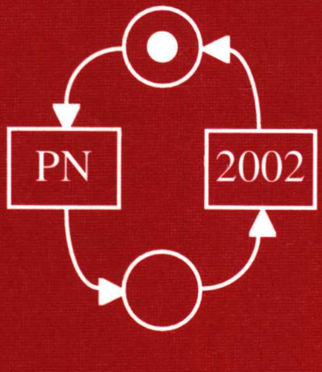
LNCS 2360

Javier Esparza
Charles Lakos (Eds.)

Application and Theory of Petri Nets 2002

23rd International Conference, ICATPN 2002
Adelaide, Australia, June 2002
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Javier Esparza
University of Edinburgh, Division of Informatics
James Clerk Maxwell Building, The King's Buildings
Mayfield Road, Edinburgh EH9 3JZ, United Kingdom
E-mail: jav@dcs.ed.ac.uk

Charles Lakos
University of Adelaide, Department of Computer Science
North Terrace, Adelaide, SA, 5005, Australia
E-mail: Charles.Lakos@adelaide.edu.au

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Application and theory of Petri nets 2002 : 23rd international conference ;
proceedings / ICATPN 2002, Adelaide, Australia, June 24 - 30, 2002.
Javier Esparza ; Charles Lakos (ed.). - Berlin ; Heidelberg ; New York ;
Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; Vol. 2360)
ISBN 3-540-43787-8

CR Subject Classification (1998): F.1-3, C.1-2, G.2.2, D.2, D.4, J.4

ISSN 0302-9743
ISBN 3-540-43787-8 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

© Springer-Verlag Berlin Heidelberg 2002
Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN 10870245 06/3142 5 4 3 2 1 0

Preface

This volume contains the proceedings of the 23rd International Conference on Application and Theory of Petri Nets. The aim of the Petri net conferences is to create a forum for discussing progress in the application and theory of Petri nets. Typically, the conferences have 100-150 participants – one third of these coming from industry while the rest are from universities and research institutions. The conferences always take place in the last week of June.

The conference and a number of other activities are co-ordinated by a steering committee with the following members: G. Balbo (Italy), J. Billington (Australia), G. De Michelis (Italy), S. Haddad (France), K. Jensen (Denmark), S. Kumagai (Japan), T. Murata (USA), C.A. Petri (Germany; honorary member), W. Reisig (Germany), G. Rozenberg (The Netherlands; chairman), and M. Silva (Spain).

Other activities before and during the 2002 conference included tool demonstrations, extensive introductory tutorials, two advanced tutorials on "Workflow Management: Models, Methods, and Systems" and "Model Checking", and two workshops on "Software Engineering and Formal Methods" and "Formal Methods Applied to Defence Systems". The tutorial notes and workshop proceedings are not published in this volume, but copies are available from the organizers. The proceedings can be found at `http://www.jrpit.flinders.edu.au/CRPIT.html`.

The 2002 conference was organized by the Computer Systems Engineering Centre, School of Electrical and Information Engineering at the University of South Australia, Adelaide, Australia with assistance from the Department of Computer Science, Adelaide University, Adelaide, Australia. We would like to thank the members of the organizing committee (see next page) and their teams.

We would like to thank very much all those who submitted papers to the Petri net conference. We received a total of 45 submissions from 18 different countries. This volume comprises the papers that were accepted for presentation. Invited lectures were given by W. van der Aalst, J. Desel, I. Hayes, C. Lakos, P.S. Thiagarajan, and A. Yakovlev (whose papers are included in this volume).

The submitted papers were evaluated by a program committee. The program committee meeting took place in Edinburgh, Scotland. We would like to express our gratitude to the members of the program committee, and to all the referees who assisted them. The names of these are listed on the following pages.

We would like to acknowledge the local support of Marco Kick, Bill Orrok, and Claus Schröter. Finally, we would like to mention the excellent co-operation with Springer-Verlag during the preparation of this volume.

