

Teaching Coloured Petri Nets: Examples of Courses and Lessons Learned

Søren Christensen and Jens Bæk Jørgensen

Department of Computer Science, University of Aarhus
Aabogade 34, DK-8200 Aarhus N, Denmark
{schristensen,jbj}@daimi.au.dk

Abstract. In this paper, we describe and discuss three different courses in which Coloured Petri Nets (CPN) is used: (1) an introductory course on distributed systems and network protocols; (2) an advanced course on CPN; (3) a course on industrial application of CPN. Courses (1) and (2) are taught at the Department of Computer Science, University of Aarhus and course (3) is given for professional software engineers. For each course, we briefly present contents, format, and role of CPN. Then we describe a number of lessons learned from teaching the three courses. We have two aims in mind: In the first place, we want to share our specific experiences with other teachers. Secondly, we want to contribute to a more general discussion and exchange of ideas on Petri nets and education.

1 Introduction

Coloured Petri Nets (CPN) [10] has had a place in the curriculum at the Department of Computer Science, University of Aarhus (henceforth abbreviated with the Danish acronym *DAIMI* [32]) for the last twenty years. The main reason is that formal modelling languages like CPN are suitable for many educational activities within computer science. Another contributing factor is the presence of professor Kurt Jensen [28], whose PhD work around 1980 defined the first version [9] of the CPN language and laid the foundation for the research of the CPN Group [26] at DAIMI.

In the 1980's, CPN was used at DAIMI as a general system description language in the introductory first-year course taken by 150-200 students each year. One of the main purposes of using CPN was to teach students that making abstract system descriptions (or models) is an important activity in computer science. A number of lectures on CPN were given, the students read some introductory material, and they were required to solve exercises on CPN. Examples of exercises were to model the flow of customers through the local canteen and to model a traffic light. These were non-trivial exercises, especially because at that time, tool support for CPN (and other kinds of Petri nets) was scarce. Models were drawn on paper and simulations were carried out by playing the token game with coins or drawing pins on sheets of paper. Another main purpose of using CPN was to introduce the students to formal semantics of programming

languages via CPN, e.g., the semantics of various language constructs of Pascal [12].

Around 1990, DAIMI began to offer advanced courses on the CPN language itself. There were two kinds of courses: (1) application-oriented courses, where students constructed and analysed fairly large CPN models (made possible by the emergence of the Design/CPN tool [27] in 1989); (2) theoretical courses, where students immersed in the mathematical foundation of CPN and typically pursued formal verification methods like state spaces or place invariants.

In the 1990's, CPN made up about half of the curriculum in a third-year course on distributed systems; the mid-1990's incarnation of that course is described in detail in the paper [6]. Two textbooks were used: Jensen's CPN book [10] and Tanenbaum's book on distributed operating systems [24]. The main emphasis of the CPN part of the course was on CPN modelling and use of the Design/CPN tool as vehicles for design and analysis of distributed systems, but students were also thoroughly introduced to the mathematical foundation and to formal verification methods of CPN.

We, the authors of this paper, are members of the CPN Group at DAIMI. We have seen CPN in education from different perspectives, starting with our first encounter in the early and mid 1980's, when we were students at DAIMI. From around 1990, we have experienced CPN from the other side of the table: as teachers in various computer science courses. This paper is based on our experiences with using CPN in education. We have two aims: In the first place, we want to share our specific experiences with other teachers, and hopefully provide some kind of inspiration (and perhaps save computer science students at other universities from teachers making the same errors as we did). Secondly, we want to contribute to a more general discussion and exchange of ideas on Petri nets and education.

In Sect. 2, we describe three courses we have taught and which have used CPN. In Sect. 3, we report some lessons learned. We draw some conclusions in Sect. 4.

2 Examples of Courses

Currently, CPN is used at DAIMI in two different courses:

- The *distributed systems course*: a third-year introductory course on distributed systems and network protocols [33].
- The *advanced course on CPN*: a graduate course in which CPN is studied as a language in its own right.

We describe these two courses more thoroughly in this section. In addition, we describe:

- The *industrial application of CPN course*: a course on application of CPN held for a group of engineers from a software company.

