Queueing Networks and Markov Chains

Modeling and Performance Evaluation with Computer Science Applications

Second Edition

Gunter Bolch
Stefan Greiner
Hermann de Meer
Kishor S. Trivedi

WILEY-INTERSCIENCE
A JOHN WILEY & SONS, INC., PUBLICATION

This Page Intentionally Left Blank

Queueing Networks and Markov Chains

This Page Intentionally Left Blank

Queueing Networks and Markov Chains

Modeling and Performance Evaluation with Computer Science Applications

Second Edition

Gunter Bolch
Stefan Greiner
Hermann de Meer
Kishor S. Trivedi

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Queueing networks and Markov chains : modeling and performance evaluation with computer science applications / Gunter Bolch ... [et al.].—2nd rev. and enlarged ed.
p. cm.
"A Wiley-Interscience publication."
Includes bibliographical references and index.
ISBN-13 978-0-471-56525-3 (acid-free paper)
ISBN-10 0-471-56525-3 (acid-free paper)
1. Markov processes. 2. Queuing theory. I. Bolch, Gunter.

QA76.9E94Q48 2006
004.2'401519233—dc22 200506965

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Contents

Preface to the Second Edition

Nearly eight years have passed since the publication of the first edition of this book. In this second edition, we have thoroughly revised all the chapters. Many examples and problems are updated, and many new examples and problems have been added. A significant addition is a new chapter on simulation methods and applications. Application to current topics such as wireless system performance, Internet performance, J2EE applications, and Kanban systems performance are added. New material on non-Markovian and fluid stochastic Petri nets, along with solution techniques for Markov regenerative processes, is added. Topics that are covered briefly include self-similarity, large deviation theory, and diffusion approximation. The topic of hierarchical and fixed-point iterative models is also covered briefly. Our collective research experience and the application of these methods in practice for the past 30 years (at the time of writing) have been distilled in these chapters as much as possible. We hope that the book will be of use as a classroom textbook as well as of use for practicing engineers. Researchers will also find valuable information here.

We wish to thank many of our current students and former postdoctoral associates:

- Dr. Jörg Barner, for his contribution of the methodological background section in Chapter 1. He again supported us a lot in laying out the chapters and producing and improving figures and plots and with intensive proofreading.

- Pawan Choudhary, who helped considerably with the simulation chapter, Dr. Dharmaraja Selvamuthu helped with the section on SHARPE, and Dr. Hairong Sun helped in reading several chapters.

- Felix Engelhard, who was responsible for the new or extended sections on distributions, parameter estimation, Petri nets, and non-Markovian systems. He also did a thorough proofreading.

- Patrick Wüchner, for preparing the sections on matrix-analytic and matrix-geometric methods as well as the MMPP and MAP sections, and also for intensive proofreading.

- Dr. Michael Frank, who wrote and extended several sections: batch system and networks, summation method, and Kanban systems.

- Lassaad Essafi, who wrote the application section on differentiated services in the Internet.

Thanks are also due to Dr. Samuel Kounev and Prof. Alejandro Buchmann for allowing us to use their paper "Performance Modelling and Evaluation of Large Scale J2EE Applications" to produce the J2EE section, which is a shortened and adapted version of their paper.

Our special thanks are due to Prof. Helena Szczerbicka for her invaluable contribution to Chapter 11 on simulation and to modeling methodology section of Chapter 1. Her overall help with the second edition is also appreciated.

We also thank Val Moliere, George Telecki, Emily Simmons, and Whitney A. Lesch from John Wiley & Sons for their patience and encouragement.

The support from the Euro-NGI (Design and Engineering of the Next Generation Internet) Network of Excellence, European Commission grant IST-507613, is acknowledged.

Finally, a Web page has been set up for further information regarding the second edition. The URL is `http://www.net.fmi.uni-passau.de/QNMC2/`

Gunter Bolch, Stefan Greiner, Hermann de Meer, Kishor S. Trivedi

Erlangen, Passau, Durham, August 2005

Preface to the First Edition

Queueing networks and Markov chains are commonly used for the performance and reliability evaluation of computer, communication, and manufacturing systems. Although there are quite a few books on the individual topics of queueing networks and Markov chains, we have found none that covers both of these topics. The purpose of this book, therefore, is to offer a detailed treatment of queueing systems, queueing networks, and continuous and discrete-time Markov chains.

