled process needs not be sound. There may be firing sequences that do not terminate properly.

Within this chapter, several ways are proposed to transform a relaxed sound and robust specification into a sound specification. The possible solutions are compared with respect to their suitability in the context of workflow modeling.

In order to transform a relaxed sound process description into a sound process description, it is necessary to restrict the set of all possible firing sequences to only sound ones. The restricted set of firing sequences must not only be sound but must also belong to a winning strategy that can be enforced against possible interactions from the environment. Assume there is a complete winning strategy that covers all desired sound firing sequences. Then the goal is reached as soon as we find a Petri net with a reachability graph isomorphic to the robust fragment.

Applying methods from the area of Petri net synthesis [GRX02a, BDC02, CKLY98, DR96, NRT92, ER90], there are two solutions to the problem.

The first applies the methods proposed in [NRT92, CKLY98] and generates a Petri net on the basis of the robust fragment. The result is a WF-net with a behavior isomorphic to the fragment. The disadvantage of this approach is the differences between the derived and the primary WF-net. They only coincide in their transition inscriptions; place labels as well as the layout is lost.

The second approach is based on the application of results of [GRX02a, GRX02b]. Here, approaches from Petri net synthesis were adapted for Petri net controller synthesis. Rather than constructing a new net, it is proposed to compute changes of the primary net so as to restrict its behavior to the one described by the robust fragment. This solution increases the possibility of recognizing the primary process description within the resulting one.

This chapter is organized as follows. First, main results from Petri net synthesis are reviewed. In Section 6.2 these results are applied to workflow modeling. They are used to synthesize sound WF-nets with isomorphic or bisimilar behavior w.r.t. the computed fragment (first solution).

In Section 6.3 the synthesis approach is refined for controller synthesis. Its application to workflow modeling is described in Section 6.4. Here, the focus is on the change of the primary WF-net, making it a sound WF-net (second solution). At the end of the chapter the two approaches are compared (Section 6.5) and related work is discussed (Section 6.6). To support a better understanding of the revised theory, we again introduce a running example. Applying these methods to workflow modeling, we will also refer to an example introduced in Chapter 5.

6.1 Petri net synthesis

The *synthesis problem* for Petri nets is tackled by deciding whether a given graph is isomorphic to the reachability graph of some Petri net.

The synthesis problem was first addressed in [ER90]. Subsequently, in [NRT92] it was shown that an elementary net system can be synthesized on the basis of regions from a (sequential) transition system satisfying some separation conditions, namely from an elementary transition system. Thereafter, the synthesis problem has been solved for other classes of Petri nets [Muk92, BDC02, BBD95, YMLA96, BMPV96]. The solutions all use regions.

Regions may be interpreted as *atomic* nets, i.e. nets consisting of a single place together with its input and output transitions. A Petri net can be composed using atomic nets and joining them at common transitions.

Every transition system can be broken down into a finite number of subsets of regions. Fitting together the atomic nets which correspond to the regions, however, does not necessarily result in Petri nets with isomorphic behavior to the primary transition system. The challenge was to decide constructively the existence of a Petri net with a reachability graph isomorphic to a given transition system. Here, the main ideas for Petri net synthesis on the basis of regions are introduced.

6.1.1 Preliminaries

We will start by recalling transition systems and providing various concepts to compare them.

Transition system

A Transition System (TS) was defined (Def. 2.22) as a quadruple $TS = (V, L, E, v_{in})$, where V is a nonempty set of states, L is a set of events (labels), $E \subseteq V \times L \times V$ is a transition relation, and v_{in} is an initial state. A TS is finite if V and L are finite.

In the following we will only consider finite transition systems. Furthermore it is assumed that every transition system $TS = (V, L, E, v_{in})$ satisfies the following axioms:

(A1) No self-loops: $\forall (v, l, v') \in E : v \neq v'$;

(A2) Every event has an occurrence: $\forall l \in L : \exists (v, l, v') \in E$;

(A3) Every state is reachable from the initial state: $\forall v \in V : v_{in} \xrightarrow{*} v$.

An example of a transition system is shown in Figure 6.1.

Some concepts are introduced for the comparison of the behavior described by transition systems.

Definition 6.1 (Split-morphism).
Let $TS_1 = (V_1, L_1, E_1, v_{in_1})$ and $TS_2 = (V_2, L_2, E_2, v_{in_2})$ be two transition systems. A split-morphism h from TS_1 to TS_2 is a pair (h_V, h_L) of total mappings, h_V being bijective and h_L being surjective,

$$h_V : V_1 \longrightarrow V_2$$
$$h_L : L_1 \longrightarrow L_2$$

which satisfies: $((v, l, v') \in E_1 \Leftrightarrow (h_V(v), h_L(l), h_V(v')) \in E_2$.

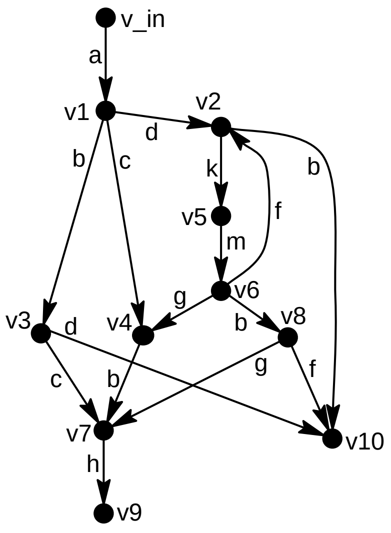


Figure 6.1: A transition system

Definition 6.2 (Isomorphism).
A split-morphism $h = (h_V, h_L)$ from TS_1 to TS_2 is an isomorphism *if h_L is bijective.*

Two transition systems are said to be isomorphic if there is an isomorphism between them. The concept of split-morphism will be used when an event is represented by different instances in a transition system. Later on, when deriving Petri nets, this splitting will result in different transitions with the same label.

Another concept of equivalence between transition systems is the concept of bisimulation [GW96, Mil80].

Definition 6.3 (Bisimulation).
Let $TS_1 = (V_1, L, E_1, v_{in_1})$ and $TS_2 = (V_2, L, E_2, v_{in_2})$ be two transition systems with the same set of events. TS_1 and TS_2 show bisimulation if there exists a binary relation R between V_1 and V_2, $R \subseteq V_1 \times V_2$ such that

- *The initial states of TS_1 and TS_2 are related by R: $(v_{in_1}, v_{in_2}) \in R$*
- *If $R(v_1, v_2)$ and $(v_1, l, v_1') \in E_1$, then there is a state v_2' such that $(v_2, l, v_2') \in E_2$.*
- *If $R(v_1, v_2)$ and $(v_2, l, v_2') \in E_2$, then there is a state v_1' such that $(v_1, l, v_1') \in E_1$.*

Two transition systems are said to be *bisimilar* if they can simulate each other, i.e. there is a bisimulation between them.

Regions

The core notion within Petri net synthesis is the concept of a region. Various versions are used to differentiate between several classes of transition systems. A comprehensive survey describing the gradual development of the theory of regions can be found in [BD98].

The basic notion of a region is the following:

Definition 6.4 ((elementary) region).
Let $TS = (V, L, E, v_{in})$ be a transition system. Then $r \subseteq V$ is a region of TS iff the following two conditions are satisfied:
event l always exits r:

$$(v, l, v') \in E \wedge v \in r \wedge v' \notin r \Rightarrow \forall (v_1, l, v_1') \in E : v_1 \in r \wedge v_1' \notin r$$

event l always enters r:

$$(v, l, v') \in E \wedge v \notin r \wedge v \in r \Rightarrow \forall (v_1, l, v_1') \in E : v_1 \notin r \wedge v_1' \in r$$

A region is a subset of states. The fundamental property of a region is that *all* transitions labeled with the same event l have the same ”entry/exit” relationship. The event may either always *enter* the region, or always *exit* the region, or never *cross* the regions boundaries.

Each transition system TS has two trivial regions: the set of all states, V, and the empty set. Further on, we will consider only nontrivial regions. The set of nontrivial regions of TS will be denoted by R_{TS}. For each state $v \in V$, we define the set of nontrivial regions containing v, denoted by R_v. A region r' is said to be a subregion of r iff $r' \subset r$. A region r is *minimum* if there is no other region r' which is a subregion of r.

Let us consider again the transition system shown in Figure 6.1. The set of states $r_2 = \{v1, v2, v4, v6\}$ is a region, since all transitions labeled with a and m enter r_2, and all transitions labeled with b and k exit r_2. Other regions are $r_1 = \{v\_in\}$, $r_3 = \{v1, v3\}$, $r_4 = \{v2, v10\}$, $r_5 = \{v5\}$, $r_6 = \{v6, v8\}$, $r_7 = \{v4, v7\}$, $r_8 = \{v3, v7, v8, v10\}$ and $r_9 = \{v9\}$. All of these regions are minimum, but are not necessarily mutually exclusive. Correspondingly, the sets R_{vi} for $i = in, 1..10$ are as follows: $R_{v\_in} = \{r_1\}, R_{v1} = \{r_2, r_3\}, R_{v2} = \{r_2, r_4\}, R_{v3} = \{r_3, r_8\}, R_{v4} = \{r_2, r_7\}, R_{v5} = \{r_5\}, R_{v6} = \{r_2, r_6\}, R_{v7} = \{r_7, r_8\}, R_{v8} = \{r_6, r_8\}, R_{v9} = \{r_9\}, R_{v10} = \{r_4, r_8\}$, We will furthermore define *pre-regions* and *post-regions* of an event. A region $r \in R_{TS}$ is a pre-region of event l, if there is a transition labeled with l which exits $r : \exists(v,l,v') \in E : v \in r \wedge v' \notin r$. The set of pre-regions of an event l is denoted by $\circ l$. A region $r \in R_{TS}$ is a post-region of event l, if there is a transition labeled with l which enters $r : \exists(v,l,v') \in E : v \notin r \wedge v' \in r$. The set of post-regions of an an event l is denoted by $l\circ$.

Pre-regions and post-regions for the transition system in Figure 6.1 are as follows:

pre-regions: $\circ a = \{r_1\}, \circ b = \{r_2\}, \circ c = \{r_3\}, \circ d = \{r_3\}, \circ f = \{r_6\},$
$\circ\, g = \{r_6\}, \circ h = \{r_7, r_8\}, \circ k = \{r_2, r_4\}$ and $\circ m = \{r_5\}$

post-regions: $a\circ = \{r_2, r_3\}, b\circ = \{r_8\}, c\circ = \{r_7\}, d\circ = \{r_4\}, f\circ = \{r_4\},$
$g\circ = \{r_7\}, h\circ = \{r_9\}, k\circ = \{r_5\}$ and $m\circ = \{r_2, r_6\}$

With the notion of a region we are now able to define an *elementary transition system.*

Definition 6.5 (Elementary transition system).
A transition system $TS = (V, L, E, v_{in})$ is elementary (ETS) [NRT92] if in addition to (A1)-(A3), it satisfies the following two axioms about regions:

(A4) State separation axiom (SSA): $\forall v, v' \in V : R_v = R'_v \Rightarrow v = v'$

(A5) Event Separation axiom (ESA): $\forall v \in V \forall l \in L : \circ l \subseteq R_v \Rightarrow v \xrightarrow{l}$

The state separation axiom (A4) implies that two different states must belong to different sets of regions. The event separation axiom (A5) implies that if state v is included in all pre-regions of an event such as l, then l must be enabled in v. Conversely, if an event l is not enabled in a state v then there is a pre-region of event l which does not contain state v: $\forall v \in V \forall l \in L : v \overset{l}{\not\longrightarrow} \Rightarrow v \notin r$ for some pre-region of l: $r \in \circ l$.

The transition system in Figure 6.1 is elementary, since all axioms (A1)-(A5) are satisfied. For example, state $v1$ is separated from any other state (A4). This state is included in regions r_2 and r_3 and there is no other state which is covered by the same set of regions. To illustrate (A5), let us consider event c with $\circ c = \{r_3\}$, $r_3 = \{v1, v3\}$, $R_{v1} = \{r_2, r_3\}$, and $R_{v3} = \{r_3, r_8\}$. The two states, $v1$ and $v3$ satisfy condition $\circ c \subseteq R_{v1}$ and $\circ c \subseteq R_{v3}$ respectively. (A5) holds as both states $v1$ and $v3$ have an exit arc labeled by event c.

Another class of transition systems is based on excitation regions [CKLY98]. These are regions that are related to transitions while normal regions in a transition system are related to places in the corresponding Petri net. An excitation region (ER) for event l is a maximum set of states in which transition l is enabled:

Definition 6.6 (Excitation region (ER)).
A set of states V' is called an excitation region for event l, denoted $ER(l)$, if it is a maximum set of states such that, for every state $v \in V'$ there is a transition $v \xrightarrow{l}$.

Referring again to the transition system in Figure 6.1, examples for excitation regions are $ER(a) = \{v\_in\}, ER(b) = \{v1, v2, v6\}$ and $ER(c) = \{v1, v3\}$.

Based on this region-notation, the class of excitation-closed transition systems is defined. Therefore, the axioms (A4) and (A5) are replaced by axioms (A4') and (A5').

Definition 6.7 (Excitation-closed TS (ECTS)).
A transition system $TS = (V, L, E, v_{in})$ is called excitation-closed [CKLY98] if in addition to (A1)-(A3), it satisfies the following two axioms about excitation regions:

(A4') Excitation closure: For each event $l : \bigcap_{r \in \circ l} r = ER(l)$

(A5') Event effectiveness: For each event $l : \circ l \neq \emptyset$

The TS from Figure 6.1 is excitation-closed. If we consider e.g. event h: Its pre-regions are non-empty: $\circ h = \{r_7, r_8\}$, therefore, axiom (A5') is satisfied. The excitation region $ER(h) = \{v7\}$ coincides with $r_7 \cap r_8$ ($r_7 = \{v4, v7\}, r_8 = \{v3, v7, v8, v10\}$), therefore, axiom (A4') is also satisfied.

The relation between ETS and ECTS is very close. The following theorem, cf. [CKLY98], establishes a connection between them.

Theorem 6.8 (Relation ETS and ECTS).
1. If a TS is elementary, then it is excitation-closed.
2. Let $TS = (V, L, E, v_{in})$ be an ECTS. Then, there is an elementary transition system TS', and TS and TS' are bisimilar.

Beside elementary and excitation closed transition system, we will also refer to *general* transition systems. For their definition we introduce the notion of a *general region*. General regions, which have been introduced in various forms [Muk92, DS93, BMPV96] and [KCK$^+$96], are multisets where regions are sets.

Definition 6.9 (General region).
A general region of a transitions system (V, L, E, v_{in}) is a multiset $r : V \longrightarrow I\!N$ iff: $\forall(v1, l, v1'), (v2, l, v2') \in E : r(v1) - r(v1') = r(v2) - r(v2')$.

In other words, a multiset r is a region if and only if every transition labeled with event l changes the "rank of membership" uniformly; $\forall l \in L, \forall(v, l, v') \in E : r(v) - r(v') = k$, where k is a constant.

If $r(v) - r(v') = k \leq 0$, it is said that transition (v, l, v') increases r by k. If $r(v) - r(v') = k \geq 0$, it is said that transition (v, l, v') decreases r by k.

A region r is said to be a *k-preregion* of event l if there is a transition labeled with l which decreases r by k. A region is a *k-postregion* of event l if there is a transition labeled with l which increases r by k.

Based on this region-notation the class of general transitions systems is defined. Here, axioms (A4) and (A5) are replaced by axioms (A4") and (A5").

Definition 6.10 (General TS).
A transition system $TS = (V, L, E, v_{in})$ is called a general TS if in addition to (A1)-(A3), it satisfies the following two axioms about general regions:

(A4") State separation axiom (SSA): $\forall v, v' \in V : R_v = R'_v \Rightarrow v = v'$

(A5") Event separation axiom (ESA): $v \not\xrightarrow{l} \Rightarrow r(v) < k$ for some k-preregion of l.

General regions are an extension of the concept of elementary regions; every elementary region is a general region which changes the rank of membership by $k = 1$. Therefore, every elementary TS is a general TS. Setting $k = 1$, the separation axioms, characterizing a general TS, corresponds to the separation axioms of elementary TSs.

The following sub-section recapitulates how different types of Petri nets are synthesized on the basis of corresponding transitions systems.

6.1.2 Synthesizing safe Petri nets

Let $TS = (V, L, E, v_{in})$ be a transition system. If TS is elementary, i.e. it satisfies the state and the event separation axioms (A4) and (A5), then TS is isomorphic with the reachability graph of a safe and pure Petri net. In [NRT92] the following theorem was proved correct.

Theorem 6.11 (Elementary TS/Petri net).
The reachability graph of a safe *and* pure *Petri net is always an elementary transition system and vice versa, i.e. if a transition system is elementary, then a Petri net with a reachability graph isomorphic to the transition system can be constructed.*

The key idea behind the synthesis of a corresponding Petri net is the interpretation of regions as atomic nets with a single place, filled by entering transitions and emptied by exiting transitions. The atomic nets are composed by joining common labeled transitions, forming the desired Petri net.

Incorporating all regions of the transition system, the synthesized net has the peculiar property that it is maximum with respect to the number of places among all the net systems whose reachability graph is isomorphic to the primary transition system. In [DR96] an *admissible* set of regions is said to be sufficient to derive a Petri net with isomorphic behavior to the transition system. An *admissible* set of regions is a subset of regions large enough to satisfy both separation axioms (SSA and ESA). In [Ber93] it was shown that the set of minimum regions is an admissible set of regions. The net constructed from all minimum regions is unique and called a minimum saturated net.

Synthesis algorithm

The algorithm for synthesizing a minimum saturated Petri net, as proposed in [NRT92], works as follows. For every event $l \in L$ a transition labeled with l is generated in the Petri net. For each minimum region $r_i \in R_{TS}$, a place is generated in the Petri net. The flow relation of the Petri net is derived from the pre- and post-region of the events: $F_{TS} = \{(r,l)|r \in R_{TS} \wedge l \in L \wedge r \in \circ l\} \cup \{(l,r)|r \in R_{TS} \wedge l \in L \wedge r \in l\circ\}$.

Place r_i contains a token in the initial marking iff the corresponding region r_i contains the initial state v_{in} of the ETS.

The TS shown in Figure 6.1 is elementary. The synthesized Petri net computed on the subset of minimum regions is shown in Figure 6.2.

The algorithm does not provide a solution if the given transition system is not elementary. In this case, a safe Petri net with isomorphic behavior cannot be synthesized.

In [CKLY98] the scope of the algorithm was broadened towards excitation closed TS. This implies an optimization of the algorithm and, even more importantly, makes it possible to cover the full class of TSs by means of transition splitting.

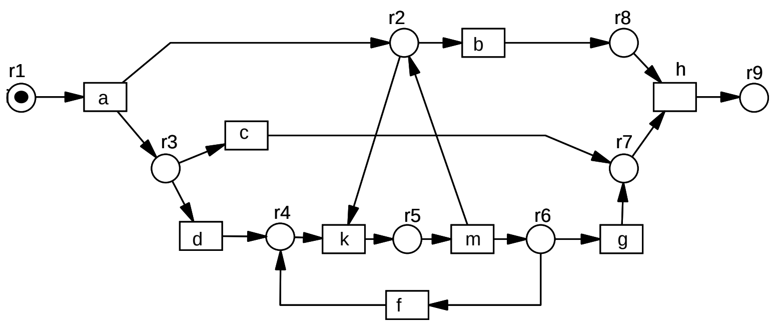


Figure 6.2: A synthesized Petri net

The use of ETSs produced a Petri net with a reachability graph *isomorphic* to the TS. Changing the degree of correspondences, the extended method produces a Petri nets whose reachability graph is bisimilar to the primary TS.

The input of the new algorithm is a TS. The output is a safe and pure Petri net whose reachability graph is bisimilar to the TS. In a first step, the TS is transformed into a split-morphic ECTS. This is done by splitting labels of transitions. On the basis of the derived ECTS, a minimum saturated Petri net is synthesized.

The new algorithm was described in detail in [CKLY98] and was implemented within the tool *Petrify* [CKK$^+$97]. An example illustrating the approach is given in the Appendix, see Section 9.5.

Another possibility to cope with non-elementary TS is the use of *general* regions.

6.1.3 Synthesizing general Petri nets

The approach proposed in [NRT92] has been extended to *general regions* in order to cope with the synthesis of general Petri nets[1], e.g. [Muk92, BDC02, KCK$^+$96].

It is apparent that every place p of net system $S = (PN, M_i)$ determines an associated general region of the reachability graph $RG = (V, E)$, such that $r(M) = M(r)$ for every reachable marking $M \in V$. Conversely, every general region r of a transition system (V, L, E, v_{in}) determines an atomic net $N' = (\{r\}, L, F', M_0')$ where $F'(r, l) = k$ iff r is a *k-preregion* of l, $F'(l, r) = k$ iff r is a *k-postregion* of l and $M_0'(r) = r(M_0)$.

Here again, the net assembled in this way is a Petri net with a reachability graph

[1]Place/Transition nets with arc weights: $F : (P \times T) \cup (T \times P) \longrightarrow I\!N$

isomorphic to the primary transition system if and only if the transition system satisfies the two separation axioms (A4") and (A5")

Theorem 6.12 (General TS $\rightarrow$ Petri net).
A transition system $TS = (V, L, E, v_{in})$ is isomorphic to the reachability graph of some marked general Petri net if and only if it satisfies the state and event separation axioms (A4") and (A5").

For a proof of this theorem, the reader is referred to [Muk92, BDC02].

The set of generalized regions has particularly good algebraic properties which made it possible to develop polynomial algorithms solving the synthesis problem for bounded place/transition nets without loops [BBD95]. The algorithmic idea is based on the relation between general regions and row-vectors in the incidence matrix of the corresponding Petri net. The algorithmic solution was extended later on to general Petri nets [BD96] and was implemented within the tool *Synet* [Cai97].

Having introduced existing results from Petri net synthesis, the actual benefit is their application for the modeling of workflow. In the next section we will exploit the described methods, generating a sound process description on the basis of a relaxed sound and robust process description.

6.2 Applying Petri net synthesis to workflow modeling

We are now in a position to generate a sound WF-system on the basis of a relaxed sound and robust WF-system. The sound WF-system only supports parts of the behavior of the relaxed sound WF-system, namely those firing sequences that are sound and belong to the winning strategy that can be enforced against possible interactions from the environment.

Constructing a sound WF-system (PN', i) on the basis of the relaxed sound and robust WF-system (PN, i) comes down to synthesizing a Petri net on the basis of a robust fragment $SG = (V_{SG}, E_{SG})$. The only further assumption is that $PN = (P, T, F)$ is pure (i.e. no self loops). This is no restriction as any non-pure Petri net can be transformed into a pure net without altering relevant behavior [Lau02].

Let $SG = (V_{SG}, E_{SG})$ be a robust fragment of the reachability graph $RG_S = (V_{RG}, E_{RG})$ of the WF-system $S = (PN, i)$: $SG \subseteq RG_S$. SG is a transition system (V, L, E, v_{in}) with: $V = V_{SG} \subseteq R_{PN}(M_i)$ as set of states, and $E = E_{SG} = E_{RG} \cap (V_{SG} \times T \times V_{SG})$ as set of labeled edges, and $v_{in} = i$ is the initial state. The set of labels L corresponds to a subset of the transitions of PN: $L = \{t|(M, t, M') \in E'\}$. Note that L includes at least all non-controllable transitions $L \cap T_{NC} = T_{NC}$, follows as SG is complete.

The fragment satisfies the standard axioms of a TS (A1)-(A3).

(A1) **no self-loops:** this follows as the WF-net was assumed to be pure (cf. Section 3.4.1).

(A2) **every event has an occurrence:** follows from the definition of SG.

(A3) **every state is reachable from the initial state:** follows as the fragment is robust (cf. Requirement 5.2 from Def. 2).

Two cases are differentiated. If the computed fragment SG is elementary, the basic algorithm [NRT92] can be applied synthesizing a Petri net with isomorphic behavior. If the computed fragment is not elementary, the extended algorithm from [CKLY98] is applied. Note, if the fragment is not excitation closed, transition splitting is carried out before the actual synthesis. This procedure leads to a sound WF-system. We will prove the following theorem.

Theorem 6.13.
Let PN be a pure WF-net with input place i. Let RG_S be the reachability graph of the system $S = (PN, i)$ and $SG = (V_{SG}, E_{SG})$ be a robust fragment, $SG \subseteq RG_S$. Let $PN' = (P', T', F')$ be the Petri net synthesized based on SG. Then PN' is a WF-net and $S' = (PN', i)$ is sound.

Proof a) PN' is a WF-net

One source- and one sink place: The existence of one source and one sink place can be deduced from the construction. The fragment does contain a state i and a state o which have either only outgoing arcs (i) or only incoming arcs (o). Therefore, there are two minimum regions $r_i = \{i\}$ and $r_o = \{o\}$ reflecting this “entry/exit” relationship. The labels of the state transitions exiting state i (or entering o) must not occur a second time as this would mean that place i (o) would be marked again. Minimum regions are mapped on places within the synthesized Petri net. As furthermore the flow relation is derived from the pre- and post-region of the events, PN' has exactly one source place and one sink place.

