ELSEVIER

Performance Evaluation 22 (1995) 3–21

PERFORMANCE EVALUATION
An International Journal

DSPNexpress: a software package for the efficient solution of deterministic and stochastic Petri nets ☆

Christoph Lindemann

GMD Institute for Computer Architecture and Software Technology (GMD-FIRST), Technical University of Berlin, Rudower Chaussee 5, 12489 Berlin, Germany

Abstract

This paper describes the analysis tool DSPNexpress which has been developed at the Technische Universität Berlin since 1991. The development of DSPNexpress has been motivated by the lack of a powerful software package for the numerical solution of deterministic and stochastic Petri nets (DSPNs) and the complexity requirements imposed by evaluating memory consistency models for multicomputer systems. The development of DSPNexpress has gained by the author's experience with the version 1.4 of the software package GreatSPN. However, opposed to GreatSPN, the software architecture of DSPNexpress is particularly tailored to the numerical evaluation of DSPNs. Furthermore, DSPNexpress contains a graphical interface running under the X11 window system. To the best of the author's knowledge, DSPNexpress is the first software package which contains an efficient numerical algorithm for computing steady-state solutions of DSPNs.

Keywords: Software packages for stochastic Petri nets; Numerical methods for transient analysis of Markov chains; Performance and dependability modeling

1. Introduction

During the last decade several classes of Petri net models in which transition firings are augmented with time have been proposed in order to define a unified modeling technique for both formal description and quantitative analysis of systems with concurrency. Such Petri nets have been broadly accepted for modeling computer and communication systems due to the availability of appropriate software packages which completely automate their solution process. A variety of such analysis tools have been presented in the literature (e.g. GreatSPN [5,7], SPNP [8], UltraSAN [10]).

Deterministic and stochastic Petri nets (DSPNs [2]) are particular well suited for modeling computer systems and communication networks because this modeling tool incorporates both

☆ This work has been supported by the Federal Ministry for Research and Technology of Germany (BMFT) under grant ITR9003.

SSDI 0166-5316(93)E0035-4

deterministic and stochastic delays. Under the restriction that in no marking of the DSPN two or more deterministic transitions are concurrently enabled, Ajmone Marsan and Chiola showed how to the derive the steady-state solution of a DSPN by the method of the embedded Markov chain (EMC). Previously, the applicability of DSPNs for modeling complex systems has been severely hampered by the high computational effort of the numerical algorithm for computing the transition probability matrix of the EMC implemented in the package GreatSPN1.4. In the version 1.5 of GreatSPN the numerical solution component for DSPNs has been removed and a software module for timed interactive simulation has been added [7].

The development of the analysis tool DSPNexpress has been motivated by the lack of a software package for an efficient numerical analysis of DSPNs and the complexity requirements imposed by evaluating memory consistency models for multicomputer systems [15]. The development of DSPNexpress has gained by the author's experience with the version 1.4 of the software package GreatSPN [5], which has been available in source code. However, opposed to GreatSPN1.4, the software architecture of DSPNexpress is particularly tailored to the numerical evaluation of DSPNs. During the generation of the reachability graph of a DSPN, the Markov chains defined by exponential transitions competitively or concurrently enabled with a deterministic transition are derived. These Markov chains are subsequently called *subordinated Markov chains* (SMCs) of deterministic transitions. For each connected component of such a SMC, the transient state probabilities and the mean sojourn times in their states are efficiently calculated by the numerical algorithm introduced in [13]. The former values define the transition probabilities of the EMC. The latter values are used to convert the steady-state solution of the discrete-time EMC to the marking probability vector of the corresponding DSPN with continuous-time stochastic behavior. Thus, these mean sojourn times have been referred to as *conversion factors* in [2].

The separate transient evaluation of each connected component of a SMC is related to the decomposition approach on the net level proposed by Ajmone Marsan et al. [3]. In their decomposition approach, *DSPN subnets with independent behavior* have to be identified by means of structural analysis on the net level. To obtain the transition probabilities of the EMC of a DSPN this approach requires a proper combination of the transient quantities calculated separately for each subnet. The algorithm implemented in DSPNexpress employs a depth-first-search algorithm for deriving the generator matrices of connected components of each SMC from the reachability graph of tangible markings of a DSPN. The transient analysis of a SMC yields immediately the corresponding transition probabilities of the EMC underlying the DSPN. The interaction between software modules of DSPNexpress is performed mostly by interprocess communication by means of sockets. As a consequence, the system overhead required by reading from or writing to files is substantially reduced. Moreover, this allows a parallel execution of the transient analysis of SMCs on a cluster of workstations. Due to this efficient numerical DSPN solution algorithm DSPNexpress is able to calculate steady-state solutions of complex DSPNs with reasonable computational effort on a modern workstation. To the best of the author's knowledge DSPNexpress is the first software package with this feature. Moreover, the package DSPNexpress has a graphical interface running under the release 5 of the X11 window system [16].

The remainder of this paper is organized as follows. In Section 2 three related software packages are described in order to motivate the development of DSPNexpress. Section 3

introduces the design of the software architecture and the numerical solution modules of DSPNexpress. In Section 4 the features of the graphical interface of DSPNexpress are described. To illustrate the applicability of DSPNexpress for solving complex DSPNs some performance curves are presented in Section 5. Finally, concluding remarks are given.

2. Motivation for developing DSPNexpress

The following describes three software packages which are closely related to DSPNexpress. Since 1986, Giovanni Chiola has been developing at the University of Torino the software package *Gr*aphical *E*ditor and *A*nalyzer for *T*imed and *S*tochastic *P*etri *N*ets (GreatSPN [5,7]). In 1989 the version 1.4 of the software package GreatSPN has become available. One major contribution of GreatSPN lies in its user-friendly graphical interface which allows to interactively validate a model [5]. This feature clearly constitutes an asset in the modeling process. Moreover, GreatSPN has been the only software package which has contained a numerical solution algorithm for DSPNs. However, the version 1.4 of GreatSPN contained rather inefficient numerical solution algorithm for calculating transient solutions of GSPNs and the steady-state solution of DSPNs. In particular, the numerical DSPN solution component of GreatSPN severely hampered their applicability for modeling complex systems. In a recent paper the version 1.5 of GreatSPN has been introduced [7]. In this newest version of the software package GreatSPN the numerical solution module for DSPNs has been removed and a software module for interactive timed simulation has been added. The availability of this component constitutes one of the major contributions of GreatSPN1.5. Moreover, several "compiling techniques" [6] have been implemented which yield a reduction of run time and memory space required for the reachability graph construction. However, the numerical solution algorithm for calculating transient solutions of GSPN models is still based on an adaptive matrix exponentiation method which has proven to be rather inefficient [12].

At Duke University, Kishor Trivedi together with his former students Gianfranco Ciardo and Jogesh Muppala have been developing the *S*tochastic *P*etri *N*et *P*ackage (SPNP [8]). The version 2.0 of SPNP contains robust and orthogonal software modules for calculating the steady-state solution of a GSPN. One of the main contributions of the development of SPNP is the implementation of algorithms for an automated composition of large GSPNs and the sensitivity analysis of GSPNs to model parameters. Moreover, SPNP can process very general reward specifications which are useful for an integrated performance and dependability evaluation of fault-tolerant systems. The version 3.0 of SPNP contains also an efficient software module for transient analysis of GSPNs which is based on the randomization technique with left-truncation. But SPNP3.0 does not contain a numerical solution algorithm for DSPNs. Users of SPNP have to specify the GSPN to be evaluated via an alphanumerical interface in a C-based specification language.

Recently, William Sanders together with a group of students introduced the software package UltraSAN [10]. The package UltraSAN contains numerical solution components for transient and steady-state analysis of stochastic activity networks (SANs) in which all timed activities have exponentially distributed delays. SANs are similar to stochastic Petri nets which include timed and immediate transitions (there called activities). UltraSAN contains also a

simulator for transient and steady-state analysis of SANs incorporating timed activities with arbitrary distributed delays (e.g. deterministic). Furthermore, the package UltraSAN incorporates a software module for an automated generation of a *reduced base model* of a hierarchically-defined SAN [10]. The reduced base model is tailored to the computation of the particular reward measures of the SAN and typically contains considerably fewer states than the SAN. Moreover, UltraSAN contains a user-friendly graphical interface running under X11 windows. These features are the main contributions of the package UltraSAN.

The lack of a powerful software package for an efficient numerical evaluation of deterministic and stochastic Petri nets motivated the author to develop a new software package particularly tailored for this purpose. This analysis tool has been called DSPNexpress. In the following the main features of the software package DSPNexpress are outlined.

(1) DSPNexpress contains an efficient numerical algorithm for calculating the state transition probabilities of the EMC of a DSPN and the corresponding conversion factors. A similar algorithm is employed for calculating transient solutions of a GSPN. These numerical algorithms are based on the randomization technique improved by a stable calculation of Poisson probabilities and have been introduced in [13] and [12].

(2) The DSPN solution module of DSPNexpress corsiders each connected component of a Markov chain subordinated to a deterministic transition of a DSPN, separately, for calculating the corresponding transition probabilities of the EMC and the conversion factors. This leads to a considerable reduction of the computational effort and the memory requirements of the DSPN solution algorithm. Moreover, it allows invoking multiple instances of the appropriate procedure which may run in parallel on a cluster of workstations.

(3) A numerical solution approach for dealing with marking-dependent firing delays of deterministic transitions is implemented in DSPNexpress. It has been shown in [14] how the DSPN solution process introduced in [2] can be extended in order to cope with marking-dependent firing delays of deterministic transitions. The basic idea is to scale each row of the generator matrix of the SMC of this deterministic transition by the delay specified for the corresponding marking. Thus, the general approach for dealing with marking-dependent firing delays introduced in [1] has been tailored to the deterministic case. This extension to the modeling power of DSPNs is useful for representing state-dependent deterministic service times which occur in fault-tolerant systems with gracefully degradable performance [14].