In addition to introducing the basics of these subjects, we have endeavored to:

- Provide some in-depth numerical solution algorithms.
- Incorporate a rich set of examples that demonstrate the application of the different paradigms and corresponding algorithms.
- Discuss stochastic Petri nets as a high-level description language, thereby facilitating automatic generation and solution of voluminous Markov chains.
- Treat in some detail approximation methods that will handle large models.
- Describe and apply four software packages throughout the text.
- Provide problems as exercises.

This book easily lends itself to a course on performance evaluation in the computer science and computer engineering curricula. It can also be used for a course on stochastic models in mathematics, operations research and industrial engineering departments. Because it incorporates a rich and comprehensive set of numerical solution methods comparatively presented, the text may also well serve practitioners in various fields of applications as a reference book for algorithms.

With sincere appreciation to our friends, colleagues, and students who so ably and patiently supported our manuscript project, we wish to publicly acknowledge:

- Jörg Barner and Stephan Kösters, for their painstaking work in keying the text and in laying out the figures and plots.
- Peter Bazan, who assisted both with the programming of many examples and comprehensive proofreading.
- Hana Ševčíková, who lent a hand in solving many of the examples and contributed with proofreading.
- János Sztrik, for his comprehensive proofreading.
- Doris Ehrenreich, who wrote the first version of the section on communication systems.
- Markus Decker, who prepared the first draft of the mixed queueing networks section.
- Those who read parts of the manuscript and provided many useful comments, including: Khalid Begain, Oliver Düsterhöft, Ricardo Fricks, Swapna Gokhale, Thomas Hahn, Christophe Hirel, Graham Horton, Steve Hunter, Demetres Kouvatsos, Yue Ma, Raymond Marie, Varsha Mainkar, Victor Nicola, Cheul Woo Ro, Helena Szczerbicka, Lorrie Tomek, Bernd Wolfinger, Katinka Wolter, Martin Zaddach, and Henry Zang.

Gunter Bolch and Stefan Greiner are grateful to Fridolin Hofmann, and Hermann de Meer is grateful to Bernd Wolfinger, for their support in providing the necessary freedom from distracting obligations.

Thanks are also due to Teubner B.G. Publishing House for allowing us to borrow sections from the book entitled **Leistungsbewertung von Rechensystemen** (originally in German) by one of the coauthors, Gunter Bolch. In the present book, these sections are integrated in Chapters 1 and 7 through 10.

We also thank Andrew Smith, Lisa Van Horn, and Mary Lynn of John Wiley & Sons for their patience and encouragement.

The financial support from the SFB (Collaborative Research Centre) 182 ("Multiprocessor and Network Configurations") of the DFG (Deutsche Forschungsgemeinschaft) is acknowledged.

Finally, a Web page has been set up for further information regarding the book. The URL is `http://www4.cs.fau.de/QNMC/`

GUNTER BOLCH, STEFAN GREINER, HERMANN DE MEER, KISHOR S. TRIVEDI

Erlangen, June 1998

1

Introduction

1.1 MOTIVATION

Information processing system designers need methods for the quantification of system design factors such as performance and reliability. Modern computer, communication, and production line systems process complex workloads with random service demands. Probabilistic and statistical methods are commonly employed for the purpose of performance and reliability evaluation. The purpose of this book is to explore major probabilistic modeling techniques for the performance analysis of information processing systems. Statistical methods are also of great importance but we refer the reader to other sources [Jain91, Triv01] for this topic. Although we concentrate on performance analysis, we occasionally consider reliability, availability, and combined performance and reliability analysis. Performance measures that are commonly of interest include throughput, resource utilization, loss probability, and delay (or response time).

The most direct method for performance evaluation is based on actual measurement of the system under study. However, during the design phase, the system is not available for such experiments, and yet performance of a given design needs to be predicted to verify that it meets design requirements and to carry out necessary trade-offs. Hence, abstract models are necessary for performance prediction of designs. The most popular models are based on discrete-event simulation (DES). DES can be applied to almost all problems of interest, and system details to the desired degree can be captured in such simulation models. Furthermore, many software packages are available that facilitate the construction and execution of DES models.