Strongly connected: The system $S' = (PN', i)$ does not contain any dead transitions, i.e. for every transition $t \in T'$ there is a marking M' reachable from i which enables t. This follows as for every transition of PN' there is a corresponding label in the robust fragment.

The reachability graph $RG_{S'}$ is isomorphic or bisimilar to the robust fragment SG which contains sound firing sequences only, i.e. every marking reachable from the initial marking i eventually leads to o ($i \xrightarrow{*} M \xrightarrow{*} o$). We short-circuit the derived Petri net by adding a transition $t^* \notin T'$, such that $\overline{PN'} = (P', T' \cup \{t^*\}, F' \cup \{(o, t^*), (t^*, i)\})$. Firing this transition the system is reset to its initial state.

For the short-circuited system $S = (\overline{PN'}, i)$ it now holds that from every reachable marking M a marking M' is reachable ($M \xrightarrow{*} M'$) which enables t, i.e. the short-circuited net is live. The system $S = (PN', i)$ is safe ([NRT92, CKLY98]). In its final marking place o contains one token and all the other places are empty ($M(o) = 1$ and for all $p \in P \setminus \{o\} : M(p) = 0$). Short-circuiting the net as described above, the system remains bounded. System $S = (\overline{PN'}, i)$ is live and bounded. With [DE95] Theorem 2.25 it can be concluded that S is strongly connected.

No arc weights: follows as the synthesized Petri net is safe.

Proof b) (PN', i) is sound Soundness of (PN', i) follows as the short-circuited system $(\overline{PN'}, i)$ is live and bounded, cf. Theorem 2.48.

Summarizing the above points, it follows that (PN', i) is a sound WF-system.

□

Note There may be transitions in the primary WF-net PN which do not occur in the resulting Petri net PN'. Starting from a transition system, the synthesis algorithm generates a Petri net transition (labeled with l) for every event $l \in L$. If the set of labels of the fragment equals the set of transitions of the primary WF-system ($L = T$) the set of generated transitions T' equals T. If some controllable transitions were not covered by the robust fragment ($L \subset T$) the resulting sound WF-system PN' will have less transitions than the primary WF-system. There are examples were $L \subset T$. The reader is referred to the appendix (cf. Section 9.5) for an Illustration.

Using the generated sound WF-system to support the execution of the actual process at run-time a reliable execution can be guaranteed.

Example: Consider again the revised example in Figure 5.6 (cf. Page 96). The WF-net "Planning trip" was robust, and relaxed sound but was not sound. The robust fragment $SG = (V_{SG}, E_{SG})$ was shown in Figure 5.8 on Page 97. The fragment contains a t-labeled state transition for every transition $t \in T$: $L = T$.

The fragment is an ETS. The WF-net synthesized on the basis of the fragment is shown in Figure 6.3. The WF-net was generated using the tool *Petrify* [CKK$^+$97].

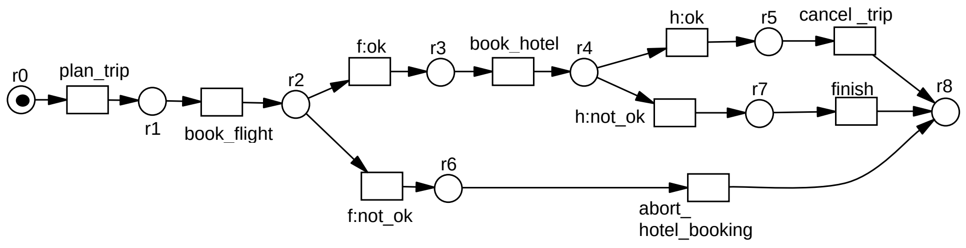


Figure 6.3: The synthesized Petri net "Planning Trip"

The resulting WF-system is sound covering parts of the behavior of the primary process description. This is a necessary prerequisite to use the derived WF-net as workflow specification.

Still, the derived WF-net is possibly inadequate for communication purposes. The behavior of the primary WF-system became restricted. The derived behavior should be confirmed by a domain expert before the workflow-specification is put to use. Therefore, the final process description should be discussed again between domain experts. This may cause difficulties as the derived WF-net probably bears little resemblance with the primary WF-net. This is due to a number of points:

Loss of place names: Within region computation, the labels of the old places are replaced by new arbitrary region names.

Loss of places Implicit places[2] which have been part of the primary WF-net are omitted in the synthesized net.

Augmentation of elements Synthesized nets are always safe. Synthesized sound WF-net may become quite large. This is because for a k-bounded place in the primary WF-net, k places would be synthesized in the resulting WF-net.

[2]Implicit or redundant places are places that do not restrict the firing of transitions

Loss of layout Starting from the reachability graph of the WF-net, further important information such as layout and position of net-elements is not incorporated in the drawing of the new WF-net.

Change of order In the resulting WF-net even the ordering of transitions may have changed. Transitions that were primarily ordered with respect to their appropriate organizational unit are reordered with respect to their actual occurrence. These rearrangements can also be observed comparing the primary WF-net “Planning trip” (cf. Figure 5.6) with the synthesized WF-net shown in Figure 6.3.

The only guaranteed correspondence between primary and resulting WF-net is for the transition labels. But if transition splitting was necessary, a transition in the primary WF-net may be replaced by several transitions which are assigned with deviated labels. Furthermore, if the set of labels of the fragment does not equal the set of primarily used transitions ($L \subset T$), some transition labels will be missing in the generated WF-net.

The altered appearance of the resulting process description complicates the recognition of the primary modeled process. The domain experts have to understand a new process description in order to discuss and agree on the final behavior.

The growth of the resulting WF-net could be circumvented by the use of general regions. Then the scope of the algorithm is broadened towards the synthesis of general Petri nets. The drawback is that starting from a WF-net which has no arc-inscriptions the derived Petri net may have arc-inscriptions and may thus no longer be a WF-net. Changes due to loss of information (place names, layout, appropriate organizational unit) could still not be avoided.

To remedy this problem a second procedure is proposed. The objective is not to generate a new sound process description but to enhance the primary WF-net, such that the behavior is restricted. The latter approach has the advantage that domain experts can recognize the WF-net more easily, as only some changes have to be considered.

In order to identify these changes, we again consider methods proposed in the literature; this time regarding the synthesis of Petri nets controllers. This research field applies results from Petri net synthesis generating only these parts of a net which guarantee some constraints specified in advance.

6.3 Petri net controller synthesis

The objective of Petri net controller synthesis is to compute a set of new places for a given Petri net which supervise or control the behavior of the Petri net, avoiding the entry into forbidden states[3]. The introduced places are called controller places (e.g. [YMLA96]) or monitors (e.g. [GDS92]).

Contrary to Petri net synthesis, where a whole net is synthesized, in this application domain only some designated places, namely the controller places, have to be synthesized. Adding these places to the primary net, the behavior is restricted. As these places are not contained in the primary net there are no corresponding regions so far. 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 [GRX02b]. We will here look more closely at the approach proposed in [GRX02b, GRX02a].

The information needed for the computation of regions is derived from the state transitions transgressing the *legal behavior*. The legal behavior corresponds to the partial reachability graph from where all desired states remain reachable.

It is clear that state transitions leaving the legal behavior have to be prevented. Remember the event separation axiom (A5/A5”) stating that if an event l is not enabled in a state v then there is a pre-region of event l which does not contain state v. Therefore, the controller places p_{c_i} correspond to these pre-regions.

Let us consider one controller place p_c. If the legal behavior is an ETS it holds for region r_{p_c} that it is pre-region of event l and $r(M) = 0$. If the legal behavior is a general TS, region r_{p_c} must be a k-preregion of event l and $r(M) \leq k$.

In the following we will sketch the algorithm that was proposed [GRX02b, GRX02a] to compute the corresponding regions. The algorithm is based on the use of general regions and exploits the relation between general regions and row-vectors in the incidence matrix of the corresponding Petri net.

Let $SG \subset RG$ be a subgraph of the reachability graph, describing the legal behavior. Let Ω be the set of pairs (M, t) corresponding to state transitions that leave the subgraph. Ω is called the set of separation instances. It is assumed that for each separation instance $(M, t) \in \Omega$ one additional control place p_c is necessary in order to prevent its occurrence. It will be seen that in practice the number of new places is much smaller than the number of state transitions to be inhibited.

[3]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.

In order to influence the firing of transition t it is clear that the control place p_c must be in its preset. In order to inhibit the enabling of t in M, it must hold that

$$M(p_c) + \mathbf{C}(p_c, t) < 0. \qquad \text{[Event separation condition]}(6.1)$$

where $\mathbf{C}$ is the incidence matrix of the system (PN, i) and the entry $\mathbf{C}(p_c, t)$ corresponds to the change of the marking of the place p_c caused by the occurrence of transition t.

Relation 6.1 is the event separation condition of (M, t). Note that different event separation instances may have common solutions. As a result, the number of places needed to solve all event separation instances is generally much smaller than the number of event separation instances.

The introduction of new places need not change the legal behavior. Therefore, other equations, such as the *Marking equation lemma* (cf. Lemma 2.26), as well as the general property of T-invariants (cf. Definition 2.27) should still hold. Restricted to the controller place p_c it must hold:

$$M'(p_c) = M(p_c) + \mathbf{p_c}\cdot\vec{\sigma} \geq 0, \forall\vec{\sigma} : M \xrightarrow{\sigma} M' \quad \text{[Marking equation lemma]}(6.2)$$

$$\mathbf{p_c} \cdot Y = 0, \forall \text{ T-invariants } Y \text{ of } PN \qquad \text{[General property of T-invariants]}(6.3)$$

Solving the given set of equations 6.1, 6.2, and 6.3, a solution is derived for the row vector $\mathbf{p_c}$. Note, that the equation system has a solution only if the subgraph satisfies the two separation axioms SSA (A4”) and ESA (A5”).

A row-vector of the incidence matrix describes the in- and output relation of one place. Therefore, the solution provides all information needed to introduce the controller place p_c. The aggregation of all controller places is called a synchronization pattern: $SP = (P_c, T, F_c)$ where P_c is the set of controller places, T is the set of transitions, and $F_c : (P_c \times T) \cup (T \times P_c) \longrightarrow I\!N$ is the flow relation, $F_c(p, t) = |\mathbf{p}(t)|$ if $\mathbf{p}(t) < 0$ (0 otherwise) and $F_c(t, p) = \mathbf{p_c}(t)$, if $\mathbf{p_c}(t) > 0$ (0 otherwise).

The algorithm computing a synchronization pattern by solving the above equation system was implemented within the tool Synet [Cai97].

Integrating the synchronization pattern into the primary Petri net $PN = (P, T, F)$ a new Petri net is derived with a behavior isomorphic to the partial reachability graph denoting the desired behavior: $PN_{sync} = SP \cup PN = (P \cup P_c, T, F \cup F_c)$[4].

[4]We assume PN to be a general Petri net. This is no restriction as every Petri net without arc weights can be reformulated as a general Petri net setting all arc weights to 1.

Example: The transition system in Figure 6.4 a) denotes the reachability graph of the Petri net from Figure 6.2. Let the legal behavior be the subgraph given in Figure 6.4 b). The subgraph is an ETS.

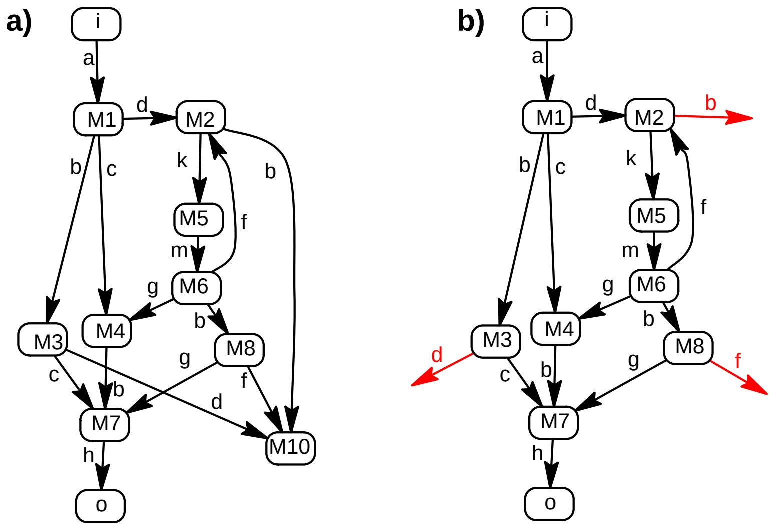


Figure 6.4: a) The reachability graph RG b) An assumed legal behavior $SG \subseteq RG$

The deduced set of separation instances Ω, is $\Omega = \{(M_2, b), (M_3, d), (M_8, f)\}$. The following three event separation conditions are derived:

1. $M_2(p_{c_1}) + \mathbf{C}(p_{c_1}, b) < 0$
2. $M_3(p_{c_3}) + \mathbf{C}(p_{c_3}, d) < 0$
3. $M_8(p_{c_2}) + \mathbf{C}(p_{c_2}, f) < 0$

According to the Marking equation lemma (cf. Equation 6.2), the following equations must hold for all p_{c_i} with $i = 1, 2, 3$:

- $M_1(p_{c_i}) = i(p_{c_i}) + \mathbf{C}(p_{c_i}, a) \geq 0$
- $M_2(p_{c_i}) = M_1(p_{c_i}) + \mathbf{C}(p_{c_i}, d) \geq 0$
- $M_3(p_{c_i}) = M_1(p_{c_i}) + \mathbf{C}(p_{c_i}, b) \geq 0$
- $M_4(p_{c_i}) = M_1(p_{c_i}) + \mathbf{C}(p_{c_i}, c) \geq 0$
- $M_5(p_{c_i}) = M_2(p_{c_i}) + \mathbf{C}(p_{ci}, k) \geq 0$
- $M_6(p_{c_i}) = M_5(p_{c_i}) + \mathbf{C}(p_{c_i}, m) \geq 0$
- $M_8(p_{c_i}) = M_6(p_{c_i}) + \mathbf{C}(p_{c_i}, b) \geq 0$
- $M_4(p_{c_i}) = M_6(p_{c_i}) + \mathbf{C}(p_{c_i}, g) \geq 0$
- $M_7(p_{c_i}) = M_4(p_{c_i}) + \mathbf{C}(p_{c_i}, b) \geq 0$
- $M_7(p_{c_i}) = M_3(p_{c_i}) + \mathbf{C}(p_{c_i}, c) \geq 0$
- $M_7(p_{c_i}) = M_8(p_{c_i}) + \mathbf{C}(p_{c_i}, g) \geq 0$
- $o(p_{c_i}) = M_7(p_{c_i}) + \mathbf{C}(p_{ci}, h) \geq 0$

According to the T-invariants (cf. 6.3) of the primary system, the following equations must hold for all p_{c_i} with $i = 1, 2, 3$:

- $\mathbf{C}(p_{c_i}, a) + \mathbf{C}(p_{c_i}, b) + \mathbf{C}(p_{c_i}, c) + \mathbf{C}(p_{c_i}, h) = 0$
- $\mathbf{C}(p_{c_i}, a) + \mathbf{C}(p_{c_i}, b) + \mathbf{C}(p_{c_i}, d) + \mathbf{C}(p_{c_i}, k) + \mathbf{C}(p_{c_i}, m) + \mathbf{C}(p_{c_i}, g) + \mathbf{C}(p_{c_i}, h) = 0$
- $\mathbf{C}(p_{c_i}, k) + \mathbf{C}(p_{c_i}, m) + \mathbf{C}(p_{c_i}, f) = 0$

Solving the set of equations for either of the three separation instances results in one control place p_c. One solution of the equation system is:

$$\mathbf{C}(p_c, a) = 1, \mathbf{C}(p_c, b) = -1, \mathbf{C}(p_c, d) = -1, \mathbf{C}(p_c, m) = 1.$$

A second solution for p_c is:

$$\mathbf{C}(p_c, a) = 1, \mathbf{C}(p_c, b) = -1, \mathbf{C}(p_c, d) = -1, \mathbf{C}(p_c, k) = 1.$$

The resulting synchronization pattern[5] (first solution) is

$$SP = (\{p_c\}, T, \{(a, p_c), (m, p_c), (p_c, b), (p_c, d), (p_c, f)\})$$

The same result produced through the tool Synet is shown in Figure 6.5

Integrating the synchronization pattern SP with the Petri-net from Figure 6.2 leads to the new net shown in Figure 6.6. The behavior of the resulting process description is isomorphic to the legal behavior SG.

[5]Note, arc weights have been omitted in this notation, because they are all equal to 1.

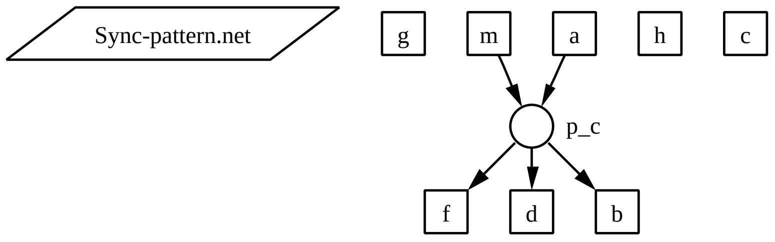


Figure 6.5: The synthesized synchronization pattern

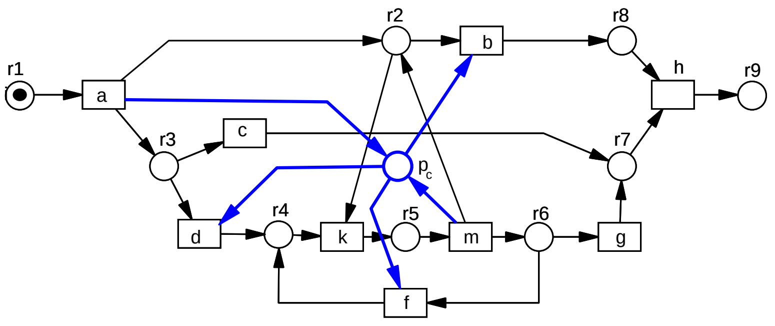


Figure 6.6: Integrating the derived pattern into the primary WF-net

6.4 Applying controller synthesis to workflow modeling

The use of the proposed algorithm for the synthesis of a sound WF-net is obvious. What was called legal behavior corresponds to the behavior described by a robust fragment SG. We furthermore require that the robust fragment covers all controllable transitions, i.e. contains a t-labeled state transition for every transition $t \in T_{CON}$. Consequently the set of labels of the robust fragment coincides with the set of transitions of the primary WF-system: $L = T$. This further requirement is reasonable as in Section 5.5 it was argued that controllable transitions that are not covered by the robust fragment probably denote a modeling deficiency.

Applying the algorithm to that fragment, a set of controller places is synthesized, which determines the desired synchronization pattern $SP = (P_c, T, F_c)$. Finally the synchronization pattern is joined with the primary WF-net in order to obtain the controlled Petri net: $PN_{sync} = SP \cup PN = (P \cup P_c, T, F \cup F_c)$.

Figure 6.7 illustrates the application of controller synthesis for workflow modeling. Starting with a relaxed sound and robust process description a Petri net is constructed fulfilling the conditions that characterize a sound process description: *option to complete, proper termination,* and *no dead transitions,* see Def. 2.47.

We will prove the following theorem.

Theorem 6.14. *Let $S = (PN, i)$ be a relaxed sound and robust WF-system. Let $SG = (V_{SG}, E_{SG})$ be a robust fragment of the reachability graph: $SG \subseteq RG_S$ with $L = \{t | (M, t, M') \in E_{SG}\} = T$. Let PN' be the Petri net that was constructed inserting the computed synchronization pattern $SP = (P_c, T, F_c)$ into PN. Then (PN', i) fulfills the three properties characterizing a sound WF-net:*

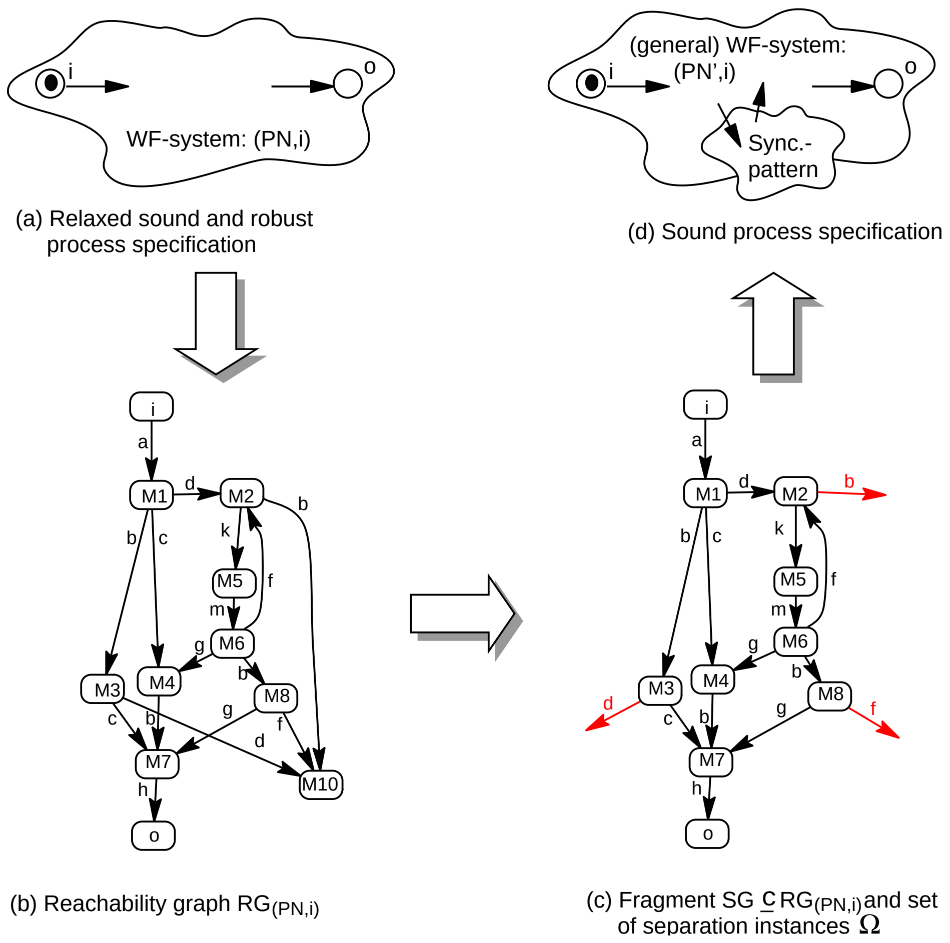
(i) For every state M reachable from state i, there is a firing sequence leading from state M to state o (option to complete).

(ii) State o is the only state reachable from state i with at least one token in place o (proper termination)

(iii) There are no dead transitions in S.

Proof The first and second condition hold, as the behavior of the resulting Petri net was restricted to the robust fragment, i.e. contains sound firing sequences only. The third property requiring no dead transitions can be assured as it was additionally required that all transitions are covered by the fragment ($L = T$).

□



i
o
WF-system: (PN,i)
(a) Relaxed sound and robust process specification
(general) WF-system: (PN',i)
Sync.-pattern
(d) Sound process specification
i, a, M1, d, M2, b, k, M5, f, m, M6, g, M4, b, M8, M3, c, M7, d, M10, h, o
(b) Reachability graph RG(PN,i)
(c) Fragment SG ⊆ RG(PN,i) and set of separation instances Ω

Figure 6.7: Applying controller synthesis for workflow modeling

The resulting Petri net is not necessarily a WF-net. Remember that it was derived by unifying the primary net with a synchronization pattern. The resulting Petri net differs from the primary net only w.r.t to some additional places and the corresponding flow relation. These controller places limit the behavior of the primary net system to sound firing sequences only. Following an analogous argumentation as used in the proof of Theorem 6.13 we can conclude that short-circuiting the synthesized system, we derive a live, bounded and therefore strongly connected net-system having one source and one sink place. Still, the arc inscriptions belonging to the newly inserted places may be greater than 1. This follows as the synthesis of the controller places goes back to the use of general regions. An example illustrating the case, where a Petri net with arc-inscriptions is generated, is given in the Appendix 9.5.

Applying the algorithm to the "Planning trip"- example (cf. Figure 5.8 a) the following synchronization pattern is generated:
$SP = (\{p_c\}, T, \{(f : ok, p_c), (p_c, book\_hotel)\}$. Figure 6.8 shows its integration into the revised process description from Figure 5.6 on Page 96. The resulting

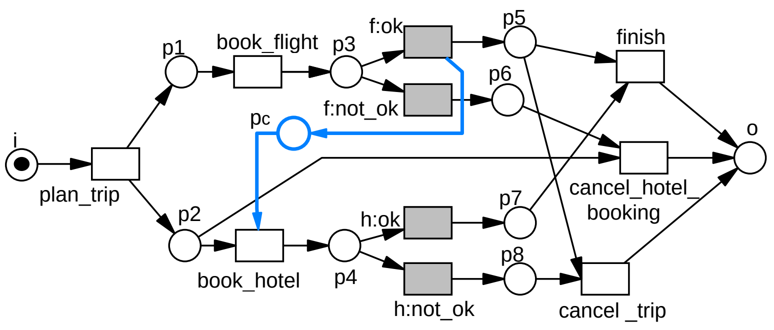


Figure 6.8: WF-net "Planning trip" with integrated synchronization pattern

specification is sound. There are no firing sequences that deadlock or do not terminate properly. Using this process specification as basis for the workflow-controller a reliable process execution at run-time can be guaranteed.

