# WebSPN: Non-Markovian Stochastic Petri Net Tool

**A. Bobbio §, A. Puliafito *, M. Scarpa §, M. Telek †**

§ Dipartimento di Informatica
Università di Torino, 10149 Torino, Italy

* Istituto di Informatica e Telecomunicazioni
Università di Catania, 95125 Catania, Italy

† Department of Telecommunications
Technical University of Budapest, 1521 Budapest, Hungary

§ {bobbio,scarpa}@di.unito.it; * ap@iit.unict.it; † telek@hit.bme.hu

**Abstract**

This paper describes a new modeling tool for the analysis of non-Markovian stochastic Petri nets *(SPN)* that relax some of the restrictions present in currently available packages. This tool, called WebSPN, provides a discrete time approximation of the stochastic behaviour of the mrking process which results in the possibility to analyze a wider class of *PN* models with *prd* and *prs* concurrently enabled generally distributed transitions. WebSPN makes wide use of Java technology and it is easily accessible from any node connected with the Internet as long as it possesses a Java-enabled Web browser.

**Keywords:** Non-Markovian *SPN*, transient analysis, Java technology, Word Wide Web.

## 1 Introduction

The analytical approach to the evaluation of systems is being increasingly viewed as an integral part of the process of design, analysis and tuning of computer systems. Analytical models give results whose accuracy depends on the designer's ability and on the level of detail of the model employed. Furthermore, once the model has been developed, its solution is generally very quick, so an accurate analysis of the system can be made with the variation of all the model parameters.

Special model specification techniques are needed that help analysts to describe their systems in such a way that the models can be understood at the level of the system designer, rather than at the mathematical level. Software environments that support these specification techniques for analytical models are needed. Such environments (tools) should allow for the easy specification and efficient solution of the models. Furthermore, they should allow for the control of the numerical solution of the models as well as for a suitable presentation of the results.

Petri nets are commonly viewed as a valid tool for the qualitative and quantitative study of systems. Many Petri nets modeling tools have been proposed or developed recently (e.g. ESP [10], GSPN [5], SPNP [7], DSPNExpress [15], TimeNet [11], UltraSAN [9]).

Some of the above tools have also implemented the possibility of including some non-Markovian feature. The main restrictions, in the already existing tools, are associated with generally distributed firing time transition with *prs* preemption policy and with concurrently enabled general transitions. The first restriction can be relaxed by the analytical results available for the analysis of *PN* with non-overlapping *prs* general transitions [4], and there is an active research to find the proper way to analyze *PN* with concurrently active general transitions [15, 2]. However, none of the existing modeling tools incorporates the capability of analyzing these models automatically. The only possible approach for the analysis of *PN* models, with *prs* and *prd* general transitions, is the Phase type *(PH)* approximation. With this technique, the marking process of the non-Markovian *SPN* is approximated byan expanded Markov chain. The main drawback of the *PH* approximation consists in the very large state space of the expanded Markov chain, mainly if the random firing times have a low coefficient of variation.

In this paper, we present a new modeling tool for the analysis of non-Markovian stochastic Petri nets that relax some of the restrictions present in currently available modeling packages. This tool, called WebSPN, provides a discrete time approximation of the stochastic behaviour of the marking process which results in the possibility to analyze a wider class of Petri net models with *prd* and *prs* concurrently enabled generally distributed transitions. WebSPN is completely developed in Java and implements some powerful communication mechanisms which allow a generic user connected to the Internet to remotely access the tool regardless of the type of his/her hw/sw platform, which can be Unix, Windows95, Mac etc... The only requirement is a Web browser which can execute code written in Java. WebSPN also provides authentication services in order to allow only authorized users to use the tool.

The rest of the paper is organized as follows: in Section 2 we will outline the features of Java and its use on the Web; in Section 3 we will review the main concepts of non-Markovian stochastic Petri nets while in Section 4 we briefly outline the analytical approach adopted for the solution of the model. Section 5 shows how it is possible to use the Java technology for the Web sharing of WebSPN. Section 6 provides an application example, presents the graphical user interface of WebSPN and reports some comparative numerical results. Conclusions are given in Section 7.

## 2 The Java approach

Java is an object-oriented, portable, interpreted, multithreaded programming language developed by Sun [13]. Although it is a new language it is simple and familiar because a number of its constructs are taken from C++, eliminating some of the more complex aspects which are often source of errors. This feature ensures high programmer productivity right from the start. Another fundamental characteristic is the security which a language designed for use in a network environment has to guarantee. It also provides both compile-time and run-time checking mechanisms which allow the development of reliable and robust applications.