April 2002

Javier Esparza and Charles Lakos

Organizing Committee

Carolyn Bellamy
Jonathan Billington (chair)
Pierre Dauchy
Guy Gallasch
Bing Han
Robin King
Lars Michael Kristensen
Charles Lakos
Lin Liu
Pauline Olsson
Patrick O'Sullivan
Chun Ouyang
Laure Petrucci

Tools Demonstration

Lars Michael Kristensen (chair)

Program Committee

Wil van der Aalst (The Netherlands)
Gianfranco Balbo (Italy)
Eike Best (Germany)
Jonathan Billington (Australia)
Gianfranco Ciardo (USA)
Jordi Cortadella (Spain)
Philippe Darondeau (France)
Giorgio De Michelis (Italy)
Javier Esparza (UK; co-chair; theory)
Claude Girault (France)
Nisse Husberg (Finland)
Lars Michael Kristensen (Denmark)
Sadatoshi Kumagai (Japan)
Charles Lakos (Australia; co-chair; applications)
Madhavan Mukund (India)
Wojciech Penczek (Poland)
Vladimiro Sassone (UK/Italy)
Karsten Schmidt (Germany)
Sol Shatz (USA)
Enrique Teruel (Spain)
Alex Yakovlev (UK)
Wlodek Zuberek (Canada)

Referees

Alessandra Agostini
Adrianna Alexander
Gerard Berthelot
Eric Badouel
Marek A. Bednarczyk
Simona Bernardi
Luca Bernardinello
Andrzej M. Borzyszkowski
Roberto Bruni
Benoît Caillaud
Felice Cardone
Søren Christensen
Paolo Ciancarini
Robert Clariso
José-Manuel Colom
Deepak D'Souza
Michel Diaz
Claude Dutheillet
Susanna Donatelli
Emmanuelle Encrenaz
Robert Esser
Joaquín Ezpeleta
David de Frutos-Escrig
Hans Fleischhack
Giuliana Franceschinis
Rossano Gaeta
Steven Gordon
Marco Gribaudo
Bing Han
Keijo Heljanko
Zhaoxia Hu
Guy Juanole
Jorge Júlvez
Jens Bæk Jørgensen
Victor Khomenko
Ekkart Kindler
Mike Kishinevsky
Hanna Klaudel
Maciej Koutny

K Narayan Kumar
Johan Lilius
Lin Liu
Louise Lorentsen
Thomas Mailund
Axel Martens
José Meseguer
Toshiyuki Miyamoto
Kjeld Høyer Mortensen
Alix Munier
Tadao Murata
Marko Mäkelä
Chun Ouyang
Emmanuel Paviot-Adet
Denis Poitrenaud
Giuseppe Pappalardo
Elina Parviainen
Enric Pastor
Olli-Matti Penttinen
Laure Petrucci
Agata Półrola
Lucia Pomello
Pierre-Olivier Ribet
Laura Recalde
Wolfgang Reisig
Elvinia Riccobene
Diego Rodríguez
Matteo Sereno
Manuel Silva
Radu I. Siminiceanu
Pawel Sobocinski
Jeremy Sproston
Jiří Srba
Peter Starke
Christian Stehno
K.V. Subrahmanyam
Ichiro Suzuki
Maciej Szreter
Cecile Bui Thanh
P.S. Thiagarajan
Fernando Tricas
Teemu Tynjälä
François Vernadat
Kimmo Varpaaniemi
Eric Verbeek
José-Luis Villarroel
Harro Wimmel
Jozef Winkowski
Bozena Wozna
Fei Xia
Xiaolan Xie
Haiping Xu
Hideki Yamasaki

Sponsoring Institutions

The Australian Defence Science and Technology Organization
The Adelaide City Council
The University of South Australia
Division of Informatics, University of Edinburgh

Table of Contents

Making Work Flow: On the Application of Petri Nets to Business Process Management

Wil M. P. van der Aalst *

Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract. Information technology has changed business processes within and between enterprises. More and more work processes are being conducted under the supervision of information systems that are driven by process models. Examples are workflow management systems such as Staffware, enterprise resource planning systems such as SAP and Baan, but also include many domain specific systems. It is hard to imagine enterprise information systems that are unaware of the processes taking place. Although the topic of business process management using information technology has been addressed by consultants and software developers in depth, a more fundamental approach has been missing. Only since the nineties, researchers started to work on the foundations of business process management systems. This paper addresses some of the scientific challenges in business process management. In the spirit of Hilbert's problems[1], 10 interesting problems for people working on Petri-net theory are posed.