The principal drawback of DES models, however, is the time taken to run such models for large, realistic systems, particularly when results with high accuracy (i.e., narrow confidence intervals) are desired. A cost-effective alternative to DES models, analytic models can provide relatively quick answers to "what if" questions and can provide more insight into the system being studied. However, analytic models are often plagued by unrealistic assumptions that need to be made in order to make them tractable. Recent advances in stochastic models and numerical solution techniques, availability of software packages, and easy access to workstations with large computational capabilities have extended the capabilities of analytic models to more complex systems.

Analytical models can be broadly classified into state-space models and non-state-space models. Most commonly used state-space models are Markov chains. First introduced by A. A. Markov in 1907, Markov chains have been in use in performance analysis since around 1950. In the past decade, considerable advances have been made in the numerical solution techniques, methods of automated state-space generation, and the availability of software packages. These advances have resulted in extensive use of Markov chains in performance and reliability analysis. A Markov chain consists of a set of states and a set of labeled transitions between the states. A state of the Markov chain can model various conditions of interest in the system being studied. These could be the number of jobs of various types waiting to use each resource, the number of resources of each type that have failed, the number of concurrent tasks of a given job being executed, and so on. After a sojourn in a state, the Markov chain will make a transition to another state. Such transitions are labeled with either probabilities of transition (in case of discrete-time Markov chains) or rates of transition (in case of continuous-time Markov chains).

Long run (steady-state) dynamics of Markov chains can be studied using a system of linear equations with one equation for each state. Transient (or time dependent) behavior of a continuous-time Markov chain gives rise to a system of first-order, linear, ordinary differential equations. Solution of these equations results in state probabilities of the Markov chain from which desired performance measures can be easily obtained. The number of states in a Markov chain of a complex system can become very large, and, hence, automated generation and efficient numerical solution methods for underlying equations are desired. A number of concise notations (based on queueing networks and stochastic Petri nets) have evolved, and software packages that automatically generate the underlying state space of the Markov chain are now available. These packages also carry out efficient solution of steady-state and transient behavior of Markov chains. In spite of these advances, there is a continuing need to be able to deal with larger Markov chains and much research is being devoted to this topic.

If the Markov chain has nice structure, it is often possible to avoid the generation and solution of the underlying (large) state space. For a class of queueing networks, known as product-form queueing networks (PFQN), it is possible to derive steady-state performance measures without resorting to the underlying state space. Such models are therefore called non-state-space models. Other examples of non-state-space models are directed acyclic task precedence graphs [SaTr87] and fault-trees [STP96]. Other examples of methods exploiting Markov chains with "nice" structure are matrix-geometric methods [Neut81] (see Section 3.2).

Relatively large PFQN can be solved by means of a small number of simpler equations. However, practical queueing networks can often get so large that approximate methods are needed to solve such PFQN. Furthermore, many practical queueing networks (so-called non-product-form queueing networks, NPFQN) do not satisfy restrictions implied by product form. In such cases, it is often possible to obtain accurate approximations using variations of algorithms used for PFQNs. Other approximation techniques using hierarchical and fixed-point iterative methods are also used.

The flowchart shown in Fig. 1.1 gives the organization of this book. After a brief treatment on methodological background (Section 1.2), Section 1.3 covers the basics of probability and statistics. In Chapter 2, Markov chains basics are presented together with generation methods for them. Exact steady-state solution techniques for Markov chains are given in Chapter 3 and their aggregation/disaggregation counterpart in Chapter 4. These aggregation/disaggregation solution techniques are useful for practical Markov chain models with very large state spaces. Transient solution techniques for Markov chains are introduced in Chapter 5.

Chapter 6 deals with the description and computation of performance measures for single-station queueing systems in steady state. A general description of queueing networks is given in Chapter 7. Exact solution methods for PFQN are described in detail in Chapter 8 while approximate solution techniques for PFQN are described in Chapter 9. Solution algorithms for different types of NPFQN (such as networks with priorities, nonexponential service times, blocking, or parallel processing) are presented in Chapter 10.