(4) The organization of DSPNexpress exploits the property that each GSPN can be considered as a DSPN without deterministic transitions. As a consequence, a unified solution process for both DSPN and GSPN models is provided by DSPNexpress.

(5) The interaction between software modules of the DSPN solution process is mostly performed by interprocess communication with sockets [11] rather than by employing input/output files. This yields a substantial reduction of the system overhead required by I/O operations from and to the disk in case of evaluating complex DSPNs.

(6) In DSPNexpress, sparse implementations of the direct Gaussian elimination [17] and the iterative successive overrelaxation (SOR) method with the improved adaptive computation of the relaxation factor as proposed in [9,18] have been implemented. Depending on the properties of the transition probability matrix of the EMC the appropriate numerical

Table 1
Comparison of SPNP3.0, GreatSPN1.5, UltraSAN2.0 and DSPNexpress1.3

	SPNP3.0	GreatSPN1.5	UltraSAN2.0	DSPNexpress1.3
Structural analysis	reachability graph construction for GSPNs allows the preservation of vanishing markings	reachability graph construction for confusion-free nets extensive features for structural analysis available	reachability graph construction for SANs	reachability graph construction for confusion-free nets some features for structural analysis available
SPN extensions	mark.-dep. arc multiplicities, enabling functions, rewards	N/A	cases and gates, rewards	mark.-dep. firing delays for deterministic transitions
Numerical steady-state analysis for GSPNs	iterative near optimal SOR or Gauß–Seidel	iterative Gauß–Seidel	near optimal SOR with convergence monitoring sparse implementation of direct Gaussian Elimination	iterative near optimal SOR or Gauß–Seidel sparse implementation of direct Gaussian Elimination
Numerical transient analysis for GSPNs	randomization enhanced by left-truncation and steadystate check	adaptive matrix exponentiation	randomization enhanced by a stable calculation of Poisson probabilities	randomization enhanced by a stable calculation of Poisson probabilities
Numerical steady-state analysis for DSPNs	N/A	N/A	N/A	efficient solution module which may run in parallel
Stochastic simulation of SPNs with arbitrary timing	N/A	discrete event simulator timed interactive simulation for quantitative validation	discrete event simulator	N/A
Model composition or decomposition	support of automated GSPN composition	N/A	automated reduced base model construction of SANs	N/A
Non-standard techniques	automated sensitivity analysis calculation of mean time to absorption	N/A	calculation of accumulated reward measures in a time interval	N/A
Interaction between software modules	input/output files	input/output files interprocess communication employed in timed simulator	input/output files interprocess communication employed in timed simulator	reliable interprocess communication by means of sockets
User interface	alphanumerical interface using a C-based language	graphical interface running under Open Windows	graphical interface running under X11	graphical interface running under X11

method for solving the linear system of its global balance equations is chosen. If an iterative method is selected, the convergence will be monitored. In case of bad convergence the algorithm retries the calculation of the state probability vector by the Gaussian elimination method.

(7) All software modules of DSPNexpress are implemented in the programming language C to ensure good portability of the package. DSPNexpress employs the UNIX™ tools lex, yacc, csh, rsh, make, and several UNIX™ system calls (e.g. clock and exec).

(8) The reachability graph construction is performed for confusion-free DSPNs by a software component which is based on the method proposed in [4]. The coding structure of the search tree is organized as proposed in [10]. The Markov chains subordinated to the deterministic transitions are derived during the reachability graph construction.

(9) Checking for syntax errors in specifications of user-defined reward measures and marking-dependent firing delays is performed at the beginning of the DSPN solution process. Thus, syntax errors in specifications are detected before any useless computation is started.

(10) DSPNexpress has a user-friendly graphical interface running under the X11 window system which is described in Section 4.

In Table 1 DSPNexpress is compared with the software packages SPNP, GreatSPN, and UltraSAN according to their description in [7], [8], and [10], respectively. If a specific feature is not available in one of the packages, the appropriate entry is labeled with N/A.

3. Organization of the software package DSPNexpress

Similar to the other analysis tools for stochastic Petri nets mentioned above the package DSPNexpress is organized as several sets of software modules which are stored in separate directories of a UNIX file system. We originally developed the package DSPNexpress for Sun™ workstations under SunOS.4.1, but the package has recently been ported to DEC™ and HP™ workstations under ULTRIX4.2 and HP-UX9.0, respectively. All software modules of DSPNexpress are implemented in the programming language C.

To exploit the power of the numerical DSPN solution algorithm of DSPNexpress for solving complex DSPNs the package should run on machines with at least 16 MByte main memory. DSPNexpress allows a multi-user mode by including a link to the global directory DSPNexpress1.3 in the path expression in each user's shell profile. Only the model descriptions and the user-defined settings for DSPNexpress are stored in a local directory at each user's account. Figure 1 shows the organization of the global part of DSPNexpress from which the solution programs may be invoked by users. Each user of DSPNexpress has to include a link to DSPNexpress in the path expression in the shell profile from which DSPNexpress is called. A directory *src* consists of program components in source code and a makefile. A directory *bin* of the same anchor directory contains the corresponding object code of these software modules. The directory *DSPNINTERFACE* contains the software modules which implement the graphical interface of DSPNexpress1.3. The subdirectory *include* contains precompiler commands for defining constants and including system files of the X11 programming library. The subdirectory *help* contains text files of the help messages. The features of the graphical interface are

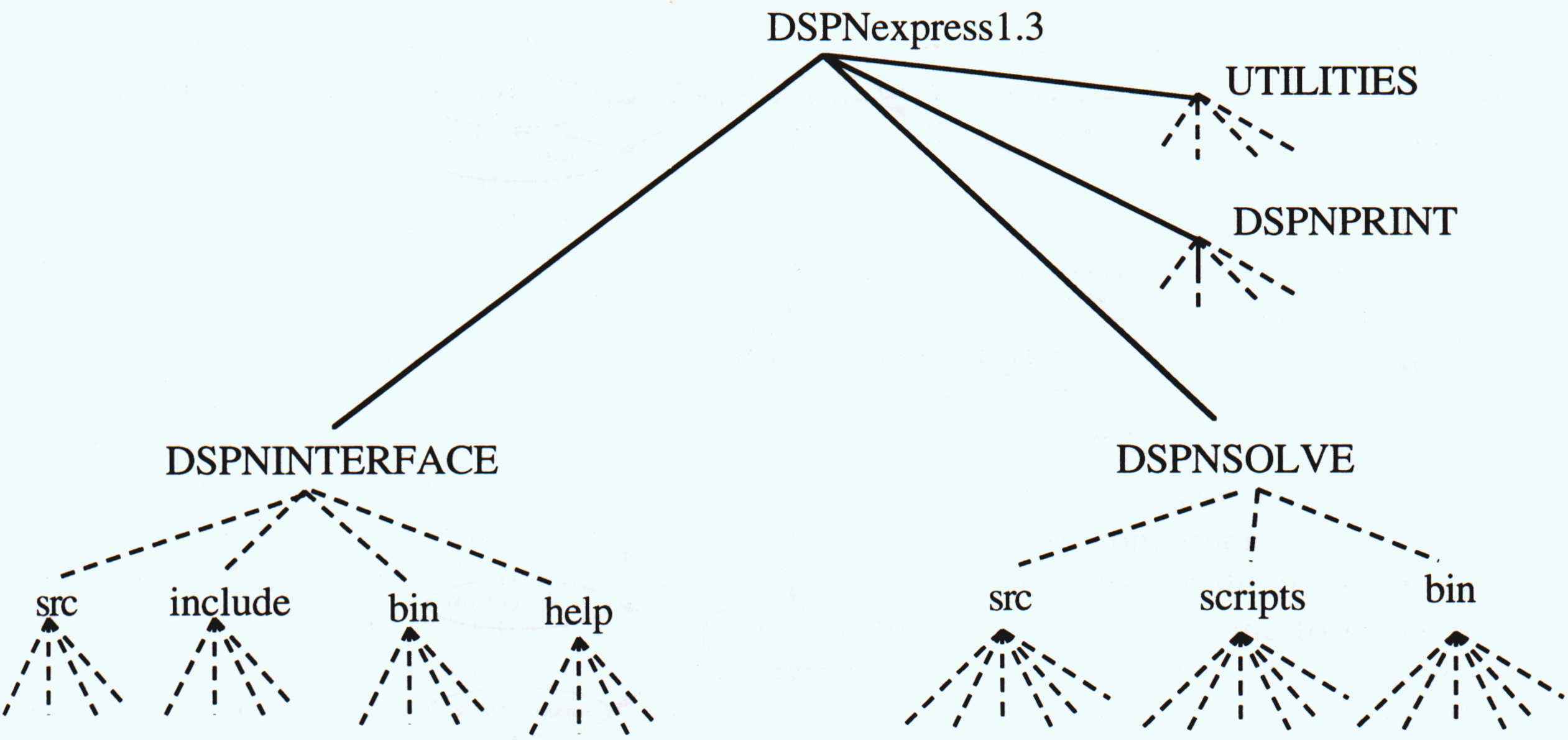

Fig. 1. Organization of DSPNexpress at the master account.

described in Section 4. The directory *DSPNSOLVE* contains software modules which implement the numerical solution process of DSPNs. The subdirectory *scripts* consists of shell scripts which call the executable programs of the subdirectory *bin*. The directory *DSPNPRINT* contains a software module for generating LaTeX source or postscript files of the textual and graphical description of a DSPN. The directory *UTILITIES* contains useful utility programs such as a shell script *INSTALL* which performs the installation of the package by a single command.