1 Introduction

The goal of this paper is to show the relevance, architecture, and Achilles heel of business process management systems. This way we hope to interest Petri-net researchers in some of the scientific challenges in this domain. The definition of a business process management system used throughout this paper is: *a generic software system that is driven by explicit process designs to enact and manage operational business processes.* The system should be process-aware and generic in the sense that it is possible to modify the processes it supports. The process designs are often graphical and the focus is on structured processes that need to handle many cases.

In the remainder of this paper, we will first put business process management and related technology in its historical context. Then, we will discuss models for process design. Since business process management systems are driven by explicit models, it is important to use the right techniques. Next, we will discuss

* Part of paper is taken from my inaugural lecture "Making Work Flow: On the Design, Analysis, and Enactment of Business Processes" [6].

[1] Note that by no means we are suggesting that the problems in this paper are of the same stature as the 23 problems raised by David Hilbert in 1900.

techniques for the analysis of process models. We will argue that it is vital to have techniques to assert the correctness of workflow designs. Based on this we will focus on systems for process enactment, i.e., systems that actually make the "work flow" based on a model of the processes and organizations involved. Finally, we will pose 10 interesting problems in the spirit of Hilbert's problems [31].

2 Business Process Management from a Historical Perspective

Only the wisest and stupidest of men never change.
Confucius

To show the relevance of business process management systems, it is interesting to put them in a historical perspective. Consider Figure 1, which shows some of the ongoing trends in information systems. This figure shows that today's information systems consist of a number of layers. The center is formed by the operating system, i.e., the software that makes the hardware work. The second layer consists of generic applications that can be used in a wide range of enterprises. Moreover, these applications are typically used within multiple departments within the same enterprise. Examples of such generic applications are a database management system, a text editor, and a spreadsheet program. The third layer consists of domain specific applications. These applications are only used within specific types of enterprises and departments. Examples are decision support systems for vehicle routing, call center software, and human resource management software. The fourth layer consists of tailor-made applications. These applications are developed for specific organizations.

In the sixties the second and third layer were missing. Information systems were built on top of a small operating system with limited functionality. Since no generic nor domain specific software was available, these systems mainly consisted of tailor-made applications. Since then, the second and third layer have developed and the ongoing trend is that the four circles are increasing in size, i.e., they are moving to the outside while absorbing new functionality. Today's operating systems offer much more functionality. Database management systems that reside in the second layer offer functionality which used to be in tailor-made applications. As a result of this trend, the emphasis shifted from programming to assembling of complex software systems. The challenge no longer is the coding of individual modules but orchestrating and gluing together pieces of software from each of the four layers.

Another trend is the shift from data to processes. The seventies and eighties were dominated by data-driven approaches. The focus of information technology was on storing and retrieving information and as a result data modeling was the starting point for building an information system. The modeling of business processes was often neglected and processes had to adapt to information technology. Management trends such as business process reengineering illustrate the increased emphasis on processes. As a result, system engineers are resorting to a more process driven approach.