2.1 Distributed Systems Course

Since the late 1990's, CPN has been used as an ingredient of a third-year introductory course on distributed systems and network protocols. The course runs over 15 weeks and is attended by close to 100 students. The course gives a credit of 10 ECTS points and consists of two parts of equal size. The first part introduces basic concepts and design techniques for distributed systems. The second part introduces the basic ideas behind computer networks and network protocols, including a detailed coverage of the Internet protocols. The textbooks currently used are Coulouris et al's on distributed systems [7] and Stallings' on networks and network protocols [23].

The format of the course is a combination of lectures for all students and tutorials where the students are divided into classes of approximately 20 students, and where an older student is available as teaching assistant. In average, there are three hours of lectures and three hours of tutorials each week. Students are expected to use a total of up to 15 hours on the course each week.

Three two-hour lectures on CPN are given early in the course. The first informally introduces the basic concepts of CPN. The second gives some practical hints on the construction of CPN models and it introduces basics of the Standard ML language [17]; some elementary Standard ML programming skills are necessary in order to properly apply the CPN tool, which is introduced in the third lecture in an extensive tool demonstration. The CPN literature we have used over the years is excerpts from Jensen's book [10] (chapter 1 and parts of chapter 3) and the practitioner's guide to CPN [15]. The tools which have been applied are Design/CPN and the newer CPN Tools [21, 30].

CPN is used to give the students a better understanding of distributed systems and network protocols than can be provided by the textbooks alone. Thus, the role of CPN is to be a supplement to the main curriculum. We feel that the exercises are a weak part of the two textbooks we use. Therefore, we need additional exercises and CPN helps us to achieve this. CPN exercises are put forward throughout the course. An example of a CPN exercise aiming at making the students better understand fundamental mechanisms of distributed file systems as described in the textbook (chapter 8 of Coulouris et al's book [7]) is: (1) Make a CPN model of caching in Sun's Network File System (NFS); (2) based on your model, discuss the problem of cache consistency and advantages and drawbacks of the NFS solution.

In addition to smaller exercises, we have asked the students to solve a larger mandatory project on CPN over a time period of three weeks. The project varies from year to year. In 2003, the students were asked to design a protocol ensuring reliable communication over an unreliable communication channel with appropriate use of timers, sequence numbers, retransmissions, etc.

CPN is a suitable language for this course because it allows the students to explicitly describe their interpretation of the highly prose-based and always slightly ambiguous and sometimes even vague presentation of algorithms, protocols etc. from the textbooks. In particular, CPN facilitates the students' comprehension of traditionally hard-to-understand issues related to concurrency, resource sharing,

synchronisation, and conflicts. CPN models give a solid foundation for discussions between students and between students and teaching assistants.

2.2 Advanced Course on CPN

CPN is the subject of an advanced course, which runs over 12-15 weeks and is typically attended by 10-15 students. The course gives a credit of 10 ECTS points and comprises three parts. The second part is rather special; it consists in participation in an international workshop on CPN being held in Aarhus (for the 2002 incarnation of that workshop, see [31]). Thus, from the students' perspective, the workshop is an integrated part of the course. The first part consists of introductory lectures and student presentations of the scientific papers on CPN, which are accepted for the workshop. Together, the first and the second parts occupy about a month of calendar time, in which the students work intensively with the course. The third part, which takes approximately two months, consists in carrying out a project. The literature used in the course is excerpts from volumes 1 and 2 of Jensen's books [10, 11], the practitioner's guide [15], and workshop proceedings (which in 2002 were [8]).

The students choose between two categories of projects: One category is application-oriented project, i.e., creation and analysis of CPN models of domains of interest for the students. An example of such a project from 2002 is a group of students who worked together with the large Danish company Danfoss to model and investigate the behaviour of control software for an industrial embedded system. The other category comprises theoretical projects. Examples of such projects are study of methods for verification by means of state space analysis. The students read relevant literature and sometimes do small practical exercises using various tools (depending on availability and quality of such tools). All students interested in this subject study basic state space analysis. Subsequently, some choose to pursue more advanced methods, e.g., state space analysis using equivalence classes or symmetries [11], state space analysis using stubborn sets [25], or state space analysis by the sweepline method [5].

2.3 Industrial Application of CPN Course

In addition to teaching CPN to computer science students, we have given several CPN courses for software engineers from the industry over the years. These courses are tailored for particular companies and vary in contents and format. Examples of CPN projects where a course has been an integrated part are analysis of audio/video transmission protocols at Bang and Olufsen as described in [4], design of alarm systems at Dalcotech [20], and analysis of car control systems at Peugeot Citroen [18].