The Java *multithreading* capability provides the means to build applications with many concurrent threads of activity. Although recent operating systems provide libraries for multithreading, their dependence on a particular architecture prevent them from being used by a large number of programmers. Java, on the other hand, supports multithreading at the language level, providing sophisticated synchronization primitives.

A feature that has permitted the language to spread is the possibility of inserting small Java applications (*applets*) into a Web page. A Java-enabled Web browser recognises a particular tag on

a normal *html* page identifying the *url* of the applet; when it interprets this tag it sends a request to the Web server from which it will receive the bytecode of the applet. Then the applet is executed on the local client inside the browser itself.

A basic concept of the Java architecture is that of *class loaders*. The Java runtime system is allowed to load classes without any knowledge of the underlying file system. All Java Virtual Machines include a default class loader which loads the classes of the local file system, but Java allows the user to define his own class loader, thus overriding the behaviour of the default loader. This makes it possible to load classes from various sources, as is done, for example, in applets, which load classes from a Web server using the http protocol.

# 3 Non-Markovian stochastic Petri nets

We define a non-Markovian *SPN* as a stochastically timed *PN* in which the time evolution of the marking process cannot be mapped into a *Continuous Time Markov Chain (CTMC)*. Recent research on this subject is documented in [6, 12, 3, 8]. In the spirit of many modeling formalism, in which the complexity of the solution must be hidden to the modeler, a complete set of specifications must be given at the *PN* level, in order to univocally define the underlying marking process. Therefore, the way in which the future evolution of the marking process depends on its past history needs to be specified at the *PN* level.

A consistent way to introduce memory into a *SPN* is provided in [1] and extended in [4]. Each timed transition $t_g$ is assigned a general random firing time $\gamma_g$ with a cumulative distribution function $G_g(t)$. A clock, associated to each individual transition, counts the time in which the transition has been enabled. An *age variable* $a_g$ associated to the timed transition $t_g$ keeps track of the clock count. A timed transition fires as soon as the memory variable $a_g$ reaches the value of the firing time $\gamma_g$. In the original view [1], two main firing policies were introduced:

- *Preemptive resume (prs)*: the associated clock counts the time according to an age memory policy: the age variable $a_g$ is reset only when $t_g$ fires.
- *Preemptive repeat different (prd)*: the associated clock counts the time according to an enabling memory policy: the age variable $a_g$ is reset each time $t_g$ is disabled or fires.

In the described individual memory models, a *prs* transition cannot be disabled and restarted before firing, while a *prd* transition is restarted each time the corresponding transition is disabled or fires. It should be noted that the *prd* policy is the only considered in the available tools modeling non-Markovian *SPN* [15, 11, 9].

# 4 Algorithm description

The algorithm developed to provide an approximation of the transient and steady state behaviour of a non-Markovian SPN is based on a discretization of the time. The preemption policies considered in WebSPN are *prs* and *prd* type as discussed above.

The approximation of the continuous time model at equispaced discrete time points involves the analysis of the system behaviour over a time interval based on the system state at the beginning of the interval and the past history of the system. The length of the constant discretization interval must be chosen as a function of the random time variables of the model in such a way that the

piece-wise discretized functions provide a sufficiently accurate approximation of the corresponding continuous functions. In the case of a *prd* or a *prs* general transition, the past history is condensed into a single age variable, so that the remaining firing time can be computed conditioned on the time already spent in an enabling condition (represented by the age variable). In the developed analytical approach the value of the continuous age variables are discretized as well.

Based on these considerations an elementary step of the approximation method is as follows:

- analysis of the behavior of the marking process inside a single time interval. This analysis is based on the the associated age variables at the beginning of the interval and on the state reached by the system at the end of the previous interval;
- evaluation of the values of the associated variables at the end of the current time interval.
- determination of the transition probabilities over the reachable states of the discretized state space at a fixed time (the lenght of the discretization interval).

The system behaviour is approximated by a Discrete Time Markov Chain (DTMC) over an expanded state space determined by the cross product of the system states (the markings of the Petri net) and the discretized values of the associated age variables. This approach is very similar to the *PH* expansion method [10] in several senses. The main difference is that, in this case, the system behavior is approximated by an expanded DTMC while in the *PH* approximation case an expanded CTMC is obtained. The present approach inherits some similarities also from the supplementary variable approach [12], since the supplementary (age) variables are constrained to assume values in a discretized set.