6.5 Appraisal of results

The objective of this chapter was the generation of a sound WF-system on the basis of a relaxed sound and robust WF-system. This was achieved by applying results from Petri net synthesis. But the quality of the results differs depending on the

TS and the approach used. The best result is achieved if the controller synthesis approach [GRX02b] is applied to an elementary TS which additionally satisfies the requirement $L = T$. The result is a WF-system which differs from the primary specification only by some additional places. Its behavior is isomorphic to the robust fragment of the primary WF-system.

The same approach applied to a non-elementary but general TS results in a potentially general Petri net. Advantage is again the high resemblance with the primary WF-net. The resulting net is still in accordance with the structural WF-net properties. It has exactly one source- and one sink place and the short circuited net is strongly connected. The resulting net differs from the primary WF-net only by some additional places. Still, the introduced places may be linked by weighted arcs. The behavior of the resulting Petri net is again isomorphic to the robust fragment of the primary WF-net.

An approach that always leads to a sound WF-net is the full synthesis as described in [CKLY98]. It is applied if the robust fragment of the WF-system

- is non-elementary and a change towards general WF-nets is considered inadequate,
- does not satisfy the requirement $L = T$, or
- does not satisfy the separation axioms of a general TS

The result is a safe WF-net with a behavior bisimilar to the robust fragment. The disadvantage of this approach is the possibly high dissimilarity with the primary WF-net.

Assertions about possible assignments of a robust fragment The robust behavior of a relaxed sound and robust WF-system (PN, i) was computed within the robustness algorithm, cf. Chapter 5 and coincides with a fragment of the reachability graph. The computed fragment is a transition system, as it satisfies the axioms (A1)-(A3).

So far, there is no result showing that the fragment always satisfies further axioms, and hence coincides with a special TS, e.g. a general TS. The robust fragment is not necessarily elementary, even if the reachability graph was elementary. Furthermore, the robust fragment does not necessarily contain a t-labeled state transition for every transition $t \in T_{CON}$. There are counter-examples for both cases although artificially constructed ones. The reader is referred to the Appendix for illustration. The counter-examples do not match any known, realistic process description. All relaxed sound and robust WF-systems investigated in the course of the work on the thesis produced fragments satisfying the axioms of elementary transition systems and covered all controllable transitions. Since we only have two artificially constructed counter-examples it is not possible at this stage to make any generalizations about the class of non-compliant WF-nets. This could be an interesting topic for further research.

6.6 Related work

The following section is subdivided into three parts. The first two parts review literature on Petri net synthesis and Petri net controller synthesis. In the last part we investigate two other approaches for the synthesis of process controllers and outline shortcomings of the provided algorithms.

Petri net synthesis The synthesis problem was first addressed in [ER90]. Subsequently, in [NRT92] it was shown that an elementary net system can be synthesized on the basis of regions from a (sequential) transition system satisfying some separation conditions, namely from an elementary transition system.

The approach was enhanced in [CKLY98] to synthesize safe net systems on the basis of excitation regions. These are regions that are related to transitions while normal regions in a transition system are related to places in the corresponding Petri net. This approach is not limited to elementary TS but covers the full class of TS by means of transition splitting. The behavior of the resulting Petri net is not necessarily isomorphic to the TS but bisimilar.

Furthermore, the synthesis problem has been solved for other classes of Petri nets, cf. [Muk92, DS93, BBD95, BMPV96, KCK^{+}96, Dar00] and [BDC02]. Here the concept of a general region is used. General regions are multisets where regions are sets. A comprehensive review of the theory of regions can be found in [BD98].

Petri net controller synthesis The objective of this research area is the synthesis of a controller supervising a plant (e.g. given in terms of a discrete event systems (DES)), so that the entering of forbidden states is avoided.

In the original work [RW87], controller synthesis is based on finite state machines (FSM). FSMs provide a general framework for establishing fundamental properties of DES control problems. However, they are not convenient or intuitive to model practical systems, because of the large number of states that have to be introduced to present several interacting subsystems, and because of the lack of structure [HK94]. More efficient models have been introduced in the DES literature, among them Petri nets. For a survey of DES control using Petri nets the reader is referred to [HK94],[Giu96], and more recently [CDLX02].

There are several approaches that synthesize controllers for specific subclasses of Petri nets, such as [HK94] for marked graphs, [BNRL$^+$95] for state machines and [HGZ96] for nets satisfying some transition conflict condition (similar to the free-choice property). In [GDS92] it was proven that for the subclass of safe and conservative Place/Transition nets any forbidden marking specification can be enforced by a set of additional places called monitors. The forbidden marking specifications are formulated in terms of general mutual exclusion constraints (GMEC). A different approach was proposed in [YMLA96]. Here a set of monitors is computed on the basis of the net's place invariants. It is shown that a wide variety of forbidden marking specifications can be reformulated in terms of place invariants. Still, for some transformations the approach is limited to safe Petri nets only.

In [GRX02b] an approach is presented synthesizing controllers for general Place/ Transitions nets. Specifications that can be enforced by the approach are expressed as sets of forbidden states. The proposed approach consists of two main steps. It first determines the desired behavior of the reactive system using a Ramadge-Wonham-like approach. It then uses the theory of regions as proposed in [BBD95] to design a Petri net controller, which is again a set of control (or monitor) places. The Petri net controller is synthesized if the desired behavior of the reactive system (partial reachability graph) satisfies the event- and state separation axioms of a general TS. No statement has been made how this requirement restricts the set of enforceable specifications.

Algorithms computing the set of separation instances The application of the controller synthesis method is based on the prior computation of the set of separation instances, Ω. Within the described setting Ω could easily be determined as state transitions leaving the robust fragment SG. There are two further approaches proposing the use of controller synthesis for the improvement of process descriptions, namely [GRX02a] and [Pet00]. Here, slightly different procedures are proposed to determine the set of separation instances.

The separation instances are computed by a backwards search starting in the forbidden states, such as deadlocks or other states forbidden by the specifications (e.g. more than one object using a certain resource). Note that forbidden states in our setting would refer to states from which no extension towards state o (proper termination) exists.

Starting from the forbidden states, so-called, dangerous markings are detected. In [GRX02b] a dangerous marking is defined as a marking that is forbidden or that is reachable from a forbidden marking via the firing of only non-controllable transitions. In [Pet00] a dangerous marking is the marking that enables the *last* controllable transition, before enabling a sequence of uncontrollable transitions, that leads to a forbidden state. The separation instances then correspond to state transitions leading into dangerous markings ([GRX02b]) or state transitions leaving the dangerous markings ([Pet00]).

In both approaches the dangerous markings are markings that lead to forbidden markings via a sequence of uncontrollable transitions. Markings reachable via several controllable transitions are not considered dangerous. This causes a problem as controllable transitions may lead into dangerous markings as well. In these cases both procedures lead to an improper set of separation instances.

Chapter 7

Installing different scheduling strategies

Transforming a relaxed sound WF-system into a sound WF-system, the behavior is restricted to a subset of the sound firing sequences. The choice for a certain subset determines a strategy, which in turn determines the efficiency of the process execution. The fragment computed through the robustness algorithm in Chapter 5 so far only determined pessimistic strategies. Implementing the computed strategy, applying the techniques presented in Chapter6, the process execution is sequentialized. In this chapter we will discuss alternative strategies and their implementation.

The chapter starts with a general overview of factors affecting the efficiency of the process execution. Section 7.2 addresses the determination and implementation of different scheduling strategies. In Section 7.3 the installment of optimistic strategies is described in more detail. In Section 7.4 the proposed method is appraised w.r.t its support in finding optimization potential. Finally, related work is discussed.

7.1 Optimization potential

One of the most significant objectives of workflow management is the improvement of the overall performance of the system. Both the qualitative properties (e.g. prevention of congestion of remaining orders, prevention of deadlocks), and quantitative properties (e.g. maximum throughput, minimum average delay, optimum employment of resources) have to be considered.

The qualitative properties are closely related to the correctness of the underlying process description. To guarantee a smooth processing of the process at run-time, the workflow specification should be sound. The quantitative properties are determined through the selection of a scheduling strategy and the determination of system parameters.

7.1.1 Determination of system parameters

System parameters are adjustable parameters that influence the efficiency of the execution. Important examples are number and assignment of employed resources. It is clear that the execution of a process can be improved if more people work on it or a faster machine is used, but higher costs must be accepted. However, the question of resource allocation was beyond the scope of this thesis. It was determined at the outset that the focus would be on the control flow aspects. However, the author has emphasized some resource allocation issues in [DFZ02, DFZ00b, DFZ00a].

7.1.2 Scheduling strategies

Another potential for the optimization of the process execution is the choice of a suitable scheduling strategy fixing the final dispatching rule. The choice is difficult because in most cases it will not be possible to find a strategy that suits all situations.

In general, strategies can be optimistic or pessimistic. Pessimistic strategies wait for decisions to be taken in advance in order to avoid faulty situations. Following a pessimistic strategy, the process execution is sequentialized. In contrast, optimistic strategies support parallel execution of depending threads but accept additional costs in some cases through the need for recovery.

The decision for a certain strategy is based on expert knowledge or long term statistical evaluations. It depends, among other things, on costs and duration of tasks. Making use of the operational semantics of WF-nets, the decision for a specific strategy can be supported. Therefore, the process description must be enhanced with further information such as the duration and costs of tasks. Simulating the behavior of the workflow system, a trade off is possible between incurred costs and duration.

The installment of a certain strategy should preferably be one of the last steps in the modeling of workflows, as corresponding information (the occurrence probability of a certain failure, costs of failure compensation, or priorities) will often only become available then, and may even change during the run-time. Their late incorporation allows flexibility if priorities change. It will not be necessary to revise the whole procedure starting from new requirements, but modeling results from earlier phases may be reused.

Starting from a relaxed sound process description, we will now investigate how different scheduling strategies become implemented. Beside the implementation of a fixed strategy, the proposed procedure also facilitates the identification of useful strategies.

7.2 Strategy determination and implementation

A relaxed sound process description determines a set of desired executions. Still, it does not describe "how" the desired executions are achieved. This decision is left to a strategy.

We have seen that a strategy corresponds to a special fragment of the reachability graph (cf. Def. 5.2). In Chapter 6 it was shown how a strategy was implemented, restricting the behavior of the relaxed sound WF-system to the corresponding fragment. The derived WF-system can be used as input for a WF-controller. If the implemented strategy was complete and winning, the WF-controller could, by following the prescribed rules, guarantee a sound process execution at run-time independent from the moves of the environment.

If the primary relaxed sound WF-system is robust, there is a complete winning strategy, i.e. a fragment of the reachability graph which satisfies the corresponding requirements (contains only sound firing sequences, covers all non-controllable transitions, has only controllable state transitions leading out, cf. Def. 5.2, 5.3 and Def. 5.4).

Still, there may be sound firing sequences in the primary WF-system which are not supported by any robust fragment. These executions, although sound, would not be supported if the corresponding strategy becomes implemented. The problem with these executions is that proper termination cannot be guaranteed because if the environment interferes the system may end in a deadlock.

We will illustrate this by means of an example. Consider the process description given in Figure 7.1(a). There are two choices: The upper choice between `taskA1` and `taskB1` is controllable. The lower choice, which is non-controllable, decides between `taskA2` and `taskB2`. But the choices are not independent. The following two transitions either join the A-tasks (`joinA`) or the B-tasks (`joinB`). A mixed execution of A- and B-tasks leads to a deadlock.

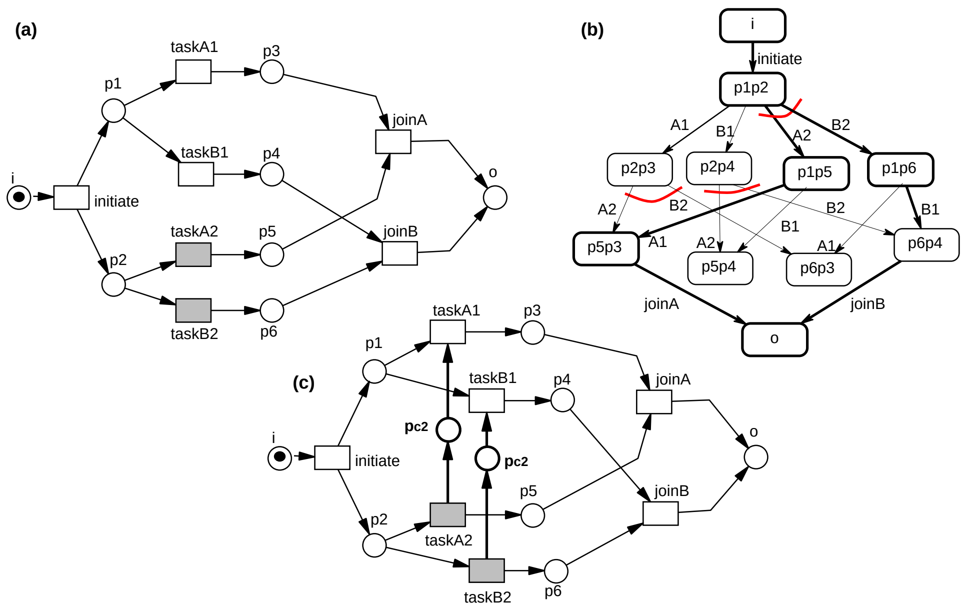


Figure 7.1: (a) Relaxed sound & robust WF-system (b) Reachability graph (c) Sound WF-system

The WF-system is relaxed sound and robust and covers all transitions. Figure 7.1(b) shows the reachability graph of the WF-system, highlighting the robust fragment. Implementing the strategy that corresponds to the robust fragment (applying the controller synthesis approach described in Section 6.4), the WF-system shown in Figure 7.1(c) is derived. The resulting WF-system is sound, so it may be used as basis for the execution support of the process at run-time.

Implementing the strategy, a synchronization pattern was incorporated into the primary WF-net. Through the synchronization pattern, the two transitions `taskA1` and `taskA2`, and transitions `taskB1` and `taskB2` become synchronized. As a consequence, all sound but parallel firing sequences of the relaxed sound WF-system have been eliminated. Using this process description as input for a WFMS, the execution of the process becomes serialized. The non-controllable choice would always be awaited before the controllable choice is processed.

Scenario I: Assume that the determination of the non-controllable choice takes a long time. Then this pessimistic scheduling would be annoying. It could be more efficient to support the parallel sound firing sequences and to consider compensation of tasks if a deadlock is reached.

Scenario II: Assume that the determination of the non-controllable choice takes a long time and one of the outcomes of the non-controllable choice (say `taskA2`) is much more likely than the other task (`taskB2`). In this case, a suitable strategy would support only one of the parallel but sound firing sequences, again considering compensation when a deadlock is reached.

In the following we will discuss how an optimistic approach is implemented.

7.3 Installing an optimistic strategy

Investigating the behavior of the relaxed sound and robust WF-system there are sound firing sequences which are not covered by the robust fragment[1]. These executions, although sound, would not be supported if the strategy that corresponds to the robust fragment becomes implemented. If the domain experts consider these executions to be considerably more efficient than the ones covered, another strategy must be found. The new strategy should cover these sound executions. It does

[1]This holds also for sound firing sequences containing controllable transitions not covered by the robust fragment, see Section 5.5.

not suffice to merely combine the desired executions. The corresponding fragment will not satisfy the properties of a strategy. It is not self-contained with respect to non-controllable transitions, cf. Def 5.2. There are non-controllable transitions that lead from the fragment to states that indicate a deadlock.

In order to support the desired set of sound executions, the fragment must be enhanced. New behavior must be incorporated that makes it possible to recover from deadlocks. This is achieved by adding tasks to the process description which compensate the results of previous tasks. For their specification further information must be compiled, regarding:

- the states from which compensation is possible,
- compensating tasks, and
- states to which the process is rolled back after compensation.

The specification of the compensating tasks cannot be automated but must be done by domain experts. The knowledge for the recovery behavior is based on the application context in combination with efficiency considerations and cannot be determined by a predefined set of rules.

Once the recovery behavior has been specified it is incorporated into the primary WF-net. The result of this enhancement must again be a relaxed sound and robust process description.

The robust fragment is computed on the basis of the enhanced WF-system. It now contains the desired firing sequences. It also contains some new sound firing sequences which enable recovery if a (former) deadlock is reached. Implementing the strategy that corresponds to the derived robust fragment, a sound WF-system is computed. We will illustrate this again by means of examples. Reviewing the two scenarios we will adapt the process description shown in Figure 7.1 such that, the demanded executions are supported by a strategy. We will start with the second scenario. Here, it was assumed that one result of the non-controllable choice was more probable than the other one.

Scenario II: Figure 7.2(a) shows the WF-system from Figure 7.1(a) now enhanced by a compensation task `changeAB`. By means of this task it is possible to recover from the state `p6p3`, which was a deadlock in the primary WF-system. The WF-system is relaxed sound and robust. The reachability graph of the system and the robust fragment (highlighted) are shown in Figure 7.2(b).

Implementing the strategy that corresponds to the robust fragment, the WF-system shown in Figure 7.2(c) is derived. The new system supports the optimistic executions demanded in the second scenario. As a consequence, some additional, less efficient executions are accepted too.

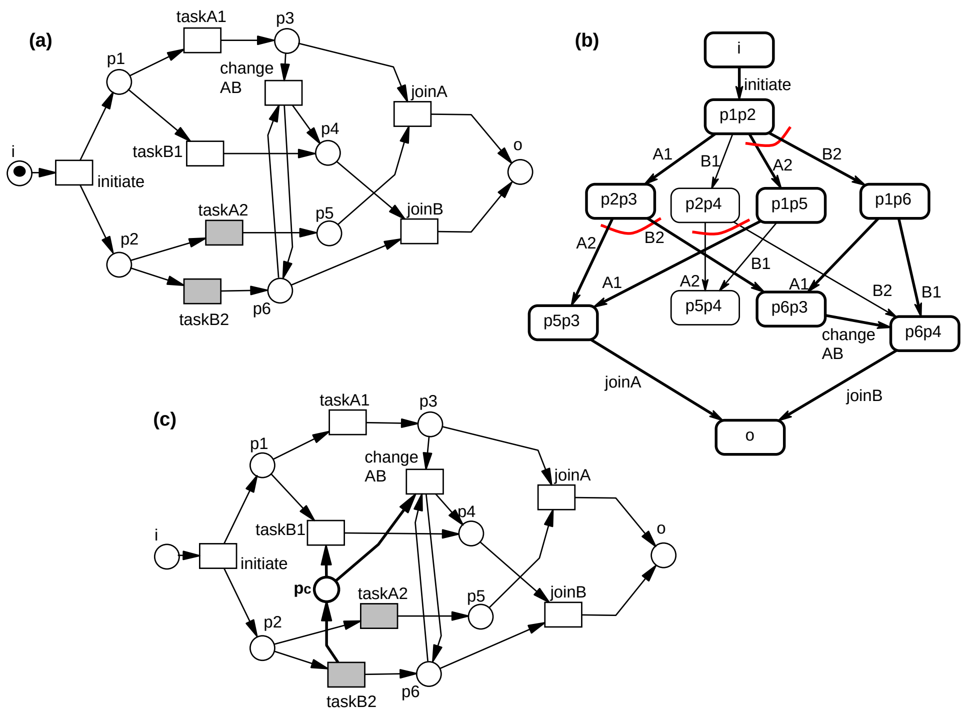


Figure 7.2: (a) WF-system with integrated recovery behavior
(b) Reachability graph (c) Sound WF-system

Scenario I: In this case it was assumed that compensation is not very expensive and parallel executions should be generally supported. Figure 7.3(a) shows the WF-system from Figure 7.1(a) now enhanced by two compensation tasks changeAB and changeBA. These two tasks reverse the decision made. If a deadlock is reached in the primary description, one of the compensation task can be executed, leading to a state from which proper termination is guaranteed. Here the robust fragment, depicted in Figure 7.3(b) coincides with the reachability graph. Therefore, no synchronization pattern has to be computed. The WF-system is already sound.

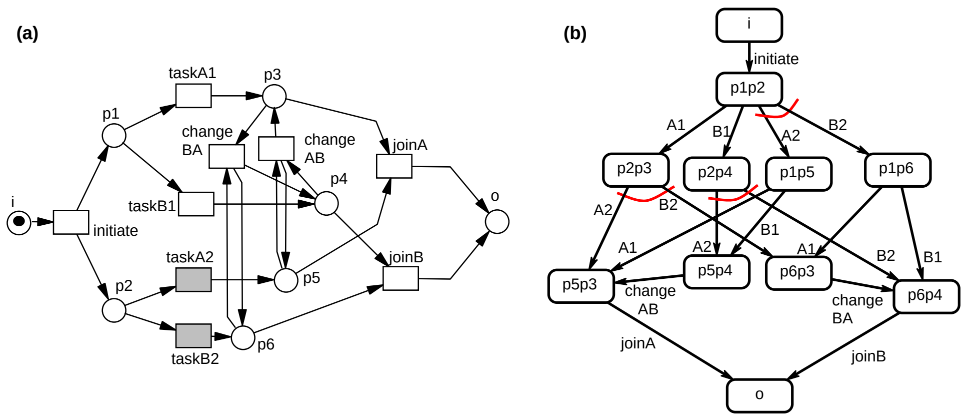


Figure 7.3: (a) WF-system with recovery behavior
(b) Corresponding reachability graph

In both cases, we have seen that through the appropriate adaption of the primary process description other than pessimistic strategies can be supported. The recovery integration was only shown exemplarily. As necessary information is specific for any process, this cannot become automated. Still, the investigation of the possible behavior of the relaxed sound process may help to find an optimum set of executions. If the set does not already form a robust fragment, the user needs to adapt the process description such that, the desired set of executions determines a robust fragment. New tasks must be incorporated to ensure that non-controllable transitions that were not covered by any strategy are incorporated. This can be achieved e.g. by compensation of controllable tasks. After the enhancement the WF-system must be again relaxed sound and robust.

Note that the introduction of additional behavior can also be used to improve relaxed sound and robust process descriptions where the robust fragment does not cover all controllable transitions ($L \subset T$).

As soon as the WF-system is relaxed sound and the set of desired executions determines a robust fragment, the fragment can be implemented automatically. The result is a sound WF-system.

In this chapter we have seen how existing results, e.g. the implementation of a strategy (cf. Chapter 6) and the modeling towards relaxed sound and robust process descriptions (cf. Chapter 3 to Chapter 5) can be reused to detect and implement optimization potential. The relaxed sound process description can be seen as an incomplete body that must be enhanced for further use. The way of enhancement determines the final execution policy. The benefit of the method is increased flexibility in the modeling of workflow. Existing process descriptions can be reused under changing priorities or different prerequisites. Only the scheduling strategy has to be adapted.

7.4 Appraisal of results

The optimization of existing processes is the core of all business process re-engineering projects. The flexible adaption of workflow specifications according to changes in the application domain is another important issue. However, there is hardly any method that supports the user in finding optimization potential. Most approaches merely propose investigating the performance evaluation of process descriptions under varying parameters in a trial and error process. We could summarize the drawbacks of this wide range of existing methods in the following points:

- in box-solutions which play only within a given setting
- trial and error
- real improvement is left to the modelers
- support is limited to invitations such as to look for possible parallelism

The key problem here is that creative import is left solely to the modeler. While this input will always be important, by introducing support as described in the previous chapter(s) the modeler will also be able to find non-immanent solutions (outside the box).

The proposed approach presents subsets of possible executions among which the application developer may choose a suitable one. If a non-robust subset is chosen that relates to an optimistic strategy we can assume that this strategy had not been thought of before. Its implementation needs some further investigation, namely the enhancement of the primary specification by some recovery behavior. Even here, the user can count of some support through the provided process model. The compensation tasks to be added need lead to states that are covered by the robust fragment. Feedback is provided again through the tests for relaxed soundness and robustness.

We are confident that the strategy implementation on top of a relaxed sound process description provides a good starting point for the development of a tool supporting the identification of optimization potential in a methodical manner.

7.5 Related work

Transactional workflows In the presented approach the recovery specification was left to the modeler. Specific knowledge about the states from which compensation is possible, compensating tasks, and states to which the process is rolled back after compensation, is assumed.

The specification, analysis and support of recovery is a core aspect within transaction management. Introducing the possibility to recover from deficient states by compensating corresponding transitions, transactional properties (e.g. ACID) have to be considered. Transaction models determine a set of transactional properties and describe how they are enforced. Transaction models have been initially developed to be used in database systems. Since, in [SR93] transactional workflows are introduced as workflows with transaction support, a lot of work has been done to integrate workflows and transaction models. Much effort was put in the investigation of advanced transaction models [Elm92] and their capabilities to support workflow applications, e.g. [AAEA$^+$96, KMO98, GPS99]. The general idea, exploited in the mentioned approaches, is to assign existing transaction properties to workflow activities. The use of several transaction models, e.g. SAGA or nested transaction provides high flexibility because activities can be grouped as desired. However, properties like atomicity, isolation and recovery cannot be defined independently, but must follow either the SAGA or nested transaction model.

