Petri Net Tools - A Comparative Study

Dietmar Wikarski (Editor)

Abstract

This report gives a comparative overview of the majority of the available software tools for Petri net editing, analysis, dynamisation (interpretation, simulation and animation), generation and transformation.

After a general introduction into the terminology and classification of Petri nets and an explanation of the tool assessment criteria and classification features, an overview is given in form of a table encompassing 53 Petri net tools, most of them running on UNIX- or PC-based architectures.

As a refinement, a more comprehensive study is given in the form of detailed tool portraits for 16 UNIX-based tools, which are candidates for use in a hypothetical Petri net toolkit.

Email: dwikar@cs.tu-berlin.de

Table of Contents

1 Introduction

Over the past few decades, Petri nets have established themselves as an accepted formalism for modelling, analysis, simulation, control and evaluation of the behaviour of distributed systems. There have been various periods, during which the theory and its application developed at different speeds. An invariant property over all of these periods has been that the theoretically demonstrated application potential of Petri nets always considerably outweighed the extent of the actual applications.

One of the reasons for this was - and still is - the level of development of most of the available tools, which we still consider to be insufficient. Although there are a number of implementations at universities and research organizations, neither they nor the commercially available Petri net tools are used to a degree which corresponds to the true application potential of Petri nets.

For this too, there are a number of reasons of which we will only name one here: the various tools, which are generally based on different net model worlds, are rarely compatible with one another. Desirable synergetic effects have thus seldom been achieved and the entire burden of further development for each individual tool remained with those who began work on it.At the Fraunhofer ISST, an application-oriented research institution, Petri nets are now being used in several projects in the fields of business process modelling, workflow management, and for the modelling of production and control processes. There are also plans to use Petri nets for the design and performance evaluation of software-intensive technical systems. From this perspective, it makes good sense to develop Petri net technology which enables reuse and multiple use of tools and methods. The net tools used at the Fraunhofer ISST to date have all been developed at the Institute itself or under its supervision (CorMan, MoPEd, Process Net Interpreter, Process Net Presenter); software from other sources has only been used to a very limited degree (e.g. INA and Design/CPN). This will have to change in the future. It was decided to create a development framework for software systems - "Petri Net Toolkit" for short - which is intended to systematically combine and integrate those modelling and description techniques which are scientifically proven, complementary, and transferable. In order to ensure comprehensive tool support for the Petri Net Toolkit, public domain tools are to be used for the most part, although a small number of commercial tools will be included. This idea is currently being worked on by the DFG Research Group - Petri Net Technology, a joint research project of the Fraunhofer ISST with the Technical University Berlin and the Humboldt University Berlin, under the leadership of Professors Herbert Weber (coordinator), Hartmut Ehrig and Wolfgang Reisig.

First of all, this report was produced in order to identify suitable Petri net tools fulfilling specific requirements. But the results of this survey may also be interesting for developers of Petri net tools. For this reason - and to encourage the further spread of Petri net technology - these results are being made generally available in the form of this technical report.

This report originates from an ISST Internal Report [HaW95] from December 1995. Apart from the translation into English - many thanks to Nick Grindell for this strenuous work - the details of many of the featured tools have been updated (including detailed descriptions of DesignCPN and PEP which were not featured in the Internal Report) - thanks to the friendly cooperation and contributions of Alexander Borusan, Peter Deussen (PED, PEP, Prod), Olaf Fricke (DesignCPN), Monika Heiner (INA), Angela Mölders (Chromos, DesignCPN) and Jochen Spranger (TimeNet).

1.1 Terminology and Classification

A *net* is a bipartite, directed graph, i.e. a graph with two types of nodes where adjacent nodes are of different type. One of these types is used to symbolize *local states* and is represented by round or oval symbols, while the other type describes *local changes* of these local states and is represented by rectangular symbols (including bars and squares). Although there are many semantically differentiated names for the two types of node, we will be using the widely accepted terms *place* and *transition* to refer to the round and rectangular symbols respectively. The term *Petri net* is generally understood to mean a net with *marked* places whose transitions are subject to a given *firing rule* so that the behaviour of a distributed system can be modelled by the local changes in the local states of a corresponding Petri net. The changeable markings of places are represented graphically by so-called *tokens* (black or coloured dots or other symbols or tuples of symbols). A good basic introduction to Petri nets and some of their applications in systems design can be found in [Rei85].

In recent years, a large number of *net classes* with varying options for expression and abstraction have been developed. What they all have in common is the underlying net structure ("net", see above) and the fact that its dynamic behaviour is determined locally by one or more firing rules. The fact that the net elements are equipped with a wide range of "labellings" which determine its behaviour in a detailed way, often makes it difficult to relate any two net classes to one another. The following rough classification into *net types*[1], which are explained below, provides a framework for the classification of net tools in subsequent chapters. (Additionally, a net type "general" will be used in the following to denote the ability of a tool to deal with all of these types of Petri nets).

⇨ low-level nets	(**LL**)	"pure": without time
⇨ hierarchical low-level nets	(**HLL**)	
⇨ high-level nets	(**HL**)	
⇨ hierarchical high-level nets	(**HHL**)	
⇨ timed nets	(**Tim**)	generally with time concept
⇨ stochastic nets	(**Sto**)	
⇨ function nets	(**Fun**)	
⇨ queueing Petri nets	(**QPN**)	
⇨ object Petri nets	(**OPN**)	with and without time concept

Note that the system of classification shown here cannot claim to be complete or final. However, it is compatible with most known classifications and allows systematic ordering of the relations between net types. In the following we will first characterize the net types listed above and then state some of the relations between a few of them.

1. Note that we assume that every particular net class may be assigned to just one of these net types. This idea works surprisingly well but may also seem artificial in certain cases.

Low-level nets(LL) are characterized by the fact that only identical (i.e. "black") tokens appear on the places. All other net elements (places, transitions and arcs) can be given more detailed characteristics by labelling, e.g. with capacities (for places), priorities (for transitions) and weights (for arcs).

In *high-level nets*(HL) tokens may be individually identifiable. The labels on the input arcs[1] set the necessary conditions for the flow of token types, while those on the output arcs[2] denote the types of the tokens generated. In the case of nets containing cyclical structures, high-level nets help achieve more compact representation. The best-known examples of untimed high-level nets (in the sense of net classes) are predicate/transition nets, coloured nets and algebraic high-level nets.

