Jörg Desel
Wolfgang Reisig
Grzegorz Rozenberg (Eds.)

LNCS 3098 Tutorial

Lectures on Concurrency and Petri Nets

Advances in Petri Nets

Springer

Jörg Desel Wolfgang Reisig
Grzegorz Rozenberg (Eds.)

Lectures on Concurrency and Petri Nets

Advances in Petri Nets

Springer

Volume Editors

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

Wolfgang Reisig
Humboldt-Universität zu Berlin
Institut für Informatik
Unter den Linden 6, 10099 Berlin, Germany
E-mail: reisig@informatik.hu-berlin.de

Grzegorz Rozenberg
Leiden University
Leiden Institute of Advanced Computer Science (LIACS)
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
E-mail: rozenberg@liacs.nl

Library of Congress Control Number: Applied for

CR Subject Classification (1998): F.1, F.2, F.3, C.2.4, C.2, H.3, H.4, C.1

ISSN 0302-9743
ISBN 3-540-22261-8 Springer-Verlag Berlin Heidelberg New York

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

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11013105 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3098

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

Editorial Board

Lecture Notes in Computer Science

For information about Vols. 1–3005

please contact your bookseller or Springer-Verlag

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004.

Vol. 3109: S.C. Sahinalp, S. Muthukrishnan, U. Dogrusoz (Eds.), Combinatorial Pattern Matching. XII, 486 pages. 2004.

Vol. 3105: S. Göbel, U. Spierling, A. Hoffmann, I. Iurgel, O. Schneider, J. Dechau, A. Feix (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. XVI, 304 pages. 2004.

Vol. 3099: J. Cortadella, W. Reisig (Eds.), Applications and Theory of Petri Nets 2004. XI, 505 pages. 2004.

Vol. 3098: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets. VIII, 849 pages. 2004.

Vol. 3096: G. Melnik, H. Holz (Eds.), Advances in Learning Software Organizations. X, 173 pages. 2004.

Vol. 3094: A. Nürnberger, M. Detyniecki (Eds.), Adaptive Multimedia Retrieval. VIII, 229 pages. 2004.

Vol. 3093: S.K. Katsikas, S. Gritzalis, J. Lopez (Eds.), Public Key Infrastructure. XIII, 380 pages. 2004.

Vol. 3092: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering. XVI, 358 pages. 2004.

Vol. 3091: V. van Oostrom (Ed.), Rewriting Techniques and Applications. X, 313 pages. 2004.

Vol. 3089: M. Jakobsson, M. Yung, J. Zhou (Eds.), Applied Cryptography and Network Security. XIV, 510 pages. 2004.

Vol. 3086: M. Odersky (Ed.), ECOOP 2004 – Object-Oriented Programming. XIII, 611 pages. 2004.

Vol. 3085: S. Berardi, M. Coppo, F. Damiani (Eds.), Types for Proofs and Programs. X, 409 pages. 2004.

Vol. 3084: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering. XIV, 596 pages. 2004.

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 pages. 2004.

Vol. 3079: Z. Mammeri, P. Lorenz (Eds.), High Speed Networks and Multimedia Communications. XVIII, 1103 pages. 2004.

Vol. 3078: S. Cotin, D.N. Metaxas (Eds.), Medical Simulation. XVI, 296 pages. 2004.

Vol. 3077: F. Roli, J. Kittler, T. Windeatt (Eds.), Multiple Classifier Systems. XII, 386 pages. 2004.

Vol. 3076: D. Buell (Ed.), Algorithmic Number Theory. XI, 451 pages. 2004.

Vol. 3074: B. Kuijpers, P. Revesz (Eds.), Constraint Databases and Applications. XII, 181 pages. 2004.

Vol. 3073: H. Chen, R. Moore, D.D. Zeng, J. Leavitt (Eds.), Intelligence and Security Informatics. XV, 536 pages. 2004.

Vol. 3072: D. Zhang, A.K. Jain (Eds.), Biometric Authentication. XVII, 800 pages. 2004.

Vol. 3070: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004. XXV, 1208 pages. 2004. (Subseries LNAI).

Vol. 3068: E. André, L. Dybkj{\}ae r, W. Minker, P. Heisterkamp (Eds.), Affective Dialogue Systems. XII, 324 pages. 2004. (Subseries LNAI).

Vol. 3067: M. Dastani, J. Dix, A. El Fallah-Seghrouchni (Eds.), Programming Multi-Agent Systems. X, 221 pages. 2004. (Subseries LNAI).