The local structure of DSPNexpress is depicted in Fig. 2. The directory *Models* contains the specification files of DSPNs which have been inserted previously via the graphical interface of DSPNexpress. The file *.DSPNexpress* contains user-defined settings for DSPNexpress such as the error tolerance with which numerical solution shall be calculated and the names of workstations which are used for the parallel execution of the transient analysis of the SMCs in the DSPN solution process.

The software modules of the DSPN solution process and their interfaces are depicted in Fig. 3. The five software modules of the numerical solution component of DSPNexpress are drawn

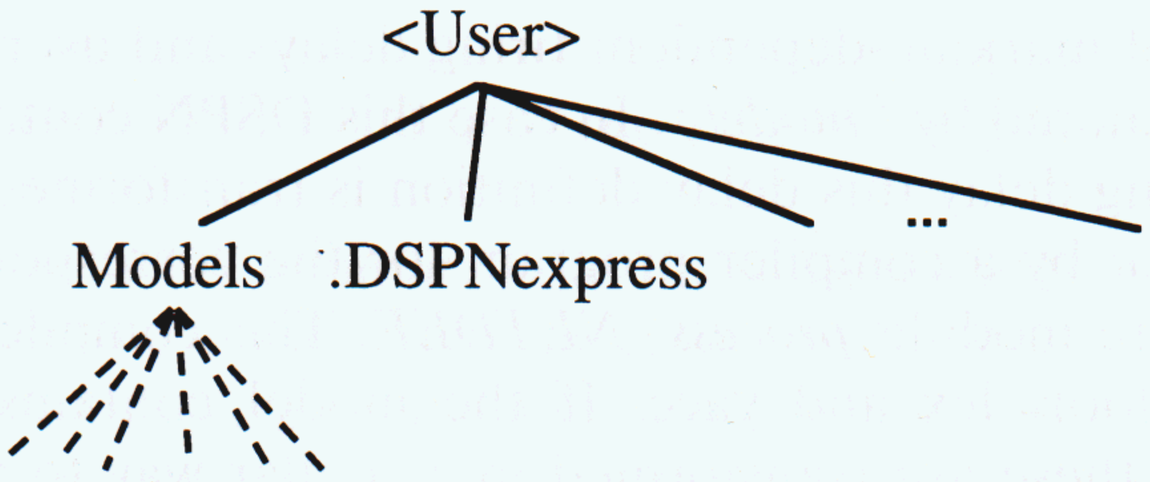

Fig. 2. Local directory structure of DSPNexpress at each user's account.

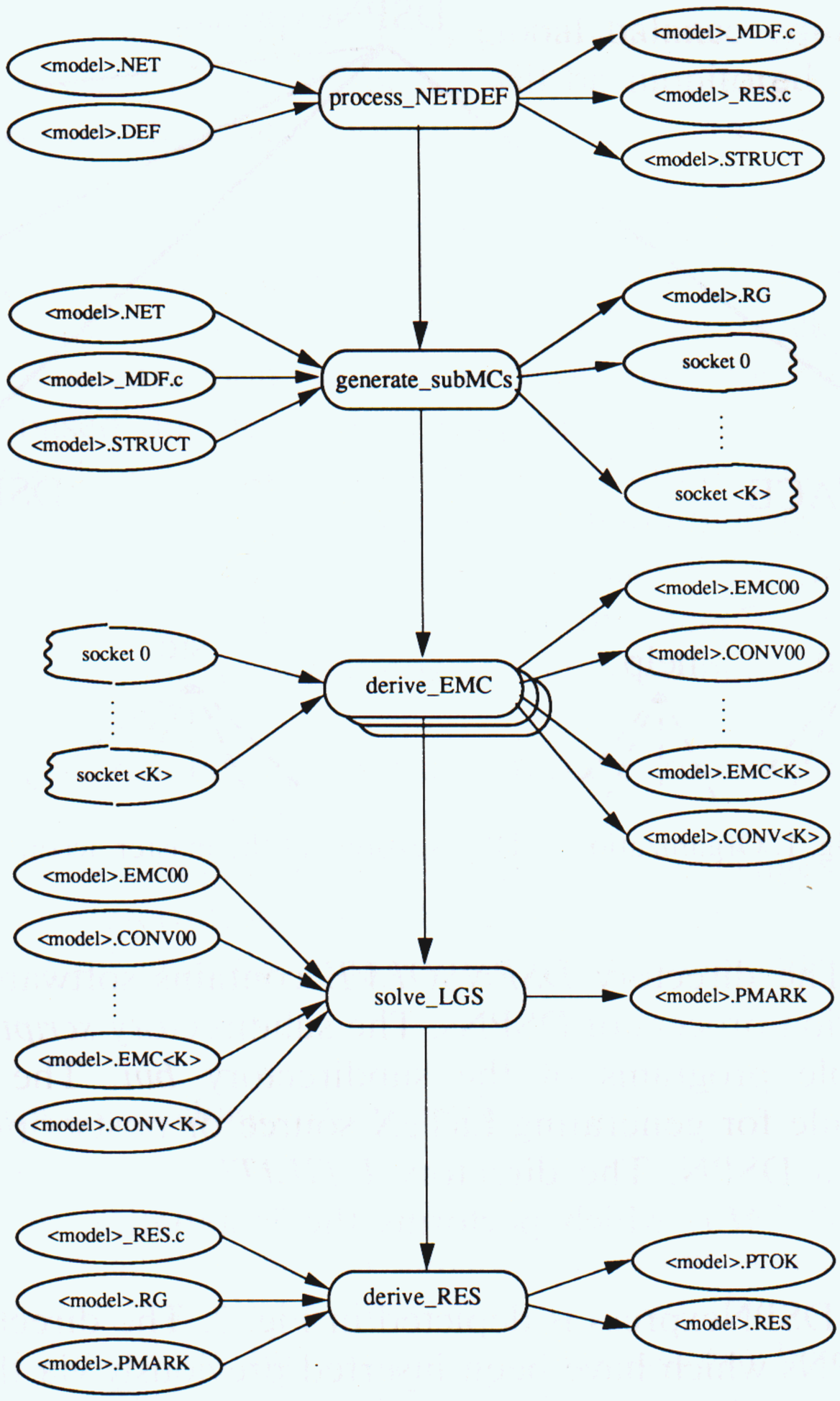

Fig. 3. The software modules of the DSPN solution process of DSPNexpress.

in rounded rectangles connected to their input and output data structures by directed arcs. Data structures stored in files are drawn as ellipses, whereas data structures transferred via UNIX sockets [11] are drawn as semi-ellipses. In each software module memory management functions for data structures are implemented dynamically and exploit their sparsity.

The files ⟨*model*⟩*.NET* and ⟨*model*⟩*.DEF* contain the internal description of a DSPN and also the specifications of marking-dependent firing delays and user-defined steady-state reward measures of a DSPN denoted by ⟨*model*⟩. In case this DSPN contains a timed transition with a marking-dependent firing delay this delay definition is transformed into a C program stored in the file ⟨*model*⟩*_MDF.c* by a compiler program for the corresponding specification language contained in the software module *process _NETDEF*. This compiler program was generated by employing the UNIX tools lex and yacc. If the model contains some user-defined reward measure specifications, these are transformed in a similar way to a C program which is stored in the file ⟨*model*⟩*_RES.c*. Furthermore, the software module *process _NETDEF* contains a

program component which performs some structural analysis of a DSPNs such as the computation of minimum-support-place-invariants and extended conflict sets of immediate transitions. The results of the structural analysis are stored in the file ⟨*model*⟩.*STRUCT*. The generation of the reduced reachability graph of a DSPN and the derivation of the SMCs is performed by the software module *generate_subMCs*. In case the DSPN contains some marking-dependent firing delays the executable program derived from the file ⟨*model*⟩_*MDF.c* is compiled and linked at run-time. Subsequently, the reduced reachability graph of the DSPN is constructed employing the approach introduced in [17]. The reduced reachability graph is stored a sparse data structure in the file ⟨*model*⟩.*RG*.

The software module *derive_EMC* contains software implements the computation of the transition probability matrix of the EMC and the matrix of conversion factors underlying a DSPN. This software module consist of the two program components *proc_Mexp* and *proc_Mdet*. The program component *proc_Mexp* calculates the rows of the transition probability matrix of the EMC and the matrix of conversion factors corresponding to states in which only exponential transitions are enabled. In case a DSPN contains $K \geqslant 1$ deterministic transitions, the software component *proc_Mdet* calculates the transient quantities of a SMC according to the computational formulas introduced in [13]. The software component *proc_Mexp* gets its input through the socket 0. The computed rows of the transition probability matrix of the EMC and the matrix of conversion factors are written in the files ⟨*model*⟩.*EMC00* and ⟨*model*⟩.*CONV00*. The generator matrices of the kth SMC is transferred via the socket k ($1 \leqslant k \leqslant K$) to an instance of the software component *proc_Mdet*. The corresponding probability matrix is also transferred via this socket to the same executable instance of *proc_Mdet*. Since the numerical solution method of DSPNs requires that in no marking two or more deterministic transitions are concurrently enabled, multiple instances of the software component *proc_Mdet* may run quasi-parallel on a single workstation or may run in parallel on different machines of a cluster of workstations.

The software module *solve_LGS* computes the solution of the linear system of global balance equations of the EMC underlying a DSPN. First, the transition probability matrix of the EMC is put together by reading the files ⟨*model*⟩.*EMC00*, ⟨*model*⟩.*EMC01*, ..., ⟨*model*⟩.*EMC*⟨*K*⟩. The matrix of conversion factors is obtained by reading the files ⟨*model*⟩.*CONV00*, ⟨*model*⟩.*CONV01*, ..., ⟨*model*⟩.*CONV*⟨*K*⟩. The linear system is solved by either a direct or an iterative method depending on the size of the transition probability matrix. The direct solver is a sparse implementation of the Gaussian elimination method introduced in [17] which has been revised and reimplemented in the programming language C. The iterative solver is an implementation of the SOR method for stochastic matrices as proposed in [18] with some improvements in adapting the relaxation factor according to [9]. Subsequently, the software module implements the transformation of the steady-state solution vector of the EMC to the steady-state marking probability vector of the DSPN. This marking probability vector is stored in the file ⟨*model*⟩.*PMARK*.