Typically, between two and six engineers participate in the course, which runs over a total of six full days, divided into two parts each comprising three days in one week and three days in another week. The course is very application-oriented and no introduction to the formal, mathematical foundation of CPN is

given. The attendees use most of their time doing practical hands-on exercises in small groups.

In the first part, the basic CPN concepts and a CPN tool are introduced. The first day covers the most fundamental CPN concepts (corresponding to chapter 1 of Jensen's book [10]). The second day covers hierarchical CPN models (corresponding to parts of chapter 3 of Jensen's book [10]) and on the third day, CPN models with time are introduced. The introductions are not traditional lecture-like presentations, but integrated parts of extensive tool demonstrations. It is always shown how to create, edit, and simulate CPN models (it is often shown how to carry out simple state space analysis as well). The basic functionality of the tool is explained and step-by-step instructions on how to carry out modelling tasks are given. Throughout the first part, the engineers do practical hands-on exercises with the tool. As example, on the first day, they make various modifications of a small model of a simple communications protocol, e.g., modify a stop-and-wait protocol to become a more general sliding-window protocol.

On the last day of the first part, much time is allocated for discussion and determination of the more specific contents of the second part. It is crucial that the engineers make this choice themselves. They identify problems in their domain which they would like to address using CPN. CPN instructors often start to outline model drafts. In the time between the first and the second part, the engineers and the CPN instructors continue to think about how CPN can be used for the particular problem that the engineers want to solve. This thinking is crucial preparation for the second part in which the engineers spend most of the time creating and analysing larger domain-specific models. The CPN instructors are available to help the engineers, who, thus, are in a good position to work efficiently. Quite often, one engineer and one CPN instructor sit together in front of a computer and build models together.

3 Lessons Learned

In this section, we describe and discuss a number of lessons we have learned from teaching the three courses described above. The sources include course evaluation forms that students fill out after having attended a course and feedback from teaching assistants.

3.1 On Literature

As mentioned in the previous section, in the distributed systems course and the industrial application course, we have tried two possibilities for introductory literature on CPN: excerpts of Jensen's book [10] and the practitioner's guide [15].

In the distributed systems course, we have experienced that Jensen's book work better than the practitioner's guide. Students seem to prefer the thorough, step-wise introduction of the basic CPN concepts given in Jensen's book via easily understandable place-transition nets (PT nets) and small examples of CPN models. In the industrial application course, we have better experiences with

the practitioner's guide, which is written to be directly appealing to industrial software engineers. It is non-formal, emphasises the application aspects of CPN, and advocates CPN as a way to address common software development problems. As an example, many industrial software engineers have scalability as a main concern; some may be reluctant to use of formal methods at all because they have seen approaches that do not scale well (e.g., when they studied computer science some years ago). One of the main purposes of the practitioner's guide is to demonstrate that CPN scales well to the size of problems that the software industry is dealing with. Therefore, the practitioner's guide introduces the basic CPN concept via a quite large example model and without PT nets, which do not scale well to modelling of industrial systems.

In the advanced course, the students who attend are particularly interested in CPN and want a thorough and broad coverage of the language. Therefore, we use both Jensen's books [10, 11] for a well-founded introduction of the basic concepts and analysis methods, and the practitioner's guide to set the stage for large-scale modelling. In addition, we use workshop proceedings from the current year. Typically, the range of subjects covered by the papers in these proceedings is quite broad. Reading the papers gives the students an introduction to research in CPN and to the process of writing and publishing scientific papers. Workshop papers often describe early results and work in progress. They may later mature into conference and journal papers, after more research, writing, and rewriting. Therefore, it is often possible for students to find errors and shortcomings and to propose constructive improvements. It is useful for the students to see the authors present their papers at the workshop and to compare this with the presentation that they (the students) gave themselves earlier in the course. And the students are in an excellent position to ask questions and to engage in discussions.

3.2 On Tools

The choice of which tool to use in a particular course may have a high impact on the quality of the course as experienced by the students.