The main steps of the implemented solution method are the following:

- generation of the reachability graph (with tangible and vanishing states), and reduction of the reachability graph to tangible states, only;
- generation and analysis of the expanded DTMC;
- evaluation of the final measures at the net level, based on the solution of the expanded DTMC.

# 5 Design Issues

The use of Java technology allows to share the WebSPN tool through the Web. The tool can be accessed from any node connected with the Internet, as long as it possesses a Java-enabled Web browser. To make the tool available only to authorized users, adequate security mechanisms based on public and private key algorithms are included, which provide authentication services and, if required, encryption. In this way, only users with regular licences can access the applications in question. The approach proposed was successfully used to port the Sharpe tool (Symbolic Hierarchical Automated Reliability/Performance Evaluator) onto the Web [18, 17].

## 5.1 Communication Mechanisms

The approach we followed can be seen as an extension of the client/server programming paradigm. The client, in fact, (1) processes locally the request to be sent to the server, who (2) executes it at a different time, at the end of which (3) it notifies the client of the results of the calculation. Unlike

the classical approach, however, the client does not need to possess any specific software; through a simple Web interface it loads the software using the mechanisms provided by Java. The immediate advantage is the simplicity of access to the application and the total absence of a preliminary phase to distribute and install the interface software. During connection with the server the client loads the software required for the graphic interface, as well as the modules necessary for future communication sessions. The application provider can then update the application, modifying the interface as he likes, without having to provide potential clients with updated versions of the software. The user, in turn, has the certainty of always using the latest version of the software, and can also count on optimal installation and an execution speed that is not always possible with his own computing resources. The immediate retrievability on the Web ensures the complete availability of applications that otherwise would probably remain known only to a limited number of potential users. In this sense, an immediate use for the design choice we propose is in teaching, to allow students easy, economic access to the modeling tool.

The only requirement for the client is a Java-enabled Web browser, while the server needs the following software modules:

- Web server;
- Java Virtual Machine;
- Application to be made available on the network;
- Java applet of the user interface;
- Software module to run the communication session with the client.

The last module, entirely developed in Java, comprises two submodules. One of these, in particular, is transferred to the client when the latter forwards a request for access to the server and provides the client with the mechanisms needed to run the communication sessions just started. The second submodule, on the other hand, is always in execution on the server and deals with accepting requests from various clients, robustly managing the various connections with clients, and sending clients the results put out by the server. It also keeps a memory of the correspondence between clients and the applications they use.

### 5.2 Security and Access Control

Network sharing of WebSPN requires the creation of suitable security mechanisms to regulate access, reserving access to previously authorized users. It is therefore necessary to provide authentication services and, if required, encryption. The techniques we will use are based on public and private key algorithms [14, 19].

Below we will refer to a server S and a generic client C that wants to use the services provided by S. In our structure the server S also represents the authority issuing certificates for the clients' public keys. The communication protocol can be subdivided into the following 3 stages, shown in Fig. 1: *registration, initialization* and *data transfer*.

**Registration Stage:** In this stage of the security protocol C and S agree on the type of services offered/required and the access modes. C then generates two keys, a private one and a public one, giving the latter to S through a safe channel (typically not through the same communication channel

that subsequent connections will use) and keeping the private one safely (the key is often encrypted with an alphanumerical password and then memorized).

**Initialization Stage:** This stage starts when C decides to link up with S to use a certain application. Neglecting the loading of the applet through Web, as any security mechanisms are incorporated in the Web browser, this stage can be schematically represented as follows:

- C sends a connection request to S;
- S sends a signed message indicating that the request has been received;
- C replies sending a signed message containing the security services required (confidentiality and authentication or authentication alone), and a code identifying its public key;
- S checks C's signature on the message received and if it is recognized, sends C a signed acknowledgement and starts to transfer the data; if the authentication stage fails the connection is immediately interrupted.

**Data Transfer Stage:** during this stage each message sent is treated according to the security services requested. If confidentiality and authentication have been requested the message will be composed of two fields, one in which the data is encrypted and another containing the sender's signature.