Nets which provide means of expressing the refinement of places or transitions and the composition of partial nets into coarsened nodes (places, transitions or special symbols) are referred to as *hierarchical* Where the underlying structure is a low-level net, i.e. all tokens are identical, then the corresponding net type is called *hierarchical low-level nets (HLL)*. Modular process nets [Wik95] and (low-level) object nets [Wik90] are examples of this type of net class. Hierarchical nets with individually identifiable tokens are called *hierarchical high-level nets (HHL)* and include a large number of different classes [Hub95].

For the net types referred so far, which have for good reason no time concept, the following set inclusion statements apply:

$$\begin{array}{ccc} \mathrm{LL} & \subset & \mathrm{HLL} \\ \cap & & \cap \\ \mathrm{HL} & \subset & \mathrm{HHL} \end{array}$$

All of these net types can be extended by some concept of "time". This always involves an essential change of the firing rules. In the aforementioned cases of nets without a time concept, it is assumed that a transition with concession (i.e. there are enough tokens on the preplaces and few enough on the postplaces) *may* fire, but that it doesn't *have* to (and certainly not immediately). If, on the other hand, a (global) time axis is introduced, then it is always assumed that as soon as a given transition gets concession, some kind of change will occur or begin. Either all the tokens responsible for the concession are removed from their places immediately (whereupon corresponding delay times start to run) or the token in question is allocated a "clock" which determines its availability.

The explicit assumption of a time concept allows quantitative statements to be made on the basis of net models, e.g. for predicting performance or reliability. If all the delay parameters are constants or intervals, then the term used is *timed nets*. However, if they are distributed according to stochastic distribution functions or if probabilities for the firing of conflicting transitions are stated explicitly, then one speaks of *stochastic nets*. The two best-known examples of stochastic nets are GSPN[3] (exponentially distributed delays and immediate firing of transitions) [Mar84] and DSPN[4] (additional deterministic delays) [Mar87].
Since then, with the development of more and more sophisticated analytical and numerical solution procedures for stationary and transient probability distributions, these classes have been undergoing constant extension (see for example [Ger94]).

1. Arcs from places to transitions
2. Arcs from transitions to places
3. Generalized Stochastic Petri Nets
4. Deterministic and Stochastic Petri Nets

Function nets are special high-level nets in which data types are assigned to the places and jobs are assigned to the transitions. Jobs are software code segments which are executed when transitions fire. One example of a development on from function nets (including time and additional node types) is FUNSOFT nets [Gru91].

Queueing nets were developed independently of the Petri nets used for evaluating the performance of distributed systems [Laz84]. The nodes of the graphs on which these "nets" are based are operating stations. These consist of a queue and one or more operators. However, if one considers an operating station to be a partial net composed of places and transitions, then it makes sense to include queueing nets among the timed Petri nets. The parameters are the distribution functions for inter-arrival times and for sojourn (sometimes called "dwelling") times at the service stations. New developments are using this approach to embed queueing nets (QN) in Petri nets (PN),resulting in so-called QPN's (queueing Petri nets) [Bau93].

The eight types of nets introduced up to now can be uniquely divided into nets with and nets without a time concept.[1] Another type of nets, the so-called *object (Petri) nets* need to be treated differently, since they are characterized by the existence of an object concept in addition to the usual net elements. There have been two lines of development: On the one hand, objects are seen as a generalization of tokens. The corresponding object nets are usually viewed as generalized high-level nets, e.g. [FLSW93]. On the other hand, certain partial nets can be viewed as (net) objects, see e.g. [Wik90]. Generalizations which combine both lines of development include OPNs [Lak95] and HOONs [LWH95].

1.2 Application of Petri Nets

Today, Petri nets are used in diverse areas of computer science and their fields of application. Typical examples include:

- business process modelling
- workflow management
- verification, validation and analysis of programs
- software design
- program generation
- software re-engineering
- implementation of knowledge bases

How can the practical success of this deeply theory-based class of models be explained? We consider the following properties of net-based model classes to be the reasons for the success of Petri nets to date and the justification for believing in the continuing spread of their influence:

1. However, it is noticeable that in many of the tools examined below, time delays are often added to the net classes defined without delays in "pure form" (from LL to HHL) without mentioning that this has been done and without changing the name of the net class.

- Petri nets have a very simple syntax due to the small number of descriptive elements, making them easy to learn.
- They can be used as a self-similar ("fractal") modelling tool for the representation of a single system at various levels of abstraction (i.e. the same means of description and analysis can be used at every level of abstraction).
- Concurrency (parallelism) and causality of events can be adequately modelled in a single model.
- Petri nets are both visualisable and equipped with formal semantics.
- Concurrent behaviour can be dynamically represented by the flow of tokens.
- Consistent locality principles apply with regard to the representation both of system states and of their changes.

Due to their particular strengths in the modelling of concurrent system behaviour, Petri nets are predestined for use in the analysis of behaviour in distributed systems. Using Petri nets, the behaviour of a distributed system can be:

modelled - to understand or specify it[1]
assessed - to analyse or predict it
implemented - by generating software or via systems control.

In order to increase the efficiency and quality of the above-mentioned applications, it makes sense to aim to use as many of the methods and tools already developed for a particular class of applications in other areas too, so as to achieve multiple use, particularly in the case of tools.

However, experience has shown that the quality obtained with the different tools can vary strongly even between different phases of a single application (e.g. modelling, analysis, simulation, assessment, control). The combination of different tools for individual phases is therefore a useful strategy.

As we have already seen, even on the level of net formats, most tools are not compatible with each other and so far there is no "standard format" to support the exchange of net specifications between tools. In Autumn 1994, as a response to this situation, at a Petri net workshop organised by the Humboldt Universität, the task was set to develop a standard protocol for the coupling of net tools which would function independently of the tools' underlying net classes. At least one proposal was submitted which fulfilled the named requirements to a large degree [Bau94]. In the meantime, an ISO standardisation proposal is under way [ISO96].[

1. The corresponding model for the first case is usually called descriptive and in the second case prescriptive.

2 Assessment Criteria

What a "good" or "high quality" system is, usually depends on the needs and on the view of its (potential) user(s). In this sense, different tools are not generally comparable. Rather, it has to be defined for what goal and for which kind of user they are intended.

