Jörg Desel
Barbara Pernici
Mathias Weske (Eds.)

LNCS 3080

Business Process Management

Second International Conference, BPM 2004
Potsdam, Germany, June 2004
Proceedings

Springer

F273-53
B793
2004

Jörg Desel   Barbara Pernici
Mathias Weske (Eds.)

Business Process Management

Second International Conference, BPM 2004
Potsdam, Germany, June 17-18, 2004
Proceedings

华南理工大学图书馆
外国教材
中心藏书章

E200404169

Springer

Volume Editors

Jörg Desel
Katholische Universität Eichstätt-Ingolstadt, Lehrstuhl für Angewandte Informatik
Ostenstr. 14, 85072 Eichstätt, Germany
E-mail: joerg.desel@ku-eichstaett.de

Barbara Pernici
Politecnico di Milano, Dipartimento di Elettronica e Informazione
via Beato Angelico 23/1, 20133 Milano, Italy
E-mail: barbara.pernici@polimi.it

Mathias Weske
University of Potsdam, Hasso-Plattner-Institute for Software Systems Engineering
Prof.-Dr.-Helmert-Straße 2-3, 14480 Potsdam, Germany
E-mail: mathias.weske@hpi.uni-potsdam.de

Library of Congress Control Number: 2004107855

CR Subject Classification (1998): H.3.5, H.4.1, H.5.3, K.4.3, K.4.4, K.6, J.1

ISSN 0302-9743
ISBN 3-540-22235-9 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable to prosecution under the German Copyright Law.

Springer-Verlag is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 11013068 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3080

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
University of Surrey, Guildford, UK
Jon M. Kleinberg
Cornell University, Ithaca, NY, USA
Friedemann Mattern
ETH Zurich, Switzerland
John C. Mitchell
Stanford University, CA, USA
Moni Naor
Weizmann Institute of Science, Rehovot, Israel
Oscar Nierstrasz
University of Bern, Switzerland
C. Pandu Rangan
Indian Institute of Technology, Madras, India
Bernhard Steffen
University of Dortmund, Germany
Madhu Sudan
Massachusetts Institute of Technology, MA, USA
Demetri Terzopoulos
New York University, NY, USA
Doug Tygar
University of California, Berkeley, CA, USA
Moshe Y. Vardi
Rice University, Houston, TX, USA
Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

此为试读，需要完整PDF请访问：www.ertongbook.com

Preface

In recent years the management of business processes has emerged as one of the major developments to ease the understanding of, communication about, and evolution of process-oriented information systems in a variety of application domains. Based on explicit representations of business processes, process stakeholders can communicate about process structure, content, and possible improvements. Formal analysis, verification and simulation techniques have the potential to show deficits and to effectively lead to better and more flexible processes. Process mining facilitates the discovery of process specifications from process logs that are readily available in many organizations.

This volume of Springer's Lecture Notes in Computer Science contains the papers presented at the 2nd International Conference on Business Process Management (BPM 2004) which took place in Potsdam, Germany, in June 2004. From more than 70 submissions BPM 2004 received, 19 high-quality research papers were selected.

BPM 2004 is part of a conference series that provides a forum for researchers and practitioners in all aspects of business process management. In June 2003, the 1st International Conference on Business Process Management took place in Eindhoven, The Netherlands. Its proceedings were published as Volume 2678 of Lecture Notes in Computer Science by Springer-Verlag. A previous volume (LNCS 1806) on Business Process Management was based on four events devoted to this topic.

This book presents a significant view on the state of the art in business process management, ranging from formal approaches to descriptions of tools for the design of processes. The topics addressed by the contributions cover areas like business process modeling, formal models, as well as analysis and verification of business processes, process mining and workflow management, and, moreover, case studies from various domains including medicine, technology, and logistics.