Vol. 3066: S. Tsumoto, R. S lowiński, J. Komorowski, J.W. Grzymala-Busse (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004. (Subseries LNAI).

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic in Computer Science. X, 275 pages. 2004. (Subseries LNAI).

Vol. 3064: D. Bienstock, G. Nemhauser (Eds.), Integer Programming and Combinatorial Optimization. XI, 445 pages. 2004.

Vol. 3063: A. Llamosí, A. Strohmeier (Eds.), Reliable Software Technologies - Ada-Europe 2004. XIII, 333 pages. 2004.

Vol. 3062: J.L. Pfaltz, M. Nagl, B. Böhlen (Eds.), Applications of Graph Transformations with Industrial Relevance. XV, 500 pages. 2004.

Vol. 3061: F.F. Ramas, H. Unger, V. Larios (Eds.), Advanced Distributed Systems. VIII, 285 pages. 2004.

Vol. 3060: A.Y. Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004. (Subseries LNAI).

Vol. 3059: C.C. Ribeiro, S.L. Martins (Eds.), Experimental and Efficient Algorithms. X, 586 pages. 2004.

Vol. 3058: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 233 pages. 2004.

Vol. 3057: B. Jayaraman (Ed.), Practical Aspects of Declarative Languages. VIII, 255 pages. 2004.

Vol. 3056: H. Dai, R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining. XIX, 713 pages. 2004. (Subseries LNAI).

Vol. 3055: H. Christiansen, M.-S. Hacid, T. Andreasen, H.L. Larsen (Eds.), Flexible Query Answering Systems. X, 500 pages. 2004. (Subseries LNAI).

Vol. 3054: I. Crnkovic, J.A. Stafford, H.W. Schmidt, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 311 pages. 2004.

Vol. 3053: C. Bussler, J. Davies, D. Fensel, R. Studer (Eds.), The Semantic Web: Research and Applications. XIII, 490 pages. 2004.

Vol. 3052: W. Zimmermann, B. Thalheim (Eds.), Abstract State Machines 2004. Advances in Theory and Practice. XII, 235 pages. 2004.

Vol. 3051: R. Berghammer, B. Möller, G. Struth (Eds.), Relational and Kleene-Algebraic Methods in Computer Science. X, 279 pages. 2004.

Vol. 3050: J. Domingo-Ferrer, V. Torra (Eds.), Privacy in Statistical Databases. IX, 367 pages. 2004.

Vol. 3049: M. Bruynooghe, K.-K. Lau (Eds.), Program Development in Computational Logic. VIII, 539 pages. 2004.

Vol. 3047: F. Oquendo, B. Warboys, R. Morrison (Eds.), Software Architecture. X, 279 pages. 2004.

Vol. 3046: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1016 pages. 2004.

Vol. 3045: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1040 pages. 2004.

Vol. 3044: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1140 pages. 2004.

Vol. 3043: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1180 pages. 2004.

Vol. 3042: N. Mitrou, K. Kontovasilis, G.N. Rouskas, I. Iliadis, L. Merakos (Eds.), NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications. XXXIII, 1519 pages. 2004.

Vol. 3040: R. Conejo, M. Urretavizcaya, J.-L. Pérez-de-la-Cruz (Eds.), Current Topics in Artificial Intelligence. XIV, 689 pages. 2004. (Subseries LNAI).

Vol. 3039: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 1271 pages. 2004.

Vol. 3038: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 1311 pages. 2004.

Vol. 3037: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 745 pages. 2004.

Vol. 3036: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 713 pages. 2004.

Vol. 3035: M.A. Wimmer (Ed.), Knowledge Management in Electronic Government. XII, 326 pages. 2004. (Subseries LNAI).

Vol. 3034: J. Favela, E. Menasalvas, E. Chávez (Eds.), Advances in Web Intelligence. XIII, 227 pages. 2004. (Subseries LNAI).

Vol. 3033: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVIII, 1076 pages. 2004.

Vol. 3032: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVII, 1112 pages. 2004.

Vol. 3031: A. Butz, A. Krüger, P. Olivier (Eds.), Smart Graphics. X, 165 pages. 2004.

Vol. 3030: P. Giorgini, B. Henderson-Sellers, M. Winikoff (Eds.), Agent-Oriented Information Systems. XIV, 207 pages. 2004. (Subseries LNAI).