Since there are many practical problems that may not be analytically tractable, discrete-event simulation is commonly used in this situations. We introduce the basics of DES in Chapter 11. For the practical use of modeling techniques described in this book, software packages (tools) are needed. Chapter 12 is devoted to the introduction of a queueing network tool, a stochastic Petri net tool, a tool based on Markov chains and a toolkit with many model types, and the facility for hierarchical modeling is also introduced. Each tool is described in some detail together with a simple example. Throughout the book we have provided many example applications of different algorithms introduced in the book. Finally, Chapter 13 is devoted to several large real-life applications of the modeling techniques presented in the book.

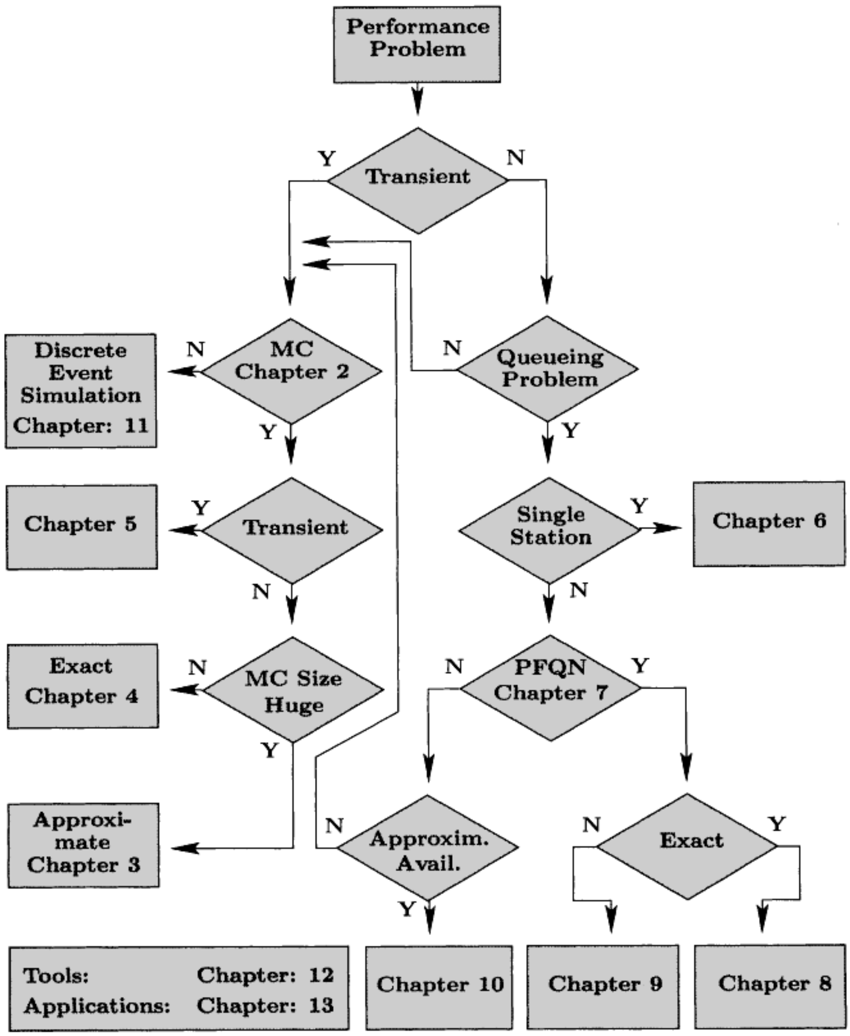

Fig. 1.1 Flowchart describing how to find the appropriate chapter for a given performance problem.

1.2 METHODOLOGICAL BACKGROUND

The focus of this book is the application of stochastic and probabilistic methods to obtain conclusions about performance and reliability properties of a wide range of systems. In general, a system can be regarded as a collection of components which are organized and interact in order to fulfill a common task [IEEE90].