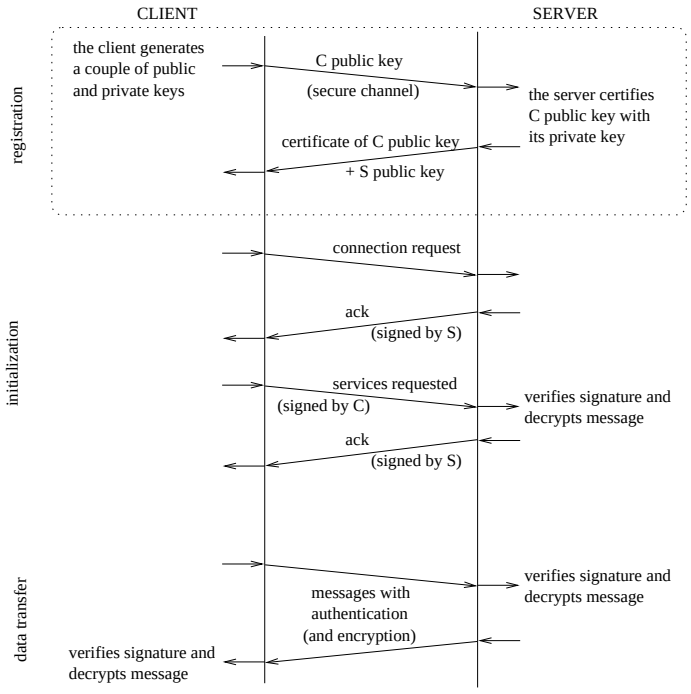


Figure 1: Security protocol

# 6 Application example

A simple application example is completely developed to illustrate the use of the graphical user interface and to show the numerical capabilities of the implemented discrete time approximation method in the presence of both *prd* and *prs* memory policies.

## 6.1 The Graphical User Interface

To load the graphical user interface of WebSPN it is necessary to link up with the Web page containing the link for the applet and click on the relative icon: the subsequent loading and execution of the interface onto the local machine is quite transparent to the user.

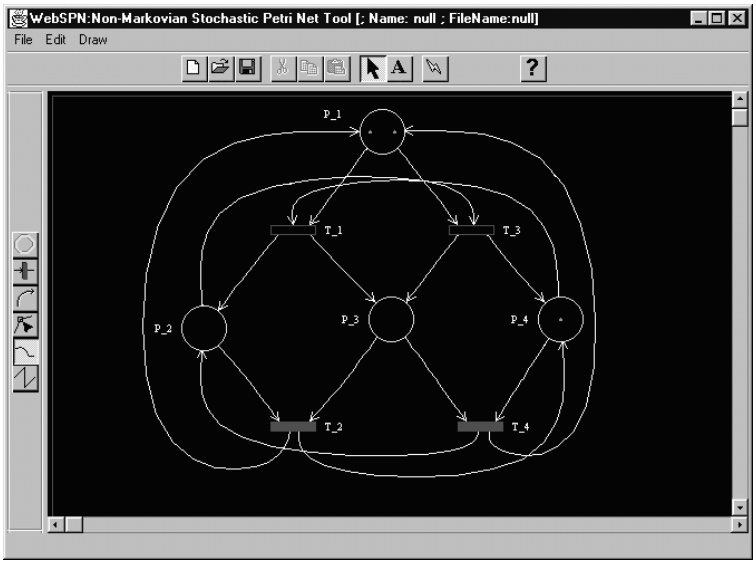


Figure 2: Main display of WebSPN

The main WebSPN display, shown in Fig. 2, has the following four zones: a) *Menu panel*; b) *Control panel*; c) *Design area*; d) *Status panel*.

*The Menu Panel* offers the usual choices. Besides the submenus *File* and *Edit* there is also the *Draw* submenu with items which allow the user to select the graphic primitives to be used in the specification phase. More immediate use of the graphic functions is provided by a series of push buttons on the left hand side of the display.

*The Control Panel* can be used to activate a series of functions to create, load and save a model (there are also the classical cut, copy and paste functions) and others to activate the solution of the model and to add some text.

*The Design Area* is where the user is allowed to draw his model. Significant graphic symbols and the associated dialogue boxes are enabled.

*The Status Panel* gives run-time indications regarding the status of the interface, signalling the occurrence of any event that may be of interest to the user.

With a double click on a generic primitive, the user gains access to a dialogue box with which it is possible to specify the properties of the primitive. In the case the primitive is a place, a name can be assigned as well as the initial number of tokens. If a transition is selected, then a dialog box will appear which offers the possibility to change its default name and orientation and to specify its timing (immediate, exponential or general). If the transition is defined as generally distributed, then the following choices are available: deterministic, uniform and Weibull (only the deterministic case is implemented in the current version of the tool). Moreover, for any general transition a memory policy (among *prd* and *prs*) must be assigned.