Vol. 3029: B. Orchard, C. Yang, M. Ali (Eds.), Innovations in Applied Artificial Intelligence. XXI, 1272 pages. 2004. (Subseries LNAI).

Vol. 3028: D. Neuenschwander, Probabilistic and Statistical Methods in Cryptology. X, 158 pages. 2004.

Vol. 3027: C. Cachin, J. Camenisch (Eds.), Advances in Cryptology - EUROCRYPT 2004. XI, 628 pages. 2004.

Vol. 3026: C. Ramamoorthy, R. Lee, K.W. Lee (Eds.), Software Engineering Research and Applications. XV, 377 pages. 2004.

Vol. 3025: G.A. Vouros, T. Panayiotopoulos (Eds.), Methods and Applications of Artificial Intelligence. XV, 546 pages. 2004. (Subseries LNAI).

Vol. 3024: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3023: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 611 pages. 2004.

Vol. 3022: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3021: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 633 pages. 2004.

Vol. 3019: R. Wyrzykowski, J.J. Dongarra, M. Paprzycki, J. Wasniewski (Eds.), Parallel Processing and Applied Mathematics. XIX, 1174 pages. 2004.

Vol. 3018: M. Bruynooghe (Ed.), Logic Based Program Synthesis and Transformation. X, 233 pages. 2004.

Vol. 3017: B. Roy, W. Meier (Eds.), Fast Software Encryption. XI, 485 pages. 2004.

Vol. 3016: C. Lengauer, D. Batory, C. Consel, M. Odersky (Eds.), Domain-Specific Program Generation. XII, 325 pages. 2004.

Vol. 3015: C. Barakat, I. Pratt (Eds.), Passive and Active Network Measurement. XI, 300 pages. 2004.

Vol. 3014: F. van der Linden (Ed.), Software Product-Family Engineering. IX, 486 pages. 2004.

Vol. 3012: K. Kurumatani, S.-H. Chen, A. Ohuchi (Eds.), Multi-Agnets for Mass User Support. X, 217 pages. 2004. (Subseries LNAI).

Vol. 3011: J.-C. Régin, M. Rueher (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. XI, 415 pages. 2004.

Vol. 3010: K.R. Apt, F. Fages, F. Rossi, P. Szeredi, J. Váncza (Eds.), Recent Advances in Constraints. VIII, 285 pages. 2004. (Subseries LNAI).

Vol. 3009: F. Bomarius, H. Iida (Eds.), Product Focused Software Process Improvement. XIV, 584 pages. 2004.

Vol. 3008: S. Heuel, Uncertain Projective Geometry. XVII, 205 pages. 2004.

Vol. 3007: J.X. Yu, X. Lin, H. Lu, Y. Zhang (Eds.), Advanced Web Technologies and Applications. XXII, 936 pages. 2004.

Vol. 3006: M. Matsui, R. Zuccherato (Eds.), Selected Areas in Cryptography. XI, 361 pages. 2004.

Preface

The very first model of concurrent and distributed systems was introduced by C.A. Petri in his seminal Ph.D. thesis in 1964. Petri nets has remained a central model for concurrent systems for 40 years, and they are often used as a yardstick for other models of concurrency. As a matter of fact, many other models have been developed since then, and this research area is flourishing today.

The goal of the 4th Advanced Course on Petri Nets held in Eichstätt, Germany in September 2003 was to present applications and the theory of Petri Nets in the context of a whole range of other models. We believe that in this way the participants of the course received a broad and in-depth picture of research in concurrent and distributed systems.

It is also the goal of this volume to convey this picture. The volume is based on lectures given at the Advanced Course, but in order to provide a balanced presentation of the field, some of the lectures are not included, and some material not presented in Eichstätt is covered here. In particular, a series of introductory lectures was not included in this volume, as the material they covered is well established by now, and well presented elsewhere (e.g., in W. Reisig and G. Rozenberg, eds., "Lectures on Petri Nets," LNCS 1491, 1492, Springer-Verlag, 1997 – these two volumes are based on the 3rd Advanced Course on Petri Nets).

We believe that this volume will be useful as both a reference and a study book for the reader who is interested in obtaining an up-to-date overview of research in concurrent and distributed systems. It will be also useful for the reader who is specifically interested in Petri nets. Although the material presented in this volume is based on the Eichstätt course, the papers included here were written after the course, and therefore they have taken into account numerous comments made by the participants and fellow lecturers during the course. Because of this, and because, to start with, the lecturers were asked to present their material in a tutorial fashion, this volume is very suitable as an auxiliary reading for courses on concurrency and/or Petri nets, and especially useful as the underlying book for a seminar covering this research area.