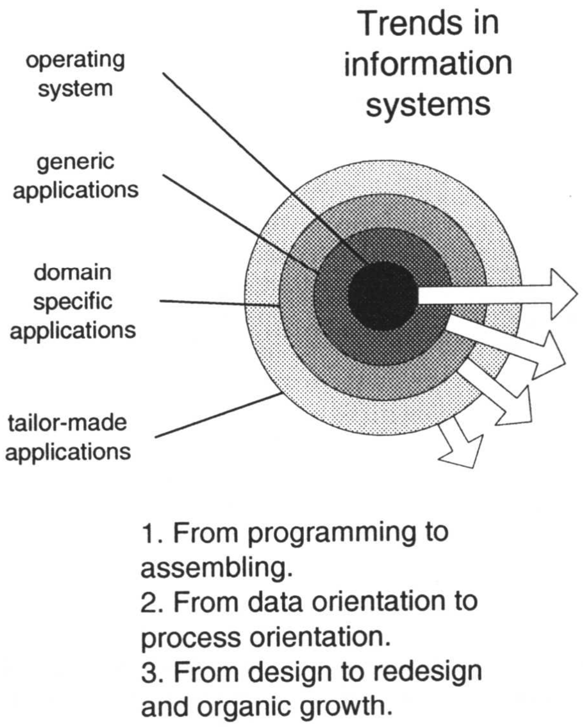

Trends in information systems
operating system
generic applications
domain specific applications
tailor-made applications
1. From programming to assembling.
2. From data orientation to process orientation.
3. From design to redesign and organic growth.

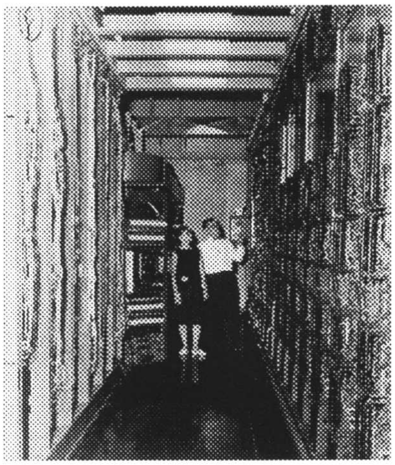
Fig. 1. Trends relevant for business process management

The last trend we would like to mention is the shift from carefully planned designs to redesign and organic growth. Due to the omnipresence of the Internet and its standards, information systems change on-the-fly. As a result, fewer systems are built from scratch. In many cases existing applications are partly used in the new system. Although component-based software development still has it problems, the goal is clear and it is easy to see that software development has become more dynamic.

The trends shown in Figure 1 provide a historical context for business process management systems. Business process management systems are either separate applications residing in the second layer or are integrated components in the domain specific applications, i.e., the third layer. Notable examples of business process management systems residing in the second layer are workflow management systems [33,36] such as Staffware, MQSeries, and COSA, and case handling systems such as FLOWer. Note that leading enterprise resource planning systems populating the third layer also offer a workflow management module. The workflow engines of SAP, Baan, PeopleSoft, Oracle, and JD Edwards can be considered as integrated business process management systems. The idea to isolate the management of business processes in a separate component is consistent with the three trends identified. Business process management systems can be used to avoid hard-coding the work processes into tailor-made applications and thus support the shift from programming to assembling. Moreover, process orientation, redesign, and organic growth are supported. For example, today's workflow management systems can be used to integrate existing applications and support process change by merely changing the workflow diagram. Give these observations, we hope to have demonstrated the practical relevance of business process management systems. In the remainder of this paper we will focus more on the scientific importance of these systems. Moreover, for clarity we will often restrict the discussion to clear cut business process management systems such as workflow management systems.

An interesting starting point from a scientific perspective is the early work on office information systems. In the seventies, people like Skip Ellis [24], Anatol Holt [32], and Michael Zisman [46] already worked on so-called office information systems, which were driven by explicit process models. It is interesting to see that the three pioneers in this area independently used Petri-net variants to model office procedures. During the seventies and eighties there was great optimism about the applicability of office information systems. Unfortunately, few applications succeeded. As a result of these experiences, both the application of this technology and research almost stopped for a decade. Consequently, hardly any advances were made in the eighties. In the nineties, there again was a huge interest in these systems. The number of workflow management systems developed in the past decade and the many papers on workflow technology illustrate the revival of office information systems. Today workflow management systems are readily available [36]. However, their application is still limited to specific industries such as banking and insurance. As was indicated by Skip Ellis it is important to learn from these ups and downs [26]. The failures in the eighties can be explained by both technical and conceptual problems. In the eighties, networks were slow or not present at all, there were no suitable graphical interfaces, and proper development software was missing. However, there were also more fundamental problems: a unified way of modeling processes was missing and the systems were too rigid to be used by people in the workplace. Most of the technical problems have been resolved by now. However, the more conceptual problems remain. Good standards for business process modeling are still missing and even today's workflow management systems enforce unnecessary constrains on the process logic (e.g., processes are made more sequential).