Moreover, the situation with respect to quality is very special in our case. Usually, quality assurance - which aims at user satisfaction - is done in the process of constructing (designing, implementing, testing etc.) of a tool. Many of the tools considered in the following have been built in a different situation, namely for the needs of their creators and their colleagues or just for the demonstration of skills and technology. Nonetheless we will try to give a framework for a holistic assessment and evaluation of these tools which is based on a general framework for systems evaluation. As we learned from literature later, this framework has some similarity with the paradigm of *acceptability* by Shackel as described in [ShR91], pp. 21 ff. According to this paradigm (which is first of all described as a basis for the *usability framework*), acceptability of an "informatics system" is a result of balance between *utility* (will it do what is needed successfully?), *usability* (will the users actually work it successfully?) and *likability* (will the users feel it is suitable?) against *cost* (what are the capital and running costs and what are the social and organisational consequences?).

This section begins with a presentation of some general criteria and their structuring for the assessment of open systems in general and of software tools in particular. Based on this assessment framework, the specific criteria for the assessment of Petri net tools are listed and classified. The result is a reasonably "canonical" classification of the various criteria which later forms the basis for the assessment table as given in Section 3.3.

2.1 General Criteria

Let us start with a rough natural scheme for assessing "how good" a service or product is, by separating the individual assessment criteria into the categories "cost"[1] and "benefit"[2].

This rough classification can be refined by introducing another division - independent of cost/ benefit considerations - into system-related ("technical") aspects on the one hand (defined by the set of services required and services provided) and user-related ("social") aspects on the other (to be obtained by evaluating the necessary costs and potential benefit).

---

1. *The term "cost" is used here in the sense of "expense" which is required for the use of the service or product in question., i.e. cost need not necessarily be expressible in terms of money, but as every effort and all prerequisites which are necessary for "running" the system under consideration.*

2. *A similar, often used term is "quality", i.e. the degree of user satisfaction. Concerning the quality of products requiring a sophisticated production process - in particular software - a distinction is currently made between two basic aspects: firstly the production of quality products - where the production process forms the object of the assessment ([Dun93]) - and secondly the quality assessment of products or services itself. Although these two aspects are closely related, we will be clearly concentrating in the following on the second only, since we receive the tools as "finished products" and have no means of examining or influencing the processes involved in their production.*

In the following, we will denote the set of required services as *infrastructure*, whereas *functionality* will denote the set of provided services.

Note that service definitions (as usually understood) declare only *what* is done on request by the system providing the service, but not *how* (fast, user-satisfying etc.)[1]. In order to be more general and to include also the "cost" side of the coin, we use here the term "socially assessable" aspects to include both the costs ("*usage costs*") as well as the benefits ("*usability*") which a user of the considered product experiences when using its services.
As already mentioned above, *usage costs* should take into account the entire effort required in order to be able to use a tool. Alongside the purely financial costs in the case of commercial tools this can also include the effort for testing, for adaptation to personal requirements and for locating a suitable supplier.
According to ISO 9241-11 [ISO95] *usability* is the degree to which a product can be used effectively, efficiently and satisfactorily by specified users to achieve a specified goal in a specified context[2]. E.g. response times, user friendliness and other, often "subjective", criteria which fall under this quite clearly defined criterion group.

A graphical representation of these relations is given in Fig. 1. Note that the purely technical/ functional aspects of the system in question - to be placed in the inner circle of Fig. 1 - are always *countable* (as services), whereas the socially assessable aspects - to be placed in the outer circle of Fig.1 - are *evaluable*, for which the *social environment* (which is defined using indices such as time, money, effort, satisfaction, etc.) must be taken into account.

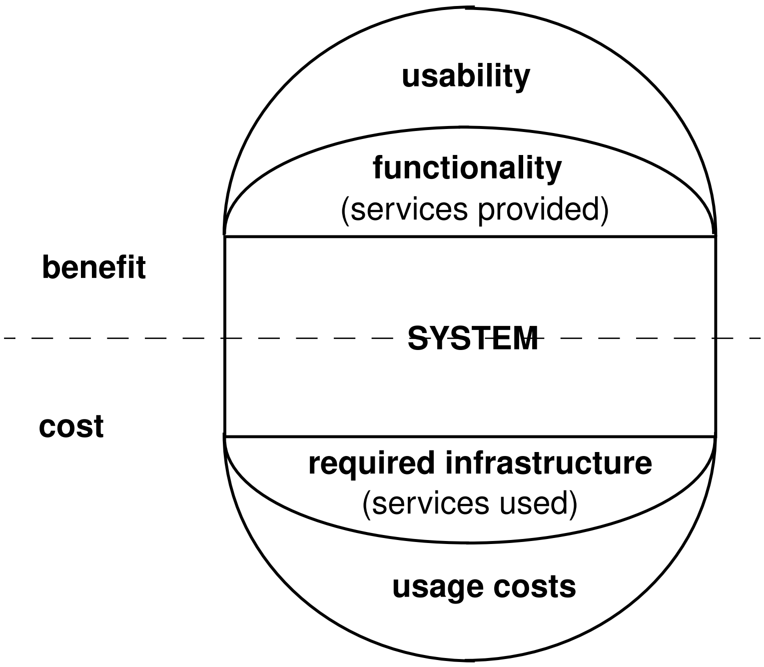

Fig. 1: Schematic classification of assessment criteria

1. *This "how" is recently more precise defined by the attributes of "quality of service".*

2. *Effectiveness is understood to mean the accuracy and completeness with which the user achieves the specified goals, whereas efficiency is determined by the amount of resources required.*

The general assessment framework of Fig. 1 will be applied in the following for a hierarchical classification of the features of the Petri net tools under consideration in the table in Section 3.3.

As *functionality* (or provided services) of a Petri net tool one considers usually the presence of components such as graphical editors, analysers, simulators and generators. But considering these tools as open systems which should cooperate with other tools - be it other Petri net tools, data bases or application systems like workflow management systems - the availability of appropriate interfaces becomes also crucial. The supported net classes have to considered as a kind of functionality, too, but this feature is treated separately on account of its outstanding characterisation function.

The other part of the technical/functional aspects - the *required infrastructure* - includes all necessary services required from a tool's technical environment. These requirements are usually not typical only for Petri net tools. They include, in particular, the operating system(s) and any libraries that may be needed. Note that for evaluation purposes, we have to distinguish carefully between alternatively required services (e.g., a tool can be run on a UNIX or on a MS Windows environment) and simultaneously required services (e.g. the UNIX operating system with the X11R5 window system): The first case (several services connected by "and") corresponds to more requirements to the environment (higher costs,...) whereas the second case (several services connected by "or") corresponds to less requirements to the environment and thus lower costs. Note that the second case corresponds to the adaptability of the system (to different infrastructures) whereas the first represents just a refinement of the service(s) specification.