Once the specification stage is completed, the user passes to the analysis stage simply by pressing the *Analyze* key which opens a dialogue box in which the user specifies the evaluation indices he wants.

Pressing the *Ok* button the graphic representation is converted into text format using an internal specification syntax. The ASCII file thus created is then transferred to the server where the management module begins execution of a new WebSPN instance. Once the processing is completed the results are transmitted back to the client.

Another possibility offered by WebSPN is saving the graphic description of a model in the *Xfig* format. *Xfig* is a public domain tool for vector graphics on Unix, frequently used in academic environments. The picture in this format can be easily manipulated on different platforms and converted into one of the many available graphic formats (e.g., PostScript).

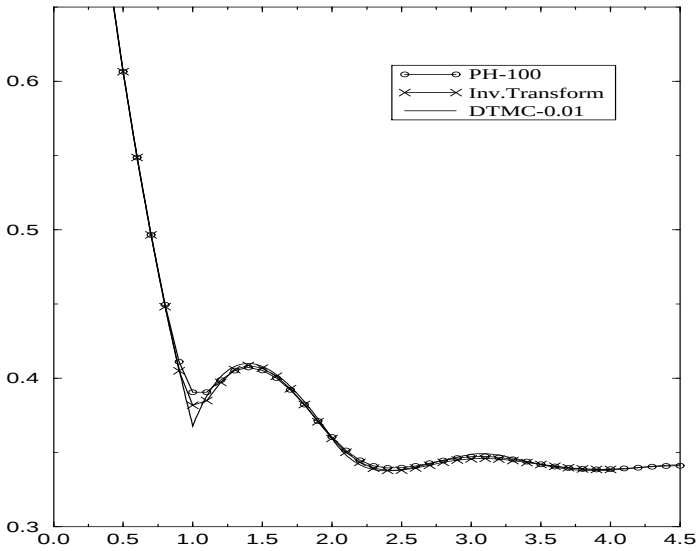


Figure 3: Transient probability of State 1 with *prd* preemption policy

## 6.2 Numerical Results

The Petri net considered in this example is shown in Figure 2. This PN models an M/D/1/2/2 queue with preemptive LILO service policy. The preemption policy can be either *prd* or *prs*. A detailed description of the model is in [3]. Transitions $t_1$ and $t_3$ represent the job submitting time and are assumed to be exponential with rate $\lambda = 0.5$. Transitions $t_2$ and $t_4$ model the service

duration and are assumed to be deterministic with duration $d = 1$, and with associated *prd* or *prs* memory policy. The example was already evaluated with the PH expansion method and with the numerical inverse transformation method for the *prd* policy in [3], and with the numerical inverse transformation method for the *prs* policy in [4].

The calculated measure is the probability versus time of the initial marking (state 1) which represents the server in the empty state and both the customers in the thinking phase. On Figure 3 and 4, the results of the transient analysis of the three methods, the numerical inversion, the PH approximation and the discretization, are reported with *prd* and with *prs* preemption policy, respectively.

As can be observed, the different numerical approaches result in different numerical properties of the evaluated measure. Among the three analyzed numerical methods, the DTMC approximation, implemented in WebSPN, better captures the sharp changes in the transient behaviour, for both the considered preemption policies.