Other approaches, aiming at the integration of workflows and transaction models, focus on the specification, analysis and support of transaction state dependencies, e.g. [ASSR93, GH94, GHM96, RSS97, AAH98, LR99, DDGJ01]. Within these approaches transactional properties can be defined independently from a specific transaction model. The transaction state dependencies are used to describe transactional properties of single tasks but also of intertask dependencies. In [Reu89, ASSR93, GH94, GHM96, AAH98] intertask dependencies are also used to implicitly specify the control-flow dependencies between tasks.

In contrast [DDGJ01] and [LR99] propose to specify the transactional and control-flow dependencies separately. Here transactional properties are annotated to the workflow specification, introducing the concept of *spheres*. All tasks contained in a common sphere share the same corresponding transactional property, e.g. an atomicity sphere describes the property that the contained tasks either all execute successfully, or all have to be compensated.

In case the transactional properties (as well as the workflow) are specified in a formal manner, e.g. using Petri nets ([AAH98, DDGJ01]) or CTL ([ASSR93]), the enforceability of the intertask dependencies can be verified. In our terminology this would correspond again to robustness of the corresponding WF-system. This means, in the positive case, a controller can be constructed supervising the compliance of the transactional properties at run-time. For the construction of the controller again the adapted synthesis methods (described in Chapter 6) could be applied.

Chapter 8

The process model

We are now in a position to tie together all the elements introduced in the previous chapters in a final process model for the specification of workflow processes.

The process model was designed to guide the modeler from an intuitive but informal process description towards a formal workflow specification. The resulting process description provides a sound specification of the functional process aspects. By adding further aspects, the resulting process description can be used as input for a WFMS.

The process model bridges the gap between different abstraction levels. In contrast to other approaches this is done not only by refining tasks but also by refining applied correctness measures.

The chapter is organized as follows. In the first part, cf. Section 8.1, we motivate the objective, illustrate the process model on a general level and explain modeling decisions. In the second part, cf. Section 8.2, we describe the proposed process model in more detail. The single steps are illustrated using a running example. Finally, related work is discussed.

8.1 Objective and Overview

The main objective of the proposed procedure is to support the modeler in defining the functional aspects of a workflow specification.

The modelers are experts within their domain, but are assumed to have relatively little modeling knowledge. Therefore the proposed process model initially supports the use of an intuitive but semiformal modeling language. Such languages provide a set of graphical elements and do not restrict their combination by too many rules. A greater freedom in modeling raises the acceptability of languages but imports ambiguity and vagueness into the derived process description.

At the start of process modeling this is an advantage. Here the objective is to find a common level of understanding. Semi-formal semantics make it easier to agree on a process description, as different interpretations are possible. In contrast, the final process description, to be used as basis for a WFMS, should support a consistent and unambiguous interpretation. Here, the modeling technique used should have a formal foundation.

The proposed process model supports the use of different modeling languages for different purposes. It is clear that this places considerable emphasis on the transformation between the different process descriptions. The procedure supported starts with modeling business processes from a user perspective. For this purpose, the use of a semi-formal modeling technique is proposed. The result of this first step is a process description which possibly contains ambiguities and/or deficiencies.

The revision of the primary specification is supported, applying a pragmatic correctness check. In preparation for this, the primary process description is mapped onto Petri nets. This way formal semantics is attached to the primary specification. Note, the transformation does not eliminate ambiguities inherent to the primary modeling technique but makes them explicit.

The transformed process description is tested for some pragmatic correctness criteria, namely relaxed soundness and robustness. By applying them, we refrain from the claim of producing process specifications that satisfy soundness right from the beginning.

If the primary process description contained deficiencies, the analysis will give precise feedback to the modeler. This way revision of the process description is supported until the required correctness properties are satisfied.

In the next steps of the process model, the description is enhanced to fit the requirements posed by the use in a WFMS. The modeler is guided in eliminating ambiguities, fixing an execution strategy, and changing the perspective of the description towards monitoring.

Altogether, the proposed process model consists of five steps (Figure 8.1), namely 1. “Business process modeling (EPCs)”, 2. “Transformation into WF-systems”, 3. “Correctness check and feedback”, 4. “Strategy determination & implementation”, and 5. “Control refinement”. They are now reviewed.

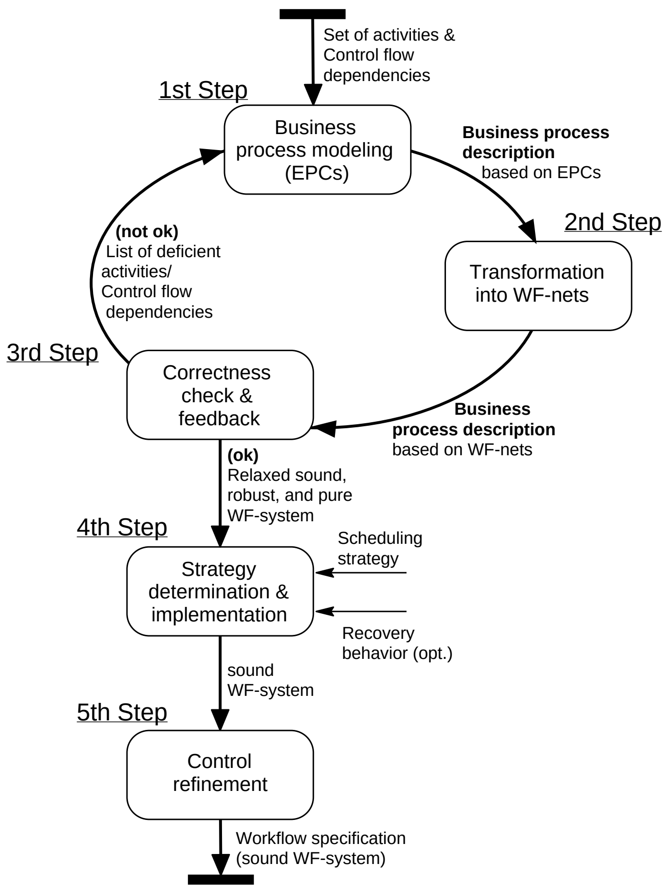

Set of activities & Control flow dependencies
1st Step
Business process modeling (EPCs)
Business process description based on EPCs
2nd Step
Transformation into WF-nets
(not ok) List of deficient activities/ Control flow dependencies
3rd Step
Correctness check & feedback
Business process description based on WF-nets
(ok) Relaxed sound, robust, and pure WF-system
4th Step
Strategy determination & implementation
Scheduling strategy
Recovery behavior (opt.)
sound WF-system
5th Step
Control refinement
Workflow specification (sound WF-system)

Figure 8.1: A process model for workflow modeling

1st step: Business process modeling (EPCs) For the modeling of business processes, we chose Event-driven Process Chains (EPCs) of the Architecture of Integrated Information Systems (ARIS) described in [Sch94]. The use of EPCs is not essential. Any other semiformal modeling technique could be used equally well. EPCs were chosen as they are widely accepted in practice. This is based on their use to describe the SAP reference models [KT97] and their comprehensive tool support through the ARIS tool set.

EPCs provide a set of graphical elements. These can be combined in a fairly free manner. The result is a process description providing a pattern describing a set of desired process executions.

In the original publications concerning EPCs neither a comprehensive and consistent syntax nor corresponding semantics are defined. EPCs leave room for interpretation and hence ambiguities. An ambiguous process description may be desired in the beginning, where the main focus is on communication.

2nd step: Transformation into WF-systems In the second step, the primary specification is transformed into a Petri net. This way the process description is provided with formal semantics. As a suitable Petri net-type we chose Workflow nets (WF-nets), cf.[Aal98]. In contrast to other existing approaches, the proposed transformation between the two techniques does not resolve ambiguities but makes them explicit i.e. all intended behavior is preserved. The transformation is well defined and therefore enables the transfer of properties of the WF-net to the original process description.

3rd step: Correctness check & feedback Petri nets are supported by a wide variety of analysis techniques and tools. These can now be applied to the resulting WF-net. Beside standard tests such as the check for liveness and boundedness, we particularly propose checks for relaxed soundness and robustness. Relaxed soundness guarantees some minimum requirements that the resulting WF-system should satisfy. Due to the possibly pure input, this criterion provides an adequate means to conclude the correctness of the primary process description. Whereas relaxed soundness makes a statement about the internal behavior, robustness states whether the interaction with the environment can be managed in a satisfying manner.

4th step: Strategy determination & implementation The derived process description is a relaxed sound and robust WF-system. WF-systems are generally suitable for workflow specification and they provide a precise formal foundation in combination with operational semantics.

However, it is not advisable to use the derived relaxed sound and robust WF-system as input format for a workflow management system. Relaxed soundness and robustness only state that at least all intended behavior has been described correctly. They do not guarantee that deficient executions do not exist.

To guarantee a reliable execution of the process at runtime, the process description will be augmented such that, it becomes sound. The necessary enhancements are determined by the choice of a certain scheduling strategy. The decision in favor of a certain strategy (pessimistic or optimistic) is made by the modeler. The enhancements are computed automatically. The resulting process description is sound.

5th step: Control refinement A workflow management system monitors activities performed by actors (people or software). Monitoring comprises activating tasks by assigning them to certain actors and waiting for the tasks to complete. To use the sound process description as input format for the workflow management system, it has to be refined, reflecting the change from modeling to monitoring. In the last step, task-modeling transitions are replaced by a refining subnet. Within the subnet a distinction is made between the initiation of the task, the processing of the corresponding activity and the completion of the task.

The derived specification is a sound WF-system specifying the functional aspects of a process. By adding further aspects, the process description can be used as input for a WFMS. Operating on the basis of the derived workflow specification, smooth and reliable execution of the process can be guaranteed.

We will now provide detailed information of each step and illustrate these with the help of a new running example.

8.2 Detailed description and illustration

8.2.1 1st step: Business process modeling (EPCs)

The graphical elements and the modeling rules of EPCs were introduced in Chapter 3. We will not repeat this but merely recall some information about the semantics. We refer to the primary publications concerning EPCs [Sch94] and act on the assumption that EPCs do not have operational semantics. In our interpretation, an EPC specification describes a set of accepted executions. Deficient executions are only described implicitly as the set of executions which do not fit the described pattern.

Modeling with EPCs, the application developer describes "what the execution (should) look like", i.e. a set of accepted executions. The modeler does not think about all possible executions but merely specifies a set of accepted executions. In other words: EPCs do not state "how" the described executions are achieved but just which executions are desirable. We will explain the EPC semantics we use by means of an example.

Fig. 8.2 shows an EPC modeling the process "Handling of incoming order". It represents a reduced version of a real-life process of a telephone company. The process models the ordering of a mobile phone which involves two departments: accounts and sales.

The process starts with the event `new order`. The execution is split into two parallel threads (`AND_split`), the right one models the accounting, whereas the left one models the sales. In the accounts department, first the creditworthiness of the customer is checked (`check_credit`). The result of this is either `ok` or `not_ok`. If the result is positive the payment is arranged (`arrange_payment`). Otherwise, the order is canceled. The left path models the tasks on the sales side. After the order was recorded (`record_order`), it either follows the paths `pick`, `wrap`, and `deliver`, or `cancel`.

The two AND-connectors at the end make sure that only executions are accepted where both the accounts and the sales departments, either cancel or proceed with the order. The process "Handling an incoming order" is finished by archiving the information (`archive`).

An accepted execution of the EPC from Figure 8.2 is

- `check_credit`, `arrange_payment`, `record_order`, `pick`, `wrap`, `deliver`, `archive`
  but also
- `record_order`, `cancel`, `check_credit`, `notify_cancel`, `archive`.

The EPC only describes executions where the two departments work together correctly: they either both accept the order (`AND_accept`) or both reject it (`AND_cancel`). Hence, the execution `check_credit, notify_cancel,`

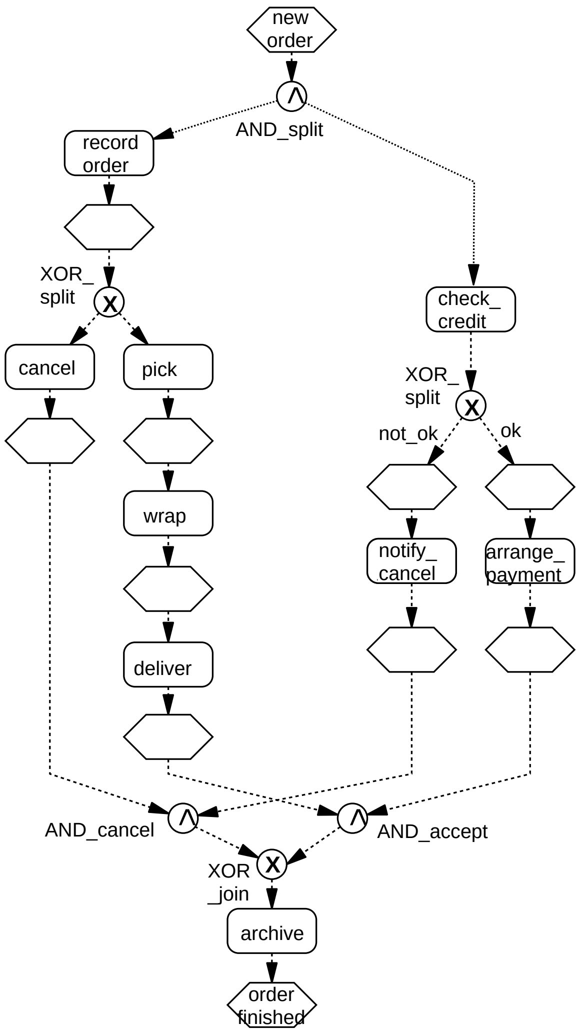


Figure 8.2: EPC: “Handling an incoming order”

`record_order, pick, wrap, deliver, archive` is not described by the EPC.

The EPC does not determine "how" the accepted executions are achieved. It does not stipulate the order of the two possible choices. So, the EPC from Figure 8.2 accepts executions where the two departments work in parallel as well as executions where the two departments work sequentially. In an early design phase, this abstraction is beneficial, as it relieves the designer from thinking about efficiency aspects of the execution for the time being.

To investigate the correctness of the process description, we transform the EPC into a WF-net.

8.2.2 2nd step: The transformation of EPCs into WF-nets

The transformation of EPCs into WF-nets was introduced in Chapter 3. It takes place in three steps. First, elements of the EPC are mapped onto Petri net-modules. Second, the modules are combined to form a complex process description. In the last step the resulting Petri net is enhanced to fit the WF-net definition. The last step is only necessary if the EPC has more than one input and/or output event. The transformation is well-defined. By applying the proposed rules, each EPC is assigned to exactly one WF-net.

Figure 8.3 shows the application of the rules to the EPC from Figure 8.2.

In contrast to EPCs, Petri nets do have operational semantics (execution semantics). A Petri net specification also describes "how" an execution is reached. Whereas an EPC only describes a set of accepted executions, a Petri net describes all possible behavior. This difference has been neglected in previous attempts of mapping EPCs to Petri nets. As a result, all EPCs have been considered deficient if the corresponding Petri nets were deficient, cf. [Aal99, LSW98, MR00]. As a consequence, the modeling facilities of EPCs have often been restricted such that, it was possible to give useful feedback in case of errors.

The EPC in Figure 8.2 describes the set of executions where either both departments proceed, or both departments cancel the order, which describes reasonable behavior. The EPC should therefore be considered correct. Previous approaches would reject this process description, because of deficiencies of the corresponding Petri net. In the following we see that the Petri net is vulnerable to deadlocks.

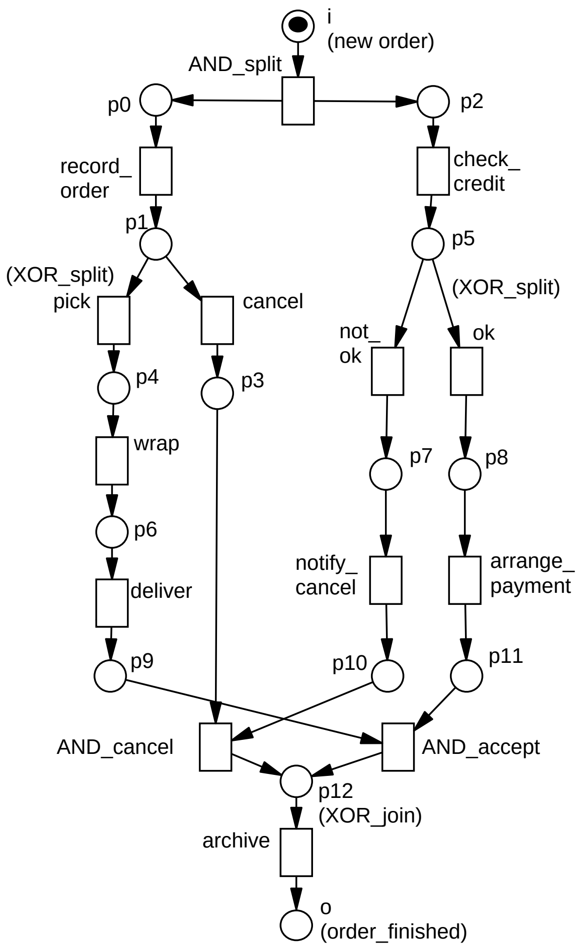


Figure 8.3: WF-net: “Handling an incoming order”

8.2.3 3rd step: Correctness check & feedback

As WF-nets are a special type of Petri nets, many analysis techniques can be applied. The question is which correctness properties are reasonable to check. The WF-net is the result of the transformation of an EPC. So far, we only know that the WF-net derived is pure (cf. Section 3.4.1). EPCs make it possible to model sequences, parallel threads, alternatives and cycles, and these constructs are found again in the WF-net. A possible failure that may occur through the introduction of cycles is the unboundedness of the resulting WF-net. Since modelers try to match realistic scenarios within their descriptions, we can assume that unbounded behavior is not desired. Unbounded places in the WF-net should therefore be discovered through the analysis phase, and the function which corresponds to the faulty transition should be notified to the modeler. Boundedness can be verified using standard Petri net techniques (coverability analysis).

In the next step, we check the WF-system for relaxed soundness. In Chapter 3 it was shown that in the context of modeling with EPCs, relaxed soundness is more useful than the check for soundness. Requiring soundness, rather than relaxed soundness, means that many EPCs are discarded although they represent reasonable behavior.

Relaxed soundness

As we have seen, relaxed soundness is a more pragmatic criterion which only checks the resulting WF-system for some minimum requirements. Relaxed soundness requires finding enough sound firing sequences, so that each transition is contained in one of them. If the WF-system is not relaxed sound, it is not sound either.

Applying the new criterion, we do not claim to be able to model in a precise and sound manner right from the beginning. Having looked at various process specifications by domain experts, we are convinced that such an approach will find acceptance from the modelers.

The check for relaxed soundness has been implemented within Petri net tools, such as LoLA [Sch99] (Low Level Petri Net Analyzer) or Woflan [VA00]. The algorithms determine whether the WF-system is relaxed sound or not and in the negative case return a list of deficient transitions.

The results of the correctness check of the WF-net can be transferred directly to the primary specification. If the result of the relaxed soundness check is positive, we can conclude that the EPC represents reasonable behavior. If the WF-net is furthermore well-structured, we can conclude with Theorem 4.12 that the specification is sound.

If the result is negative, then there are transitions that are not contained in any sound firing sequence. According to the proposed transformation, the deficient transitions either correspond to a function or to a connector within the EPC. This means that either the function or one of the possible choices described by a connector are not included in an execution that terminates properly. It can be concluded that the corresponding part in the EPC needs improvement. This way, precise feedback is provided which will help the modeler to improve the process description until the corresponding WF-system fits the property.

Non-controllable choice robustness

As a further criterion, we require the resulting WF-system to be non-controllable choice robust (short: robust). A necessary prerequisite for this second check is the indication of non-controllable transitions in the WF-net. In general, this step needs the intervention of the modeler. Using EPCs it cannot become automated due to the imprecise event concept. Recall that an event either triggers a function or marks the termination of it. Therefore there is a distinction between the trigger-event and the supply-event [Sch94]. Within the modeling, this distinction is blurred by the use of a simplifying event node (recall that events and functions are depicted as alternating nodes).

This simplification suggests that both events coincide. If this is not the case, the modeler often emphasizes the need for an extra input (e.g. to indicate the time distance between two functions), by introducing an extra event. This input event has one outgoing and no incoming arc, and thus breaks one of the syntactical rules[1].

When trying to identify non-controllable transitions in the WF-net, these extra events can be used as pointer, as they indicate non-controllable transitions of type *event*. This kind of pointer was illustrated in Figure 8.4 a) also using an example from an earlier chapter.

Non-controllable transitions of type *decision* are introduced in the WF-net if a decision based on external information was modeled in the EPC. Within EPCs, decisions are modeled with the help of an XOR-connector. A further hint to detect decisions based on external information is given through the inscriptions of the function preceding and the events following the XOR connector. Whereas the function is denoted with notions such as “test”or “check”, the succeeding events

[1]Remember that only start events have no incoming arc

often refer to the decision criteria (e.g. $> amount$). A pointer of this kind was illustrated in Figure 8.4 b).

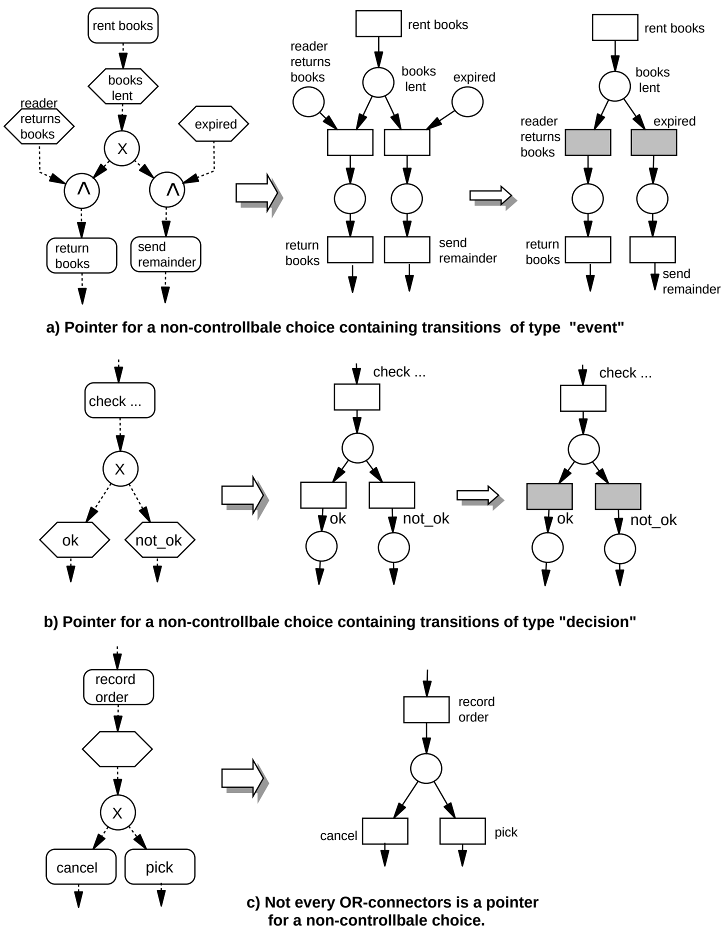


Figure 8.4: Pointers that can be used to discover non-controllable transitions

Figure 8.4 c) gives an example of an XOR-connector which is not transformed into a non-controllable choice. The XOR-connector models the decision between the functions `pick` and `cancel`. The transitions derived through the EPC-PN transformation are of type *task*. Hence, the choice is controllable.

The order of the EPC-PN transformation was the following: 1. Mapping EPC elements to Petri net-modules, 2. Module combination, and 3. Adding unique input/output places.

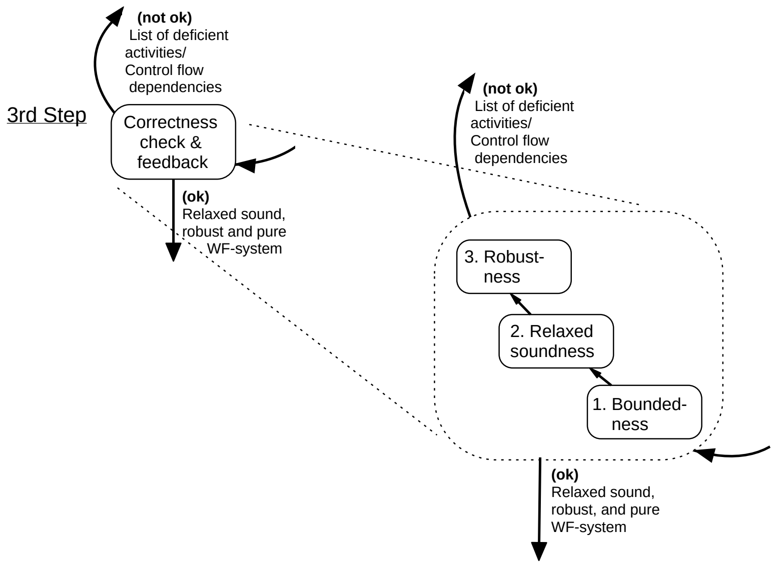
The described step, identifying the non-controllable transitions, must be inserted between steps two and three. A benefit of the additional step is that the resulting net may be smaller, as places that must be merged in the last step will be reduced.