Reactive systems and nondeterminism: The real-world systems treated in this book usually contain at least one digital component which controls the operation of other analog or digital components, and the whole system *reacts* to stimuli triggered by its *environment.* As an example, consider a computer communication network in which components like routers, switches, hubs, and communication lines fulfill the common task of transferring data packets between the various computers connected to the network. If the system of interest is the communication network only, the connected computers can be regarded as its environment which triggers the network by sending and receiving data packets. The behavior of the systems studied in this book can be characterized as *nondeterministic* since the stimulation by the environment is usually unpredictable. In case of a communication system like the Internet, the *workload* depends largely on the number of active users. When exactly a specific user will start to access information on the WWW via a browser is usually not known in advance. Another source of nondeterminism is the potential failure of one or several system components, which in most cases leads to an altered behavior of the complete system.

Modeling vs. Measurement: In contrast to the *empirical* methods of measurement, i.e., the collection of output data during the observation of an executing system, the *deductive* methods of model-based performance evaluation have the advantage to be applicable in situations when the system of interest is not yet existing. Deductive methods can thus be applied during the early design phases of the system development process in order to ensure that the final product meets its performance and reliability requirements. Although the material presented in this book is restricted to modeling approaches, it should be noticed that measurement as a supplementary technique can be employed to validate that the conclusions obtained by model-based performance evaluation can be translated into *useful* statements about the real-world system.

Another possible scenario for the application of modeling is the situation in which measurements on an existing system would either be too dangerous or too expensive. New policies, decision rules, or information flows can be explored without disrupting the ongoing operation of the real system. Moreover, new hardware architectures, scheduling algorithms, routing protocols, or reconfiguration strategies can be tested without committing resources for their acquisition/implementation. Also, the behavior of an existing system under a variety of anticipated workloads and environments can be evaluated very cost-effectively in advance by model-based approaches.

1.2.1 Problem Formulation

Before a meaningful model-based evaluation can commence, one should carefully consider what performance metric is of interest besides the nature of the system. This initial step is indispensable since it determines what is the appropriate formalism to be used. Most of the formalisms presented in the following chapters are suitable for the evaluation of specific metrics but inappropriate for the derivation of others. In general, it is important to consider the crucial aspects of the application domain with respect to the metrics to be evaluated before starting with the formalization process. Here, the *application context* strongly determines the kind of information that is meaningful in a concrete modeling exercise.

As an illustrative example, consider the power outage problem of computer systems. For a given hardware configuration, there is no ideal way to represent it without taking into consideration the software applications which run on the hardware and which of course have to be reflected in the model. In a real-time context, such as flight control, even the shortest power failure might have catastrophic implications for the system being controlled. Therefore, an appropriate reliability model of the flight control computer system has to be very sensitive to such a (hopefully) rare event of short duration. In contrast, the total number of jobs processed or the work accomplished by the computer hardware during the duration of a flight is probably a less important performance measure for such a safety-critical system. If the same hardware configuration is used in a transaction processing system, however, short outages are less significant for the proper system operation but the throughput is of predominant importance. As a consequence thereof, it is not useful to represent effects of short interruptions in the model, since they are of less importance in this application context.

Another important aspect to consider at the beginning of a model-based evaluation is how a reactive real-world system — as the core object of the study — is triggered by its environment. The stimulation of the system by its environment has to be captured in such a way during formalization so it reflects the conditions given in the real world as accurately as possible. Otherwise, the measures obtained during the evaluation process cannot be meaningfully retransformed into statements about the specific scenario in the application domain. In the context of stochastic modeling, the expression of the environment's influence on the system in the model is usually referred to as *workload modeling.* A frequently applied technique is the characterization of the arriving workload, e.g., the parts which enter a production line or the arriving data packets in a communication system, as a stochastic *arrival process.* Various arrival processes which are suitable in specific real-world scenarios can be defined (see Section 6.8).

The following four categories of system properties which are within the scope of the methods presented in this book can be identified:

Performance Properties: They are the oldest targets of performance evaluation and have been calculated already for non-computing systems like telephone switching centers [Erla17] or patient flows in hospitals [Jack54] using closed-form descriptions from applied probability theory. Typical properties to be evaluated are the mean throughput of served customers, the mean waiting, or response time and the utilization of the various system resources. The IEEE standard glossary of software engineering terminology [IEEE90] contains the following definition:

Definition 1.1 Performance: The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage.