In the case of the socially assessable aspects, we refer to the benefit side in the following as *usability*. Note that this survey examines only a selection of the aspects generally considered to constitute usability (see e.g. [ISO95]).

In general, the number of assessment criteria to be considered for each of the above-mentioned aspects is large. In order to keep our tabular overview to a manageable size, however, it is necessary to concentrate on a smaller selection. We have therefore omitted criteria whose assessment is problematic or unreliable (e.g. training costs) or which are of little importance to the practical use of the tools in our case (e.g. performance in terms of response times).

An interesting (and unintentional) by-product of the graphical representation given in Fig. 1 is that it can be seen that there is an analogy to the structuring of those domains most important to *process quality* (according to [Dun93]), i.e. *technology* on one hand - which corresponds to our "technical-functional criteria" - and *management/staff* on the other - corresponding to our "socially assessable criteria".

2.2 Classification of Features

Let us apply now the proposed framework to Petri net tools: We have separated one important aspect of functionality - the class of the nets used (**Net type**/**Net class**) - from the main group of functionality features because it forms the context for all the other features and also because it plays an important role in the potential linking of different tools.

The two groups containing the technical/functional and socially assessable criteria are listed below as they are listed in the table in Section 3.3 (the corresponding keywords are also highlighted):

Technical-Functional Criteria ("system-related")

- **Functionality:**
  - **Editor** is based on: text processing (**T**), graphics processing (**G**)
  - **Analysis** options (see Section 4.2.1):
    - general static properties: syntax check, net topology, connectedness, statistics on net elements, covering graph,... (**Sta**)
    - general dynamic properties: liveness, boundedness (**Dyn**)
    - special features: invariants (**Inv**), conflicts (**Cfl**), reduction (**Red**), reachability graph (**RG**), temporal-logical statements (**TL**)
  - **Dynamisation** features: animation (**Ani**), simulation (**Sim**), interpretation (**Int**) (see Section 4.2.2)
  - **Generation** options: **Code**: code generation, **Net**: net generation (see Section 4.2.3)
  - **DB-access** (**+/-**)
  - **Interfaces**: **Import, Export** (via filter (**F**) or call (**A**) with parameter transfer)

- (required) **Infrastructure:**
  - Operating system. In cases where more than one option is given, the tool is only mentioned once, in the operating system group with the highest priority in the order UNIX, PC, Others (cf. Chapter 3).
  - Programming system components for the creation of executable software and dynamic libraries for their execution
  - Necessity of other libraries
  - Necessity of a Window Management System
  - Implementation-related restrictions

Socially-Assessable Criteria ("user-related")

- **Usability**:
  - **Stability** (on a scale from **1** down to **5**)
  - **Operability** (on a scale from **1** down to **5**)
  - **User manual**(**+/-**)
  - **Help function** (**+/-**)
  - Functional description (+/-) (not shown in the table)
  - Publications about this tool (+/-) (not shown in the table)

- **Usage Costs**:
  - commercial (**Com**)
  - on request (**Req**)
  - available for testing (**Tst**)
  - available for university/research (**R**/**D**)
  - public domain (**Pub**)
  - generally available (**Gen**)

The assessment of *stability* and *operability* tends to be of a highly subjective nature. By adopting objective assessment criteria we have tried to reduce this subjectivity. In the case of *stability*, for example, these criteria included fault tolerance[1], crash rate and portability. For *operability* they included adherence to styleguides (menu design, button labels cf. [OSF91]), comprehensibility of error messages (and/or warnings) and learnability.

1. *E.g. whether user errors result in a warning instead of a program crash*

3 Overview

In this chapter, over 50 tools for the editing, analysis, dynamisation and generation of Petri nets are presented together in the form of a table, based on the criteria listed in Section 2. Our information about these tools comes from various different sources. The degree of detail and in some cases the credibility of this information therefore differs from tool to tool. To give the reader an idea of the status of the information provided, this chapter opens with a presentation of its sources, some of which are explained in detail. In addition to this, all abbreviations used in the table and not already mentioned above are explained.

The tools are listed in alphabetical order within each category

UNIX means "UNIX workstation architecture": This category also includes workstations with such UNIX-like operating systems as VMS. The operating system is only named (under **Infrastructure)** if the program system does *not* run on Sun OS or Solaris.
If a given tool is available both for UNIX and for other platforms (e.g. PC, MAC), then it is mentioned *here only*, but with a reference to these other platforms.

PC means "PC architecture": This category is based on the hardware architecture of the underlying system. Thus, even tools which require the Linux operating system fall within this category. The operating system is only named (under **Infrastructure)** if it is *not* MS-DOS (e.g. Linux). Otherwise, only platforms (e.g. Windows) or programming tools (e.g. Borland Pascal) which run on MS-DOS are mentioned here. If a given tool is available both for PC and for other platforms (but not for UNIX, see above), then it is mentioned *here only*, but with a reference to these other platforms.

Other Platforms: This category contains all tools not covered by the two categories above, e.g. Apple Macintosh

The reason for the priorisation of UNIX tools is due to the initial intention to provide the basis for a UNIX-based toolkit. In the meantime, this goal is not of such a high priority, but the classification can continue to be used in this sense.

3.1 Information Sources

The overview given in the table in Section 3.3 is based both on the results of our own testing and on information supplied by the tools' manufacturers or third parties.

Where a tool has been tested by us, **PT** (personal test) appears in the penultimate column (**source**). The majority of these tools are presented separately and in greater detail as "portraits" in Section 4. Those tools included in this more detailed comparison are marked with **+** in the second column of the table in Section 3.3.

If information has been taken from manuals, publications, prospectuses or tool surveys, this is also indicated in the penultimate column. The following abbreviations are used:

- software package user's manual (**Man**)
- literature (**Lit**)
- tool list from the University of Aarhus, Denmark (**Aar**)
- Petri Net Newsletter 41 (**PNN**)

Although the information content of manuals and publications can vary a great deal from tool to tool, the comprehensiveness of the above source types generally decreases as one goes down the list. Further reference is made to the PN tool list from CRIM in Canada and the tool survey from the Petri Net Newsletter 41 (April 1992) in the following. As a recent development, the tool list is now being maintained as "Petri Nets Tools Database" by the CS Department of University of Aarhus, Denmark together with the Petri nets homepage [ChS96]. Further surveys of Petri net tools can be found in the Lecture Notes [Fel85] and [Les89] published by Springer Verlag, but are not included here as they are no longer up-to-date.