In the distributed systems course, in which relatively many unexperienced users use the tool, it must be easy to learn, stable, and well documented. In the advanced course, the demands to the tools are lower for a number of reasons. In the first place, the students are older and more mature than the students, who attend the distributed systems course. Secondly, the students have a particular interest in CPN, and thirdly, they have a higher willingness to accept the inherent limitations of research prototypes of tools. In the industrial application course, the requirements to the tool are very high because industrial software engineers will inevitably compare it with top-quality commercial tools that they are used to from their everyday development work.

For the last couple of years, the choice between the new CPN Tools [21, 30] and the older Design/CPN tool has been difficult in all three courses. The trade-off between stability and being easy to learn has not been easy. Design/CPN has a quite steep learning curve; there are a number of obstacles causing troubles for unexperienced users like young students or software engineers previously unfa-

miliar with CPN. As an example, Design/CPN does not have a fully incremental syntax check which often makes it difficult to debug models. On the other hand, for the last many years, Design/CPN has been a stable and well-tested tool with many useful and nice features (and a bit old-fashioned user interface). CPN Tools alleviates many of the problems that are present with Design/CPN. In particular, it seems to be faster to become a proficient CPN Tools user than a Design/CPN user. However, sometimes we have been too eager to use new versions of CPN Tools. They have not always been tested well enough and have occasionally caused frustration for students (who do not want to spend their valuable time as alpha or beta testers of a tool, which is not sufficiently mature for a large group of unexperienced users). In 2003, CPN Tools had reached a maturity that ensured a successful use by approximately 100 students in the distributed systems course.

In all the CPN related courses we have taught, we have experienced that it is important that a long extensive tool demo is given. The students who do not attend the demo often have had severe problems getting started with the tool.

3.3 On Teacher Skills

The required level of CPN skills for the teacher or teachers varies between the three courses discussed in this paper. It must of course be solid, but is lowest for the distributed systems course, in which only basic CPN is taught and only relatively small exercises are put forward. The skill level required to run the industrial application course is higher: It is necessary that the teachers are experienced in building large CPN models and have the ability to understand the domain of interests for the engineers. The advanced course demands the highest skill level: It can probably only be taught properly by teachers who are themselves CPN researchers.

The teaching assistants for the distributed systems course are appointed by the Faculty of Science, University of Aarhus. We have taught instances of the course where the teaching assistants were not sufficiently proficient with either CPN itself or with the applied CPN tool. That problem immediately propagated on to the students, who were not being appropriately helped. To solve the problem, we now staff the weekly tutorials in the weeks in which CPN are introduced with older students who we know are well experienced with CPN and CPN tools (e.g., recruited among the CPN Group's PhD students and student programmers). In this way, help is readily available, and students do not have to spend excessive amounts of time trying to figure out themselves the peculiarities of CPN and CPN Tools (including the Standard ML programming language, which is new to the majority of the students).

In the advanced course and the industrial application course, we have always hand-picked teaching assistants to ensure that they were sufficiently experienced CPN users.

3.4 On Student Motivation

In the advanced course and the industrial application course, the attendees themselves have usually chosen that they want to learn about CPN. Therefore, we

always teach highly motivated people in these two courses. But in the much broader distributed systems course, a number of students have seen CPN as a small and irritating “appendix” that they did not have to take too seriously. This has caused these students to more or less ignore CPN in the weeks where it was introduced. As a consequence, it turned out to be very difficult and sometimes even impossible for them to solve the CPN exercises put forward in conjunction with the main curriculum. To address this problem, we now require that all students carry out a larger mandatory CPN project early in the course.

In the advanced course, we have experienced that it is very motivating for students to participate in an international workshop giving them an opportunity to meet other students and researchers from foreign universities and companies. We have also experienced that it is motivating for some students to collaborate with an industrial partner like Danfoss.

In the industrial application course, it is important that focus is on the domain-specific problems that the engineers are facing. Therefore, as we saw, an example model of something that the engineers is familiar with from their everyday work is always a central ingredient in the course; we have not met many industrial software engineers, who find the dining philosophers or similar toy examples very appealing. Moreover, it is essential to present CPN as a useful supplement to the software development techniques that the engineers are already using; software engineers do typically not take a CPN course because they (or their managers) want to make dramatic changes to their company’s software development practices. They want to improve what they are already good at. Today, this means that CPN often must be presented as a supplement to UML [19, 22], the de-facto modelling language of the software industry. CPN models may supplement, e.g., UML use cases [13], class diagrams, sequence diagrams, and collaboration diagrams. CPN may be seen as vehicle to make strong descriptions of behaviour and as an alternative to UML state machines and activity diagrams [14].