The software module *derive_RES* computes the token probability distribution in each place of the DSPN by summing up the appropriate marking probabilities. Furthermore, the mean number of firings in unit time (also called throughput) is computed for each timed transitions. If the model contains some specifications of non-standard reward measures, the C program stored in the file ⟨*model*⟩_*RES.c* is compiled and linked at run-time. The token probability

distribution in each place of a DSPN is stored in the file ⟨*model*⟩.*PTOK*. The throughput value of each timed transition values of the computed results for user-defined reward measures are stored in the file ⟨*model*⟩.*RES*.

4. The graphical user interface of DSPNexpress

The graphical interface of DSPNexpress runs under the release 5 of the X11 window system and is implemented using athena widgets of the X11 programming library [16]. It allows a user-friendly definition, modification, and quantitative analysis of DSPN models. Places (*place*), immediate transitions (*imT*), exponential transitions (*expT*), deterministic transitions (*detT*), and arcs (*arc*) of a graphical description of a DSPN are processed by selecting the corresponding object and one of the commands *add, move, delete*, or *change* with the mouse. For example, in the setting depicted Fig. 4 deterministic transitions may be inserted and the grid option is used to simplify the graphical editing. Marking parameters (*marking*), firing delays (*delay*), and tags (*tag*) associated with places and transitions are processed in a similar way. For each setting of the command line an online help is provided in the upper part of the graphical interface explaining the actions currently available. The message displayed in Fig. 4 indicates that clicking the left mouse button inserts a deterministic transition. In Fig. 4 a DSPN named *sequential* of the directory VSM is displayed. Each place and each transition of this DSPN is labeled with a tag (e.g. $T1$, $t2, \ldots,$ etc). Each timed transition is also labeled with a parameter specifying the mean value of its firing delay (e.g. write or d_locate_owner). The exponential transitions $T1$, $T4$, and $T16$ are also labeled with *inf.-serv.* to specify their enabling policy as *infinite-server*. This DSPN is used in Section 5 for illustrating the efficiency of the numerical DSPN solution component of the package DSPNexpress. The design of the graphical interface has been influenced by the interface of the version 1.4 of the package GreatSPN [5]. Opposed to the graphical interface of GreatSPN special purpose editors are provided by DSPNexpress for defining steady-state reward measures and marking-dependent firing delays. The editor for specifying user-defined reward measures is shown in Fig. 5. The definitions of eight reward measures (QueueReadMiss, QueueWriteMiss, etc.) of the DSPN *sequential* are displayed in the middle part. The Backus–Naur form of the specification language for reward measures is displayed in the lower part of this window by clicking the help button. To illustrate the capabilities of DSPNexpress the DSPN solution popup is shown in Fig. 6. In case the steady-state solution of a DSPN shall be computed the numerical method for solving the linear system of the global balance equations of its EMC may be chosen by the user by clicking in one of the toggles *iterative* or *direct*. Moreover, a user may specify whether the transient analysis of the SMCs is performed sequentially on a single workstation (*sequential*) or in parallel on a cluster of workstations (*parallel*) and whether verbose output of the solution process is displayed and stored in a logfile. Transient solutions of GSPNs are computed by clicking in the toggle *transient* and specifying an instant of time.

The popup for changing firing delays of timed transitions is shown in Fig. 7. In the current setting the firing delay of the selected transition is changed to the value of the delay parameter write with the default enabling policy single-server (*single-serv.*). Similar popups are provided

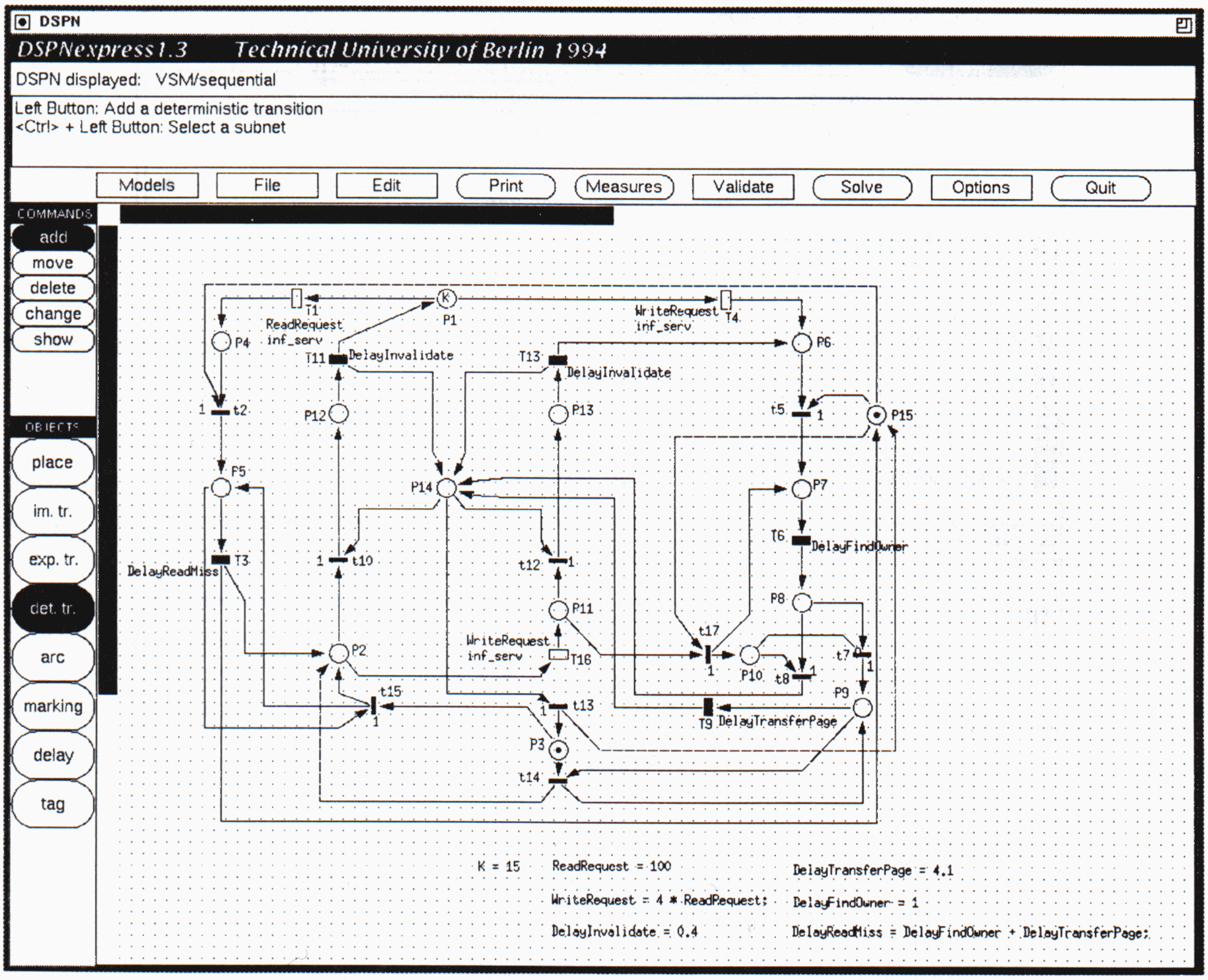

Fig. 4. User interface of DSPNexpress.

for changing the type of a transition and for changing the multiplicity or the direction of an arc. The graphical interface also provides popups for defining or modifying a delay or a marking parameter, changing the string of a tag, etc.

The editor for defining marking-dependent firing delays is shown in Fig. 8. This text editor is invoked by clicking in the button labeled with *mark.-dep.* in the popup for changing firing delays. As in the reward measure editor the Backus–Naur form of the specification language for marking-dependent firing delays is displayed in the lower part of this window by clicking the help button.

The print popup of DSPNexpress is shown in Fig. 9. This popup allows the printing of the DSPN currently being displayed in the main window of DSPNexpress and/or its stochastic description such as delay and marking parameters and definitions and computed results of user-defined reward measures. These description may be either directly printed on a laser writer or stored in files named ⟨*model*⟩*.graph* and ⟨*model*⟩*.text*, respectively, by clicking the appropriate toggles with the left mouse button.