University of Aarhus, Denmark

In addition to the Petri Net Mailing List, the Coloured Petri Nets group at the Computer Science Department, University of Aarhus, Denmark is maintaining also Petri Net WWW pages. Among a diversity of information like announcements, questions, bibliographies, addresses, and all other kinds of Petri net related information, information about Petri net tools is provided under the WWW address

http://www.daimi.aau.dk/PetriNets/tools/

The collection of Petri net tools has been running for several years under the maintenance of CRIM, Canada and has recently been shifted. It is updated by the individual tool suppliers via email or directly by using the WWW registry on the Web page (see below). The list is accessible worldwide via WWW and FTP and, as far as we know, it is the most extensive and up-to-date survey of Petri net tools available.

For completeness reasons, we propose here both the old (maintained by CRIM) as well as the new (maintained by the Aarhus University Computer Science department) questionnaires.

CRIM-Questionnaire

Questionnaire completion date, Tool name, Version, Tool completion date

Address

Contact person(s), Organisation name, Organisation address, Phone, Fax, Email, Other distributors

Environment

Computer systems (e.g. DOS, Unix, OS2, etc.), Graphical systems (e.g. Motif, OpenLook, Windows, etc.), Other software required (e.g. Run-times)

Availability

Tool fee, Maintenance fee, Maintenance services, Price for university, FTP site where to get the tool

Functionality

Kind of Petri net supported (e.g. CPN, SPN, etc.), Availability of a graphical editor (Yes, No, comments), Type of animation: (e.g. forward, backward), Type of performance analysis (ex. steady state, continuous), How are the results presented (e.g. Graphical report):, Additional Functionality

Documentation and information

References, FTP sites where to get references, Number of copy releases, Tool language, Interface language, Documentation language:

Future plans

Other remarks

Despite certain shortcomings in its maintenance (up-to-dateness of the versions described, muddling of tool names) and the unfortunate formulation of some items (e.g. under "Functionality"), this list does give a good first overview of the range of Petri net tools on offer. However, the list doesn't contain all the tools known to us (36 out of over 50) which means that our table has the advantage of being not only systematically organized but also more complete.

Petri Nets Tools Database

Availability:

Free of Charge,
Commercial,
Commercial (discounts for academic institutions)
Commercial (free for academic institutions)

Tool Features: Petri Nets Supported:

Algebraic
Coloured
P/T
Elementary
Free Choice
Stochastic
Timed

Tool Features: Tool Components:

Textual representation
Graphical editor
Token game animation
State space
Condensed state space
Place invariants
Transitions invariants
Net reductions
Structural analysis
Performance simulation
Advanced performance analysis

Environments:

Sun
HP
SGI
Linux
MS DOS
MS Windows
MacIntosh

Tool Description:

Provide an overview of your tool. Make the description such that it becomes as time stable as possible, i.e. avoid information such as version number, price, future plans, etc.

Please keep the description to at most 15 lines (approx. 200 words).

Contact Information:

Address:
Phone:
Fax:
E-mail:

Other Remarks:

Currently (in December 1996), the Petri Nets Tools Database contains only two tools (Design/CPN and ELSIR). Its further growth is expected on the basis of interactive information supply by tool providers.

Petri Net Tool Overview (Petri Net Newsletter)

The overview given in [Fel92] is organised in a similar way to the above tool lists, but with less structure. Although this gives a less clear overview, it does give more scope for characterisation, free from the constraints of rigid forms.

As this overview dates from 1992, the information it contains can already be considered out-of-date (in our fast changing times) as illustrated by the implementation of aspects which were referred to at the time as "future developments" or the subsequent "disappearance" of certain tools. This makes it difficult to judge how much valuable information the overview contains. However, since it mentions tools which are missing from the CRIM list, it represents an interesting and useful source of additional material for this survey.

3.2 Structure of the Table

The structure of the Section 3.3 is based on the assessment criteria presented in Chapter 3. In addition, the first two columns contain information on the tool's **Category** (see Section 3.1) and whether or not the tool has a **Portrait** in Section 4. The last two columns contain details of the **Source** of our information (see Section 3.1) and the **Address** of the tool's supplier. Where available, the email address is given, otherwise the street address.

In some columns "+" and "-" are used to denote "featured" and "not featured" respectively. The abbreviations for individual attributes have been explained above. Empty fields indicate that no information is available concerning the aspect in question.

3.3 Table

Category	Portrait	Name	Net type; Net class	Functionality							Infrastructure	Usability				Availability	Source	Address
				Editor	Analysis	Dynamisation	Generation	DB access	interfaces			Stability	Operability	User manual	Help function			
									Import	Export								
UNIX	+	Cabernet	HHL,Tim; ER nets	G	RG Sta	Ani Sim	-	-	Cabernet (A)	Cabernet (C) PS, Idraw (F)	g++, Motif	2	2	+	+	Gen	PT Aar	ipese4.elet.polimi.it
	+	CorMan	HLL; FUNSOFT	G	RG Sta	Ani Sim Int	-	+	-	-	X11, SunView	3	3	+	-	Req	PT	Fraunhofer ISST J.-v.-Fraunhofer-Str. 20 44227 Dortmund
	+	Design/ CPN	HHL,Tim	G	RG Inv Red	Ani Sim	-	-	-	PS(A)	X11; Linux; MAC	2	2	+	-	Gen	PT Man Lit	http:// www.daimi.aau.dk/ designCPN/
		Diogenes	LL, Tim; M nets	G	-	Ani Sim	-				VMS, GKS				+	Gen	PNN	GSI mbH
		DNAnet	Sto; GSPN	G	Inv Sta Dyn	Ani Sim	-	-		PS (F)	Motif; PC					Gen	Aar	psk@cs.uct.ac.za
		DSPN-express	Sto; DSPN	G	RG	Sim	-	-			16 MB+ RAM, X11,				-	R/D Req	Lit Aar	lind@first.gmd.de
		EASE	LL; SA nets	T	Red	-	-	-			VMS, Common Lisp					Gen	PNN	decindio@hermes. mc.dsi.unimi.it