One of the great challenges of business process management systems is to offer both support and flexibility [9,14,35]. Today's systems typically are too rigid, thus forcing people to work around the system. One of the problems is that software developers and computer scientists are typically inspired by processes inside a computer system rather than processes outside a computer. Figure 2 illustrates the typical mind-frame of people developing business process management systems. This photograph shows the Whirlwind computer, which was the first computer system to have magnetic core memory (1953). It is interesting to mention that Whirlwind was developed by Jay Forrester who also



WHIRLWIND (1953)

Architecture: 32 bit word length, duplex CPU, 75kips single address, no interrupts, 4 index registers, real time clock
Memory: magnetic core (4Kx64word) 6 microseconds cycle time; magnetic drum (150K word); 4 IBM Model 729 tape drives (~100K word each); parity checking
I/O: CRT display, keyboard, light gun, real time serial data (teletype 1300bps modem), voice line
Size: 60,000 vacuum tubes, 175,000 diodes, 13,000 transistors; CPU space 50x150 feet each; CPU weight 500,000 lbs; power consumption: 3 megawatts

Fig. 2. The Whirlwind - photo from the Timeline of Events on Computer History (©2001 IEEE)

developed the well-known Systems Dynamics approach [27]. Software engineers are typically trained in the architecture and systems software of computers like the Whirlwind and its successors. As a result, these engineers think in terms of control systems rather than support systems. This explains that few of the existing workflow management systems allow for the so-called implicit choice, i.e., a choice resolved by the environment rather than the system [13]. To solve these problems, the typical mind-frame should be changed such that the business process management system is viewed as a reactive system rather than merely a control system.

To summarize we state that, although the relevance of business process management systems is undisputed, many fundamental problems remain to be solved. In the remainder of this paper we will try to shed light on some of these problems.

3 Models for Process Design

A camel is a horse designed by committee.
Sir Alec Issigonis

Business process management systems are driven by models of processes and organizations. By changing these models, the behavior of the system adapts to its environment and changing requirements. These models cover different perspectives. Figure 3 shows some of the perspectives relevant for business process management systems [33]. The process perspective describes the control-flow, i.e., the ordering of tasks. The information perspective describes the data that are used. The resource perspective describes the structure of the organization and identifies resources, roles, and groups. The task perspective describes the

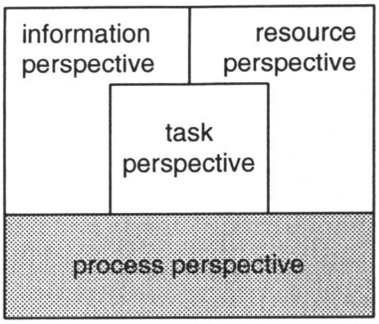


Fig. 3. Perspectives of models driving business process management systems

content of individual steps in the processes. Each perspective is relevant. However, the process perspective is dominant for the type of systems addressed in this talk.
Many techniques have been proposed to model the process perspective. Some of these techniques are informal in the sense that the diagrams used have no formally defined semantics. These models are typically very intuitive and the interpretation shifts depending on the modeler, application domain, and characteristics of the business processes at hand. Examples of informal techniques are ISAC, DFD, SADT, and IDEF. These techniques may serve well for discussing work processes. However, they are inadequate for directly driving information systems since they are incomplete and subject to multiple interpretations. Therefore, more precise ways of modeling are required.