Besides its research paper track, BPM 2004 hosted a keynote presentation by Christoph Bussler, Vice-Director of the Digital Enterprise Research Institute in Galway, Ireland. A tutorial on workflow modeling and analysis using Petri nets was given by Wil van der Aalst, Head of the Information Systems department in the Faculty of Technology Management, Eindhoven Technical University (NL). Rainer Ruggaber of SAP Corporate Research talked about SAP's involvement in European research projects. Thomas Volmering and Karl Wagner reported on a recently strengthened cooperation between SAP and IDS Scheer in the context of the new SAP software architecture, based on service technology. All these presentations are highly appreciated.

The organizers are thankful to SAP AG, IDS Scheer and Adesso AG as well as to the Hasso Plattner Institute for supporting this scientific event. A special thanks goes to Jörn Lauterjung and his colleagues from Geoforschungszentrum Potsdam, which provided the location of this year's BPM conference. We thank the local organization group at the Hasso Plattner Institute, including Hilmar Schuschel, Katrin Heinrich and Mirko Schulze, who provided the conference Web site and online registration system and also installed and maintained the CyberChair conference management software that we used during the reviewing process. The group at KU Eichstätt-Ingolstadt, in particular Dorothea Iglezakis and Birgit Eisen, collected the final versions of the research papers and prepared the camera-ready copy of this Springer Lecture Notes in Computer Science volume.

We should like to acknowledge the excellent cooperation with Alfred Hofmann of Springer-Verlag and his colleagues in the preparation of this volume.

Finally, we are grateful to all Program Committee members and additional reviewers for their contribution to the success of the conference.

June 2004

Jörg Desel
Barbara Pernici
Mathias Weske

Program Committee

Wil van der Aalst, The Netherlands
Boualem Benatallah, Australia
Christoph Bussler, Ireland
Fabio Casati, USA
Piotr Chrzastowski-Wachtel, Poland
Leonid Churilov, Australia
Peter Dadam, Germany
Jörg Desel, Germany (Co-chair)
Jan Dietz, The Netherlands
Susanna Donatelli, Italy
Schahram Dustdar, Austria
Chiara Francalanci, Italy
Dimitrios Georgakopoulos, USA
Claude Godart, France
Kees van Hee, The Netherlands
Arthur ter Hofstede, Australia
Geert-Jan Houben, The Netherlands
Stefan Jablonski, Germany
Gerti Kappel, Austria
Ekkart Kindler, Germany
Akhil Kumar, USA
Ronald M. Lee, USA
Dan C. Marinescu, USA
Massimo Mecella, Italy
Andreas Oberweis, Germany
Barbara Pernici, Italy (Co-chair)
Manfred Reichert, Germany
Colette Rolland, France
Michael Rosemann, Australia
Heiko Schuldt, Austria
Edward Stohr, USA
Gottfried Vossen, Germany
Mathias Weske, Germany (Co-chair)
Leon Zhao, USA

Referees

Michael Adams
Rainer Anzböck
Danilo Ardagna
Xin Bai
Donald Baker
Giuseppe Berio
Sami Bhiri
Jaap Boender
Cinzia Cappiello
Gerome Canals
Eugenio Capra
François Charoy
Andrzej Cichocki
Vincent Chevrier
Enzo Colombo
Fabio De Rosa
Antonio Di Leva
Sebastian Eichholz
Pascal Fenkam
Thomas Gschwind
Rachid Hamadi
Bodo Huesemann
Alexander Kaiser
Markus Kalb
Markus Klemen
Agnes Koschmider
Jens Lechtenboerger
Kirsten Lenz
Beate List
Emily (Rong) Liu
Udo Mayer
Christian Meiler
Kees van der Meer
Marco von Mevius
Sascha Mueller
Amedeo Napoli
Phillipa Oaks
Ilia Petrov
Olivier Perrin
Frank Puhlmann
Stefanie Rinderle
Nick Russell
Monica Scannapieco
Orit Schwartz
Natalia Sidorova
Carla Simone
Quan Z. Sheng
Halvard Skogsrud
Justin O'Sullivan
Alexander Tararbrin
Farouk Toumani
Marc Voorhoeve
Liangzhao Zeng
Henricus M.W. Verbeek
Marc Voorhoeve
Peter Westerkamp
Darrell Woel
Moe Wynn