				Functionality								Usability						
									interfaces									
Category	Portrait	Name	Net type; Net class	Editor	Analysis	Dynamisation	Generation	DB access	Import	Export	Infrastructure	Stability	Operability	User manual	Help function	Availability	Source	Address
UNIX		ELSIR	HL, Tim, Sto	G	-	Ani Sim	-				X11, Motif					Co m	Aar	Richard.Dykiel@ sp1.y-net.fr
	+	ExSpect	HHL,Tim; ExSpect nets	G	Inv Cfl Sta	Ani Sim	-	-	-	LaTeX, PS (F)	SPARC, OpenWindows ; PC	2	3	+	+	Co m	PT Aar PNN	exspect@bmc.nl
		FUN	Fun	G	-	Ani Sim	Code	-	-	-	; PC, MAC					Req	Lit	schwidd@ibm.net
		GraphEd	general	G	Sta	-	-	-	-	PS, FIG (F)	SunView	5	3	+	-	Gen	PT	forwiss.uni-passau.de
		GRASPIN	LL, HL	G	+	Sim	-	-			GeneraOS, Common Lisp, UX400-Board; MAC, Symbolics LM			+		Gen	PNN	nit@gmdzi.gmd.de
		GreatSPN	Sto; GSPN	G	Inv Cfl Sta Dyn	Sim	-				X11						Lit	G. Chiola, Uni. Turin Corso Svizzera 185 10149 Torino, Italy
	+	INA	HL,Tim,LL	T	Inv RG Red Cfl Sta Dyn	-	-	-	-	-	; PC	1	3	+	-	Co m	PT Aar	starke@informatik. hu-berlin.de

Category	Portrait	Name	Net type; Net class	Functionality							Infrastructure	Usability				Availability	Source	Address
				Editor	Analysis	Dynamisation	Generation	DB access	interfaces			Stability	Operability	User manual	Help function			
									Import	Export								
UNIX		INCOME	HL, Tim	G	-	Sim	-	+			Motif, Oracle; PC					Co m	Aar	hq@PROMATIS.de
		IPTES	HL, Tim	-	RG	-	-	-	Cabernet (F)	Cabernet (F)					-	Req	Lit	[Bel93]
	+	LOOPN	HL, OPN; OPN	-	-	Sim	-	-	-	-	C-Compiler	2	3	+	-	Gen	PT Aar	ftp.utas.edu.au
		Macrotec	Sto, HL	G	-	Ani Sim	-	+			X11, NeWS, SunWindows, Motif, SPNP, Gemstone DBM System					Req	Aar	keller@crim.ca
		Meta Design	general	G		Sim	-		-	Bitmap (F) DesignCPN (C)	X11; PC, MAC					Co m	Lit PNN	C.I.T. GmbH Ackerstr. 71-76 13355 Berlin
		MOBY	LL, Tim, HL	G	-	Ani Sim	-				Smalltalk 80; PC; MAC					R/D	Aar	lichtblau@informatik. uni-oldenburg.de
	+	MoPEd	HLL, Tim	G	Cfl Sta	-	-	-	PNG, INA (F)	PS, MIF, PNS,INA, PROD (F)	gcc, Motif	2	2	+	-	Req	PT	Fraunhofer ISST Kurstr. 33 10117 Berlin

Category	Portrait	Name	Net type; Net class	Functionality							Infrastructure	Usability				Availability	Source	Address
				Editor	Analysis	Dynamisation	Generation	DB access	interfaces			Stability	Operability	User manual	Help function			
									Import	Export								
UNIX		ONE	O; OBJSA-Netz	G		-	Code				Open-Window, SRI, DesignML					Gen	Aar	battiston@ hermes.dsi.unimi.it
		ORIS	Tim; Time PN, (CmTPN, MTPN)	-	RG Sta	-	-	-			; PC, Amiga					Gen	Aar	lusini@ aguirre.ing.unifi.it
		PACE	HHL, Tim, Sto	G	Sta	Ani Sim	Code	-	EPOS (F)	graph. Formate (F)	Smalltalk-80; PC; MAC; ATARI			+		Req Com	Lit PNN	GPP mbH Kolpingring 18a 82041 Oberhaching
	+	PAPETRI	LL, Tim, HL	G	RG Dyn Red	Ani Sim	-	-	-	PS (F)	OpenLook	2	2	+	-	Gen	PT Aar	ftp.cnam.fr
		PEP	LL, HL; M nets	G	RG Sta Dyn	Ani Int	Net Code	-	-	PS (F)	X11, Tcl/Tk			+	+	Gen	Man Lit	pep_help@informatik. uni-hildesheim.de
		PETSI	LL, Tim	G	-	Ani Sim	-				X11, Motif; PC				-	Com	Lit	Scientific Computers Postfach 1865 52020 Aachen
	+	PNS	LL	G	-	Ani Sim	-	-	-	-	X11R4	1	2	+	+	PD	PT Aar	ftp.informatik. uni-stuttgart.de

Category	Portrait	Name	Net type; Net class	Functionality							Infrastructure	Usability				Availability	Source	Address
				Editor	Analysis	Dynamisation	Generation	DB access	interfaces: Import	interfaces: Export		Stability	Operability	User manual	Help function			
UNIX	+	PROD	LL	-	RG Sta	-	-	-	ARA (A)	ARA (C)	C-Compiler; PC	2	3	+	-	Gen	PT Aar	saturn.hut.fi
		Prompt	HL		-	-	Code				SunView/ X11; PC					R/D	PNN	unico@trl.oz.au
		PSItool NET	HHL,Tim,Sto; NET nets	G	+	Ani Sim	-	-	-	-	X11, Motif			+		Req	PNN	itter@psise.uucp
		QPN-Tool	HL, Sto, QPN; QPN	G	RG Inv Sta	Sim	-				SunView			+		Req	Lit Aar	bause@ls4.informatik.uni-dortmund.de
		SANDS	HL, OPN; CO-OPN	G	TL	Ani	-				BIMProlog, Graphtalk, Motif, OpenLook					Gen Req	Aar	Didier.Buchs@di.epfl.ch
		SURF-2	Sto; GSPN	G	Inv Sta Dyn	-	-		-	PS (F)	OpenLook, Open-Window					Com	Aar	metge@laas.fr
	+	System Specs	HHL; Specs nets	G	Inv Cfl	Ani Sim	Code	-	-	PS, EPS (F)	X11; PC; MAC	1	2	+	+	Com	PT Lit Aar	swissbox.inet.ch
	+	TimeNET	Sto, LL; GSPN, DSPN	G	Inv Cfl	Ani Sim	-	-	DSPN-express (F)	PS, EPS, LaTeX (F)	X11	2	2	+	-	Req	PT	ck@cs.tu-berlin.de