Figure 4 shows an example of an order handling process modeled in terms of a so-called workflow net [1]. Workflow nets are based on the classical Petri-net model invented by Carl Adam Petri in the early sixties [39]. The squares are the active parts of the model and correspond to tasks. The circles are the passive parts of the model and are used to represent states. In the classical Petri net, the squares are named transitions and the circles places. A workflow net models the life-cycle of one case. Examples of cases are insurance claims, tax declarations, and traffic violations. Cases are represented by tokens and in this case the token in *start* corresponds to an order. Task *register* is a so-called AND-split and is enabled in the state shown. The arrow indicates that this task requires human intervention. If a person executes this task, the token is removed from place *start* and two tokens are produced: one for *c1* and one for *c2*. Then, in parallel, two tasks are enabled: *check_availability* and *send_bill*. Depending on the eagerness of the workers executing these two tasks either *check_available* or *send_bill* is executed first. Suppose *check_availability* is executed first. If the ordered goods are available, they can be shipped by executing task *ship_goods*. If they are not available, either a replenishment order is issued or not. Note that *check_availability* is an OR-split and produces one token for *c3*, *c4*, or *c5*. Suppose that not all ordered goods are available, but the appropriate replenishment orders were already issued. A token is produced for *c3* and task *update* becomes enabled.

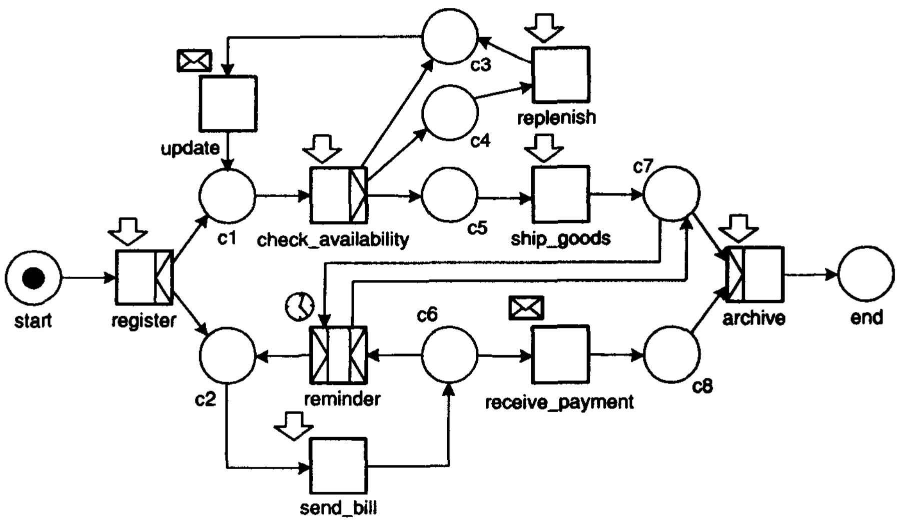


Fig. 4. WF-net

Suppose that at this point in time task *send_bill* is executed, resulting in the state with a token in *c3* and *c6*. The token in *c6* is input for two tasks. However, only one of these tasks can be executed and in this state only *receive_payment* is enabled. Task *receive_payment* can be executed the moment the payment is received. Task *reminder* is an AND-join/AND-split and is blocked until the bill is sent and the goods have been shipped. Note that the reminder is sent after a specified period as indicated by the clock symbol. However, it is only possible to send a remainder if the goods have been actually shipped. Assume that in the state with a token in *c3* and *c6* task *update* is executed. This task does not require human involvement and is triggered by a message of the warehouse indicating that relevant goods have arrived. Again *check_availability* is enabled. Suppose that this task is executed and the result is positive. In the resulting state *ship_goods* can be executed. Now there is a token in *c6* and *c7* thus enabling task *reminder*. Executing task *reminder* again enables the task.*send_bill*. A new copy of the bill is sent with the appropriate text. It is possible to send several reminders by alternating *reminder* and *send_bill*. However, let us assume that after the first loop the customer pays resulting in a state with a token in *c7* and *c8*. In this state, the AND-join *archive* is enabled and executing this task results in the final state with a token in *end*.