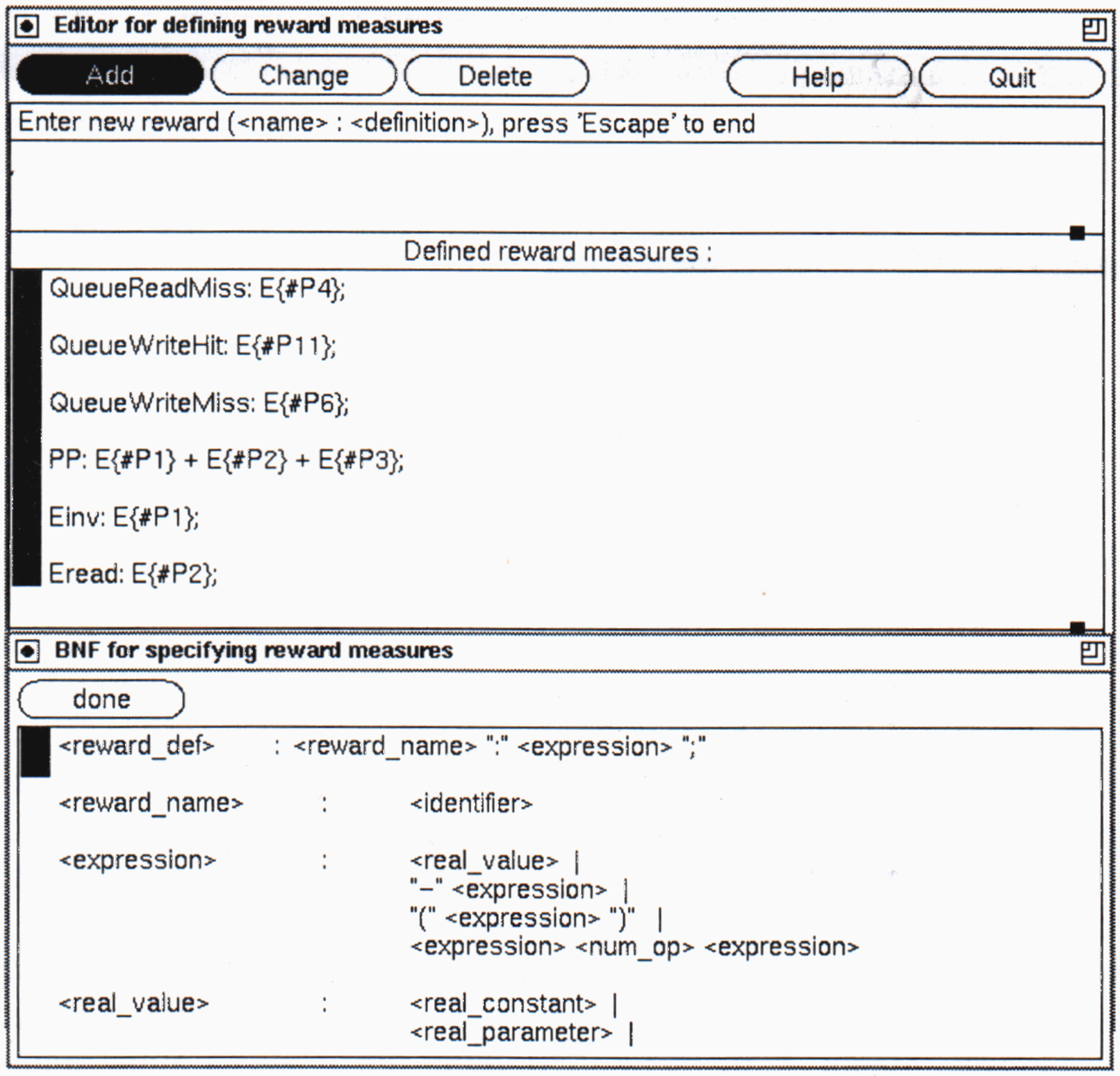

Fig. 5. Reward measure editor.

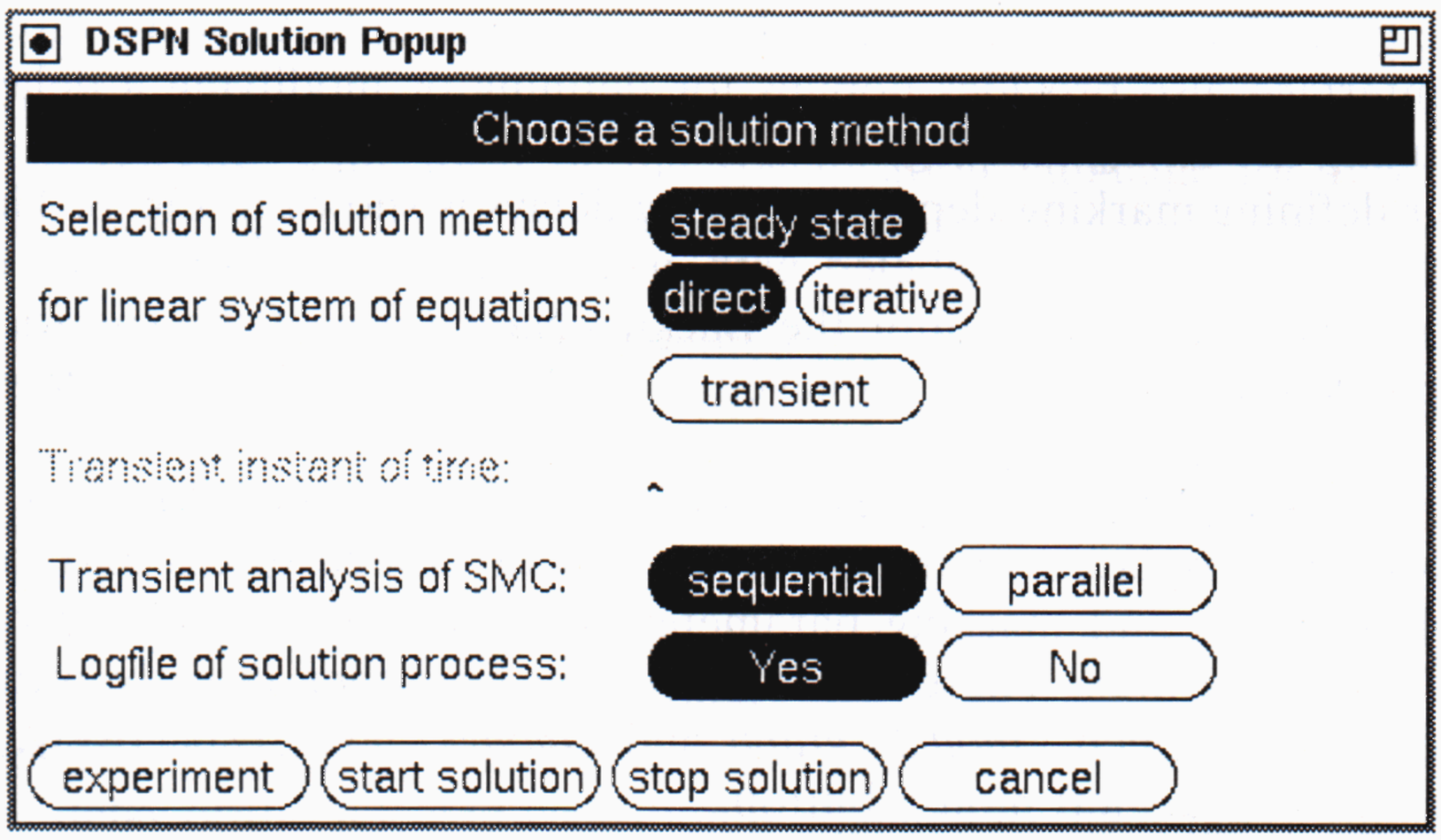

Fig. 6. DSPN solution popup.

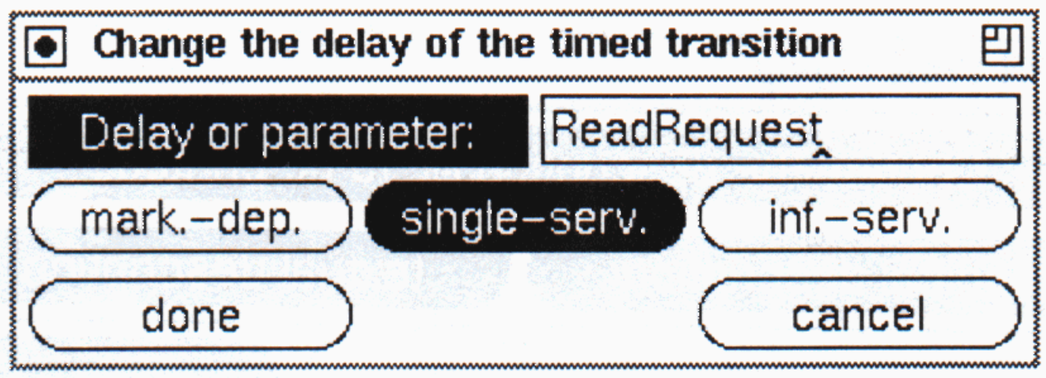

Fig. 7. Popup for changing firing delays of timed transitions.

5. Performance measurements of the DSPN solution algorithm

To illustrate the power of DSPNexpress we consider a DSPN model with several deterministic transitions and whose state space grows rather fast for increasing the number of tokens. The considered DSPN is displayed in the main window of the graphical interface of DSPNexpress shown in Fig. 2. This DSPN represents the behavior of a shared page in a multicomputer system in which virtually shared memory is implemented by the sequential memory consistency model and a write-invalidate protocol. The three main states of a shared page in the global address space, namely INVALID, SHARED and EXCLUSIVE are represented in the DSPN by places $P1$, $P2$, and $P3$, respectively. The marking parameter K of place $P1$ together with the token in place $P3$ represent the number of nodes of the multicomputer system competing