Table of Contents

Consistency in Model Integration

Kees van Hee, Natalia Sidorova, Lou Somers, and Marc Voorhoeve

Eindhoven University of Technology, Dept. Math and Comp. Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{k.m.v.hee,n.sidorova,l.j.a.m.somers,m.voorhoeve}@tue.nl

Abstract. We present a UML-inspired approach to modeling and analysis of complex systems. Different stakeholders of a system may have different views, modeled with different techniques. It is essential that the various aspect models (use cases and life cycles) provide a complete and consistent description of the total system. Our approach based on the composition and decomposition of (colored) Petri nets allows the integration of aspect models. We illustrate our approach by a case study.

1 Introduction

The analysis and engineering of complex systems cannot be performed by a single person. So, several system architects are involved, modeling various subsystems. Also the system will have several stakeholders with different views, which also requires various models for being able to validate the proposals of the architects. Different aspect models require different modeling techniques. UML [3] offers a wide range of such techniques, most of them being diagram techniques. A UML description of a moderate-size system contains hundreds of diagrams of various kinds. Each diagram models one or more aspects of the considered system. By concentrating on a few aspects at a time, validation by stakeholders becomes possible.

As the project proceeds, the aspect models will be integrated, while adding detail and rigor. This may lead to the discovery of inconsistencies. Early detection of such inconsistencies will help to reduce development costs, so the software industry is hard-pressed for methods to determine and preserve the consistency between the various models. We believe that there is no "silver bullet" for achieving this. The "honest" way is by a single model that integrates all aspects modeled so far. From this integrated model, the aspect models are derived as projections. If and only if such an integrated model can be found, the models made so far are consistent.

In this paper, we indicate how integrated models can be derived from aspect models in early stages of the development process. The key ingredients are Petri nets with synchronization and projection operators. We start with various aspect models that are combined by synchronization, resulting in an integrated model. From it, scenario's can be derived by projection in order to allow validation. There exist many tools that support such an approach, some of them, e.g. ExSpect [7] and CPN [9], allow to add pre- and postconditions, resulting in a functional prototype.

© Springer-Verlag Berlin Heidelberg 2004

此为试读，需要完整PDF请访问：www.ertongbook.com

The synchronization operator is closely related to the call mechanism for methods; the model thus can be used to support the design and implementation phases. We illustrate our proposal with a case study of the well-known library system, which is just large enough to illustrate the key aspects of our method.

In section 2, we introduce WF nets, a subclass of Petri nets used for our models and our operators for composing and decomposing them. In section 3, we describe the modeling, verification, and validation process. In section 4, we illustrate our process with a library case study, and we also show how our models can be extended, adding more functionality. We conclude with a comparison with related work.

2 Petri Net Models, Synchronization, and Projection

We assume the reader has some knowledge of "classical" place-transition nets (bipartite directed graphs), markings (distributions of tokens in places) and the interleaving firing rule. A transition may fire in marking M iff it can consume the necessary tokens from M; as a result of this firing, a new marking M' is reached, consisting of M with the consumed tokens removed and produced tokens added. A net defines a reachability relation between its markings: a marking M' is reachable from a marking M iff a finite sequence of firings exists starting in M and ending in M'.

Marked nets are too general for modeling. In [1], the class of ***WF*** (workflow) nets is defined, which can be compared to UML activity diagrams. A WF net possesses a unique source and a unique sink place. Every node of a WF net, seen as a directed graph, lies on a directed path from the source to the sink place. A WF net possesses an initial marking (one token in the source place) and a final marking (one token in the sink place). It is ***sound*** iff (1) from the initial marking it is possible for every transition t to reach a marking where t may fire and (2) the final marking is reachable from any marking M that is reachable from the initial marking. Sound WF nets are the very nets used for modeling use cases and object life cycles, c.f. [4].