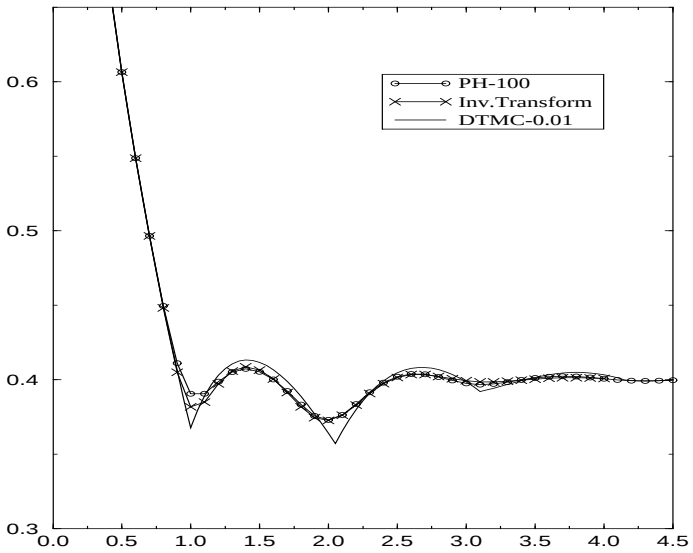


Figure 4: Transient probability of State 1 with *prs* preemption policy

# 7 Conclusion

A new modeling tool, called WebSPN, for specification and automatic solution of non-Markovian *SPN* has been described. The tool implements a time-discretization approach and allows concurrent enabling of generally distributed transitions with *prd* and *prs* preemption policies. Due to the use of the Java programming language, WebSPN is easily accessible from any node connected with the Internet as long as it possesses a Java-enabled Web browser. The tool was developed using the Java Development Kit version 1.0.2. The tool was implemented mostly on Windows95 PC and Sun/Solaris systems. The server functions were performed by a Sun SparcStation20 on which a Web Server (NCSA HTTPd 1.5.2) and the WebSPN modeling tool were installed.

## Acknowledgements

This work has been partially supported by Italian CNR under Grant No. 96.01939.CT12 and Hungarian OTKA under Grant No. T-16637.

## References

[1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, SE-15:832–846, 1989.

[2] A. Bobbio and L. Roberti. Distribution of the minimal completion time of parallel tasks in multi-reward semi-Markov models. *Performance Evaluation*, 14:239–256, 1992.

[3] A. Bobbio and M. Telek. Computational restrictions for SPN with generally distributed transition times. In D. Hammer K. Echtle and D. Powell, editors, *First European Dependable Computing Conference (EDCC-1), Lecture Notes in Computer Science*, volume 852, pages 131–148. Springer Verlag, 1994.

[4] A. Bobbio and M. Telek. Markov regenerative SPN with non-overlapping activity cycles. In *International Computer Performance and Dependability Symposium - IPDS95*, pages 124–133. IEEE CS Press, 1995.

[5] G. Chiola. *GreatSPN* 1.5 Software architecture. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation*, pages 121–136. Elsevier Science Publishers, 1992.

[6] Hoon Choi, V.G. Kulkarni, and K. Trivedi. Markov regenerative stochastic Petri nets. *Performance Evaluation*, 20:337–357, 1994.

[7] G. Ciardo, J. Muppala, and K.S. Trivedi. SPNP: stochastic Petri net package. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM89*, pages 142–151. IEEE Computer Society, 1989.

[8] G. Ciardo, R. German and C. Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. *IEEE Transactions on Software Engineering*, 20:506–515, 1994.

[9] J.A. Couvillon, R. Freire, R. Johnson, W.D. Obal, M.A. Qureshi, M. Rai, W. Sanders, and J.E. Tvedt. Performability modeling with UltrasSAN. *IEEE Software*, 8:69–80, September 1991.

[10] A. Cumani. Esp - A package for the evaluation of stochastic Petri nets with phase-type distributed transition times. In *Proceedings International Workshop Timed Petri Nets*, pages 144–151, Torino (Italy), 1985. IEEE Computer Society Press no. 674.

[11] R. German, C. Kelling, A. Zimmermann, and G. Hommel. *TimeNET - A toolkit for evaluating non-markovian stochastic Petri nets.* Report No. 19 - Technische Universität Berlin, 1994.

[12] R. German. New results for the analysis of deterministic and stochastic Petri nets. In *International Computer Performance and Dependability Symposium - IPDS95*, pages 114–123. IEEE CS Press, 1995.

[13] Arnold Gosling. *The Java Programming Language*. Addison Wesley - The Java Series, 1996.

[14] L. Hughes. *Actually Useful Internet Security Techniques*. New Riders Publishing, 1995.

[15] C. Lindemann. DSPNexpress: a software package for the efficient solution of deterministic and stochastic Petri nets. *Performance Evaluation*, 22:3–21, 1995.

[16] M.K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, C-31:913–917, 1982.

[17] A. Puliafito, O. Tomarchio, and L. Vita. Porting sharpe on the web: Design and implementation of a network computing platform using JAVA. In *Proceedings TOOL'97*, Saint Malo, (France), June 1997.

[18] R. A. Sahner, K. S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems*. Kluwer Academic Publishers, November 1995.

[19] W. Stalling. *Network and Internetwork Security Principles and Practice*. Prentice Hall, 1995.