3.5 On Integration with Main Curriculum

The discussion in this section only applies to courses in which CPN is one component amongst others; not for courses exclusively on CPN. In the scope of this paper, this means the distributed systems course, where a number of students have criticized us for not integrating CPN well enough with the main curriculum. One of their arguments is that CPN is not sufficiently used at the lectures.

We partly agree with the students. As mentioned earlier, CPN mainly play a role in the tutorials and there are a number of reasons for not using CPN more extensively at the lectures. In the first place, the course is about distributed systems and network protocols; CPN is merely a vehicle for gaining a better understanding, description, and discussion of concepts and problems of the main curriculum. More extensive use of CPN could cause the course to become a “CPN Modelling of Distributed Systems and Network Protocols” course. Secondly, the lectures would deviate more from the textbooks than we feel they should. Thirdly, there is already plenty of material for the students to digest, so

expanding the use of CPN would force us to reduce something else, which we do not believe is a good decision for this particular course.

However, finding the right balance between CPN and the rest of the curriculum is a non-trivial task that we are currently working on and have not yet solved to our full satisfaction.

4 Conclusions

In this paper, we have described three CPN related courses which we have taught and a number of lessons we have learned. Naturally, many of the lessons are of a general kind, applicable not only to the particular courses described here, but to many other kinds of courses as well: All teachers put an effort into finding good literature, making sure to have appropriate skills, and worrying about student motivations; many computer science teachers also have to deal with tools.

At this point in our writing, we would have liked to make a qualified comparative discussion of the experiences of colleagues who have taught Petri nets related courses at other universities. Berthelot and Petrucci report on experiences with education in relation to modelling, simulating, and verifying a train system using CPN and Design/CPN [2]. However, we have not been able to find many papers discussing Petri nets and education. Therefore, we encourage others to publish their experiences; we would like improve our teaching and to gain inspiration from an exchange of ideas with other teachers of Petri nets related courses.

If we take a broader perspective, going from teaching Petri nets to teaching computer science in general, there is a host of sources for more information, e.g., proceedings from the Innovation and Technology in Computer Science Education (ITiCSE) conferences, see, e.g., [3], and the ACM Curricula Recommendations [29]. Putting this paper into a broader perspective using such sources is future work.

We believe that over the years, DAIMI students and industrial software engineers have developed many useful skills from courses in which CPN has been used. This may well continue in the future. However, due to the general growth in computer science knowledge, the competition for a place in computer science curricula and for attention from the software industry is getting harder. At DAIMI, in the 1990's, CPN constituted about half of the curriculum of the distributed systems course. The amount of CPN has been decreasing; in 2003, CPN made up about 10-15 percent of the curriculum. In general, "exotic" subjects like CPN are at risk of having to leave the curriculum in order to accommodate something else, in particular in the undergraduate courses. Two examples on relatively recent additions to the undergraduate curriculum at DAIMI are a course on web technology and a course on security – that is tough competition. When use of Petri nets is considered in broader courses, there seems to always be good alternatives like process algebras (e.g., CCS [16]) and timed automata [1] in theoretical courses on concurrency and verification, and UML in software engineering oriented courses. In summary, we believe that it is worthwhile to think about good arguments to justify Petri nets in computer science curricula.

Acknowledgements

We thank Kurt Jensen, Lars M. Kristensen, and Thomas Mailund for allowing us to draw from their teaching experiences as a supplement to our own; we also thank them for discussions and helpful comments on this paper.