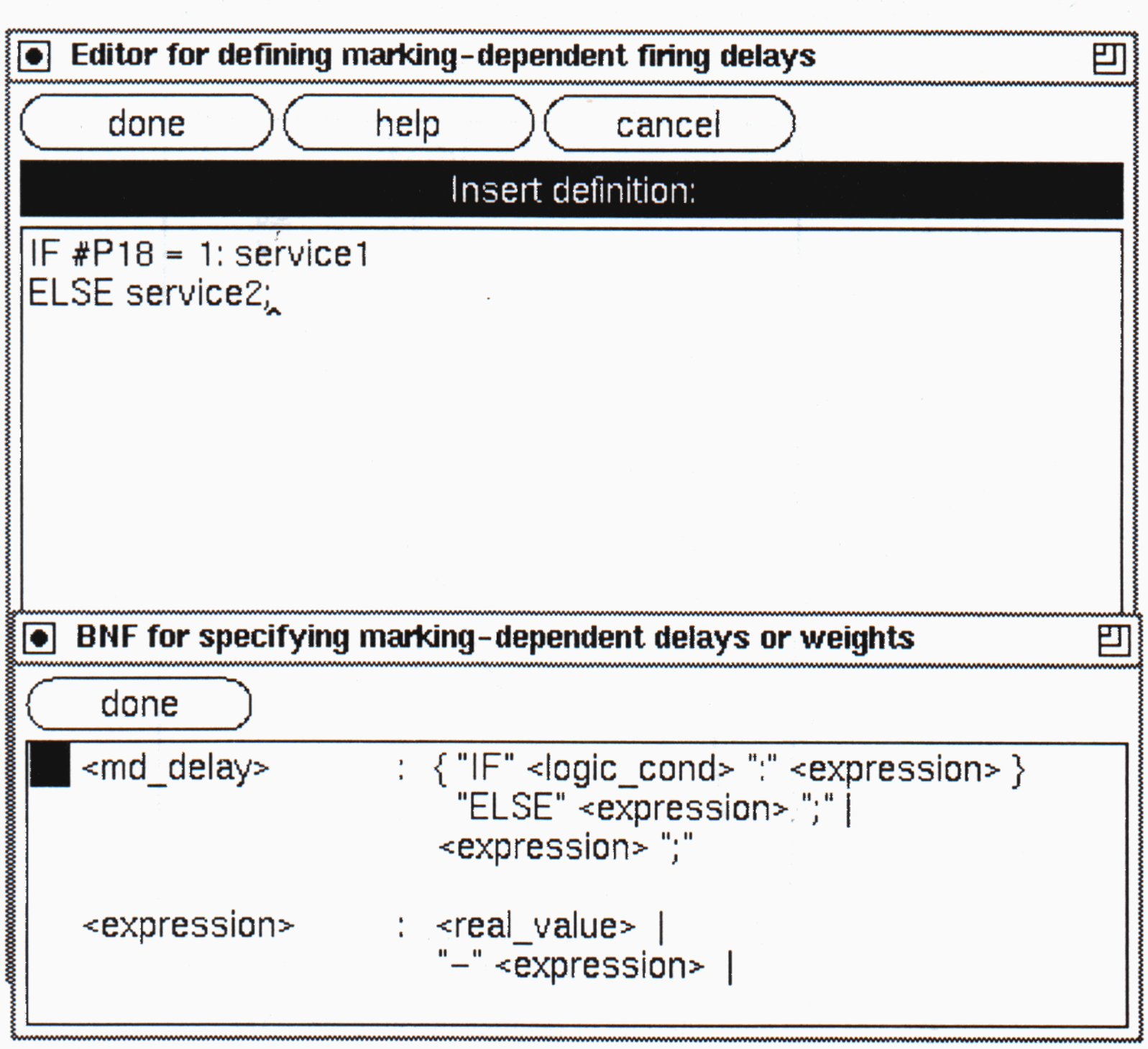

Fig. 8. Editor for defining marking-dependent firing delays.

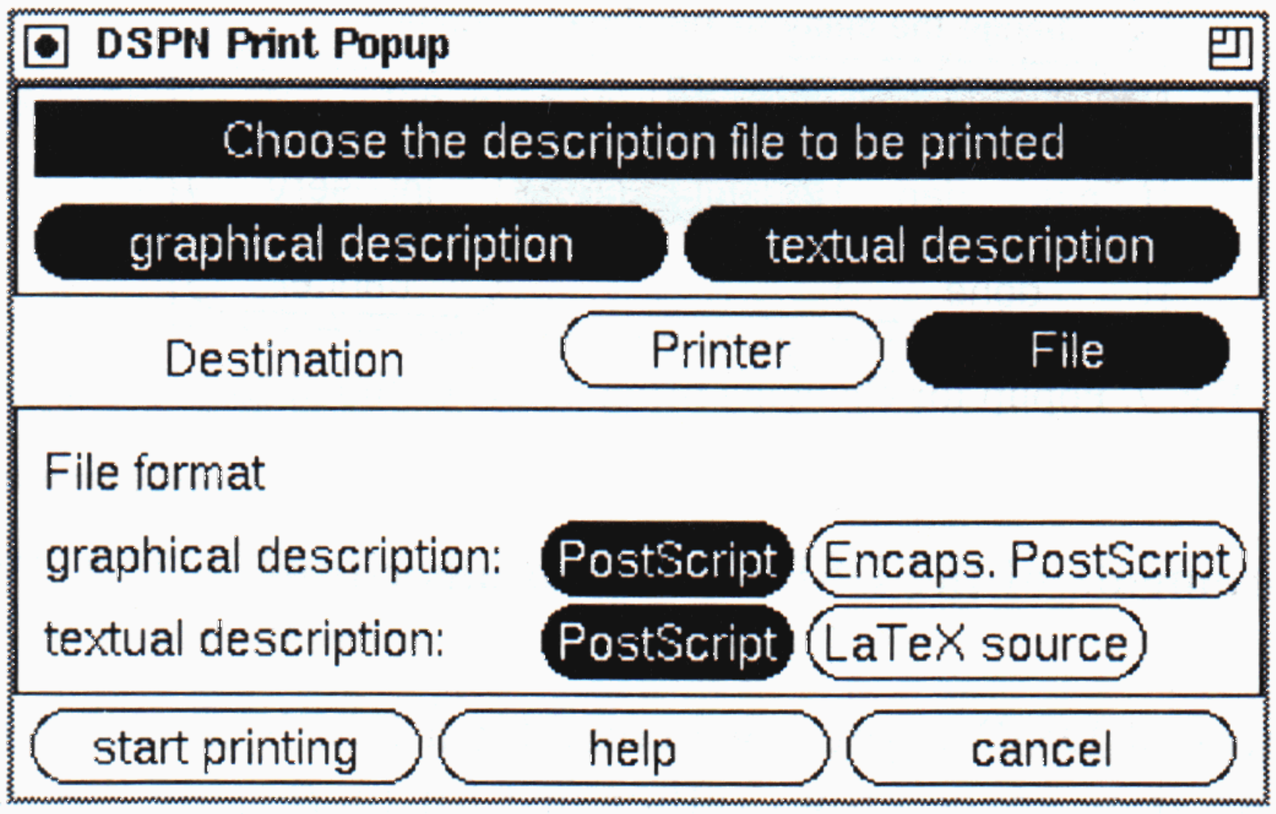

Fig. 9. DSPN print popup.

for the access to the shared page. The token in place $P15$ represents the idle state of the server this DSPN and a quantitative performance analysis of the sequential memory consistency model was presented in [15]. For increasing marking parameter K the calculation of the solution of this DSPN is severely hampered due to state space explosion. For example, in case of $K = 19$ the DSPN has 59 101 tangible markings and its EMC consists of 6 004 585 nonzero state transitions. As shown in [13], DSPNs of this complexity could not be solved in practice with the adaptive matrix exponentiation method implemented in GreatSPN1.4.

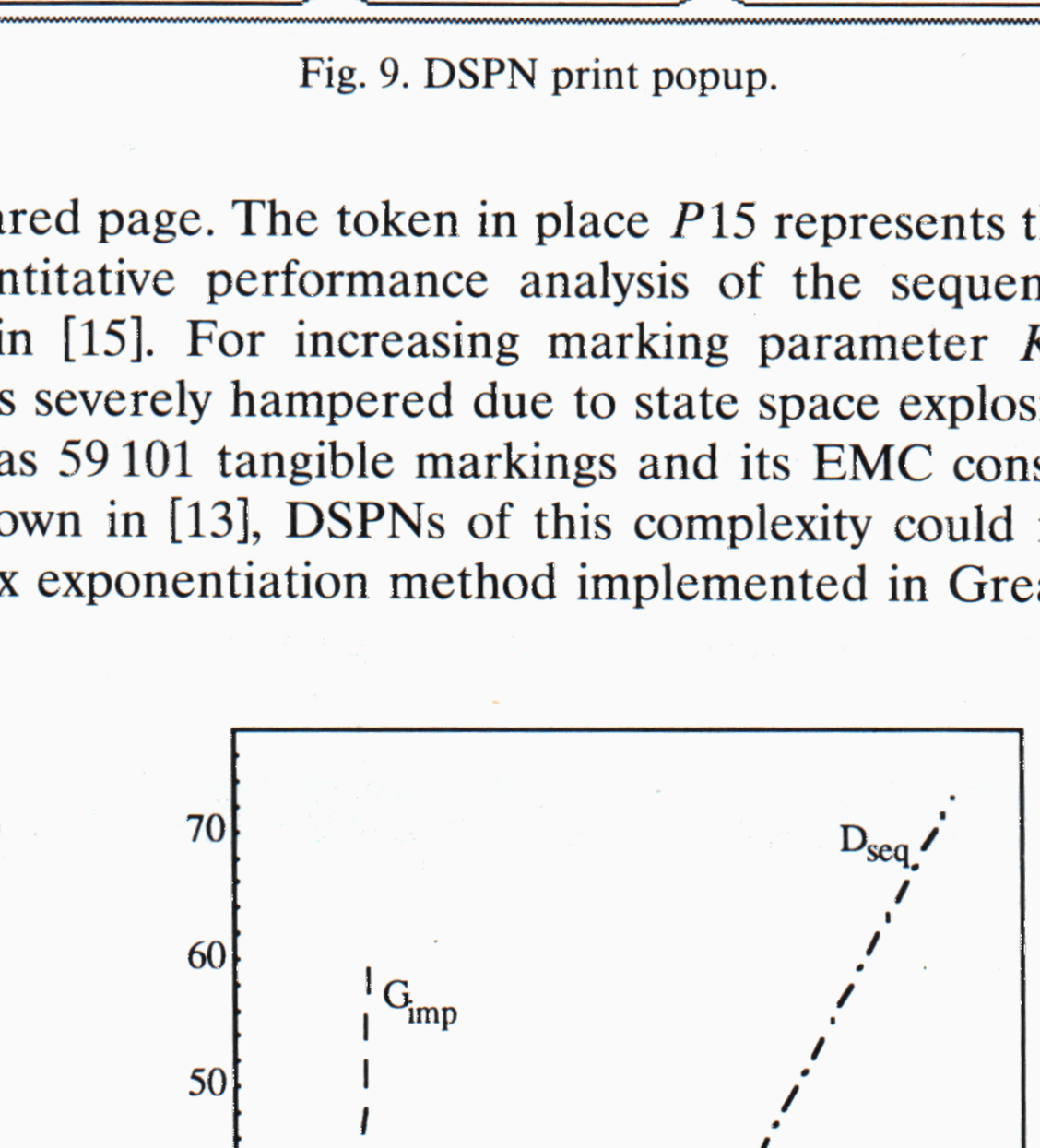

Fig. 10. CPU time versus model size.