This very simple workflow net shows some of the routing constructs relevant for business process modeling. Sequential, parallel, conditional, and iterative routing are present in this model. There also are more advanced constructs such as the choice between *receive_payment* and *reminder*. This is a so-called *implicit choice* since it is not resolved by the system but by the environment of the system. The moment the bill is sent, it is undetermined whether *receive_payment* or *reminder* will be the next step in the process. Another advanced construct is the fact that task *reminder* is blocked until the goods have been shipped. The latter construct is a so-called *milestone*. The reason that we point out both constructs is that many systems have problems supporting these rather fundamental process patterns [13].

Workflow nets have clear semantics. The fact that we are able to play the so-called token game using a minimal set of rules shows the fact that these models are executable. None of the informal informal techniques mentioned before (i.e., ISAC, DFD, SADT, and IDEF) have formal semantics. Besides workflow nets there are many other formal techniques. Examples are the many variants of process algebra [17] and statecharts [29]. The reason we prefer to use a variant of Petri nets is threefold [1]:

- Petri nets are graphical and yet precise.
- Petri nets offer an abundance of analysis techniques.
- Petri nets treat states as first-class citizens.

The latter point deserves some more explanation. Many techniques for business process modeling focus exclusively on the active parts of the process, i.e., the tasks. This is rather surprising since in many administrative processes the actual processing time is measured in minutes and the flow time is measured in days. This means that most of the time cases are in-between two subsequent tasks. Therefore, it is vital to model these states explicitly.

In recent years, the Unified Modeling Language (UML, [20]) has become the de facto standard for software development. UML has four diagrams for process modeling. UML supports variants of statecharts and its activity diagrams are inspired by Petri nets. UML combines both good and bad ideas and can be considered semi-formal. Many colleagues are trying to provide solid semantics for UML. In my opinion, it would have been better to start with a solid foundation.

4 Techniques for Process Analysis

From the errors of others, a wise man corrects his own.
Syrus

Business process management systems allow organizations to change their processes by merely changing the models. The models are typically graphical and can be changed quite easily. This provides more flexibility than conventional information systems. However, by reducing the threshold for change, errors are introduced more easily. Therefore, it is important to develop suitable analysis techniques. However, it is not sufficient to just develop these techniques. It is as least as important to look at methods and tools to make them applicable in a practical context.

Traditionally, most techniques used for the analysis of business processes, originate from operations research. All students taking courses in operations management will learn to apply techniques such as simulation, queueing theory, and Markovian analysis. The focus mainly is on *performance analysis* and less attention is paid to the correctness of models. *Verification* and *validation* are often neglected. As a result, systems fail by not providing the right support or even break down [2,42]. Verification is needed to check whether the resulting system is free of logical errors. Many process designs suffer from deadlocks and livelocks that could have been detected using verification techniques. Validation is needed to check whether the system actually behaves as expected. Note that validation is context dependent while verification is not. A system that deadlocks is not correct in any situation. Therefore, verifying whether a system exhibits deadlocks is context independent. Validation is context dependent and can only be done with knowledge of the intended business process.