Assume that the non-controllable transitions were indicated. Then the algorithm checking robustness is applied. If the algorithm aborts with the result “not robust”, then there are non-controllable transitions which may inhibit proper termination. The deficient transitions are notified to the modeler who has to revise the corresponding elements within the primary specification.

Refinement of the 3rd step

Applying the different tests, the EPC is revised until the corresponding WF-system satisfies the desired properties. As a result of the first three steps we obtain an EPC describing reasonable behavior as well as a corresponding relaxed sound and robust WF-system.

Figure 8.5 shows a high-resolution version of the 3rd step “Correctness Check and Feedback”.



(not ok) List of deficient activities/ Control flow dependencies
3rd Step
Correctness check & feedback
(ok) Relaxed sound, robust and pure WF-system
(not ok) List of deficient activities/ Control flow dependencies
3. Robustness
2. Relaxed soundness
1. Boundedness
(ok) Relaxed sound, robust, and pure WF-system

Figure 8.5: Refinement of the 3rd step: “Correctness check and feedback”

On the basis of the WF-system, the analysis can be extended in any desired way by applying a variety of results of the Petri net theory. Furthermore, the possibility to execute the WF-net can be used for simulation of the process in order to detect optimization potential.

Running example

Boundedness The resulting WF-system (cf. Figure 8.3) is bounded. It is even safe, because no place can have more than one token.

Relaxed soundness The resulting WF-system is relaxed sound. A set of sound firing sequences which contains all transitions is:

1. `AND_split, record_order, pick, wrap, check_credit, ok, deliver, arrange_payment, AND_accept, archive` and
2. `AND_split, check_credit, not_ok, notify_cancel, record_order, cancel, AND_cancel, archive`.

Following the proof of Theorem 4.12 from Chapter 4 we could conclude that if the WF-net was also well-structured, it was sound. But the WF-net contains a TP-handle, as highlighted in Figure 8.6.

The WF-system is not sound. There are firing sequences that deadlock, e.g.

- `AND_split, record_order, pick, wrap, check_credit, not_ok, deliver, notify_cancel`.

Robustness The WF-system is robust. The WF-net derived through the transformation contains one non-controllable choice. It consist of the two transitions (of type *decision*) reflecting the outcome of the task `check_credit`.

The reachability graph of the WF-system is depicted in Figure 8.7. The non-controllable state transitions are marked with a bow. The robust fragment $SG \subseteq RG$ was highlighted by showing the associated states and state transitions in bold.

The existence of the robust fragment SG states that, although the WF-system is not sound, it is possible to control the execution such that, only sound executions are chosen. Robustness states that this is possible independent of the outcome of the moves of the environment (modeled by the non-controllable choices).

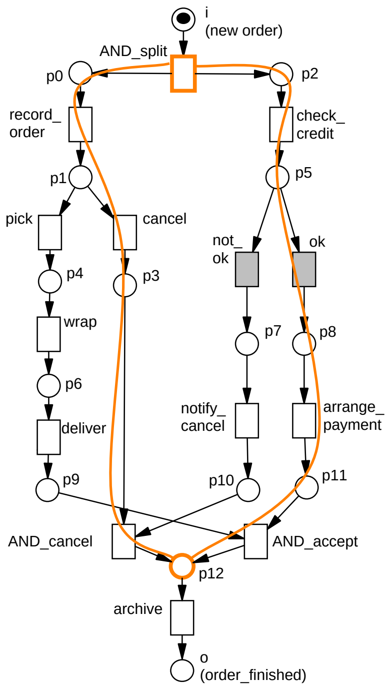


Figure 8.6: A TP-handle within the WF-net “Handling an incoming order”

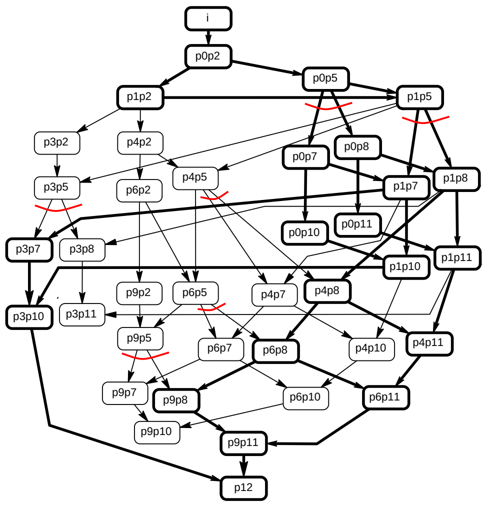


Figure 8.7: Reachability graph and robust fragment

8.2.4 4th step: Strategy determination & implementation

At this point, the specification developed only satisfies some minimum requirements. Relaxed soundness and robustness state that at least all relevant behavior has been described correctly. However, the derived specification may still allow for unsound executions, as the resulting WF-system does not have to be sound.

Within step four of the process model the process description will be augmented so that it becomes sound. The necessary enhancements are determined through the choice of a certain scheduling strategy. The decision in favor of a certain strategy is made by the modeler.

Strategy determination

The behavior of a WF-system which is relaxed sound and robust covers sound and unsound executions. To make the WF-system sound, the set of executions must be restricted to only sound ones.

The user chooses a set of desired executions. This set determines a strategy. The choice for a certain strategy cannot become automated, but depends on expert knowledge. Information that must be considered concerns the costs and duration of activities as well as the occurrence probability of certain external events or decisions.

Strategies can be optimistic or pessimistic. A pessimistic strategy would only support the executions that guarantee proper termination right from the beginning. In contrast, an optimistic strategy would also allow for executions that include the resetting of tasks.

If an optimistic strategy is chosen, the modeler is required to additionally specify possible recovery behavior, which can be added either at EPC or WF-net level. The only requirement is that when adding new tasks it must be guaranteed that the resulting WF-system is again relaxed sound and robust. More details on the determination of an optimistic strategy were provided in Chapter 7.

A certain strategy was successfully determined if the set of chosen executions determines a robust fragment.

Strategy implementation

Implementing a strategy coincides with augmenting the WF-system so that its behavior is restricted to the robust fragment. The elements that are augmented constitute a synchronization pattern - this refers to a set of places together with the corresponding flow relationship.

For the computation of the synchronization pattern we refer to results of Petri net (controller) synthesis [NRT92, CKLY98, CDLX02, GRX02a], which is based on the theory of regions.

The algorithms used for our purpose were implemented in the tools Synet [Cai97] and Petrify [CKK$^+$97]. If possible, we favor the controller synthesis approach described in Section 6.4, as it only enhances the primary process description. The algorithm, which is based on the reachability graph of a pure and bounded WF-system mainly works as follows. First, the set of forbidden state transitions is determined. It consists of all state transitions which leave the robust fragment. On the basis of this information, the algorithm computes the synchronization pattern. Its incorporation into the primary WF-system disables the forbidden state transitions, but does not change the rest of the behavior. A precise description of the algorithm, its theoretical background and prerequisites for its application were given in Chapter 6.

Bringing together the different cases, the step "Strategy determination and implementation" is refined as shown in Figure 8.8. Note, that in the refinement the mod-

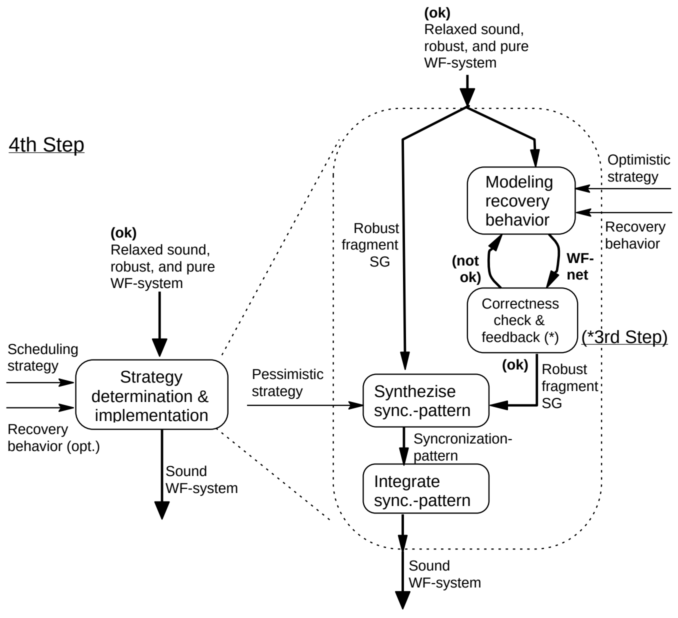


Figure 8.8: Refinement of "Strategy determination and implementation"

eler was assumed to revise the WF-net. If the modeler revises the primary EPC, the 2nd Step “Transformation into WF-nets” should be as well contained in the refinement.

Running example

We will apply the proposed steps to the running example.

Strategy determination: The pessimistic case Let us first assume that it is too expensive to reset any task. Therefore the decision is made for a pessimistic strategy. The corresponding fragment of sound executions was computed by the robustness algorithm and was shown in Figure 8.7.

Strategy implementation In the next step, the set of forbidden state transitions is determined. It coincides with the state transitions that leave the fragment. The set was illustrated in Figure 8.9(a), where it corresponds to the set of labeled state transitions. All forbidden state transitions in this example are either labeled with task `pick` or with task `cancel`.

Applying the algorithm described in Section 6.4 the computed synchronization pattern consists of two places, `pc1` and `pc2`. Incorporating them into the primary WF-net the occurrence of tasks $pick$ and $cancel$ in all indicated states will be prevented. The enhanced WF-net is shown in Figure 8.9(b). The derived WF-system is sound. Through introduction of the synchronization pattern, both processes became synchronized at the decision points.

Using this process description as basis for the workflow controller, the execution of the process becomes serialized. The customer check would always be executed before the ordering. Here, all sound but parallel firing sequences of the relaxed sound WF-system have been eliminated.

If the delivery process took a long time and the customer check took a long time, this would be very inefficient. This is especially undesirable if it is very rare that a customer check results in a `not_ok`. Such pessimistic scheduling would be annoying. It would be more efficient just to start the delivery of the order to the customer hoping the customer check will be `ok`, i.e. following an optimistic approach. Only in the rare case that `not_ok` was reported, should the order be returned to stock and canceled after all.

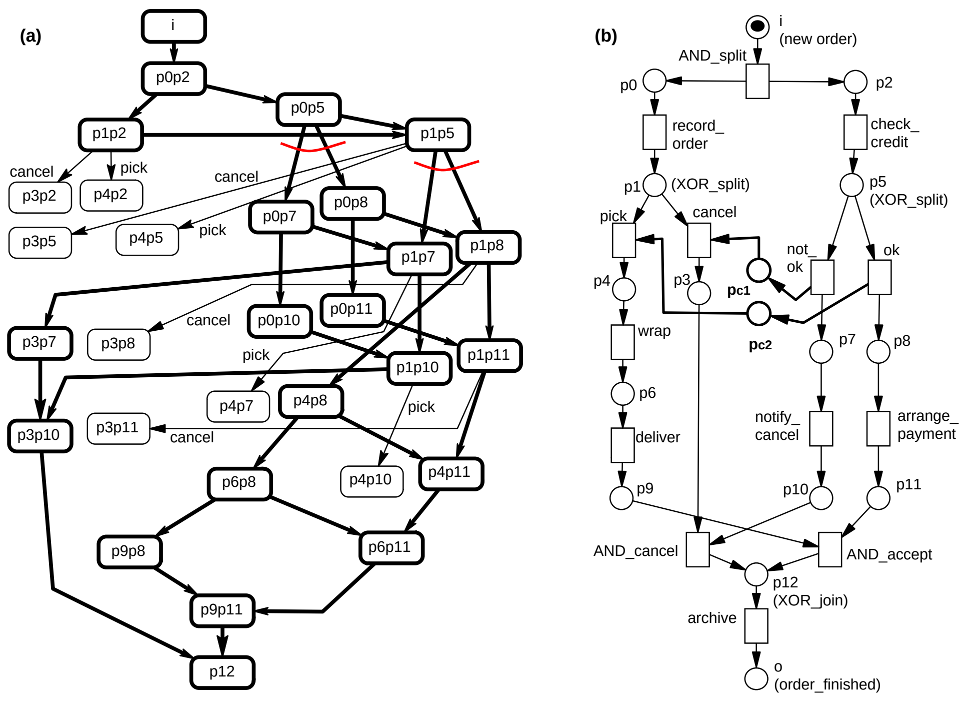


Figure 8.9: (a) Fragment with leaving state transitions,
(b) Computed sound WF-system

Strategy determination: The optimistic case If the decision was made in favor of an optimistic scheduling strategy, one additional intermediate step, “Modeling recovery behavior”, has to be performed beforehand (cf. Figure 8.8). The rest of the procedure then stays the same.

As described in Chapter 7, some additional information is necessary to add the recovery behavior. For the running example 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 compensation tasks are `return` and `unwrap`. After the item has been returned to stock, the instance should be canceled. Task `deliver` is considered to be nonreversible.

The enhanced EPC, incorporating the recovery behavior, as well as the corresponding WF-net are shown in Figure 8.10. Notice that the integrated tasks only show one possible way of modeling the recovery behavior.

The resulting WF-system is again relaxed sound and robust. The fragment that was computed, applying the robustness algorithm (cf. Chapter 5), is shown in Figure 8.11(a). All sound firing sequences of the relaxed sound WF-system are maintained. Furthermore, some additional, less efficient executions are accepted too.

Strategy implementation The synchronization-pattern is computed on the basis of the robust fragment. Finally, the synchronization pattern is added to the primary WF-net. The resulting WF-system looks as shown in Figure 8.11(b). The resulting process description is sound. There are no firing sequences that deadlock or do not terminate properly. Furthermore, none of the transitions are dead.

Using the optimistic process description as the basis for the execution support through a WFMS, the two departments can operate in parallel. The customer check by the accounts department (`check_credit`) can be executed concurrently with preparative tasks of the sales handling (`pick` and `wrap`).

8.2.5 5th step: Control refinement

A workflow system is a reactive system, so it runs in parallel with its environment and tries to enforce certain desirable effects in the environment. To be more precise, the workflow controller monitors tasks performed by actors (people or software). Monitoring comprises initiating tasks by assigning them to certain actors and waiting for the task be completed. The role of the process description is to indicate which tasks must be activated at a certain point in time.

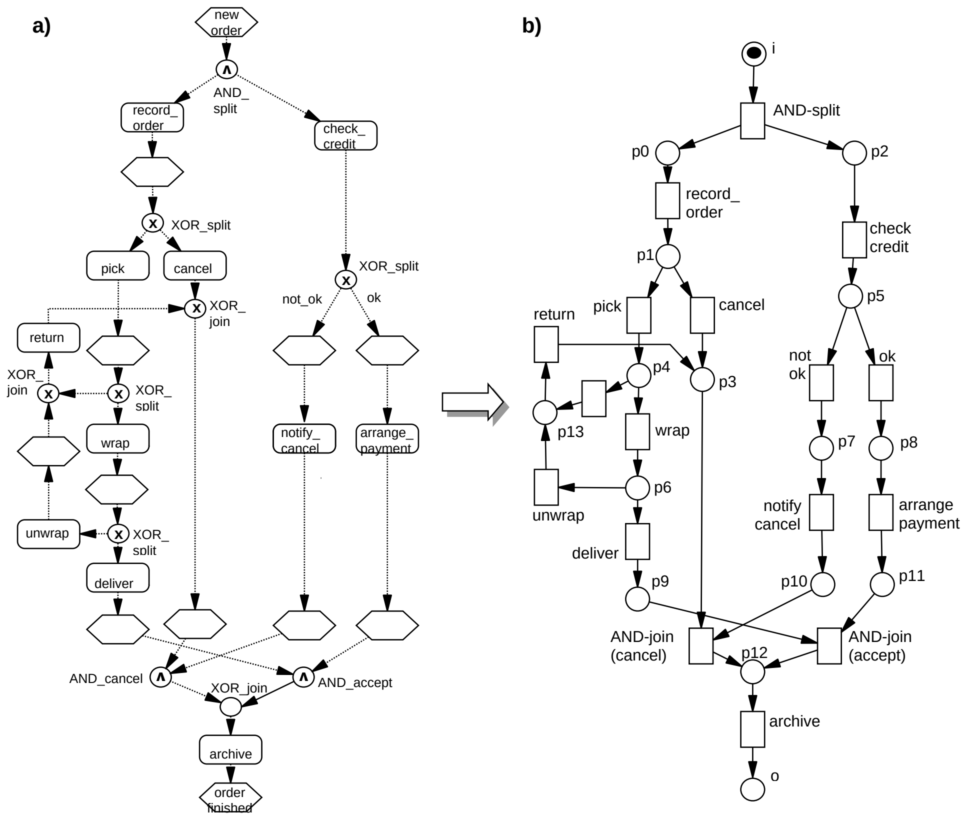


Figure 8.10: Process descriptions incorporating possible recovery behavior (a) EPC (b) WF-net

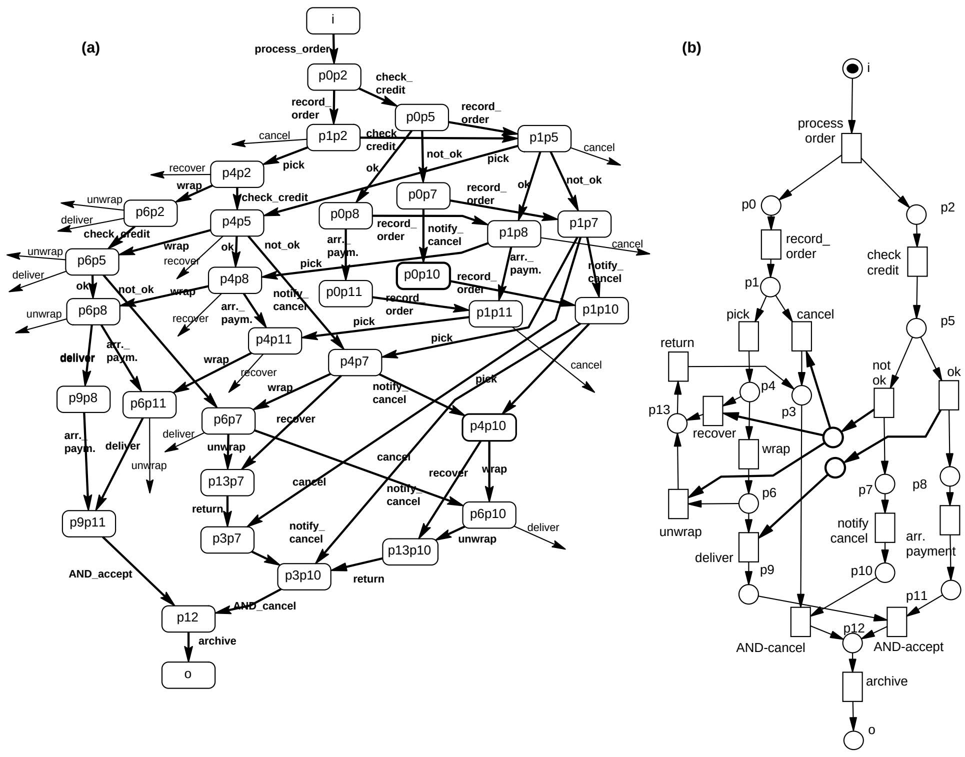


Figure 8.11: (a) Computed fragment $SG \subset RG$ with leaving state transitions, (b) sound WF-system

The process description developed so far does not fit these requirements entirely. The reason for that is twofold. It is based on the chosen perspective and on the instantaneous firing of transitions.

Perspective The process was modeled from a user point of view. So, activities have been modeled by functions, which are transformed into transitions of type *task*. Transitions are the active part within a WF-system. This contradicts the perspective of the workflow management system. The system does not execute an activity but merely initiates and monitors its processing.

Instantaneous firing Transitions fire instantaneous. This does not match with the requirement to model activities (performed by external actors) as time consuming entities. To fit these requirements the perspective of the process description must be changed towards monitoring.

Refinement of tasks-transitions

Activities have been modeled by transitions of type *task*. To adapt to the reactive setting, task transitions will be refined. Reflecting the embedding of activities within tasks, transitions of type *task* are depicted as a sequence of transitions `allocate_task`, `begin_activity`, `end_activity`, and `record_task_completion`. Figure 8.12 illustrates the described transition refinement. The transition `allocate_task` models the allocation of the task to a possible actor. This may either mean that it is “pushed” into someone’s in-basket or that the task is put on a common list, from where it can be “pulled” by any actor. The precise implementation depends on the mode of the workflow management system.

The transition `begin_activity` and `end_activity` are of type *event* as they model the external events indicating that an actor started or completed the embedded activity.

The actual processing of the task starts with transition `begin_task` and ends with `end_task`. These transition are of type *event* and hence designated as *non-controllable*. They model the external events indicating that an actor started or completed the corresponding task. The processing itself is modeled via a place. This way the instantaneous firing of transitions can be retained but now matching an acceptable abstraction. The duration implicitly assigned to the execution of a *task*-transition in a WF-net is now assigned to a place in the refined WF-net.

If the primary *task*-transition is in conflict with some other controllable transition

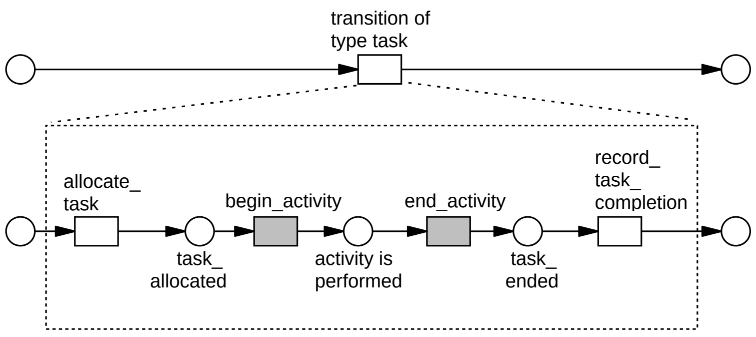


Figure 8.12: Transition refinement

and there is a marking enabling both, the choice for one of them is assumed to be non-deterministic.

Refining the transitions as proposed, soundness of the refined WF-system is maintained[2]. Note, the proposed refinement is quite basic. Requiring progress, a task is assumed to complete. A more elaborate refinement could allow a task to fail or to become withdrawn, while it is processed. Such a more elaborate refinement would encompass the need for (automatic) failure handling, in order to guarantee transactional properties, such as failure atomicity. Nevertheless, we incorporate the possibility to compensate tasks in order to recover from deficient states, e.g. deadlocks. Still, we do not provide any automatic recovery behavior, but require the modeler to explicitly indicate tasks that can be compensated and to specify the corresponding compensation task (cf. Chapter 7). Within workflow management this procedure is reasonable. Obviously, not every tasks should be automatically considered to be reversible. Referring to the running example, task `deliver` may be regarded as non-reversible, as it is unlikely (and unreasonably expensive) to get the item back unbroken. Therefore, the proposed procedure supports the requirement to decide individually on compensation.

Running example

Figure 8.13 shows the result, refining the WF-net of Figure 8.9(b). For simplicity, a short-cut notation was chosen depicting *task*-transitions as transitions subdivided into three sections: start and end white, middle grey.

[2]Refining a transition by a sequence of transitions is one of the basic operations proven to preserve properties of liveness, and boundedness, cf. [Mur89] and therefore soundness [Aal97].

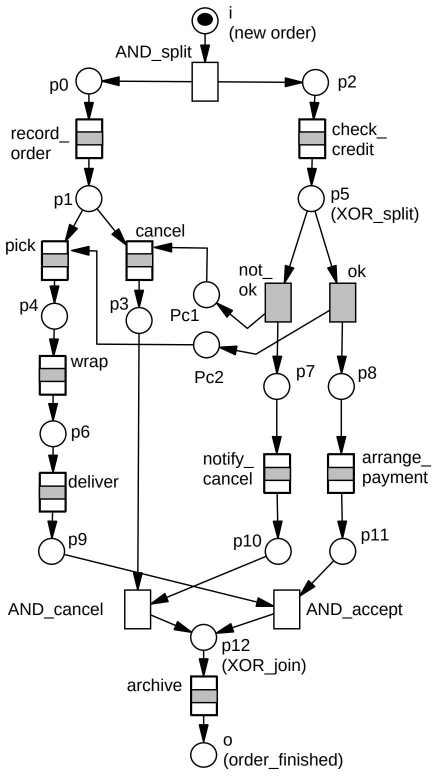


Figure 8.13: The refined WF-net

8.3 Application of results

In this chapter, a comprehensive process model for the specification of workflow was proposed. It leads from a semiformal description of the business process to a sound workflow specification.

The procedure starts with the modeling of business processes from a user perspective, using an accepted semiformal modeling technique. A special feature of the approach are the weak requirements on the first process description. In several steps including analysis, revision, and automatic refinement, the primary process description is transformed into a sound process description that may serve as input for a WFMS[3].

[3]The proposed process model covers one general issue within workflow management, namely the specification of the control-flow aspects of a process. It is clear that if the process description will

Note, for the use of a Petri net based process description as input format, one obstacle remains to be handled: the May-firing rule. In standard Petri net semantics an enabled transition may be deferred indefinitely. Clearly, this does not match realistic behavior of workflow management systems, where transitions with the initiative on the controller side (cf. Table 5.1), should not be delayed but fire immediately. In continuative work [ED03] WF-nets have been adapted to the reactive setting applying reactive semantics. It has been shown that soundness is preserved under certain conditions.