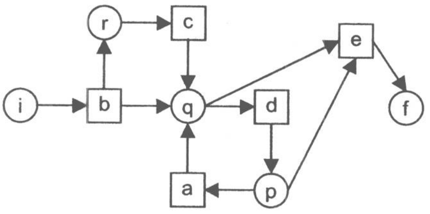

Fig. 1. Example of a WF net

In Fig. 1 a WF net is depicted. The places i,f are respectively the source and sink places. This net is sound, which can be verified by examining the reachable markings. For example, from the initial marking, a marking can be reached with only two tokens in p (e.g. by firing b, then c then twice d). From this marking, it is possible to reach the final marking by firing a, then e.

For convenience, we omit in this paper the source and sink place from the WF nets. Thus a WF net has one or more start transitions (without input places) and end transitions (without output places). The firing of transitions models the occurrence of events. If a transition bordering the source place fires, a "create" event occurs, since in a sound net this transition does not consume any tokens. Analogously, destroy events are firings of transitions bordering the sink place. The soundness property now states that whatever is caused by a single creation can eventually be undone by a single destruction.

For Petri nets there are several methods for analyzing the behavior. Some of these methods use only the structure of the Petri net and not the underlying state space. T-invariants provide such a method. A T-invariant can be computed by standard linear algebra and can be related to sequences of transitions that return the system to the state before the sequence was executed. We use T-invariants in the validation process.

The tokens in the places refer to objects; every place contains references to objects of one and the same class. Initially, we abstract from the attributes of the objects, allowing "classical" analysis of our nets. Eventually, our models will consist of high-level nets, e.g. colored nets [9], specifying pre- and postconditions for the firing of transitions. A transition will fire only if its consumption satisfies the precondition; it will then produce tokens in accordance with the postcondition.

We add operators for composing and decomposing net models, which are essential for the integration of models and for checking their consistency. The composition operator is called ***synchronization*** and is indicated by a dotted line connecting two transitions. When transitions synchronize in a high-level net, data may be exchanged in either direction. In Fig. 2, an example net with synchronization is shown at the left. The synchronization result is the net in the middle, which is obtained by transition fusion: the transitions participating in a synchronization are glued together. This mechanism resembles the synchronization within process calculi like CCS [10]. Synchronization between sound nets does not always result in a sound net; the middle net in Fig. 2 is not sound, since transition *cd* cannot fire.

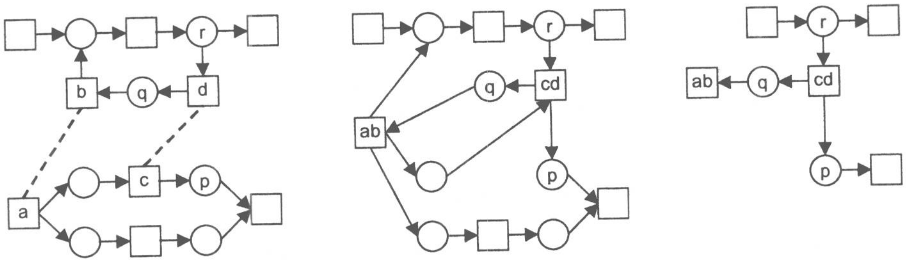

Fig. 2. Example of synchronization and projection

The decomposition operator is called ***projection***. Projection of a net w.r.t. a subset S of the net's places is obtained by removing all the places not in S plus the edges leading to and from them. Transitions that become isolated are removed as well. The right-hand net shows the projection of the middle net w.r.t. the set $\{p,q,r\}$. If N is a connected net, P its set of places, and $S \subseteq P$, then N can be obtained by synchronizing its projections w.r.t. S and $P \setminus S$.

When creating WF nets for use cases, the transitions describe ***events*** that can occur. If a net models an object life cycle, the transitions represent ***methods*** of the object's class. In the design phase, the synchronization of methods will result in one of them calling the other. The following rule describes the essence of our approach: deriving an integrated model from aspect models and checking their consistency:

- The integrated model is derived from the aspect models (use cases and life cycles) by synchronization.
- All aspect models should be derivable from the integrated model by projection.