Category	Portrait	Name	Net type; Net class	Functionality							Infrastructure	Usability				Availability	Source	Address
				Editor	Analysis	Dynamisation	Generation	DB access	interfaces			Stability	Operability	User manual	Help function			
									Import	Export								
UNIX		UltraSAN	Sto; SAN	G	+	Sim	-				X11R5					R/D Req	Aar	request@crhc.uiuc.edu
		Xpetri	LL	G	Inv Sta	Ani Sim	-				Tcl/Tk, X11, Motif					Gen	Aar	mcl@spiderman.bu.edu (not reached)
		XSimNet	Sto, HHL: SN	T	-	Ani Sim	-	-	-	-	X11	5	3	+	-	Gen	PT Aar	ftp.abo.fi
	+	WAM-Subset (PI+PP)	HLL	-	-	Ani Int	-	+	PED (F)	-	X11, Motif, CORBA	2	2	-	-	Com	PT	Fraunhofer ISST Kurstr. 33 10117 Berlin
	+	YANED	general	G	-	-	-	-	INA (F)	INA (C,F), FIG (F)	X11, Motif	3	2	-	-	Req	PT	Humboldt University Lindenstr. 54a 10117 Berlin
PC		Chromos	HLL, Tim, Sto	G	-	Ani Sim	Code	OO DB	-	INA, HPGL (C)	Windows 3.1x, Windows95 Pentium ab 100MHz	2-	2+		+	Com	Lit Aar	info@otw.de Ilmenau.de

Category	Portrait	Name	Net type; Net class	Functionality							Infrastructure	Usability				Availability	Source	Address
				Editor	Analysis	Dynamisation	Generation	DB access	interfaces: Import	interfaces: Export		Stability	Operability	User manual	Help function			
PC		Graph	HLL	G	+	Ani	-	-			Turbo Pascal 5.0				+	Req	PNN	University Rzeszów Rejtana Str. 16a 35-310 Rzeszów, Poland
		NetMan	Tim, HLL	G	-	-	-	-	-	XPN, SPNP SimNet, POSES (F)	Windows 3.1	2	3	+	+	Gen	PT Aar	gilem@rpi.edu 128.113.12.86
		PENECA PNC	HLL, Tim, Sto	G	-	Ani Sim	Code	-	-	INA, HPGL (C)	-				+	Com	Lit Aar	rsb@TheoInf. TU-Ilmenau.de
		PESIM	LL, Sto	G	RG Inv Red	Ani Sim	-	-	-	WMF, LaTeX (F)	Windows 3.1, LaTeX	2	2	+	-	Gen	PT Aar	ftp.fee.vutbr.cz
		PNTBLSIM	HL, Tim	T	RG	Ani Sim	-	-			Pascal; MAC			+		R/D	PNN	lew@uhics.ics. hawaii.edu
		POSES	Sto, Tim, extended HL	T	-	Ani Sim	-				Borland Pascal 6.0					Pub	Aar	ftp.tu-chemnitz.de
		TemPRO	HL, Tim	G		Ani Sim	-		Excel (A)	Excel (C)	Windows 3.1					Com	Aar	scil@interserv.com
		Visual SimNet	Tim, Sto HL, QPN	G	RG	Ani Sim	-	-	-	INA, graph. Formate, WinPetri (F)	-	3	2	-	+	Gen	PT Aar	ftp.inf.tu-dresden.de

Category	Portrait	Name	Net type; Net class	Functionality							Infrastructure	Usability				Availability	Source	Address
				Editor	Analysis	Dynamisation	Generation	DB access	interfaces			Stability	Operability	User manual	Help function			
									Import	Export								
other architectures		CPN/AMI	LL, HL; AMI nets	G	Red Sta Dyn	Ani Sim Int	ADA				Mac (UIF) and SunOS (Tools)					Tst R/D	Aar	ftp.ibp.fr
		PetriLab	HLL	G	-	Sim	-			MacDraw II (F)	Mac						Lit	dittrich@jupiter. informatik. uni-dortmund.de
		Product Net Machine	HL; Product nets	G	RG Sta	Ani	-				Symbolic-Lisp-Machine					Gen	Aar Lit	ftp.darmstadt.gmd.de

4 Tool Portraits

4.1 Selection of tools

In the following chapter, a selection of the software packages presented above are examined in more detail. Since the primary aim of this report was to serve as a preliminary survey for a net-based toolkit, only tools which are broadly compatible with the UNIX environment at the Fraunhofer ISST have been taken into consideration. That means that they should:

i) be able to run on the SunOS or Solaris operating system
ii) not require any commercial libraries not available at the Fraunhofer ISST
iii) be generally available or already installed at the ISST.

On closer inspection of the tools listed in the table, these conditions lead to the exclusion of the following tools (divided according to the condition they don't fulfil) from a more detailed examination:

i) Chromos, NetMan, PENECA, PESIM, POSES, TemPRO, Visual SimNet, CPN/AMI, PetriLab, Product Net Machine
ii) Diogenes, Macrotec, MOBY, ONE, QPN-Tool, PACE, SANDS
iii) DSPNexpress, ELSIR, FUN, GreatSPN, IPTES, MetaDesign, PETSI, SURF-2

As an exception, we were also able to include two commercial tools (SystemSpecs and ExSpect) as their manufacturers made them available to us for testing free of charge.

With the exception of GraphEd, UltraSAN and XSimNet, all the tools included in the table have now been accounted for. The first two of these exceptions were not given closer attention due to their uncommon model classes (graphs and stochastic activity networks respectively) and the third was left out due to the extreme (possibly platform-related) instability of the available version.

4.2 Considered Features

This section explains the criteria according to which the tools are examined in the following portraits. The titles of the sub-sections reappear in the descriptions, giving a common structure to the presentation of the different tools. The descriptions of the criteria are intended to serve as a framework of reference points so that the portraits themselves can concentrate on the features of the individual tools. As one can see, five of the six headings express typical functionality categories for Petri net tools (*net analysis, net dynamisation, generation* of Petri nets and programs, *user interface, interfaces*). Comments concerning *usability* are dealt with, where appropriate, as *ergonomic features* under the functionality heading in question. The sixth heading covers specific information about the *documentation/literature* or other publications concerning the tool.