The experiments have been performed on Sun™ 4/50 workstations with 64 MByte main memory. For the performance curves of Fig. 10 the CPU time has been measured by the UNIX™ system call *clock*. The overall computation time plotted in Figs. 11 and 12 has been measured with the UNIX system call *time*. An error tolerance of $\epsilon = 10^{-7}$ is considered for the iterative calculations. In all experiments the parameters of the DSPN are chosen as follows. The firing rate of the exponential transition $T1$ which models the mean read request rate is given by 0.01 time units. The firing delay of the deterministic transition $T3$ represent the delay for processing a read miss which is given by 5.1 time units. The firing delays associated with the deterministic transitions $T6$ and $T9$ represent the delay for finding the owner and transferring a copy of a shared page have delays of 1.0, and 4.1 time units respectively. The two deterministic transitions $T11$ and $T13$ represent the delay for invalidating copies of a shared page and have both a firing delay of 0.4 time units. All immediate transitions have an equal weight of 1. The firing priority of the immediate transitions $t10$ and $t12$ is 2, all other immediate transitions have a firing priority of 1.

In a first experiment the mean write request rate is kept fixed to 0.1 time units and the marking parameter K is varied from 4 to 19. In Fig. 10 the CPU time for deriving the EMC by GreatSPN1.4 improved by the numerical algorithm introduced in [13] (labeled with G_{imp}) is compared with the CPU time required by a sequential execution of the software module *derive_EMC* of DSPNexpress (labeled with D_{seq}). Figure 10 shows that the newly designed software architecture of DSPNexpress yields a further substantial reduction of the computational effort required for calculating the steady-state solution of this DSPN. This is due to the separate execution of the transient analysis for the five SMCs of the deterministic transitions.

Users of a performance analysis tool are also interested in the overall computation time a software package requires in order to solve a model. Thus, Figs. 11 and 12 show measurements conducted on a cluster of Sun™ 4/50 workstations running exclusively DSPNexpress as

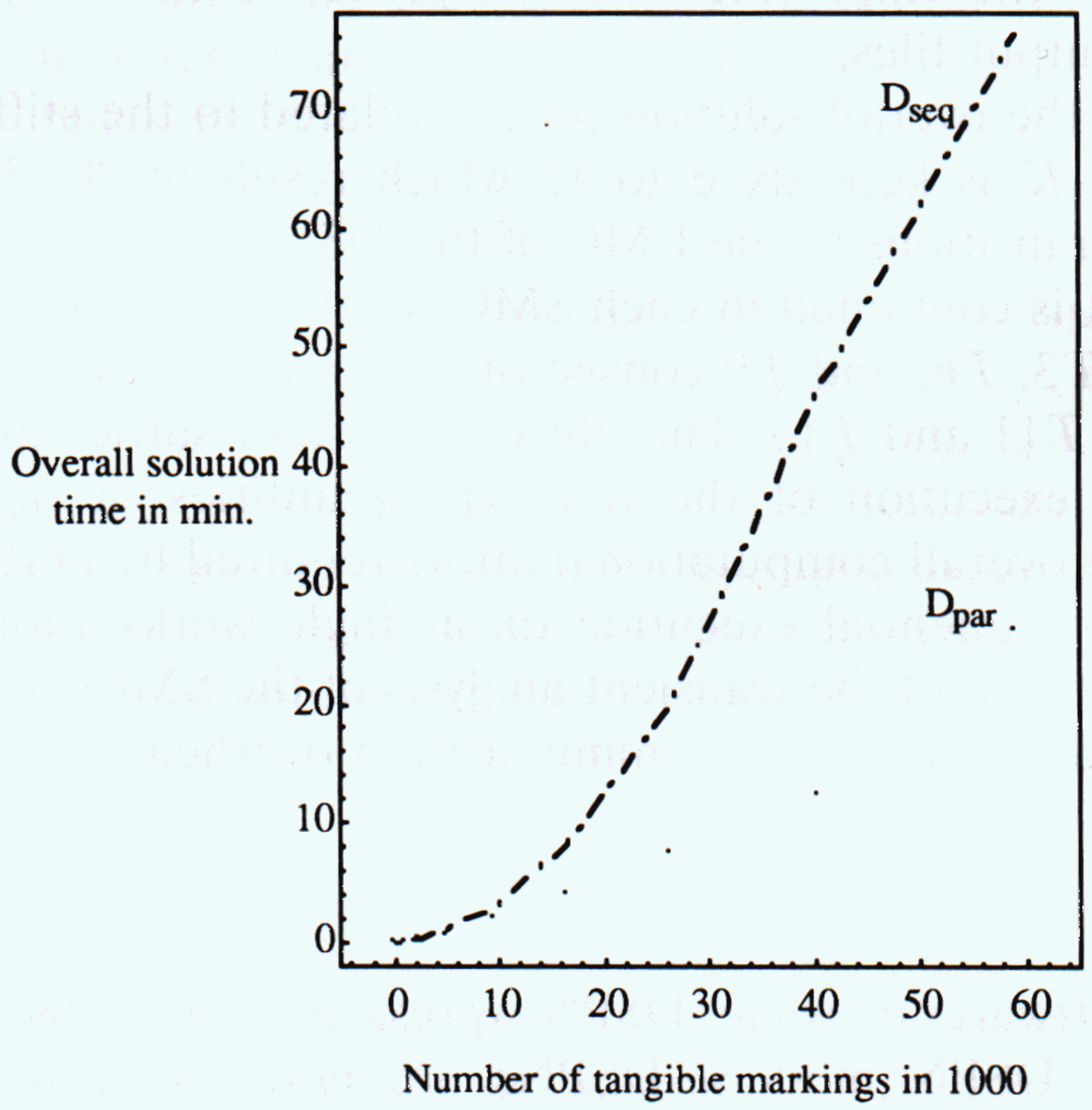

Fig. 11. Overall solution time versus model size.

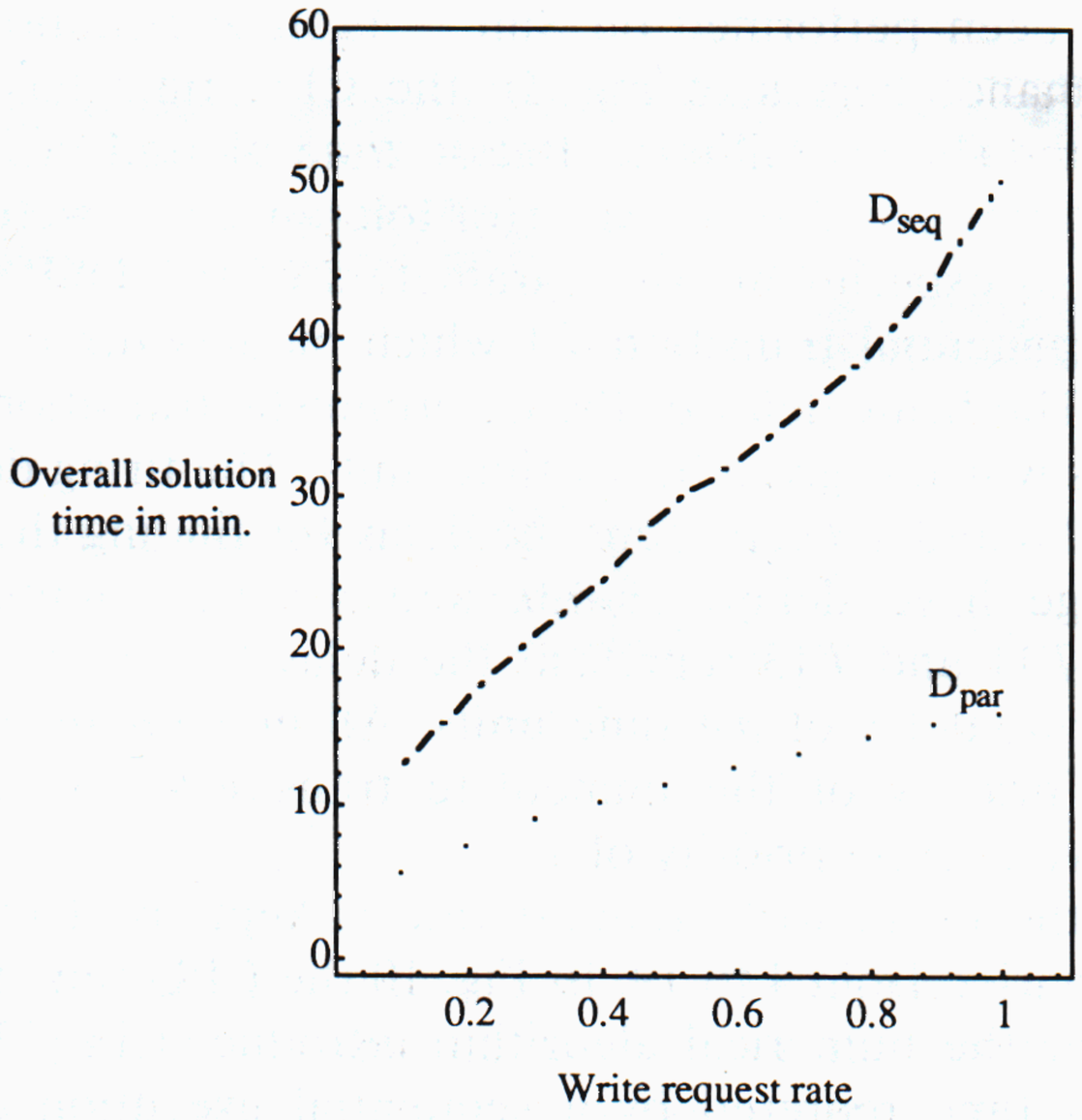

Fig. 12. Overall solution time versus stiffness index.

application. Figure 11 plots the overall computation time required by DSPNexpress for a sequential and a parallel execution of the transient analysis of the five SMCs for increasing model size. The curve corresponding to the computation time of the parallel execution is labeled with D_{par}. Note, the overall computation time required by a sequential execution of the transient analysis of the SMCs is only slightly higher than the corresponding CPU time plotted in Fig. 10. This is due to the employment of interprocess communication with sockets rather than the use of input/output files.