3 Modeling Process

We focus on deriving an integrated logical model that captures the functionality of the system, we have left out other engineering activities. In general, we have a succession of ***elicitation, modeling, verification*** and ***validation*** steps. We split the modeling step into three steps: ***process*** modeling, ***data*** modeling, and ***transformation*** modeling. In the elicitation steps the stakeholders play an important role. There are several techniques to obtain useful information from a group of stakeholders. Well-known are "brown paper sessions" where stakeholders write down individually the most important items, like issues, functions, scenario's, or objects. These items are stuck to a brown paper board and grouped by the moderator into related groups. Then the items are discussed and terminology is fixed. These sessions are repeated with different topics. Group decision support systems [11] provide computerized support. The modeling step is done by system architects, who also perform verification, possibly "on the fly" during modeling using "correctness by construction", sometimes after modeling (like verifying the integrated model). After modeling and verification comes validation with the help of stakeholders. As a result, a redesign may be needed.

The modeling, verification and validation steps are iterated until the stakeholders are satisfied with the logical model. At some stage when use cases have become stable, user interface designers can start to define screens containing forms and buttons. After having established the logical model, it is extended to accommodate for the designed user interface. We will describe the successive phases and steps in more detail. Remember that stakeholders are involved in phases 1 and 6 only.

Step 1: Elicitation

(a) Make a list of use cases, indicated by a name and some additional comments by the stakeholders.
(b) Define some allowed and explicitly forbidden scenario's (event sequences) for each use case.
(c) Identify the classes of objects that play a role in the scenario's.
(d) List relationships between object classes. The existence of these relationships is triggered by use case events that involve more than one object.
(e) Collect relevant attributes for the objects.
(f) Find static constraints that the system's state (the set of all living objects) should satisfy at all time.

Step 2: Process Modeling

(a) Create WF nets for the use cases. Each WF net should combine the allowed scenarios for one use case and disallow the forbidden ones.

(b) Create WF nets for the object life cycles. The transitions are the methods of the classes.

(c) Integrate the workflows by identifying the transitions in use cases and object life cycles that must be synchronized. If necessary, adapt use cases and/or life cycles.

Step 3: Data Modeling

(a) Construct the class model with relationships and attributes. We prefer functional relationships.

(b) Formalize the static constraints. Use logical predicates that can be translated back into natural language with increased precision. Add other common-sense static constraints.

(c) Define global variables. For each object class we define a global variable, called object store or object file. All objects that are active in the system reside in an object store. Also, other global variables like the current date or time are defined.

Step 4: Transformation Modeling

(a) Combine the process model and the data model. Establish the relationships between object classes and methods. For each class we determine whether the methods create, read, update, or destroy objects from it (a CRUD-matrix).

(b) Determine the input and output parameters of the methods: places, global variables and additional parameters, e.g. for the user interface.

(c) Determine pre- and postconditions of the methods. The end product is the high-level integrated model.

Step 5: Verification

(a) Check the soundness of all workflows: use cases, object life cycles, and the integrated model.

(b) Check that all use case nets can be derived from the integrated model by projection.

(c) Check that each relationship in the class model is created somewhere.

(d) Check the preservation of the static constraints. Some constraints may be temporarily violated during the execution of a certain sequence of transitions (a transaction) but they should be valid after the transaction.

(e) If necessary, return to modeling.

Step 6: Validation

(a) Validate the integrated model by spawning new scenarios from T-invariants of the nets.

(b) Validate all static constraints.
(c) Present the scenario's with data transformations added.
(d) If necessary, return to one of the modeling steps.

Step 7: User Interface Integration
(a) Make additional classes and methods to accommodate the user interface.
(b) Synchronize the additional methods with the existing ones. If necessary, adapt the logical model.