Still, most of the existing workflow management systems do not support Petri nets as input for the workflow controller. Exceptions are workflow management systems, such as COSA (Software Ley/COSA Solutions [SL99]) or Income (Get Process AG [Inc]). Most other workflow management systems, e.g. Action Workflow, or Staffware, use proprietary workflow languages. In order to apply the result of the proposed process model, it remains necessary to support the transformation of the derived process description into the supported file-format. This is a vendor dependent task and was hence beyond scope of the proposed process model.

Still, to provide the reader with an idea of the points which need to be clarified on a case by case basis in the course of the development for practical applications, we again refer to Staffware. We will not propose precise transformation rules translating WF-nets into Staffware descriptions but only give an impression using the running example. The following versions of the process "Handling an incoming order" are expressed in the Staffware language. As in Chapter 3, we again use the Audit Trail (AT) to observe interesting cases. The examples will show that process descriptions can be transferred without expenditure, matching the expected behavior.

Figure 8.14 shows the relaxed sound version of the process "Handling an incoming order". The choices are modeled via two nodes of type condition. The upper choice depends on the result of the task `check_credit`. The lower choice is made independently by the user.

The primary process description (cf. Figure 8.3) was only relaxed sound. The following execution led to a deadlock.

- `AND_split, record_order, check_credit, not_ok, notify_cancel, pick, wrap, deliver`

Expectedly, the corresponding execution within Staffware also deadlocks. The cor-

be used as input for a WFMS further aspects (e.g. determination and assignment of possible actors, definition of the data-flow) must be incorporated.

responding audit trail is shown in Figure 8.15. Note, Staffware does not detect the deadlock, i.e. there is no message “Case does not terminate” or the like. Instead, the system expects the case to complete.

The pessimistic case The primary, only relaxed sound WF-net was enhanced by a synchronization pattern. The result of the incorporation was a sound WF-system. Implementing the pessimistic strategy restricted the set of sound executions. Sound but parallel executions were eliminated. Within Staffware, synchronization is modeled via *wait*-steps. The Staffware description that corresponds to the pessimistic process description of Figure 8.9(b) is shown in Figure 8.16.

The introduced *wait*-step guarantees that the order is only processed (picked, wrapped, and delivered) if the check result was positive. The evaluation of the lower condition refers to the result of the task `check_credit`. This is realized with the help of an internal variable, which is set by task `check_credit` and read by the conditions.

The optimistic case The Staffware description that corresponds to the implementation of an optimistic strategy is shown in Figure 8.17. Within the optimistic strategy the synchronization between the two departments is only enforced after task `wrap`. In the Staffware description this is expressed by the *wait*-step incorporated before task `deliver`. The *conditions* following *steps* `record_order` and `pick_order` refer to the result of the task `credit_check`. If the check is positive, or has not yet been performed, the optimistic processing proceeds. If the check result is negative, processed steps are compensated.

Figure 8.18 reflects an audit, in which the task `check_credit` was executed only after task `pick` and `wrap` were already executed. The result of the check was negative. Tasks `pick` and `wrap` are reset by means of executing the compensation tasks `return` and `unwrap`. The final report is “Case terminated normally”.

In the following, we will consider briefly an extra feature of Staffware, namely the possibility to withdraw a task while it is being executed. The withdraw relation is modeled via an arc that enters the according task from the top. Within the next process description (cf. Figure 8.19) this feature was used to withdraw tasks `pick` and `wrap` as soon as the check result turns out to be negative. The new *conditions* test whether tasks `wrap` and `pick` have already been processed. This way, the point in time is determined which makes it possible to withdraw either of these tasks.

Figure 8.20 reflects an audit in which the outcome of task `check_credit` was negative. This result was computed, while task `wrap` has been processed by user *swanne*. Task `wrap` is withdrawn and task `pick` is reset. The final report is “Case terminated normally”.

From this audit it can be observed that in Staffware it is not distinguished whether a task is only allocated or already processed. The next visible state after allocating the task to an actor is the completion of the task. Therefore a withdraw, makes no difference between removing a task from the in-basket of some actor and the interruption of a perhaps almost finished task execution. This involves at least two disadvantages. First, it is not possible to compute an average processing time for a task, because the actual start of the processing is not monitored. Second, it ignores that tasks that are almost completed may have to be compensated. To remedy these problems a task refinement as proposed in the last step of the process model (cf. Section 8.2.5) should be supported.

By means of Staffware as an example, it was shown that the resulting sound process descriptions can be transferred to process descriptions supported by existing WFMSs.

8.4 Related work

There are many general publications about workflow management, e.g. [GHS95, JB96, Law97, Fis01, AH02a, Mar02]. In the literature the scope of the domain is delimited, terminology is standardized to some extent, the requirements are characterized and state of the art of the existing solutions is outlined. The authors indicate the close relation between workflow and business processes (e.g. [AH02a] requiring the re-use of existing business process descriptions. As a consequence thereof the need for different levels of abstraction within the modeling of processes (e.g. [GHS95]) is emphasized. However, in existing approaches the different levels of abstraction are either mixed or only supported through nesting of tasks.

The major contribution of this thesis is it to provide a process model that bridges the gap between different levels of abstraction proposing a cross-language procedure which first-time uses a refinement concept based on the correctness measures.

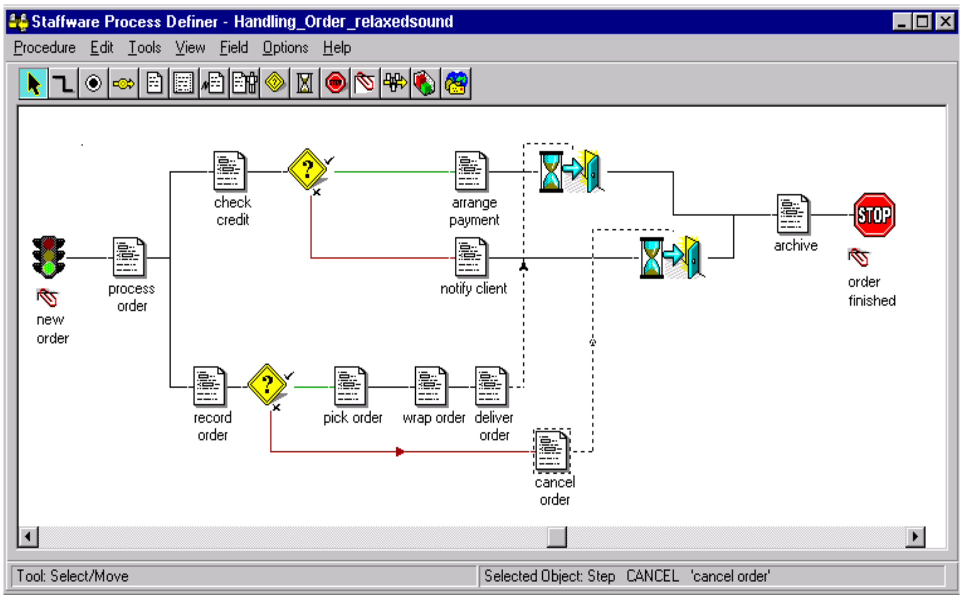
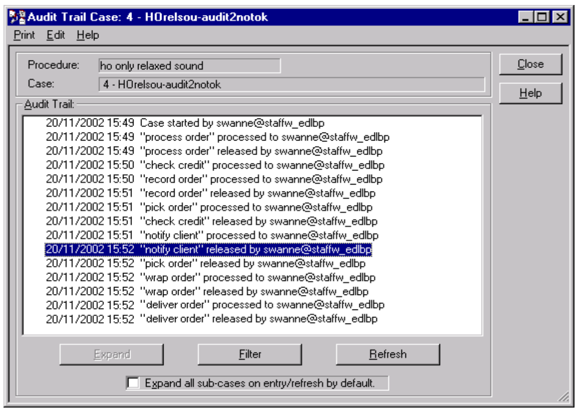


Figure 8.14: Staffware description “Handling an incoming order” (relaxed sound)



Procedure: ho only relaxed sound
Case: 4 - HOrelsou-audit2notok

Audit Trail:

```
20/11/2002 15:49  Case started by swanne@staffw_edlbp
20/11/2002 15:49  "process order" processed to swanne@staffw_edlbp
20/11/2002 15:49  "process order" released by swanne@staffw_edlbp
20/11/2002 15:50  "check credit" processed to swanne@staffw_edlbp
20/11/2002 15:50  "record order" processed to swanne@staffw_edlbp
20/11/2002 15:51  "record order" released by swanne@staffw_edlbp
20/11/2002 15:51  "pick order" processed to swanne@staffw_edlbp
20/11/2002 15:51  "check credit" released by swanne@staffw_edlbp
20/11/2002 15:51  "notify client" processed to swanne@staffw_edlbp
20/11/2002 15:52  "notify client" released by swanne@staffw_edlbp
20/11/2002 15:52  "pick order" released by swanne@staffw_edlbp
20/11/2002 15:52  "wrap order" processed to swanne@staffw_edlbp
20/11/2002 15:52  "wrap order" released by swanne@staffw_edlbp
20/11/2002 15:52  "deliver order" processed to swanne@staffw_edlbp
20/11/2002 15:52  "deliver order" released by swanne@staffw_edlbp
```

☐ Expand all sub-cases on entry/refresh by default.

Figure 8.15: Audit result of an unsound execution (deadlock)

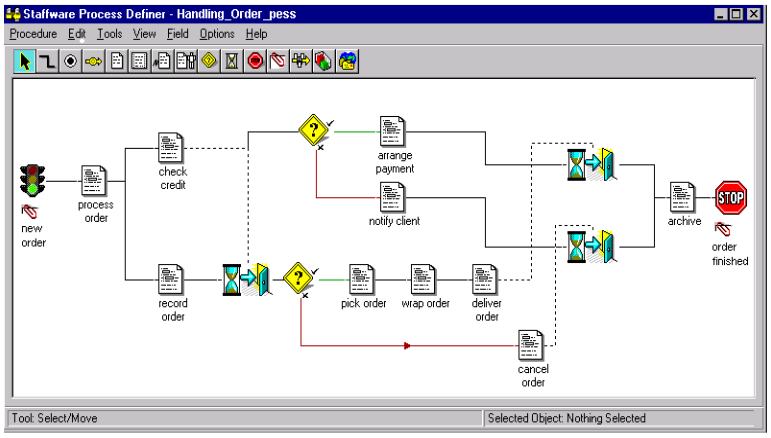


Figure 8.16: Staffware description “Handling an incoming order” (sound) (pessimistic strategy)

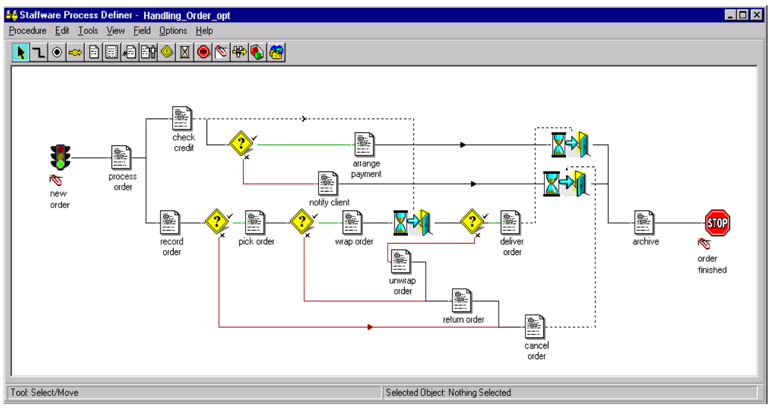
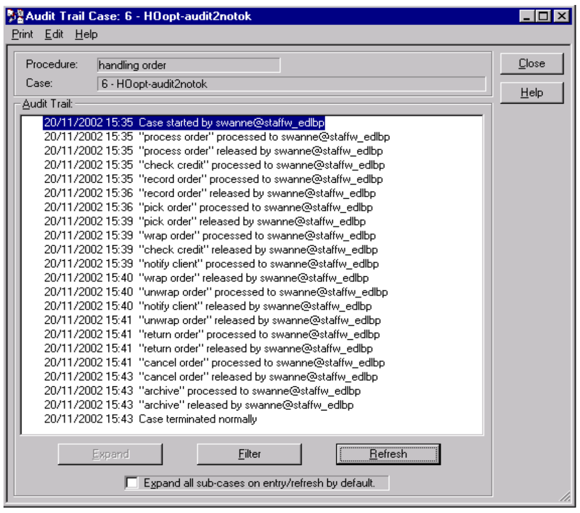


Figure 8.17: Staffware description “Handling an incoming order” (sound) (optimistic strategy)



Procedure: handling order
Case: 6 - HOopt-audit2notok

Audit Trail:

```
20/11/2002 15:35  Case started by swanne@staffw_edlbp
20/11/2002 15:35  "process order" processed to swanne@staffw_edlbp
20/11/2002 15:35  "process order" released by swanne@staffw_edlbp
20/11/2002 15:35  "check credit" processed to swanne@staffw_edlbp
20/11/2002 15:35  "record order" processed to swanne@staffw_edlbp
20/11/2002 15:36  "record order" released by swanne@staffw_edlbp
20/11/2002 15:36  "pick order" processed to swanne@staffw_edlbp
20/11/2002 15:39  "pick order" released by swanne@staffw_edlbp
20/11/2002 15:39  "wrap order" processed to swanne@staffw_edlbp
20/11/2002 15:39  "check credit" released by swanne@staffw_edlbp
20/11/2002 15:39  "notify client" processed to swanne@staffw_edlbp
20/11/2002 15:40  "wrap order" released by swanne@staffw_edlbp
20/11/2002 15:40  "unwrap order" processed to swanne@staffw_edlbp
20/11/2002 15:40  "notify client" released by swanne@staffw_edlbp
20/11/2002 15:41  "unwrap order" released by swanne@staffw_edlbp
20/11/2002 15:41  "return order" processed to swanne@staffw_edlbp
20/11/2002 15:41  "return order" released by swanne@staffw_edlbp
20/11/2002 15:41  "cancel order" processed to swanne@staffw_edlbp
20/11/2002 15:43  "cancel order" released by swanne@staffw_edlbp
20/11/2002 15:43  "archive" processed to swanne@staffw_edlbp
20/11/2002 15:43  "archive" released by swanne@staffw_edlbp
20/11/2002 15:43  Case terminated normally
```

[ ] Expand all sub-cases on entry/refresh by default.

Figure 8.18: Audit of the optimistic Staffware description

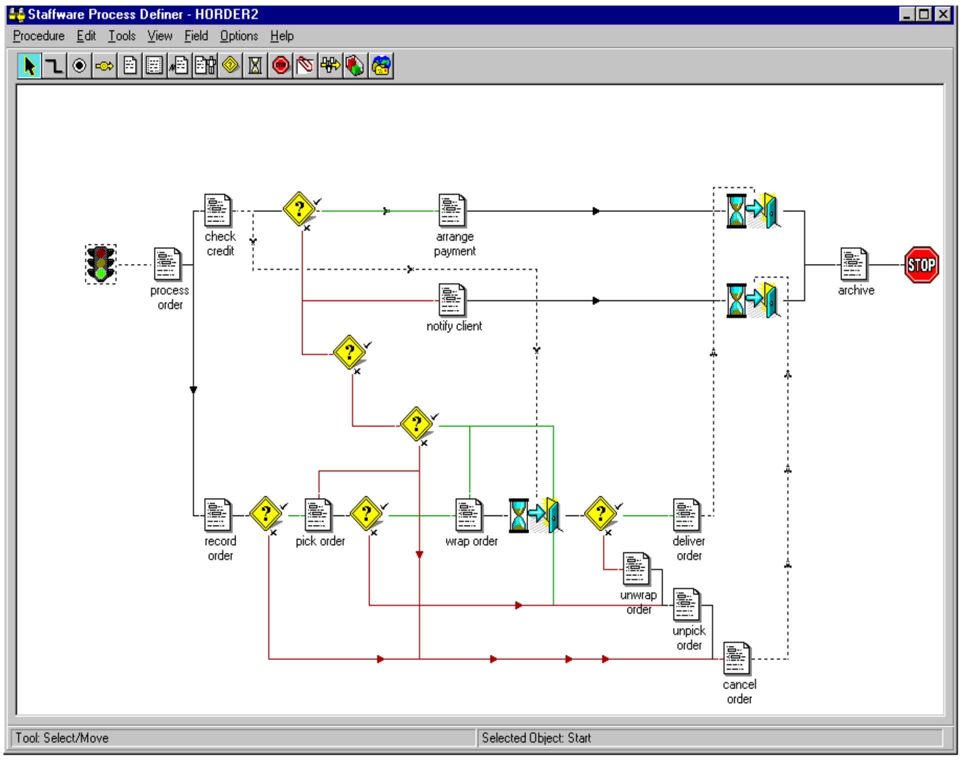
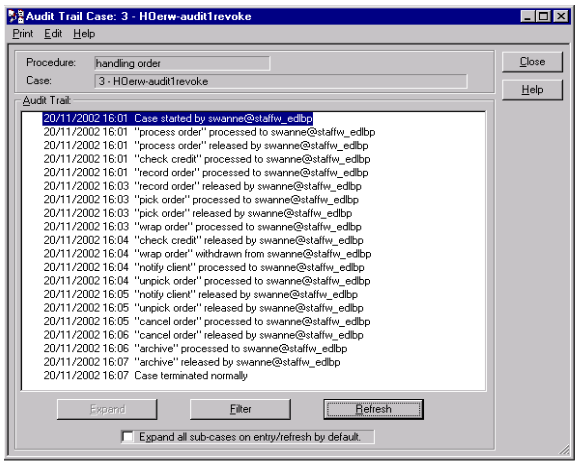


Figure 8.19: Enhanced version of the optimistic Staffware description



Procedure: handling order
Case: 3 - HOerw-audit1revoke
Audit Trail:

```
20/11/2002 16:01  Case started by swanne@staffw_edlbp
20/11/2002 16:01  "process order" processed to swanne@staffw_edlbp
20/11/2002 16:01  "process order" released by swanne@staffw_edlbp
20/11/2002 16:01  "check credit" processed to swanne@staffw_edlbp
20/11/2002 16:01  "record order" processed to swanne@staffw_edlbp
20/11/2002 16:03  "record order" released by swanne@staffw_edlbp
20/11/2002 16:03  "pick order" processed to swanne@staffw_edlbp
20/11/2002 16:03  "pick order" released by swanne@staffw_edlbp
20/11/2002 16:03  "wrap order" processed to swanne@staffw_edlbp
20/11/2002 16:04  "check credit" released by swanne@staffw_edlbp
20/11/2002 16:04  "wrap order" withdrawn from swanne@staffw_edlbp
20/11/2002 16:04  "notify client" processed to swanne@staffw_edlbp
20/11/2002 16:04  "unpick order" processed to swanne@staffw_edlbp
20/11/2002 16:05  "notify client" released by swanne@staffw_edlbp
20/11/2002 16:05  "unpick order" released by swanne@staffw_edlbp
20/11/2002 16:05  "cancel order" processed to swanne@staffw_edlbp
20/11/2002 16:06  "cancel order" released by swanne@staffw_edlbp
20/11/2002 16:06  "archive" processed to swanne@staffw_edlbp
20/11/2002 16:07  "archive" released by swanne@staffw_edlbp
20/11/2002 16:07  Case terminated normally
```

☐ Expand all sub-cases on entry/refresh by default.

Figure 8.20: Audit of the enhanced Staffware description

Chapter 9

Conclusion and future work

9.1 Conclusion

The goal of this thesis has been to provide a methodically well-founded process model for the modeling of functional workflow requirements. The process model should support and guide the modeler from a semiformal description of business processes towards sound workflow specifications, i.e. helping to bridge the gap between business process modeling and workflow specification.

We propose a process model that is based on a combination of different modeling languages. A semiformal modeling language is used as interface to the domain expert. As a prominent example, widely accepted in practice, we referred to Event-driven Process Chains (EPCs). For the definition of the workflow specification, we have used Petri nets, namely WF-nets. The strength of Petri-nets is their formal foundation and the rich background of theory and tools, which enables profound analysis.

The proposed approach acknowledges the need to describe business processes at different levels of abstraction and combines the advantages of different modeling languages that proved to fit the respective requirements.

It is clear that such a cross-language process model must direct particular attention to a smooth transformation between the techniques used. This was achieved by providing a set of rules transforming the semiformal process descriptions based on EPCs into WF-nets. The proposed transformation does not restrict the modeling facilities of the primary technique and maintains the various interpretations, making them explicit.

The key concept for the proposed process model is the use of pragmatic correctness criteria, namely *relaxed soundness* and *robustness.* They fit the correctness requirements within this first abstraction level and make it possible to provide precise feedback to the modeler.

The resulting process description cannot yet be used as a basis for the execution support. It may still contain undesired executions. Therefore, it must be refined to fit the requirements of a workflow specification. The proposed process model supports this refinement step, applying methods from Petri net synthesis. A sound WF-system is automatically generated on the basis of a relaxed sound and robust process description.

Only within this step do performance issues become relevant. Information that is incorporated relates to a certain scheduling strategy. In preparation for the refinement, the modeler must determine which of the described executions are to be supported by a WFMS.

The late determination of performance issues is especially desirable as corresponding information (the occurrence probability of a certain failure, costs of failure compensation, or priorities) will often only become available (and may even change) at run-time. Their incorporation towards the end of the proposed process model extends the possibility to reuse modeling results under changing priorities.

The gradual refinement proposed within the process model also covers the smooth transformation from processes descriptions representing a user perspective towards a monitoring perspective. WF-nets are refined to meet the requirements posed by a reactive system. The new property robustness assures that the process can interact robustly with its environment. Finally, tasks are refined in order to reflect the reactive behavior of a WFMS appropriately.

The resulting process description defines the tasks involved and determines their order. The specification is sound. Using the process description as a basis for the execution support during run-time reliable processing can be guaranteed.

9.2 Summary of main contributions

This section summarizes the main contributions. The results benefit the modeling of workflow, i.e. support the development of practical applications. Furthermore Petri net theory was enriched. Both the practical and the theoretical contribution is based on the development of the two new correctness criteria, relaxed soundness and robustness. They do not describe perfect behavior but provide a pragmatic measure for the correctness of process descriptions. The introduction of a less stringent correctness concept makes it possible to specify a process at different levels of abstraction. The process descriptions not only differ with respect to the refinement of tasks but also in the detail of the described behavior. The proposed procedure makes use of this distinction posing only loose requirements for the first modeling.

A further contribution lies in the identification of existing Petri net theory to support the missing refinement step. Applying existing algorithms it becomes possible to shift the process description on a more elaborated level of abstraction meeting stricter correctness criteria.

The contributions of the thesis are summarized in the following list.

- A comprehensive process model has been provided, for the modeling of workflow, focusing on the control flow aspects (cf. Chapter 8).
- The process model supports a semiformal modeling language as interface to the domain expert modeling the business processes (cf. Chapter 3).
- A new alleviated correctness criterion, namely *relaxed soundness*, has been introduced in order to provide an adequate quality measure for the resulting process descriptions (cf. Section 3.4). Its appropriateness was confirmed by a case study (cf. Section 3.6).
- Precise and understandable feedback is provided when relaxed soundness is not met by the process description (cf. Section 3.4).
- Transformation rules have been provided, mapping EPCs onto Petri nets. The transformation does not restrict the EPC modeling facilities. It especially supports the use of the OR-connector and the loose EPC assembling rules (cf. Section 3.2).
- Transformation rules, relaxed soundness check and feedback have been implemented in a supporting framework which is based on XML-notations (cf. Section 3.5).
- Relaxed soundness has been embedded into Petri net theory. Possible relations to existing properties were scrutinized (cf. Chapter 4).
- A further correctness criteria, namely non-controllable choice robustness has been introduced expressing the robust interaction of a WF-system with its environment (cf. Chapter 5).

- An algorithm for the robustness check has been introduced and proved to be correct and complete (cf. Section 5.4).
- Existing results from Petri net synthesis have been applied in order to generate a sound process description (cf. Chapter 6).
- The concepts introduced in the thesis have been used to identify and install different scheduling strategies (cf. Chapter 7).
- The applicability of resulting process descriptions as input for existing WFMSs has been discussed taking Staffware as an example (cf. Section 3.7.1 and Section 8.3).

9.3 Future work

There are several possible extensions of this work. First, the whole procedure could be embedded into a supporting framework. So far, several parts (transformation of EPC into WF-nets, check for relaxed soundness, transformation into a sound WF-net) are only available within separate tools.

Second, the failure feedback for the modeler could be improved. Deficiencies and improvement suggestions could be reflected directly in the primary EPC. Prerequisite for these features is the translation of Petri nets to EPCs. This backward direction is an issue of current research.

Third, we introduced two new correctness criteria, necessary as a first quality measure for process descriptions. In the discussion at the end of Chapter 3, a third property was identified. Within Staffware an accepted process description only needs to terminate, but not necessarily properly. It would be interesting to investigate this property, and possibly embed it into the proposed process model.

Fourth, the focus of the process model could be broadened towards other than the control-flow aspects.