Reliability and Availability: Requirements of these types can be evaluated quantitatively if the system description contains information about the failure and repair behavior of the system components. In some cases it is also necessary to specify the conditions under which a new user cannot get access to the service offered by the operational system. The information about the failure behavior of system components is usually based on heuristics which are reflected in the parameters of probability distributions. In [IEEE90], software reliability is defined as:

Definition 1.2 Reliability: The probability that the software will not cause the failure of the system for a specified time under specified conditions.

System reliability is a measure for the continuity of correct service, whereas availability measures for a system refer to its readiness for correct service, as stated by the following definition from [IEEE90]:

Definition 1.3 Availability: The ability of a system to perform its required function at a stated instant or over a stated period of time. It is usually expressed as the availability ratio, i.e., the proportion of time that the service is actually available for use by the Customers within the agreed service hours.

Note that reliability and availability are related yet distinct system properties: a system which — during a mission time of 100 days — fails on average every two minutes but becomes operational again after a few milliseconds is not very reliable but nevertheless highly available.

Dependability and Performability: These terms and the definitions for them originated from the area of *dependable* and *fault tolerant* computing. The following definition for dependability is taken from [ALRL04]:

Definition 1.4 Dependability: The dependability of a computer system is the ability to deliver a service that can justifiably be trusted. The service delivered by a system is its behavior as it is perceived by its user(s); a user is another system (physical, human) that interacts with the former at the service interface.

This is a rather general definition which comprises the five attributes availability, reliability, *maintainability* — the systems ability to undergo modifications or repairs, *integrity* — the absence of improper system alterations and *safety* as a measure for the continuous delivery of service free from occurrences of catastrophic failures. The term *performability* was coined by J.F. MEYER [Meye78] as a measure to assess a system's ability to perform when performance degrades as a consequence of faults:

Definition 1.5 Performability: The probability that the system reaches an accomplishment level y over a utilization interval $(0, t)$. That is, the probability that the system does a certain amount of useful work over a mission time t.

Subsequently, many other measures are included under performance as we shall see in Section 2.2. Informally, the performability refers to performance in the presence of failures/repair/recovery of components and the system. Performability is of special interest for *gracefully degrading systems* [Beau77]. In Section 2.2, a framework based on *Markov reward models* (MRMs) is presented which provides recipes for a selection of the right model type and the definition of an appropriate performance measure.

1.2.2 The Modeling Process

The first step of a model-based performance evaluation consists of the formalization process, during which the modeler generates a *formal* description of the real-world system. Figure 1.2 illustrates the basic idea: Starting from an informal system description, e.g. in natural language, which includes structural and functional information as well as the desired performance and reliability requirements, the modeler creates a formal model of the real-world system using a specific *conceptualization*. A conceptualization is an *abstract, simplified* view of the reference reality which is represented for some purpose. Two kinds of conceptualizations for the purpose of performance evaluation are presented in detail in this book: If the system is to be represented as a queueing network, the modeler applies a "routed job flow" *modeling paradigm* in which the real-world system is conceptualized as a set of service stations which are connected by edges through which independent entities "flow" through the network and sojourn in the queues and servers of the service stations (see Chapter 7). In an alternative Markov chain conceptualization a "state-transition" modeling paradigm is applied in which the possible trajectories through the system's global state space are represented as a graph whose directed arcs represent the transitions between subsequent system states (see Chapter 2). The main difference between the two conceptualizations is that the queueing network formalism is oriented more towards the *structure* of the real-world system, whereas in the Markov chain formalization the emphasis is put on the description of the system *behavior* on the underlying state-space level. A Markov chain can be regarded to "mimic" the behavior of the executing real-world system, whereas the service stations and jobs of a queueing network establish a one-to-one correspondence to the components of the real-world system. As indicated in Fig. 1.2, a Markov chain serves as the underlying *semantic* model of the high-level queueing network model.