Acknowledgements
The editors are indebted to the participants and the lecturers of the Advanced Course in Eichstätt for their enthusiastic participation in the course and for their constructive criticism of the presented material both during and after the course. We are also very much indebted to the authors of all the papers for their effort in producing all the material included here. We are grateful for the support of the Katholische Universität Eichstätt, hosting the Advanced Course, and to the Deutsche Forschungsgemeinschaft for its financial support. Many thanks to Springer-Verlag, and in particular to Mr. A. Hofmann, for the pleasant and efficient cooperation in producing this volume.

April 2004

Jörg Desel, Eichstätt
Wolfgang Reisig, Berlin
Grzegorz Rozenberg, Leiden

Table of Contents

Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow 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. Over the last decade there has been a shift from "data-aware" information systems to "process-aware" information systems. To support business processes an enterprise information system needs to be aware of these processes and their organizational context. Business Process Management (BPM) includes methods, techniques, and tools to support the design, enactment, management, and analysis of such operational business processes. BPM can be considered as an extension of classical Workflow Management (WFM) systems and approaches. This tutorial introduces models, systems, and standards for the design, analysis, and enactment of workflow processes. Petri nets are used for the modeling and analysis of workflows. Using Petri nets as a formal basis, contemporary systems, languages, and standards for BPM and WFM are discussed. Although it is clear that Petri nets can serve as a solid foundation for BPM/WFM technology, in reality systems, languages, and standards are developed in an ad-hoc fashion. To illustrate this XPDL, the "Lingua Franca" proposed by the Workflow Management Coalition (WfMC), is analyzed using a set of 20 basic workflow patterns. This analysis exposes some of the typical semantic problems restricting the application of BPM/WFM technology.

Keywords: Business process management, Workflow management, Workflow management systems, Workflow patterns, XML Process Definition Language (XPDL), Workflow verification.

1 Introduction

This section provides some context for the topics addressed in this tutorial. First, we identify some trends and put them in a historical perspective. Then, we focus on the BPM life-cycle and discuss the basic functionality of a WFM system. Finally, we outline the remainder of this tutorial.

1.1 Historical Perspective

To show the relevance of Business Process Management (BPM) systems, it is interesting to put them in a historical perspective. Consider Figure 1, which shows some of the

J. Desel, W. Reisig, and G. Rozenberg (Eds.): ACPN 2003, LNCS 3098, pp. 1–65, 2004.

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.

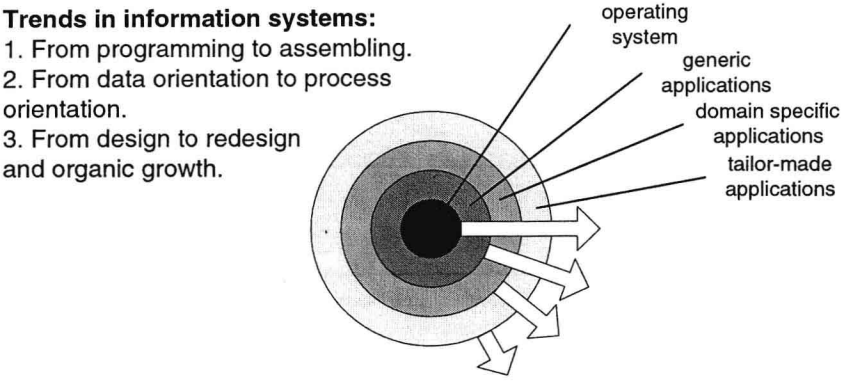

Fig. 1. Trends relevant for business process management.

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.

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 BPM. BPM 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 BPM systems residing in the second layer are WorkFlow Management (WFM) systems [12, 38, 48, 55, 57, 58, 61] such as Staffware, MQSeries, and COSA, and case handling systems such as FLOWer. Note that leading Enterprise Resource Planning (ERP) systems populating the third layer also offer a WFM module. The workflow engines of SAP, Baan, PeopleSoft, Oracle, and JD Edwards can be considered as integrated BPM systems. The idea to isolate the management of business processes in a separate component is consistent with the three trends identified. BPM 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 WFM systems can be used to integrate existing applications and support process change by merely changing the workflow diagram. Given these observations, the practical relevance of BPM is evident. Although BPM functionality is omnipresent and often hidden in larger enterprise information systems, for clarity we will often restrict the discussion to clear cut "process-aware" information systems such as WFM systems (cf. Section 1.3).