Appendix

In the appendix two artificially constructed WF-nets are investigated. The WF-systems are relaxed sound and robust, but comprise some kind of bizarre behavior which complicates the synthesis of a sound WF-system. So far no realistic process description could be found showing similar behavior. Still, as this behavior cannot be excluded, methods are provided handling these anomalies.

9.4 Synthesis of a WF-net not covering all controllable transitions

In this section we will investigate a relaxed sound and robust WF-system which contains controllable transitions not covered by the robust fragment.

The WF-system $S = (PN, i)$ shown in Figure 9.1 a) was already discussed in Section 5.5. There, it was stated that it probably depicts deficient behavior, as a choice which was considered to be controllable ($t1$ and $t2$), is followed by a non-controllable choice ($t4$ and $t5$) which *must* (in case $t1$ was chosen) revoke the first choice in order to terminate properly.

The WF-system is relaxed sound; all transitions are part of a sound firing sequence. It is furthermore robust; there exist a robust fragment (complete winning strategy for the WF-controller).

The corresponding reachability graph $RG_S(V, E)$ is depicted in Figure 9.1 b). The robust fragment $SG \subset RG_S$ has been shown in bold.

Remarkably, the robust fragment $SG = (V_{SG}, E_{SG})$ does not contain state transitions labeled with $t1$ and $t8$ although they are covered by sound firing sequences. The winning strategy suggests to always choose transition $t2$ and transition $t9$, as otherwise proper termination cannot be guaranteed. Here the set of labels $L = \{t|(M, t, M') \in E_{SG}\}$ of the robust fragment is a proper subset of the transi-

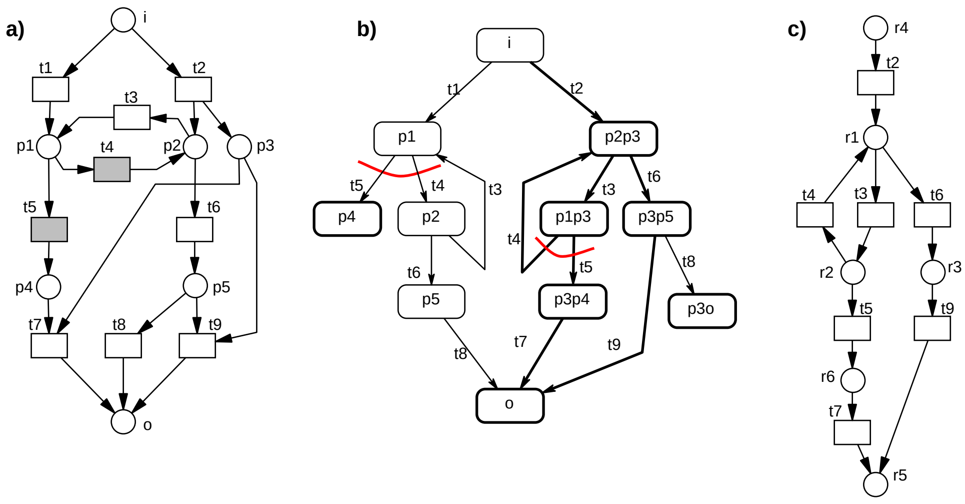


Figure 9.1: A relaxed sound and robust WF-system

tions used in the WF-system: $L \subset T$.

Applying the synthesis algorithm described in [NRT92, CKLY98] to the robust fragment SG a sound WF-system is generated. The result is shown in Figure 9.1 c). Note, as the algorithm introduces a transition for every label of the transition system, the resulting WF-system PN' does not contain transitions $t1$ and $t8$.

9.5 Synthesis of a general Petri net: Example

In this section an example is provided illustrating the synthesis of a general Petri net[1]. Consider the WF-system (PN, i) shown in Figure 9.2. The process description is pure, safe, relaxed sound and robust. The following set of sound firing sequences covers all transitions:

- *init, e, a, c, clean*
- *init, e, b, d, clean*
- *init, f, g, b, d, clean*
- *init, f, a, b, x, clean*

[1]Place/Transition nets with arc weights: $F : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$

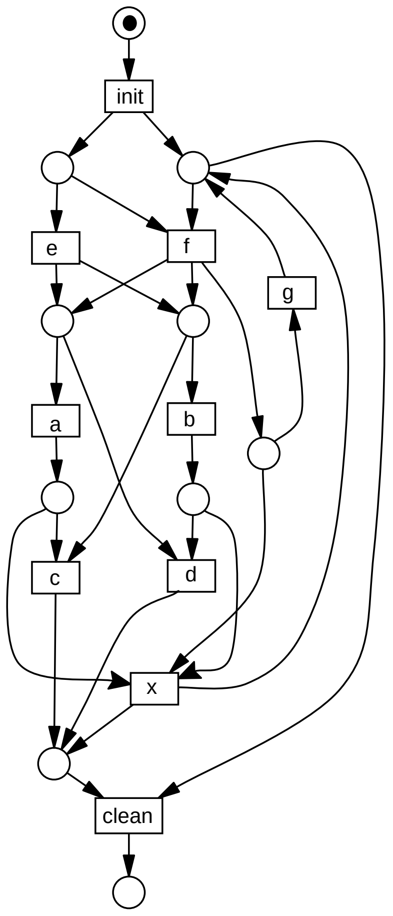


Figure 9.2: A safe, relaxed sound and robust WF-system

Still, the WF-system (PN, i) is not sound, as it may deadlock. A firing sequence that does not terminate properly is e.g.

- *init, e, b, a*

The WF-system is robust. There are not any non-controllable transitions. Therefore relaxed soundness coincides with robustness.

The reachability graph RG_{PN} is an elementary transitions system (shown in Figure 9.3).

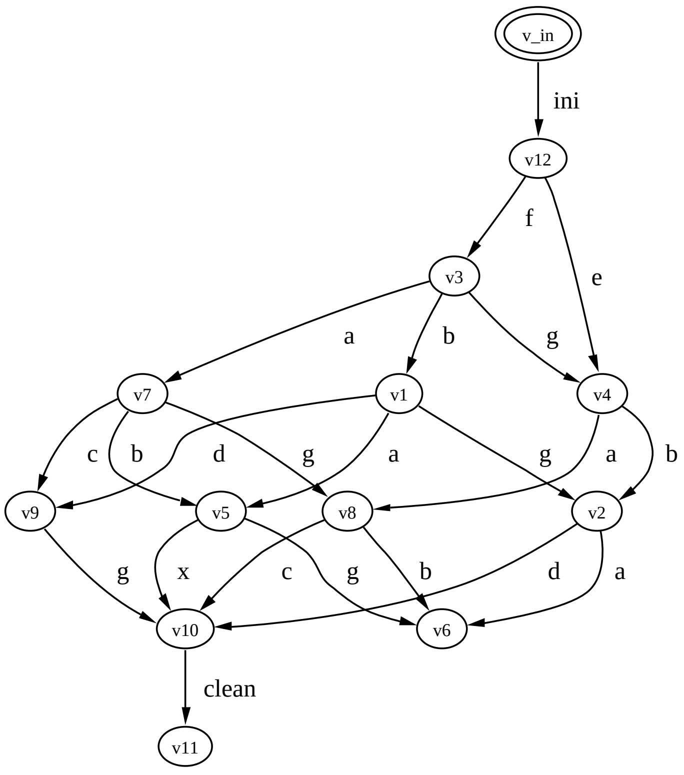


Figure 9.3: The reachability graph RG_{PN}

The robust fragment $SG \subset RG$ computed through the robustness algorithm is shown Figure 9.4. The set of minimum regions is $r_0 = \{v\_in\}, r_1 = \{v12\}, r_2 = \{v1, v2, v3, v4\}, r_3 = \{v8, v4, v3, v7\}, r_4 = \{v1, v3, v5, v7, v9\}, r_5 = \{v9, v10\}$, $r_6 = \{v5, v8, v7\}, r_7 = \{v1, v2, v5\}, r_8 = \{v11\}, r_9 = \{v4, v8, v2, v10\}$.

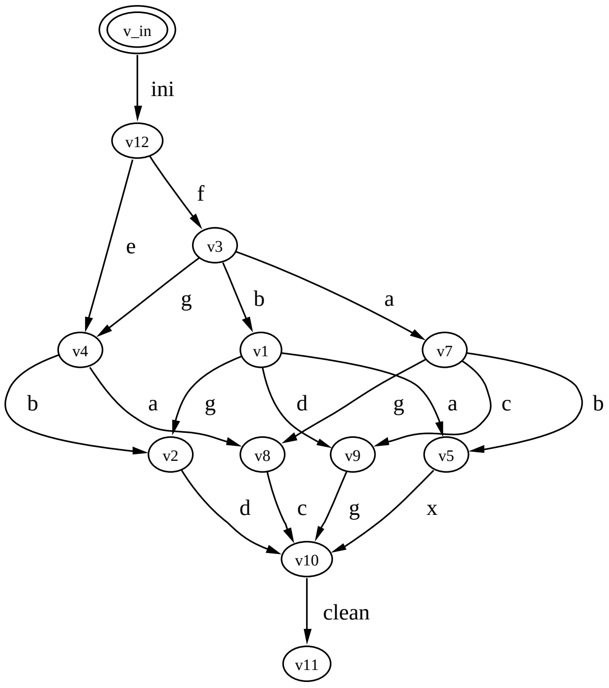


Figure 9.4: The robust fragment SG

Correspondingly, the sets R_{vi} for $i = in, 1..11$ are as follows: $R_{v_{in}} = \{r_0\}$,
$R_{v1} = \{r_2, r_4, r_7\}, R_{v2} = \{r_2, r_7, r_9\}, R_{v3} = \{r_2, r_3, r_4\}, R_{v4} = \{r_2, r_3, r_9\}$,
$R_{v5} = \{r_4, r_6, r_7\}, R_{v7} = \{r_3, r_4, r_6\}, R_{v8} = \{r_3, r_6, r_9\}, R_{v9} = \{r_4, r_5\}$,
$R_{v10} = \{r_5, r_9\}$, and $R_{v11} = \{r_8\}$. The fragment is not elementary. It does not satisfy the event separation axiom (ESA) for elementary transition systems. Consider, for example event $a \in L, \circ a = \{r_2\}$. The region r_2 is contained in R_{v1}, R_{v2} and R_{v4}. Still, $v1$ and $v2$ do not have an exit arc labeled by event c. The

fragment is not excitation closed either. It does not satisfy the excitation closure condition: $\forall l \in L : \bigcap_{r \in \circ l} r = ER(l)$. Consider again the event $a \in L$ with $ER(a) = \{v1, v3, v4\}$. $r_2 = \{v1, v2, v3, v4\} \neq ER(a)$.

This means it is not possible to synthesize a safe WF-net with a reachability graph isomorphic to the fragment.

Still, using the circumvention via transition splitting it is possible to synthesize a safe WF-net with bisimilar behavior. As the fragment SG is not excitation closed, transition splitting is used to derive a split-morphic ECTS. The derived ECTS is shown in Figure 9.5.

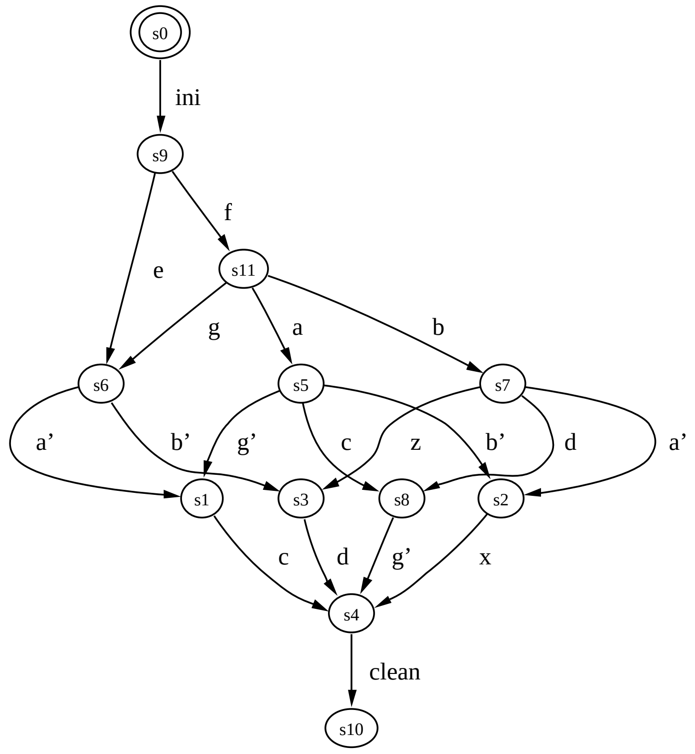


Figure 9.5: A split-morphic ECTS

On the basis of the ECTS, a safe Petri net is synthesized. The reachability graph of the synthesized Petri net is bisimilar to the subgraph SG. The synthesized Petri net is shown in Figure 9.7.

Another possibility is to abandon the safeness property and to use general regions for the synthesis. The subgraph SG satisfies the separation axioms for general transition systems. Therefore a general Place/Transition net can be synthesized.

Applying controller synthesis as proposed in [GRX02b] we first compute the set of separation instances Ω on the basis of the robust fragment of Figure 9.4: $\Omega = \{(8, b), (2, a), (5, g)\}$. In a second step the controller places are synthesized. This is done by solving the equation system determined by the event separation conditions (cf. Equation 6.1), the marking equation lemma (cf. Equation 6.2), and the general property of T-invariants (cf. Equation 6.3). The synchronization pattern $SP = (P_c, T, F_c)$ is determined by the sum of controller places. It is shown in Figure 9.6.

Joining the primary WF-net and the synchronization pattern, the Petri net shown in Figure 9.8 is derived.

The resulting Petri net is not safe. The introduced control-place is marked with two tokens. The resulting net is furthermore no WF-net. With an arc inscription greater than one, it no longer belongs to the class of ordinary Place/Transition nets but to the class of general Place/Transition nets. Still, the resulting Petri net is sound. It may therefore serve as basis to support the execution of the actual process at run-time.

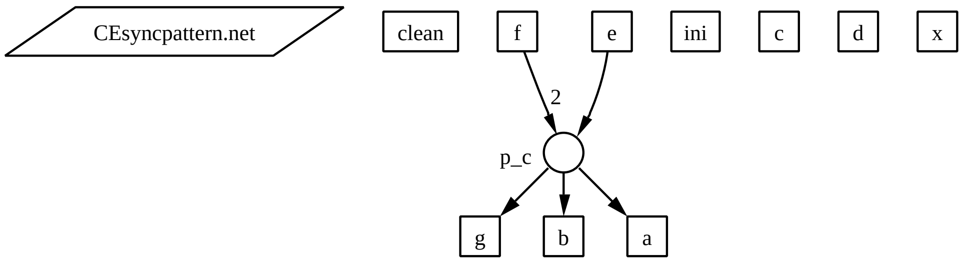


Figure 9.6: Synthesized synchronization pattern

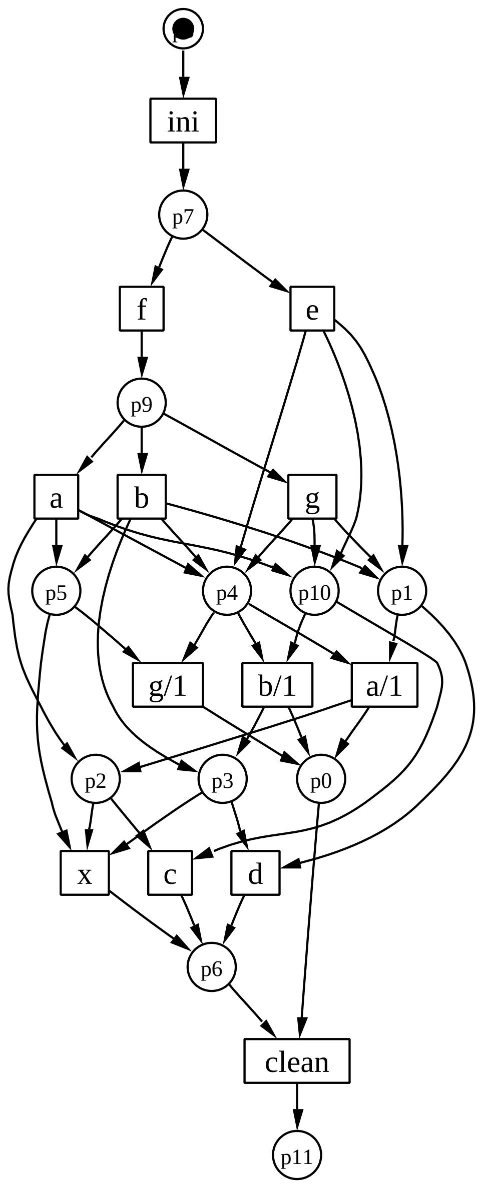


Figure 9.7: Safe WF-net with bisimilar behavior

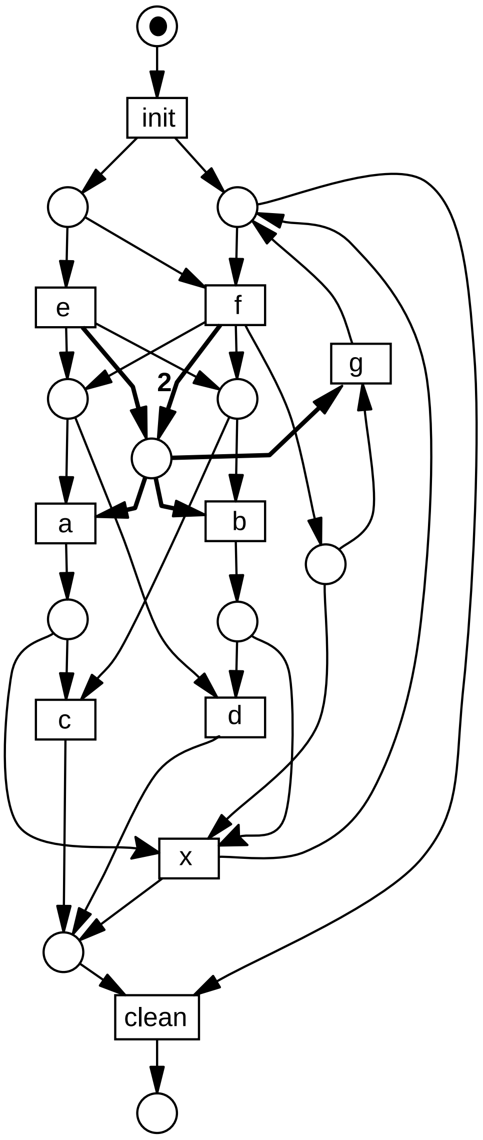


Figure 9.8: Result: A general Petri net

List of figures

1.1 A process model for workflow modeling 14

2.1 A WF-net modeling the processing of complaints 30

2.2 A simple counter-example 31

3.1 EPC elements 36

3.2 EPC: “Handling of goods” 37

3.3 Step 1: Mapping EPC elements onto Petri net-modules 40

3.4 Transformation of the OR-connector 41

3.5 Connecting Petri net-modules 43

3.6 Step 3: Adding new start and sink places 43

3.7 WF-net: “Handling of goods” 44

3.8 EPCs that are transformed into non pure Petri nets 45

3.9 Modeling restriction leads to free-choice WF-nets 46

3.10 Woflan diagnosis for process: “Handling of goods” 48

3.11 Revised EPC 52

3.12 Revised WF-net 53

3.13 WF-net modeling the interaction between a producer and a consumer 55

3.14 (a) Part of the reachability graph (b) Coverability graph 55

3.15 Staffware elements and their semantics 59

3.16 The modeling tool of Staffware 59

3.17 Staffware Audit Trail (I) showing sound execution 61

3.18 Staffware Audit Trail (II) showing sound execution 62
3.19 Audit result of an unsound execution 63
3.20 Simple pattern in Petri nets (up) and EPCs (down) 65
3.21 Multiple Merge 67
3.22 Discriminator 68
3.23 N-out-of-M join 69

4.1 Proof relations 75
4.2 Free-choice transformation rule 76
4.3 A counter-example vitiating the maintenance of liveness 76
4.4 A well-structured net showing relaxed soundness is sound 79
4.5 Relations between different Petri net-properties 79
4.6 Counter-example: T-invariants and relaxed soundness 81
4.7 Counter-example: T-coverability and relaxed soundness 81

5.1 A WF-net modeling the processing of complaints 86
5.2 Part of a library process 88
5.3 Example: “Planning trip” 90
5.4 Example: “Provide change” 91
5.5 RGs for examples: “Planning trip” and “Provide change” 95
5.6 Revised example: “Planning trip” 96
5.7 Revised example: “Provide change” 96
5.8 RGs for revised examples: “Planning trip” and “Provide change” . 97
5.9 Robustness algorithm 99
5.10 Relation of the three criteria 103
5.11 A robust WF-system which is not relaxed sound 104
5.12 A relaxed sound and robust WF-system 104
5.13 Examples of explicit choices 109

6.1 A transition system 113

6.2 A synthesized Petri net 119
6.3 The synthesized Petri net “Planning Trip” 123
6.4 Reachability graph and legal behavior 127
6.5 The synthesized synchronization pattern 129
6.6 Integrating the derived pattern into the primary WF-net 129
6.7 Applying controller synthesis for workflow modeling 131
6.8 WF-net “Planning trip” with integrated synchronization pattern . . 132

7.1 Synthesis example I 140
7.2 Synthesis example II 143
7.3 Synthesis example III 144

8.1 A process model for workflow modeling 150
8.2 Running example: EPC “Handling an incoming order” 154
8.3 Running example: WF-net “Handling an incoming order” 156
8.4 Pointers that can be used to discover non-controllable transitions . 159
8.5 Refinement of the 3rd step: “Correctness check and feedback” . . 160
8.6 Running example: TP-handle 162
8.7 Running example: Reachability graph and robust fragment 162
8.8 Refinement of “Strategy determination and implementation” . . . 164
8.9 Running example: Fragment and synthesized WF-system 166
8.10 Running example: Possible recovery behavior 168
8.11 Running example: Fragment and synthesized WF-system 169
8.12 Transition refinement 171
8.13 Running example: Refined WF-net 172
8.14 Running example: (relaxed sound) Staffware description 176
8.15 Running example: Audit of unsound execution 176
8.16 Running example: (sound) Staffware description 177
8.17 Running example: Optimistic process description 177

8.18 Running example: Audit of sound execution 178
8.19 Running example: Enhanced process description 179
8.20 Running example: Audit of enhanced process description 179

9.1 A relaxed sound and robust WF-system 185
9.2 A safe, relaxed sound and robust WF-system 186
9.3 The reachability graph RG_{PN} 187
9.4 The robust fragment SG 188
9.5 A split-morphic ECTS 189
9.6 Synthesized synchronization pattern 190
9.7 Safe WF-net with bisimilar behavior 191
9.8 Result: A general Petri net 192

Index

$Pred_{RG}(o)$, 24, 99
$Succ_{RG}(M)$, 24, 99

actor, 8, 153, 171

bisimulation, 114
bounded, 28, 32, 46, 77
  structurally bounded, 28
business activity, 7
business process, 7, 36
business process re-engineering, 7
business process description, 8, 36
business process modeling, 36, 153

case, 9, 30, 88
choice
  controllable, 89, 110
  deferred, 109
  explicit, 109, 110
  implicit, 109, 110
  non-controllable, 89, 110
choice consistent, 90
complete strategy, 95
connectors (EPC), 37
control flow aspects, 8
controllable choice, 89
controller synthesis, 135
cycle-free, 20, 32, 46

dead transition, 23, 28
decision, 87
deferred choice, 109
discriminator, 68

elementary transition system (ETS), 116
EPC, 14, 36, 152
  connectors, 37
  event, 37
  function, 37, 171
event (EPC), 37
event effectiveness, 117
event separation axiom (ESA), 116, 118
Event-driven Process Chain, *see* EPC
excitation closure, 117
excitation regions (ER), 117
excitation-closed TS (ECTS), 117
explicit choice, 109
external event, 87

fairness, 89
firing sequence, 22
  finite, 22
  sound, 51, 73
fragment, 93, 95, 122
  robust, 95
free-choice, 21, 46, 75, 79, 89
  transformation, 76
functional aspects, *see* control flow aspects

game, 93
  game graph, 93, 94
  play, 93
  player, 94

winning condition, 93
general transition system, 118

implicit choice, 109
incidence matrix, 20, 25
initiative, 109, 174
isomorphism, 114

live, 47
structurally live, 28
system, 28, 77
transition, 28
LoLA, 54, 57, 158

marked graph, 20, 74
marking, 21
Marking Equation Lemma, 25, 127
multiple choice, 70
multiple merge, 68

N-out-of-M join, 69
net system, 23
non-controllable choice, 89
non-controllable choice robustness, 95, 104

OR-connector, 41, 70
outcome, 109

P-invariant, 27
P/T net, 19, 120, 186
Parikh vector, 25
path
elementary, 19
handles, 19
in nets, 19
Petri net, 19, 30, 120, 186
Petri net controller synthesis, 126
Petri net synthesis, 118
place invariant, *see* P-invariant
Place/Transition net, 120, 186
post-region, 116
postset, 19
pre-region, 116
predecessor, 24, 99
preset, 19
process model, 150
progress, 89
proper termination, 33, 95
PT-handle, 20
pure, 20, 46, 118, 158, 165

reachability graph, 23
fragment, 93, 122
reachable marking, 22, 24
reactive systems, 86
realizable T-invariant, 26
region
post-region, 116
pre-region, 116
elementary, 115
excitation, 117
general, 118
relaxed soundness, 51, 73, 79, 104
robustness, 95, 104
algorithm, 99
routing, 88

S-component, 21
S-coverability, 21
safe, 28, 118, 186, 190
separation instance, 126
siphon, 27
soundness, 32, 33, 47, 72, 79, 104
split-morphism, 113
state machine, 20, 74
state separation axiom (SSA), 116, 118
strategy, 93, 94
complete, 95
maximum, 94
winning, 95

strongly connected, 19
subnet, 21
successor, 24
synchronizing merge, 70
Synet, 127, 165
system, 23
    bounded, 28
    deadlock free, 28
    live, 28

T-component, 21, 26
T-coverable, 21
T-invariant, 26, 81, 127
    cover, 26
    positive, 26
    realizable, 26
    semi-positive, 26
    support, 26
t-simple, 20
task, 9, 87, 153
token, 21
TP-handle, 20
transition
    controllable, 89
    dead, 28
    live, 28
    non-controllable, 89
    type, 88
transition invariant, *see* T-invariant, 81
transition system, 23, 113
    elementary, 116
    excitation-closed, 117
    general, 118
trap, 27

well-formed, 28
well-handled, 20, 47, 73
well-structured, 32, 73, 76, 79
WF-net, 30, 40
    choice consistent, 90
WFMS, 9, 153
winning condition, 95
winning strategy, 93, 95
Woflan, 48, 54, 158
workflow, 8
    controller, 9, 88
    engine, 9, 88
    management, 8
    management system, *see* WFMS
    patterns, 67
    specification, 9
    system, 9, 168
Workflow net, *see* WF-net

Bibliography

[AAEA$^+$96] G. Alonso, D. Agrawal, A. El-Abbadi, M. Kamath, R. Günthör, and C. Mohan. Advanced transaction models in workflow contexts. In *Proc. of the 12th International Conference on Data Engineering*, pages 574–583. IEEE Computer Society, 1996.