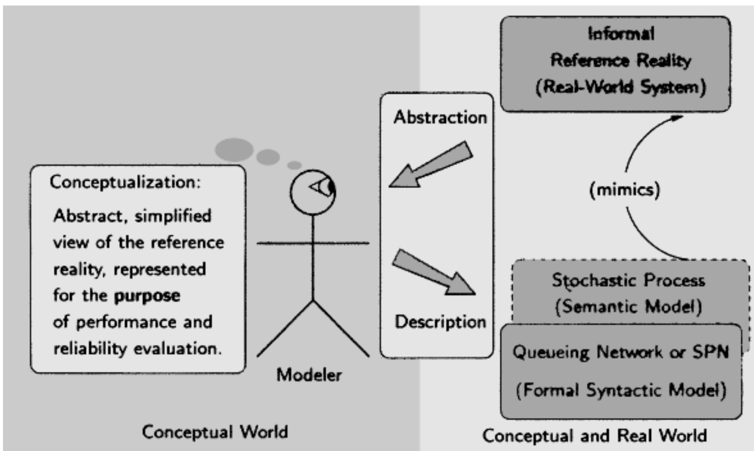

Fig. 1.2 Formalization of a real-world system.

During the formalization process the following abstractions with respect to the real-world system are applied:

- In both conceptualizations the behavior of the real-world system is regarded to evolve in a *discrete-event* fashion, even if the real-world system contains components which exhibit continuous behavior, such as the movements of a conveyor belt of a production line.

- The application of the queueing network formalism abstracts away from all *synchronization* mechanisms which may be present in the real-world system. If the representation of these synchronization mechanisms is crucial in order to obtain useful results from the evaluation, the modeler can resort to variants of stochastic Petri nets as an alternative description technique (see Section 2.3 and Section 2.3.6) in which almost arbitrary synchronization patterns can be captured.

- The *core abstractions* applied during the formalization process are the association of system *activity durations* with *random variables* and the inclusion of *branching probabilities* to represent alternative system evolutions. Both abstractions *resolve* the nondeterminism inherent in the real-world system and turn the formal queueing network or Markov chain prototype into an "executable" specification [Wing01]. For these, at any moment during their operation each possible future evolution has a well-defined probability to occur. Depending on which kind of random variables are used to represent the durations of the system activities either a discrete-time interpretation using a DTMC or a continuous-time interpretation of the system behavior based on a CTMC is achieved. It should be noted that for systems with *asynchronously* evolving components the continuous-time interpretation is more appropriate since changes of the global system state may occur at any moment in continuous time. Systems with components that evolve in a *lock-step* fashion triggered by a global clock are usually interpreted in discrete-time.

1.2.3 Evaluation

The second step in the model-based system evaluation is the deduction of performance measures by the application of appropriate solution methods. Depending on the conceptualization chosen during the formalization process the following solution methods are available:

Analytical Solutions: The core principle of the analytic solution methods is to represent the formal system description either as a single equation from which the interesting measures can be obtained as closed-form solutions, or as a set of system equations from which exact or approximate measures can be calculated by appropriate algorithms from numerical mathematics.

1. *Closed-form solutions* are available if the system can be described as a simple queueing system (see Chapter 6) or for simple product-form queueing networks (PFQN) [ChMa83] (see Section 7.3) or for structured small CTMCs. For these kind of formalizations equations can be derived from which the mean number of jobs in the service stations can be calculated as a *closed-form solution*, i.e., the solutions can be expressed analytically in terms of a bounded number of well-known operations. Also from certain types of Markov chains with regular structure (see Section 3.1), closed-form representations like the well-known Erlang-B and Erlang-C formulae [Erla17] can be derived. The measures can either be computed by ad-hoc programming or with the help of computer algebra packages such as Mathematica [Mat05]. A big advantage of the closed-form solutions is their moderate computational complexity which enables a fast calculation of performance measures even for larger system descriptions.

2. *Numerical solutions*: Many types of equations which can be derived from a formal system description do not possess a closed-form solution, e.g., in the case of complex systems of integro-differential equations. In these cases, approximate solutions can be obtained by the appli-

cation of algorithms from numerical mathematics, many of which are implemented in computer algebra packages [Mat05] or are integrated in performance analysis tools such as SHARPE [HSZT00], SPNP [HTT00], or TimeNET [ZFGH00] (see Chapter 12). The formal system descriptions can be either given as a queueing network, stochastic Petri net or another high-level modeling formalism, from which a state-space representation is generated manually or by the application of state-space generation algorithms. Depending on the stochastic information present in the high-level description, various types of system state equations which mimic the dynamics of the modeled system can be derived and solved by appropriate algorithms. The numerical solution of Markov models is discussed in Chapters 3 – 5, numerical solution methods for queueing networks can be found in Chapters 7 – 10. In comparison to closed-form solution approaches, numerical solution methods usually have a higher computational complexity.