To put the topic of this tutorial in a historical perspective it is worthwhile to consider the early work on office information systems. In the seventies, people like Skip Ellis [32], Anatol Holt [45], and Michael Zisman [78] 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 WFM systems developed in the past decade and the many papers on workflow technology illustrate the revival of office information systems. Today WFM systems are readily available [12, 38, 48, 55, 57, 58, 61]. 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 [33]. 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 WFM systems enforce unnecessary constrains on the process logic (e.g., processes are made more sequential).

1.2 BPM Life-Cycle

As indicated before, *Business Process Management (BPM) includes methods, techniques, and tools to support the design, enactment, management, and analysis of operational business processes.* It can be considered as an extension of classical Workflow Management (WFM) systems and approaches. Before discussing the differences between WFM and BPM, let us consider the *BPM life-cycle.*

The BPM life-cycle has four phases:

- *Process design*
  Any BPM effort requires the modeling of an existing ("as-is") or desired ("to-be") process, i.e., a *process design.* During this phase process models including various perspectives (control-flow, data-flow, organizational, sociotechnical, and operational aspects) are constructed. The only way to create a "process-aware" enterprise information system is to add knowledge about the operational processes at hand.
- *System configuration*
  Based on a process design, the process-aware enterprise information system is realized. In the traditional setting the realization would require a time-consuming and complex software development process. Using software from the second and third layer shown in Figure 1, the traditional software development process is replaced by a configuration or assembly process. Therefore, we use the term *system configuration* for the phase in-between process design and enactment.
- *Process enactment*
  The *process enactment* phase is the phase where the process-aware enterprise information system realized in the system configuration phase is actually used.
- *Diagnosis*
  Process-aware enterprise information system have to change over time to improve performance, exploit new technologies, support new processes, and adapt to an ever changing environment. Therefore, the *diagnosis* phase is linking the process enactment phase to the a new design phase.

Like in software life-cycle models, the four phases are overlapping (cf. Waterfall model) and the whole process is iterative (cf. Spiral model).

As is illustrated in Figure 2, the BPM life-cycle can be used to identify different levels of maturity when it comes to developing process-aware enterprise information systems. In the early nineties and before, most information systems only automated individual activities and where unaware of the underlying process. For the systems that were process-aware, the process logic was hard-coded in the system and not supported in a generic manner. Despite the early work on office automation, the first commercial WFM systems became only practically relevant around 1993 (see Figure 2(a)). The focus of these systems was on "getting the system to work" and support for enactment and design was limited. In the mid-nineties this situation changed and by 1998 many WFM

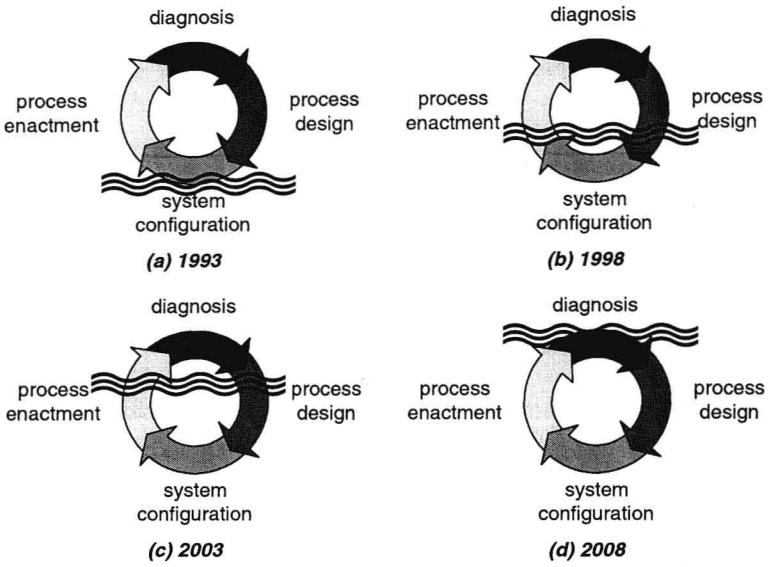

Fig. 2. The BPM life-cycle is used to indicate the maturity of BPM technology over time.

systems had become readily available (see Figure 2(b)). In these systems there was basic support for enactment and design. In the last five years these systems have been further extended allowing for more support during the design and enactment phases (see Figure 2(c)). For example, a case-handling system like FLOWer [22] allows for much more flexibility during the enactment phase than the traditional WFM systems. Today's systems provide hardly any support for the diagnosis phase. Although most BPM software logs all kinds of events (e.g., WFM systems like Staffware log the completion of activities and ERP systems like SAP log transactions), this information is not used to identify problems or opportunities for improvement. In the next five years this situation will probably change when process mining [17, 19] techniques become readily available (see Figure 2(d)).