In a third experiment the overall solution time is related to the stiffness index of the SMCs. The marking parameter K is kept fixed to 15 which result in 20976 tangible markings and 1451893 nonzero state transitions in the EMC of the DSPN. The write request rate is taken as stiffness index because it is contained in each SMC. In the considered DSPN, the SMCs of the deterministic transition $T3$, $T6$, and $T9$ consist of substantially more state transitions than the SMCs of the transitions $T11$ and $T13$. Therefore, in case of sufficiently high stiffness index or model size the parallel execution of the transient quantities of the SMCs on four or five workstations reduces the overall computational effort required by DSPNexpress to one third of the effort required by a sequential execution on a single workstation of the same type. The benefit of a parallel execution of the transient analysis of the SMCs would be even greater on a multiprocessor system due to the lower communication overhead.

6. Conclusions

In this paper the software package DSPNexpress has been introduced for an efficient *numerical* evaluation of DSPN models. DSPNs are noteworthy for performance modeling because this modeling formalism allow the representation of both exponentially distributed and

deterministic timing. To the best of the author's knowledge there is currently no other software package available which is able to calculate steady-state solutions of complex DSPNs (e.g. with 60 000 tangible markings and several millions of state transitions) in reasonable computational effort on a modern workstation.

The core of DSPNexpress constitutes the numerical DSPN solution modules in which the numerical algorithm introduced in [13] has been implemented. Moreover, DSPNexpress considers each connected component of a SMC separately for calculating the corresponding the transition probabilities of the EMC underlying the DSPN. The generator matrices of connected components of each SMC are obtained during the generation of the reachability graph by a depth-first-search algorithm. To reduce the system overhead caused by I/O operations with the disk, the interaction between software modules is mostly performed by interprocess communication by means of UNIX sockets [11] rather than by reading from and writing to data files. The separation of the transient analysis of SMCs and the employment of interprocess communication allow also a parallel execution of the transient analysis on different machines in a cluster of workstations.

The separate transient analysis of each SMC is related to the decomposition approach on the net level of Ajmone Marsan et al. [3] who additionally proposed to search for identical independent subnets. The identification of some identical subnets has the advantage that the transient analysis of such subnets has to be conducted just once. However, there may exist independent subnets of a DSPN with different topology which lead to the same transition matrix of the underlying Markov chain (e.g. one place containing K tokens as input to a single exponential transition with a mean firing delay $1/\lambda$ and a series of K places connected by exponential transitions each with a mean delay of $1/\lambda$ and the first place contains one token). Therefore, checking SMCs for isomorphic connected components is more beneficial than searching for identical independent subnets on the net level. Moreover, the separate transient analysis of a SMC directly derives the corresponding transition probabilities of the EMC underlying a DSPN. These features clearly constitutes an advantage over the separate transient analysis of independent subnets proposed in [3] which may require a considerable effort in order to determine the transition probabilities of the EMC from the transient analysis of the independent subnets of the DSPN.

Furthermore, a regular structure of the transition matrix of the EMC underlying a DSPN may be exploited. For example, in case of a $E_r/D/1/K$ queue it is sufficient to calculate one row of the transition matrix of the EMC with $r(K-1)$ entries by numerical transient analysis of its SMC. As shown in [13] the entries of the remaining rK rows of this transition matrix can be derived from the calculated $r(K-1)$ entries by exploiting the regular structure of the SMC. Currently, we are working on refining the DSPN solution algorithm of DSPNexpress in order to exploit isomorphic connected components and special structures of SMCs.

Acknowledgments

The author is grateful to Tobias Bading, Enrik Baumann, Christian Lühe, Martin Müller, and Armin Zimmermann for their effort and dedication by implementing the software components of the package DSPNexpress.

References

[1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte and A. Cumani, The effect of execution policies on the semantics of stochastic Petri nets, *IEEE Trans. Softw. Engrg.* **15** (1989) 832–846.

[2] M. Ajmone Marsan and G. Chiola, On Petri nets with deterministic and exponentially distributed firing times, in: G. Rozenberg (Ed.), *Advances in Petri Nets 1986*, Lecture Notes in Computer Science **266** (Springer, Berlin, 1987) 132–145.

[3] M. Ajmone Marsan, G. Chiola and A. Fumagalli, Improving the efficiency of the analysis of DSPN models, in: G. Rozenberg (Ed.), *Advances in Petri Nets 1989*, Lecture Notes in Computer Science **424** (Springer, Berlin, 1990) 30–50.

[4] G. Balbo, G. Chiola, G. Franceschinis and G. Molinar Roet, On the efficient construction of the tangible reachability graph of generalized stochastic Petri nets, *Proc. 2nd Int. Workshop on Petri Nets and Performance Models*, Madison, Wisc. (1987) 85–92.

[5] G. Chiola, A graphical Petri net tool for performance analysis, *Proc. 3rd Int. Workshop on Modelling Techniques and Performance Evaluation*, Paris, France (1987) 323–333.

[6] G. Chiola, Compiling techniques for the analysis of stochastic Petri nets, *Proc. 4th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Palma de Mallorca, Spain (1989) 11–24.

[7] G. Chiola, GreatSPN 1.5 software architecture, *Proc. 5th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy (1991) 117–132.

[8] G. Ciardo, J. Muppala and K.S. Trivedi, SPNP: stochastic Petri net package, *Proc. 3rd Int. Workshop on Petri Nets and Performance Models*, Kyoto, Japan (1989) 142–151.

[9] G. Ciardo, A. Blakemore, P.F. Chimento, J.K. Muppala and K.S. Trivedi, Automated generation of Markov reward models using stochastic reward nets, in: C. Meyer and R.J. Plemmons (Eds.), *Linear Algebra, Markov Chains, and Queueing Models*, IMA Volumes in Mathematics and its Applications **48** (Springer, Berlin, 1992).

[10] J. Couvillion, R. Freire, R. Johnson, W.D. Obal, M.A. Qureshi, M. Rai, W.H. Sanders and J.E. Twedt, Performability modeling with UltraSAN, *IEEE Software* **8** (1991) 69–80.

[11] S.J. Leffler, M.K. McKusick, M.J. Karels and J.S. Quarterman, *The Design and Implementation of the 4.3BSD UNIX Operating System* (Addison-Wesley, Reading, Mass. 1989).

[12] C. Lindemann, Employing the randomization technique for solving stochastic Petri net models, in: A. Lehmann and F. Lehmann (Eds.), *Proc. 6th GI/ITG Conf. on Modeling, Measurement and Evaluation of Computing Systems*, Munich, Germany (Springer, Berlin, 1991) 306–319.

[13] C. Lindemann, An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic Petri net models, *Performance Eval.* **18** (1993) 79–95.

[14] C. Lindemann and R. German, Modeling discrete event systems with state-dependent deterministic service times, *Disc. Event Dynam. Syst.* **3** (1993) 249–270.

[15] C. Lindemann and F. Schön, Evaluating sequential consistency in a virtually shared memory system with deterministic and stochastic Petri nets, *Proc. Int. Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, San Diego, Calif. (1993) 63–68.

[16] O'Reilly and Associates Inc., *The Definition Guides to the X Window System Vol. 5, X Toolkit Intrinsics Reference Manual* (1990).

[17] A.H. Sherman, NSPIV, a FORTRAN subroutine for sparse Gaussian elimination with partial pivoting, *ACM Trans. Math. Softw.* **4** (1978) 391–398.

[18] W. Stewart and A. Goyal, Matrix methods in large dependability models, Technical Report RC-11485, IBM Watson Res. Center, Yorktown Heights, 1985.

Christoph Lindemann received the degree Diplom-Informatiker (M.S. in Computer Science) from the University of Karlsruhe in 1988 and the degree Doktor-Ingenieur (Ph.D. in Engineering) from the Technical University of Berlin in 1992. Presently, he is a Research Scientist at the GMD Research Institute for Computer Architecture and Software Technology (GMD-FIRST) at the Technical University of Berlin. During the summer 1993 he was a Visiting Scientist at IBM Almaden Research Center in San Jose, California. Christoph Lindemann is recipient of a "Habilitandenstipendium" (a two-year research fellowship) from the Deutsche Forschungsgemeinschaft (German Research Council).
Dr. Lindemann is the program committee co-chair of the 6th International Workshop on Petri Nets and Performance Models. His research interests are in performance and dependability modeling, parallel computer architectures, distributed operating systems, numerical analysis and scientific computing, and applied probability.