Simulation Solutions: For many types of models no analytic solution method is feasible, because either a theory for the derivation of proper system equations is not known, or the computational complexity of an applicable numerical solution algorithm is too high. In this situation, solutions can be obtained by the application of discrete-event simulation (DES), which is described in detail in Chapter 11. Instead of solving system equations which have been derived from the formal model, the DES algorithm "executes" the model and collects the information about the observed behavior for the subsequent derivation of performance measures. In order to increase the quality of the results, the simulation outputs collected during multiple "executions" of the model are collected and from which the interesting measures are calculated by statistical methods. All the formalizations presented in this book, i.e., queueing networks, stochastic Petri nets, or Markov chains can serve as input for a DES, which is the most flexible and generally applicable solution method. Since the underlying state space does not have to be generated, simulation is not affected by the state-space explosion problem. Thus, simulation can also be employed for the analysis of complex models for which the numerical approaches would fail because of an exuberant number of system states.

Hybrid solutions: There exists a number of approaches in which different modeling formalisms and solution methods are combined in oder to exploit their complementing strengths. Examples of hybrid solution methods are mixed simulation and analytical/numerical approaches, or the combination of fault trees, reliability block diagrams, or reliability graphs, and Markov models [STP96]. Also product-form queueing networks and stochastic Petri nets or non-product-form networks and their solution methods can be combined. More generally, this approach can be characterized as intermingling of *state-space*-based and *non-state-space*-based methods [STP96]. A combination of analytic and simulative solutions of connected sub-models may be employed to combine the benefits of both solution methods [Sarg94, ShSa83]. More criteria for a choice between simulation and analytical/numerical solutions are discussed in Chapter 11.

Largeness Tolerance: Many high-level specification techniques, queueing systems, generalized stochastic Petri nets (GSPNs), and stochastic reward nets (SRNs), as the most prominent representatives, have been suggested in the literature to automate the model generation [HaTr93]. GSPNs/SRNs that are covered in more detail in Section 2.3, can be characterized as *tolerating* largeness of the underlying computational models and providing effective means for generating large state spaces.

Largeness Avoidance: Another way to deal with large models is *to avoid* the creation of such models from the beginning. The major largeness-avoidance technique we discuss in this book is that of product-form queueing networks. The main idea is, the structure of the underlying CTMC allows for an efficient solution that obviates the need for generation, storage, and solution of the large state space. The second method of avoiding largeness is to separate the originally single large problem into several smaller problems and to combine sub-model results into an overall solution. Both approximate and exact techniques are known for dealing with such multilevel models. The flow of information needed among sub-models may be acyclic, in which case a hierarchical model [STP96] results. If the flow of needed information is non-acyclic, a fixed-point iteration may be necessary [CiTr93]. Other well-known techniques applicable for limiting model sizes are *state truncation* [BVDT88, GCS+86] and *state lumping* [Nico90].

1.2.4 Summary

Figure 1.3 summarizes the different phases and activities of the model-based performance evaluation process. Two main scenarios are considered: In the first one, model-based performance evaluation is applied during the early phases of the system development process to predict the performance or reliability properties of the final product. If the predicted properties do not fulfill the given requirements, the proposed design has to be changed in order to avoid the expected performance problems. In the second scenario, the final product is already available and model-based performance evaluation is applied to derive optimal system configuration parameters, to solve capacity planning problems, or to check whether the existing system would still operate satisfactorily after a modification of its environment. In both scenarios the first activity in the evaluation process is to collect information about the structure and functional behavior of an existing or planned system. The participation of a domain expert in this initial step is very helpful and rather indispensable for complex applications. Usually, the collected information is stated informally and stored in a document using either a textual or a combined textu-