The BPM life-cycle shown in Figure 2 can also be used to define the difference between WFM and BPM. WFM focusses on the lower half of the BPM life-cycle (i.e., "getting the system to work") while BPM also includes to upper half of the life-cycle. Therefore, BPM also focusses on diagnosis, flexibility, human-centric processes, goal-driven process design, etc. Gartner expects that *Business Process Analysis* (BPA), i.e., software to support the diagnosis phase, will become increasingly important [39]. It is expected that the BPA market will continue to grow. Note that BPA covers aspects neglected by traditional WFM products (e.g., diagnosis, simulation, etc.). *Business Activity Monitoring* (BAM) is one of the emerging areas in BPA. The goal of BAM tools is to use data logged by the information system to diagnose the operational processes. An example is the ARIS Process Performance Manager (PPM) of IDS Scheer [47]. ARIS PPM extracts information from audit trails (i.e., information logged during the execution of cases) and displays this information in a graphical way (e.g., flow times, bottlenecks, utilization, etc.). BAM also includes process mining, i.e., extracting pro-

cess models from logs [17]. BAM creates a number of scientific and practical challenges (e.g., which processes can be discovered and how much data is needed to provide useful information).

1.3 Workflow Management (Systems)

The focus of this tutorial will be on WFM rather than BPM. The reason is that WFM serves as a basis for BPM and in contrast to BPM it is a mature area with well-defined concepts and widely used software products.

The Workflow Management Coalition (WfMC) defines workflow as: "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules." [55]. A Workflow Management System (WFMS) is defined as: "A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications." [55]. Note that both definitions emphasize the focus on enactment, i.e., the use of software to support the execution of operational processes.

When considering these definitions in more detail it is evident that WFM is highly relevant for any organization. However, at the same time few organizations use a "real" WFM system. To explain this we identify four categories of WFM support:

- *Pure WFM systems*
  At this point in time many WFM systems are available and used in practise. Examples of systems include Staffware Process Suite, FileNET BPM Suite, i-Flow, FLOWer, WebSphere MQ Workflow (formerly known as MQSeries Workflow), TIBCO InConcert, etc.
- *WFM components embedded in other systems*
  Many software packages embed a generic workflow component whose functionality is comparable to the pure WFM systems. For example, most ERP systems provide a workflow component. SAP WebFlow is the workflow component of SAP offering all the functionality typically present in traditional stand-alone WFM products.
- *Custom-made WFM solutions*
  Many organizations, e.g., banks and insurance companies, have chosen not to use a commercially available WFM solution but build an organization-specific solution. These solutions typically only support a subset of the functionality offered by the first two categories. Nevertheless, these systems support the definition and execution of different workflows.
- *Hard-coded WFM solutions*
  The last category refers to the situation were the processes are hard-coded in the applications, i.e., there is no generic workflow support but applications are coupled in such a way that a specific process is supported. The only way to change a process is to change the applications themselves, i.e., unlike the first three categories there is no component that is process-aware. Note that in these hard-coded system an explicit orchestration layer is missing.

At this point in time the majority of business processes are still supported by solutions residing in the third and fourth category. However, the percentage of processes supported by the first two categories is increasing. Moreover, software developers building solutions for the third and fourth category are using the concepts and insights provided by the first two categories. In this context it is interesting to refer to recent developments in the *web services* domain [68]. The functionality of web service composition languages (also referred to as "web service orchestration") like BPEL4WS, BPML, WSCI, WSWSFL, XLANG, etc. is very similar to traditional workflow languages [6, 77].

1.4 Outline and Intended Audience

The goal of this tutorial is to introduce the reader to the theoretical foundations of BPM/WFM using a Petri-net based approach. However, at the same time contemporary systems and languages are presented to provide a balanced view on the application domain.