[AAH98] N. R. Adam, V. Atluri, and W-K. Huang. Modeling and Analysis of Workflows Using Petri Net. *Journal of Intelligent Information System, Special Issue on Workflow and Process Management*, 10(2):131–158, 1998.

[Aal97] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *18th Int. Conf. on Application and Theory of Petri Nets*, volume 1248 of *LNCS*, pages 407–426. Springer-Verlag, Berlin, 1997.

[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.

[Aal99] W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.

[Aal00a] W.M.P. van der Aalst. Inheritance of Interorganisational Workflows: How to agree to disagree without loosing control? Technical Report CU-CS-899-00, University of Colorado Boulder, 2000.

[Aal00b] 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.

[Aal02] W.M.P. van der Aalst. Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. *Electronic Commerce Research*, 2(3):195–231, 2002.

[AB02] W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.

[AB03] W.M.P. van der Aalst and E. Best, editors. LNCS. Springer, 2003.

[ADK02] W.M.P. van der Aalst, J. Desel, and E. Kindler. On the Semantics of EPCs: A Vicious Circle. In M. Nüttgens and F.J. Rump, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, November 2002. Gesellschaft für Informatik, Bonn.

[AH98] W.M.P. van der Aalst and A.H.M. ter Hofstede. Verification of Workflow Task Structures: A Petri-net-based Approach. Forschungsbericht Nr. 380, Universität Karlsruhe, Institut AIFB, Karlsruhe, 1998.

[AH02a] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems.* MIT press, Cambridge, MA, 2002.

[AH02b] W.M.P. van der Aalst and A.H.M. ter Hofstede. Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages. In K. Jensen, editor, *Proceedings of the Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2002)*, volume 560 of *DAIMI*, pages 1–20, Aarhus, Denmark, 2002.

[AHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(3):5–51, 2003.

[AHT02] W.M.P. van der Aalst, K.M. van Hee, and R.A. van der Toorn. Construction rules for component-based architectures. Computing Science Reports 02/08, Eindhoven University of Technology, Eindhoven, 2002.

[AHV02] W.M.P. van der Aalst, A. Hirnschall, and E. Verbeek. An alternative way to analyze workflow graphs. In *Proc. 13th Int. Conf. Advanced Information Systems Engineering, CAiSE*, volume 2348 of *LNCS*, pages 535–552. Springer, 2002.

[ALW89] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *International Colloquium on Automata, Languages, and Programming*, number 372 in LNCS, Stresa, Italy, 1989. Springer-Verlag.

[AMP94] E. Asarin, O. Maler, and A. Pnueli. Symbolic Controller Synthesis for Discrete and Timed Systems. In *Hybrid Systems*, pages 1–20, 1994.

[ASSR93] P. C. Attie, M. P. Singh, A. P. Sheth, and M. Rusinkiewicz. Specifying and enforcing intertask dependencies. In R. Agrawal, S. Baker, and D. A. Bell, editors, *19th Int. Conf. on Very Large Data Bases*. Morgan Kaufmann, 1993.

[BBD95] E. Badouel, L. Bernadinello, and P. Darondeau. Polynomial algorithms for the synthesis of bounded nets. In *Caap'95*, volume 915 of *LNCS*, pages 647–679, 1995.

[BD96] E. Badouel and P. Darondeau. On the Synthesis of General Petri nets. Technical Report RR-3025, Inria, Institut National de Recherche en Informatique et en Automatique, November 1996.

[BD98] E. Badouel and P. Darondeau. Theory of regions. In Reisig and Rozenberg [RR98a], pages 529–586.

[BDC02] E. Badouel, P. Darondeau, and B. Caillaud. Distributing Finite Automata through Petri Net Synthesis. *Formal Aspects of Computing*, 13:447–470, 2002.

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

[Ber93] L. Bernardinello. Synthesis of Net Systems. In *Application and Theory of Petri Nets*, volume 691 of *LNCS*, pages 89–105. Springer, 1993.

[BL69] J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[BMPV96] L. Bernardinello, D. De Michelis, K. Petruni, and S. Vigna. On synchronic structure of transition systems. In J. Desel, editor, *Structures in Concurrency Theory*, pages 11–31. Springer, 1996.

[BNRL$^+$95] L. Ben-Naoum, R.K.Boel, L.Bongaerts, B. De Schutter, Y. Peng, P. Valkenaers, J.Vandewalle, and V. Wertz. Methodologies for discrete event dynamic systems: A surway. *Journal of the American Society for Horticultural Science*, 36(4), 1995.

[BP98] K. Barkaoui and L. Petrucci. Structural analysis of workflow nets with shared resources. In *Proc. Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98), Lisbon, Portugal, June 1998*, pages 82–95, 1998.

[Cai97] B. Caillaud. Synet : A Tool for the Synthesis of Bounded Petri-Nets, Applications. Technical Report RR-3155, Inria, Institut National de Recherche en Informatique et en Automatique, 1997.

[Cas98] R. Casonato. Gartner group research note 00057684, production-class workflow: A view of the market. http://www.gartner.com, 1998.

[CCPP95] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Conceptual Modeling of Workflows. *LNCS*, 1021:341–354, 1995.

[CDLX02] B. Caillaud, P. Darondeau, L. Lavagno, and X. Xie, editors. *Synthesis and Control of Discrete Event Systems.* Kluwer Academic Press, 2002.

[Chu63] A. Church. Logic, arithmetic and automata. In *Int. Cong. of Mathematicians 1962*, pages 23–35, 1963.

[CKK$^+$97] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, 1997. download from http://www.lsi.upc.es/$\tilde{j}$ordic/petrify/distrib/.

[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.

[Coa00] Workflow Management Coalition. Terminology & Glossary. WfMC-TC-1011, 2000.

[CS94] R. Chen and A. W. Scheer. Modellierung von Prozessketten mittels Petri-Netz Theorie. Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 107, University of Saarland, Saarbrücken, 1994.

[Dar00] P. Darondeau. Region Based Synthesis of P/T-Nets and its Potential. In M. Nielsen and D. Simpson, editors, *Proc. 21st Int. Conf. on Application and Theory of Petri Nets*, volume 1825 of *LNCS*, pages 16–23, 2000.

[DDGJ01] W. Derks, J. Dehnert, P. Grefen, and W. Jonker. Customized Atomicity Specification for Transactional Workflow. In H. Lu and S. Spaccapietra, editors, *Proceedings of the Third International Symposium on Cooperative Database Systems and Applications (CODAS'01)*, pages 155–164. IEEE Computer Society, 2001.

[DE95] J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge University Press, 1995.

[DE00] J. Desel and T. Erwin. Modeling, Simulation and Analysis of Business Processes. In W. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques and Empirical Studies*, volume 1806 of *LNCS*, pages 129–141. Springer, 2000.

[DFZ00a] J. Dehnert, J. Freiheit, and A. Zimmermann. Modeling and Performance Evaluation of Workflow Systems. In *Proc. 4th. World Multiconference on Systemics, Cybernetics and Informatics (SCI'2000)*, volume VIII, pages 632–637, 2000.

[DFZ00b] J. Dehnert, J. Freiheit, and A. Zimmermann. Workflow Modeling and Performance Evaluation with Colored Stochastic Petri Nets. In *Bringing Knowledge to Business Processes, Workshop in the American Association for Artificial Intelligence (AAAI) Spring Symposium Series 2000*, pages 139–141, 2000.

[DFZ02] J. Dehnert, J. Freiheit, and A. Zimmermann. Modeling and Evaluation of Time Aspects in Business Processes. *Journal of the Operational Research Society (JORS)*, 53(8):138–147, 2002.

[DGS95] W. Deiters, V. Gruhn, and R. Striemer. Der FUNSOFT-Ansatz zum integrierten Geschäftsprozessmanagement. *Wirtschaftsinformatik*, 5(37):459–466, 1995.

[DR96] J. Desel and W. Reisig. The Synthesis Problem of Petri Nets. *Acta Informatica*, 33(4):297–315, 1996.

[DR98] J. Desel and W. Reisig. Place/Transition Petri Nets. In Reisig and Rozenberg [RR98a].

[DS93] M. Droste and R. M. Shortt. Petri Nets and Automata with Concurrency Relations - An Adjunction. In M. Droste and Y. Gurevich, editors, *Semantics of Programming Langauages and Model Theory*, pages 69–87. 1993.

[ED03] R. Eshuis and J. Dehnert. Reactive Petri Nets for Workflow Modelling. In Aalst and Best [AB03], pages 296–315.

[Ell79] C. A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, pages 225–240, Boulder, Colorado, 1979. ACM Press.

[Elm92] A.K. Elmagarmid, editor. *Database Transaction Models For Advanced Application.* Morgan Kaufmann, San Mateo,CA, 1992.

[EN93] C. A. Ellis and G. J. Nutt. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *14th. Int. Conf. on Application and Theory of Petri Nets 1993*, volume 691 of *LNCS*, pages 1–16. Springer, 1993.

[ER90] A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures. Part II: State Spaces of Concurrent Systems. *Acta Informatica*, 27(4):343–368, 1990.

[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.

[ES98] A. Pnueli E. Asarin, O. Maler and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier, 1998.

[Esh02] R. Eshuis. *Semantics and Verification of UML Activity Diagrams for Workflow Modeling.* PhD thesis, University of Twente, NL, 2002.

[EW00] R. Eshuis and R. Wieringa. Requirements Level Semantics for UML Statecharts. In S. F. Smith and C. L. Talcott, editors, *Formal Methods for Open Object-Based Distributed Systems IV - Proc. FMOODS'2000, September, 2000, Stanford, California, USA*. Kluwer Academic Publishers, 2000.

[EW01a] R. Eshuis and R. Wieringa. A Comparison of Petri Net and Activity Diagram Variants. In Weber et al. [WER01], pages 93–104.

[EW01b] R. Eshuis and R. Wieringa. A Real-Time Execution Semantics for UML Activity Diagrams. In H. Hussmann, editor, *4th Int. Conf. on Fundamental Approaches to Software Engineering (FASE 2001)*, volume 2029 of *LNCS*, pages 76–90. Springer, 2001.

[Fis01] L. Fischer, editor. *Workflow Handbook 2001, Workflow Management Coalition*. Future Strategies, Lighthouse Point, Florida, 2001.

[Fre02] J. Freiheit. *Matrizen- und Zustandsraumreduzierende Verfahren zur Leistungsbewertung grosser stochastischer Petrinetze*. PhD thesis, Technische Universitat Berlin, 2002.

[GA01] P. Grefen and S. Angelov. A three-level process framework for contract-based dynamic service outsourcing. In Weber et al. [WER01].

[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.

[GH94] D. Georgakopoulos and M. F. Hornick. A framework for enforceable specification of extended transaction models and transactional workflows. *International Journal of Intelligent and Cooperative Information System*, 3(3):225–253, 1994.

[GHM96] D. Georgakopoulos, M. F. Hornick, and F. Manola. Customizing transaction models and mechanisms in a programmable environment supporting reliable workflow automation. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):630–649, 1996.

[GHS95] D. Georgakopoulos, M.F. Hornick, and A. P. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

[Giu96] A. Giua. Petri Net Techniques for Supervisory Control of Discrete Event Systems. In *First Int. Work. on Manufacturing and Petri Nets*, pages 1–3, June 1996.

[GPS99] P. Grefen, B. Pernici, and G. Sanchez. *Database support for Workflow Management - the WIDE Project.* Kluwer Academic Publishers, Boston, 1999.

[GRX02a] A. Ghaffari, N. Rezg, and X. Xie. Net Transformation and Theory of Regions for Optimal Control of Petri Nets. In *15th Triennial World Congress of the International Federation of Automatic Control*, Barcelona, Spain, 2002.

[GRX02b] A. Ghaffari, N. Rezg, and X.Xie. Live and Maximally Permissive Controller Synthesis Using the Theory of Regions. In Caillaud et al. [CDLX02], pages 155–166.

[GW96] R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[Hac72] M. Hack. Analysis of Production Schemata by Petri Nets. Technical Report MIT/LCS/TR-94, MIT, 1972.

[Har87] D. Harel. Statecharts: A Visual Formulation for Complex Systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[HGZ96] L.E. Holloway, X. Guan, and L. Zhang. A generalization of state avoidance policies for controlled petri nets. *IEEE Transactions on Automatic Control*, 41(6):804 –816, 1996.

[HK94] L. E. Holloway and B. H. Krogh. Controlled Petri nets: A tutorial survey. In *Proc. 11th Int. Conference on Analysis and Optimization of Systems: Discrete Event Systems*, pages 158–168, France, 1994.

[HN93] A.H.M. ter Hofstede and E.R. Nieuwland. Task structure semantics through process algebra. *Software Engineering Journal*, 8(1):14–20, 1993.

[HSV03] K.M. van Hee, N. Sidorova, and M. Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In Aalst and Best [AB03], pages 337–356.

[Inc] Get Process AG, Oberwil, Switzerland. *INCOME Suite.*

[JB96] S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation.* International Thomson Computer Press, London, UK, 1996.

[Jen92] K. Jensen. *Coloured Petri Nets, Volume 1: Basic Concepts.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1992.

[Jen95] K. Jensen. *Coloured Petri Nets. Volume 2: Analysis Methods.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1995.

[KCK$^+$96] M. Kishinevsky, J. Cortadella, A. Kondratyev, L. Lavagno, and A. Yakovlev. Synthesis of General Petri Nets. AIZU Technical Report 96-2-004, University of Aizu, 1996.

[KG98] A. Kumar and L. S. Ganesh. Use of Petri Nets for Resource Allocation in Projects. *IEEE Trans. on Engineering Management*, 45(1):49–56, February 1998.

[Kie02] B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows.* PhD thesis, Queensland University of Technology, Brisbane, Australia, 2002.

[Kin98] E. Kindler. Database Theory - Petri Net Theory - Workflow Theory. *Petri Net Newsletter*, (55):14–18, 1998.

[KMO98] B. Kiepuszewski, R. Muhlberger, and M. E. Orlowska. FlowBack: Providing backward recovery for workflow management systems. In *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD-98),* volume 27,2 of *ACM SIGMOD Record*, pages 555–557, New York, 1998. ACM Press.

[KMR00] E. Kindler, A. Martens, and W. Reisig. Inter-Operability of Workflow Applications: Local Criteria for Global Soundness. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *LNCS*, pages 235–253. Springer, 2000.

[KMTV00] O. Kupferman, P. Madhusudan, P. S. Thiagarajan, and M. Y. Vardi. Open systems in reactive environments: Control and synthesis. In *Proc. 11th Int. Conf. on Concurrency Theory*, volume 1877 of *LNCS*, pages 92–107. Springer-Verlag, 2000.

[KNS92] G. Keller, M. Nüttgens, and A. W. Scheer. Semantische Process-modellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, University of Saarland, Saarbrücken, 1992.

[KT97] G. Keller and R. Teufel. SAP R/3 prozessorientiert anwenden – iteratives Prozess-Prototyping zur Bildung von Wertschöpfungsketten, 1997.

[Lau02] K. Lautenbach. Reproducibility of the Empty Marking. 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.

[Law97] P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition.* John Wiley and Sons, New York, 1997.

[Ley01] F. Leymann. *Web Services Flow Language (WSFL 1.0).* Microsoft Corporation, 2001.

[LO02] K. Lenz and A. Oberweis. Interorganizational Business Process Management with XML Net. In Rozenberg and Ehrig [RE02]. to appear.

[LR99] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques.* Prentice Hall PTR (ECS Professional), Upper Saddle River, 1999.

[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.

[Mar00] C. Marshall. *Enterprise Modeling With UML: Designing Successfull Software Through Business Analysis.* Addison Wesley, Reading, Massachusetts, 2000.

[Mar01] A. Martens. Modeling Workflow in Virtual Enterprises. In Weber et al. [WER01], pages 157–162.

[Mar02] D. C. Marinescu. *Internet-Based Workflow Management: Towards a Semantic Web*, volume 40 of *Wiley Series on Parallel and Distributed Computing.* Wiley-Interscience, New York, 2002.

[McN93] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[Mil80] R. Milner. *A Calculus of Communication Systems*, volume 92 of *LNCS*. Springer, New York, 1980.

[Mil99] R. Milner. *Communicating and Mobile Systems: The Pi Calcalus*, volume 92. Cambridge University Press, Cambridge, UK, 1999.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, New York, 1992.

[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.

[MT01] P. Madhusudan and P. S. Thiagarajan. Distributed controller synthesis for local specifications. In *ICALP 2001: Annual International Colloquium on Automata, Languages and Programming*, volume 2076 of *LNCS*, pages 396–407, 2001.

[Muk92] M. Mukund. Petri Nets and Step Transition Systems. *International Journal of Foundations of Computer Science*, 3(4):443–478, 1992.

[Mur89] T. Murata. Petri Nets: Properties, Analysis, and Applications. *Proc. of the IEEE*, 77(4):541–580, April 1989.

[NR02] M. Nüttgens and F. J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In J. Desel and M. Weske, editors, *Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen*, LN in Informatics. GI-Edition, 2002.

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

[NYY92] A. Nerode, A. Yakhnis, and V. Yakhnis. Concurrent Programs as Strategies in Games. In Y. N. Moschovakis, editor, *Logic from Computer Science: Workshop-Proceedings*, volume 21 of *Mathematical Sciences Research Institute Publications*, pages 405–480. Springer, 1992.

[Obe96] A. Oberweis. *Modellierung und Ausführung von Workflows mit Petri-Netzen.* Teubener-Reihe Wirtschaftsinformatik. B.G. Teubener Verlagsgesellschaft, Stuttgart, Leipzig, 1996.

[Pet62] C. A. Petri. *Kommunikation mit Automaten.* PhD thesis, TU Darmstadt, Darmstadt, 1962.

[Pet00] L. Petrucci. Design and validation of a controller. In *Proc. 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI'2000)*, volume VIII, pages 684–688, Orlando, USA, 2000. International Institute of Informatics and Systemics.

[Pie02] R. Piedboeuf. Eine Methode zur Sicherung der Korrektheit von Geschäftsprozessen. Master's thesis, Universität Koblenz-Landau, 2002.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL '89. Proceedings of the sixteenth annual ACM symposium on Principles of programming languages*, pages 179–190, New York, 1989. ACM Press.

[PR90] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *IEEE 31st Annual Symposium on Foundations of Computer Science*, pages 746 –757, 1990.

[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.

[RE02] G. Rozenberg and H Ehrig, editors. *Petri Net Technology for Communication Based Systems*, LNCS, Advances in Petri Nets. Springer, 2002. to appear.

[Rei85] W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, 1985.

[Reu89] A. Reuter. Contracts: A means for extending control beyond transaction boundaries. In *Proc. 3rd International Workshop on High Performance Transaction Systems*, 1989.

[Rit00a] P. Rittgen. Paving the Road to Business Process Automation. In *European Conference on Information Systems (ECIS)*, Austria, 2000.

[Rit00b] P. Rittgen. Quo vadis EPK in ARIS? Ansätze zu syntaktischen Erweiterungen und einer formalen Semantik. *Wirtschaftsinformatik*, 1(42):27–35, 2000.

[Roc00] S. Roch. extended Computation Tree Logic. In H.D. Burkhard, L. Czaja, A. Skowron, and P. Starke, editors, *Workshop Concurrency, Specification & Programming*, number 140 in Informatik-Bericht, pages 225–234, Humboldt-Universität zu Berlin, 2000.

[Rod99] J. Rodenhagen. Darstellung ereignisgesteuerter Prozessketten (EPK) mit Hilfe von Petrinetzen. Master's thesis, Universität Hamburg, Fachbereich Informatik, 1999.

[RR98a] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Model*, volume 1491 of *LNCS*. Springer, 1998.

[RR98b] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets II: Applications*, volume 1492 of *LNCS*. Springer, 1998.

[RSS97] A. Reuter, K. Schneider, and F. Schwenkreis. Contracts revisited. In S. Jajodia and L. Kerschberg, editors, *Advanced Transaction Models and Architecture*. Kluwer Academic Publisher, 1997.

[Rum99] F. J. Rump. *Geschäftsprozessmanagement auf der Basis ereignisgesteuerter Prozessketten. Formalisierung, Analyse und Ausfhrung von EPKs*. Teubner, Stuttgart, 1999.

[RW87] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[Sch94] A. W. Scheer. *Business Process Engineering, ARIS-Navigator for Reference Models for Industrial Enterprises.* Springer, 1994.

[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.

[Sim02] C. Simon. A logic of actions to specify and verify process requirement. In *Proceedings of the Seventh Australian Workshop on Requirements Engineering*, 2002.

[SJ02] A. W. Scheer and W. Jost. *ARIS in der Praxis Gestaltung, Implementierung und Optimierung von Geschäftsprozessen.* Springer, 2002.

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

[SO99] W. Sadiq and M. E. Orlowska. Applying graph reduction techniques for identifying structural conflicts in process models. In M. Jarke and A. Oberweis, editors, *Proc. 11th Int. Conf. Advanced Information Systems Engineering, CAISE*, number 1626 in LNCS. Springer, 1999.

[SR93] A. P. Sheth and M. Rusinkiewicz. On transactional workflows. *Data Engineering Bulletin*, 16(2):34–40, 1993.

[Sta90] P. H. Starke. *Analyse von Petri-Netz-Modellen (in german).* Teubner, Stuttgart, 1990.

[Sta99] Staffware. *Staffware 2000 / GWD User Manual.* Staffware plc, Berkshire, United Kingdom, 1999.

[Sta00] Staffware. *Staffware Case Handler — White Paper.* Staffware PLC, Berkshire, UK, 2000.

[Tha01] S. Thatte. *Web Services for Business Process Design.* Microsoft Corporation, 2001.

[Tho95] W. Thomas. On the synthesis of strategies in infinite games. In *Symposium on Theoretical Aspects of Computer Science*, pages 1–13, 1995.

[Tho02] Wolfgang Thomas. Infinite games and verification (extended abstract of a tutorial). *LNCS*, 2404:58–65, 2002.

[VA00] H.M.W. Verbeek and W.M.P. van der Aalst. Woflan 2.0: A petri-net-based workflow diagnosis tool. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets*, volume 1825 of *LNCS*, pages 475–484. Springer, 2000.

[vU97] C. v. Uthmann. Nutzenpotentiale der Petrinetztheorie für die Erweiterung der Anwendbarkeit Ereignisgesteuerter Prozeßketten (EPK). In *Proc. Workshop Formalisierung und Analyse Ereignisgesteuerter Prozeßketten (EPK)*, pages 1–22, 1997.

[WER01] H. Weber, H. Ehrig, and W. Reisig, editors. *Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems*. Fraunhofer Gesellschaft ISST, September 2001.

[WK02] M. Weber and E. Kindler. The Petri Net Markup Language. In Rozenberg and Ehrig [RE02]. to appear.

[WW97] D. Wodtke and G. Weikum. A Formal Foundation For Distributed Workflow Execution Based on State Charts. In *Int. Conf. on Database Theory*, Delphi, Greece, 1997.

[WWKD$^+$97] G. Weikum, D. Wodtke, A. Kotz-Dittrich, P. Muth, and J. Weissenfels. Spezifikation, Verifikation und verteilte Ausführung von Workflows in MENTOR . *Informatik Forschung und Entwicklung*, 1997.

[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.