The steps are not executed in the order presented; it is important that verifications and validations are effectuated as soon as possible in order to reduce costs. For example, step 5a should immediately succeed steps 2abc for the modeled WF nets. Usually, the nets created in 2ab can be verified by hand; the net in 2c often needs tool support [14]. Step 6a can succeed step 2c after verification. Indeed, we have drawn a rather sizable WF net depicting the described process. Afterwards, the logical model is translated into specifications for software components. These components can be constructed from scratch or the can be assembled from existing components. For component selection, the scenario's are helpful.

The steps above apply to systems of moderate size. Large systems should be split into subsystems to which the above steps apply. By synchronization the subsystems are integrated as suggested in section 5 of this paper. This extra integration step should be verified and validated similarly to the description above, concentrating on the interface between the subsystems. We will illustrate the above approach with an example case study.

4 Case Study: A Library System

In the case study we consider a more or less standard library system. Stakeholders are personnel and members that lend books. Several copies of the same book may exist. Members can make reservations for books that are not available. We focus on the modeling steps, in particular the process modeling step. Therefore we treat the other steps rather superficially.

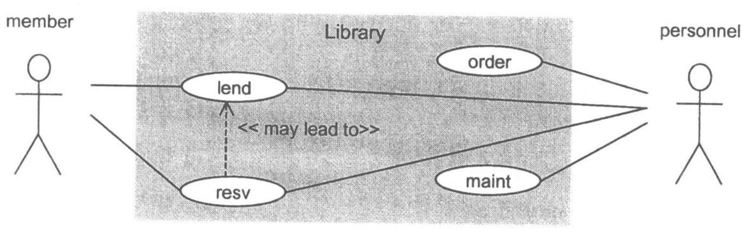

Fig. 3. Library use case diagram

4.1 Elicitation

Fig. 3 depicts typical use cases like lending a book, reserving and then lending a book, ordering books, and maintaining the member file and book catalogue. In Fig. 4 a loan/reserve use case net is given. The initial transition (event) is *s*, which creates a token in place *b* denoting the reservation by a member of a book in the catalogue. If the book is available, a loan is started (transition l_1). If the book is not available, the token stays in place *b* and if a matching book is returned, the reservation object can go to the notified state *d* by transition *n* (notification). From this state, transition l_2 can occur resulting in a loan (a token in *f*). A lent book can get lost (transition *lo*) or it will be returned (transition *re*).

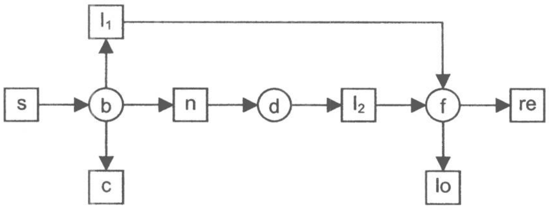

Fig. 4. Request / lend / reserve use case

Similar use cases can be found for maintenance and ordering activities. This is as complicated as it gets in our library case, but for other systems a use case may exhibit concurrent behavior, so it may have states that are distributed over various places. So far we encountered two object classes, reservations in places *b,d,* and loans in place *f*. When treating the other use cases, we encounter members, book orders, book copies, and book titles. It is necessary to distinguish book titles and copies, since several copies can exist for the same title. We determine the following classes (see Fig. 5):

MEM	library member
RSV	reservation **of** title **by** member
LOAN	loan **of** copy **by** member
TITLE	book title
BCPY	copy **of** title
ORD	order **of** title

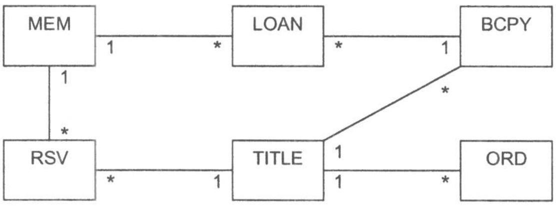

Fig. 5. Relations between object classes

此为试读，需要完整PDF请访问：www.ertongbook.com