References

1. R. Alur and D. Dill. Automata for Modelling Real-Time Systems. *Theoretical Computer Science*, 126(2):183–236, 1994.
2. G. Berthelot and L. Petrucci. Specification and Validation of a Concurrent System: an Educational Project. *Software Tools for Technology Transfer*, 3(4):372–381, 2001.
3. R. Boyle and G. Evangelidis (eds.). Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education, 2003. Thessaloniki, Greece.
4. S. Christensen and J.B. Jørgensen. Analysing Bang & Olufsen’s BeoLink Audio/Video System Using Coloured Petri Nets. In P. Azema and G. Balbo, editors, *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, volume 1248 of *LNCS*, pages 387–406, Toulouse, France, 1997. Springer-Verlag.
5. S. Christensen, L.M. Kristensen, and T. Mailund. A Sweep-Line Method for State Space Exploration. In T. Margaria and W. Yi, editors, *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, volume 2031 of *LNCS*, pages 450–464, Genova, Italy, 201. Springer-Verlag.
6. S. Christensen and K.H. Mortensen. Teaching Coloured Petri Nets – A Gentle Introduction to Formal Methods in a Distributed Systems Course. In P. Azema and G. Balbo, editors, *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, *LNCS*, pages 290–309, Toulouse, France, 1997. Springer-Verlag.
7. G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems – Concepts and Design*. Addison-Wesley, 2001.
8. K. Jensen (ed.). Proceedings of the 3rd CPN Workshop, CPN’02. Technical report DAIMI PB-560, Department of Computer Science, University of Aarhus, 2002.
9. K. Jensen. Coloured Petri Nets and the Invariant Method. *Theoretical Computer Science*, 14:317–336, 1981.
10. K. Jensen. *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, 1992.
11. K. Jensen. *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, 1994.
12. K. Jensen and E.M. Schmidt. Pascal Semantics by a Combination of Denotational Semantics and High-level Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 222 of *LNCS*, pages 297–329. Springer-Verlag, 1985.
13. J.B. Jørgensen and C. Bossen. Requirements Engineering for a Pervasive Health Care System. In *Proceedings of the IEEE International Requirements Engineering Conference (RE’03)*, pages 55–64, Monterey Bay, California, 2003. IEEE.

14. J.B. Jørgensen and S. Christensen. Executable Design Models for a Pervasive Healthcare Middleware System. In J.M. Jézéquel, H. Hussmann, and S. Cook, editors, *Proceedings of the 5th International Conference on the Unified Modeling Language (UML'02)*, volume 2460 of *LNCS*, pages 140–149, Dresden, Germany, 2002. Springer-Verlag.
15. L.M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.
16. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
17. R. Milner, R. Harper, and M. Tofte. *The Definition of Standard ML*. MIT Press, 1990.
18. G. Monvelet, S. Christensen, H. Demmou, M. Paludetto, and J. Porras. Analysing a Mechatronic System with Coloured Petri Nets. *Software Tools for Technology Transfer*, 2(2):160–167, 1998.
19. OMG Unified Modeling Language Specification, Version 1.4. Object Management Group (OMG); UML Revision Taskforce, 2001.
20. J.L. Rasmussen and M. Singh. Designing a Security System by Means of Colored Petri Nets. In J. Billington and W. Reisig, editors, *Proceedings of the 17th International Conference on Application and Theory of Petri Nets*, volume 1091 of *LNCS*, pages 400–419, Osaka, Japan, 1996. Springer-Verlag.
21. A.V. Ratzer, L. Wells, H.M. Lassen, M. Laursen, J.F. Qvortrup, M.S. Stissing, M. Westergaard, S. Christensen, and K. Jensen. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. In W. van der Aalst and E. Best, editors, *Proceedings of the 24th International Conference on Application and Theory of Petri Nets*, volume 2679 of *LNCS*, pages 450–462, Eindhoven, The Netherlands, 2003. Springer-Verlag.
22. J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.
23. W. Stallings. *Data & Computer Communications, Sixth Edition*. Prentice Hall, 2000.
24. A.S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 1992.
25. A. Valmari. A Stubborn Attack on State Explosion. *Formal Methods in System Design*, 1:297–322, 1992.
26. Home page of the CPN Group at the University of Aarhus. www.daimi.au.dk/CPNets.
27. Home page of Design/CPN. www.daimi.au.dk/designCPN.
28. Home page of Kurt Jensen. www.daimi.au.dk/~kjensen.
29. Home page of ACM Curricula Recommendations. www.acm.org/education/curricula.html.
30. Home page of CPN Tools. www.daimi.au.dk/CPNtools.
31. Home page of 3rd CPN Workshop, CPN'02. www.daimi.au.dk/CPnets/Workshop02.
32. Home page of Department of Computer Science, University of Aarhus. www.daimi.au.dk.
33. Home page of Distributed Systems Course. www.daimi.au.dk/dDist (in Danish).