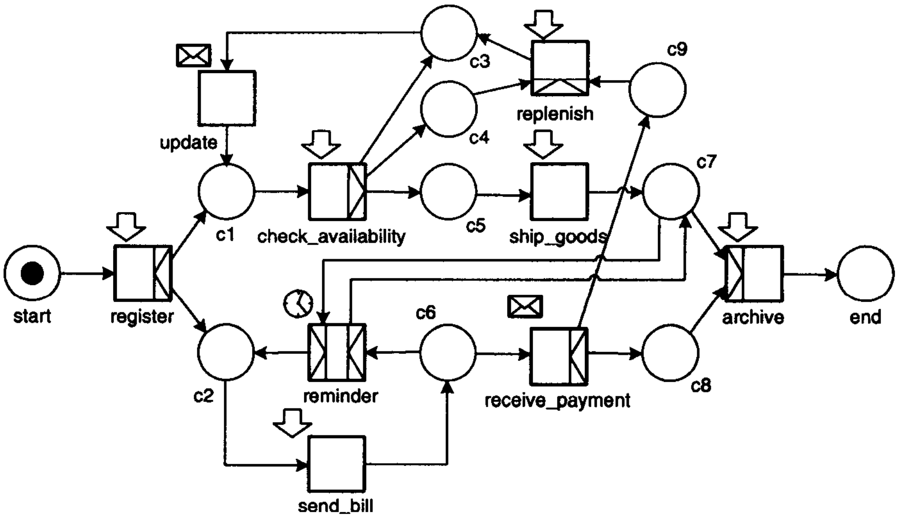


Fig. 5. An incorrect WF-net

To illustrate the relevance of validation and verification and to demonstrate some of the techniques available, we return to the workflow net shown in Figure 4. This workflow process allows for the situation where a replenishment is issued before any payment is received. Suppose that we want to change the design such that replenishments are delayed until receiving payment. An obvious way to model this is to connect task *receive_payment* with *replenish* using an additional place *c9* as shown in Figure 5. Although this extension seems to be correct at first glance, the resulting workflow net has several errors. The workflow will deadlock if a second replenishment is needed and something is left behind in the process if no replenishments are needed. These are logical errors that can be detected without any knowledge of the order handling process. For verification, application independent notions of correctness are needed. One of these notions is the so-called *soundness property* [1]. A workflow net is sound if and only if the workflow contains no dead parts (i.e., tasks that can never be executed), from any reachable state it is always possible to terminate, and the moment the workflow terminates all places except the sink place (i.e., place *end*) are empty. Note that soundness rules out logical errors such as deadlocks and livelocks. The notion of soundness is applicable to any workflow language. An interesting observation is that soundness corresponds to liveness and boundedness of the short-circuited net [1]. The latter properties have been studied extensively [41,23]. As a result, powerful analysis techniques and tools can be applied to verify the correctness of a workflow design. Practical experience shows that many errors can be detected by verifying the soundness property. Moreover, Petri-net theory can also be applied to guide the designer towards the error.

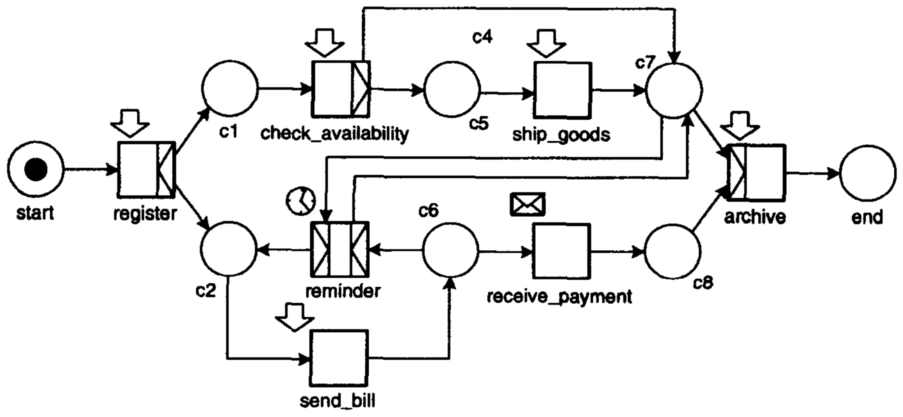


Fig. 6. A sound but incorrect WF-net

Soundness does not guarantee that the workflow net behaves as intended. Consider for example, the workflow net shown in Figure 6. Compared to the original model, the shipment of goods is skipped if some of the goods are not available. Again this may seem to be a good idea at first glance. However, customers are expected to pay even if the goods are never delivered. In other words, task *receive_payment* needs to be executed although task *ship_goods* may never be executed. The latter error can only be detected using knowledge about the context. Based on this context one may decide whether this is acceptable or not. Few analysis techniques exist to automatically support this kind of validation. The only means of validation offered by today's workflow management systems is gaming and simulation.

An interesting technique to support validation is inheritance of dynamic behavior. Inheritance can be used as a technique to compare processes. Inheritance relates subclasses with superclasses [19]. A workflow net is a subclass of a superclass workflow net if certain dynamic properties are preserved. A subclass