Section 2 shows the application of Petri nets to workflow modeling. For this purpose, the class of *WorkFlow nets* (WF-nets) is introduced, but also some "syntactical sugaring" is given to facilitate the design of workflows. Section 3 discusses the analysis of workflow models expressed in terms of Petri nets. The focus will be on the verification of WF-nets using classical analysis techniques. Section 4 discusses the typical architecture of a WFM system and discusses contemporary systems. The goal of this section is to show that the step from design to enactment, i.e., the configuration phase (cf. Figure 2), is far from trivial. In Section 5, 20 workflow patterns are used to evaluate the XML Process Definition Language (XPDL), the standard proposed by the Workflow Management Coalition (WfMC). This evaluation illustrates the typical problems workflow designers and implementers are faced with when applying contemporary languages and standards. Section 6 provides an overview of related work. Clearly, only a small subset of the many books and papers on BPM/WFM can be presented, but pointers are given to find relevant material. Finally, Section 7 concludes the tutorial by discussing the role of Petri nets in the BPM/WFM domain.

Note that parts of this tutorial are based on earlier work (cf. [2–6, 12, 15]). For more material the interested reader is referred to [12] and two WWW-sites: one presenting course material (slides, animations, etc.) `http://www.workflowcourse.com` and one on workflow patterns `http://www.workflowpatterns.com`.

This tutorial is intended for people having a basic understanding of Petri nets and interested in the application of Petri nets to problems in the BPM/WFM domain. Sections 2 and 3 are focusing more on the Petri-net side of things while sections 4 and 5 are focusing more on the application domain.

2 Workflow Modeling

In this section, we show how to model workflows in terms of Petri nets. First, we introduce the basic workflow concepts and discuss the various perspectives. Then, we define some basic Petri net notation followed by an introduction to a subclass of Petri nets tailored towards workflow modeling. We conclude this section with an exercise.

2.1 Workflow Concepts and Perspectives

Workflow processes are *case-driven*, i.e., tasks are executed for specific cases. Approving loans, processing insurance claims, billing, processing tax declarations, handling traffic violations and mortgaging, are typical case-driven processes which are often supported by a WFM system. These case-driven processes, also called *workflows*, are marked by three dimensions: (1) the control-flow dimension, (2) the resource dimension, and (3) the case dimension (see Figure 3). The control-flow dimension is concerned with the partial ordering of tasks, i.e., the workflow *process*. The tasks which need to be executed are identified and the routing of cases along these tasks is determined. Conditional, sequential, parallel and iterative routing are typical structures specified in the control-flow dimension. Tasks are executed by resources. Resources are human (e.g., employee) and/or non-human (e.g., device, software, hardware). In the resource dimension these resources are classified by identifying roles (resource classes based on functional characteristics) and organizational units (groups, teams or departments). Both the control-flow dimension and the resource dimension are generic, i.e., they are not tailored towards a specific case. The third dimension of a workflow is concerned with individual cases which are executed according to the process definition (first dimension) by the proper resources (second dimension).

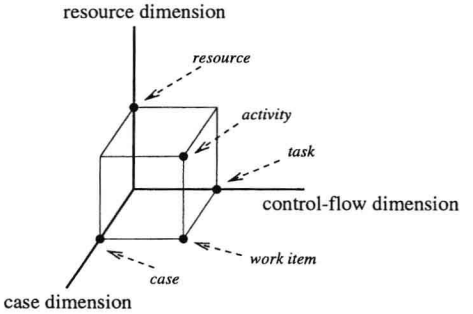

Fig. 3. The three dimensions of workflow.

The primary task of a WFM system is to enact case-driven business processes by joining several perspectives. The following perspectives are relevant for workflow modeling and workflow execution: (1) *control flow* (or process) perspective, (2) *resource* (or organization) perspective, (3) *data* (or information) perspective, (4) *task* (or function) perspective, (5) *operation* (or application) perspective. These perspectives are similar to the perspectives given in [48] and the control flow and resource perspectives correspond to the first two dimensions shown in Figure 3. The third dimension reflects the fact that workflows are case-driven and does not correspond to one of the five perspectives.

In the control-flow perspective, *workflow process definitions* (workflow schemas) are defined to specify which *tasks* need to be executed and in what order (i.e., the routing or control flow). A task is an atomic piece of work. Workflow process definitions are instantiated for specific *cases* (i.e., workflow instances). Since a case is an instantia-