4.2.1 Net Analysis

By **Analysis** we mean the process of obtaining the properties of a (net) model without generating a net's firing sequences. In contrast to simulation (see Section 4.2.2) analysis can be used to prove that a model is error-free with respect to certain internal aspects. It should be noted, however, that in order to obtain such results, sufficient knowledge of the basics of net theory is required. This cannot be compensated by any tool, however good it is. The properties of Petri nets typically checked or proved when verifying the correctness of system designs are as follows (with explanations or examples from their context within the system):

- reachability: e.g. does a system reach certain states?
- liveness: e.g. can the system reach deadlock?
- safety: e.g. may certain buffers overflow repeatedly?
- invariants: e.g. are there situations where certain local events don't lead to changes in the system state?

Let us note in advance that analysability in particular differs greatly from tool to tool, which is one indication of the need to combine tools.

4.2.2 Net Dynamisation

By **Net dynamisation** we mean the realization of the behaviour of a net by a software system. We make a distinction between the functionalities of *interpretation*, *simulation* and *animation.* The common feature of these three types of dynamisation is that nets are worked through by step-by-step firing of (sets of) transitions. Our distinction between the functionality of "dynamisers" is based on the effects caused by the firing of transitions in each case.

In the case of *interpreters*[1], the firing of transitions in a net triggers certain activities in the net's environment. Such tools are necessary, for example, for any kind of interpretation of function nets. In general, interpreters are needed wherever Petri nets are to be used for process control, e.g. in workflow management or production automation.

Simulators assist in carrying out experiments with net models. The firing sequences of nets are logged and/or statistically evaluated. These statistics can provide information both about the utilization of capacities and about properties of places (which are sometimes discovered by chance) such as conflicts and unboundedness.

Animators enable the visualisation of the token flow and of the firing of transitions usually referred to as the "token game"[2]. When animators are combined with an editor - this is the case for most of the tools examined - very user-friendly environments are obtained. Sometimes

1. in this report's sense of the word. A better, but unusual term for such tools might be "enactor".

2. for certain tools, e.g. DesignCPN, this function is called Simulation additional visualisation features are provided which enable a better representation of behaviour in terms of certain application areas. Another useful feature is the visualisation of conflicts and the provision of support for interactive conflict resolution.

Any type of interpretation must include options for the resolution of conflicts between transitions as they occur. In addition to "manual" resolution as mentioned above - where the current markings and sets of conflicting transitions are displayed for the user to choose which transition(s) should fire - it is also possible to define priorities or probabilities in advance which then determine the firing behaviour of transitions in conflict situations.

A further aspect of visualizing the interpretation process - alongside animation - concerns the representation of simulation results, for example in the form of diagrams for simulation statistics. These give an overview of a system's behaviour which can be used, among other things, to detect errors, weak points and erroneous dimensioning.

Another important consideration when assessing interpretation options is openness towards other tools. Thus, for example, the protocols generated by a simulator should ideally be available in a variety of output formats, enabling further processing to be carried out using other software packages (e.g. spreadsheet or statistics programs).

4.2.3 Generation of Petri Nets and Programs

In our table, **Generation** covers two options: the *generation of software (specifications)* out of nets and the *generation of nets* out of software. This kind of technology is only of any interest if the software used includes concurrent elements, since this is where the specific strengths of Petri nets lie. This means that the specification or programming language used must also contain the appropriate constructs (e.g. parbegin and parend).

In the first case, an executable software prototype is generated automatically from a net specification.

In the second case, an existing program specification in source code is used to generate a Petri net with the same behaviour. Usually the restriction applies that only the communication aspects are taken into account. Transformations like this have been developed, for example, for SDL, OCCAM and PDL [Les89], [Hei94].

4.2.4 User Interface

This section deals with the configurability, the usability and the special features of the **user interface**s of the tools in question.

Please note that those special features of the user interface which relate directly to analysis and interpretation are dealt with as *ergonomic features* in the corresponding sections.

When Petri net tools include a graphical editor (this is the case for most of the tools examined here) it usually also functions as the user interface into which the other tools are integrated.[1] The *usability* and *special features* of editors are therefore included under the User Interface heading in the detailed descriptions.

1. This also seems to be the most obvious and feasible solution in the case of a net-based toolkit.

Configurability

By configurability of the user interface we mean the options for adapting the tool to specific user requirements. In most cases these options are limited to changing the size and colour of symbols and characters. Other valuable features include special visualisation methods for b/w monitors, fonts, grid size and display modes (with/without additional parameters of the net elements).

Usability

This heading mainly covers our own experience with the user interface, with particular reference to styleguide conformance (menu design, button labelling, cf. [OSF91]), understandability of error announcements and warnings, and learnability.

Special Features

The features listed under this heading are special graphical functionalities. "Special" means those features which are not standard for a Petri net editor such as the rotation and mirroring of subnets and nodes, to give just two examples.

4.2.5 Interfaces

We consider a (system) **interface** between two systems or two system components to mean any properly specified option allowing one system (component) to fetch information generated by another system (component) or to pass its own information to another system (component). We distinguish between two types of interface: filters and APIs (Application Programming Interfaces). Depending on whether the information to be used is received or sent we make a further distinction between *import* and *export* interfaces.

We use *filter* to mean a program (or part of a program) which transforms structured information in a particular way into a different format without altering the semantics, i.e. so that the meaning of the information in the target format corresponds to the meaning in the source format. However, it is accepted that information can always be lost during such transformations.

In accordance with the second feature of interfaces (receiving or sending information), we distinguish between import and export interfaces:
Import filters understand foreign file formats and import the required information into the tool in question. Not all the information from the foreign format is always used for this.
Export filters know foreign formats and generate files in the requested format using their own information. Not all the information known internally is always used for this.

APIs are the second form of communication between different software tools. They allow a tool or part of a tool's functionality (e.g. output of specific data in a specific format) to be called by another tool. Such interfaces are defined by the names of the callable processes and their parameters.
If a call received by the API from another tool initiates the transfer of data to this (calling) tool, then the interface of the tool being called is an *export API*.
If, on the other hand, a call received by an API (from another tool or user) causes data to be transferred to the tool being called, the interface is an *import API*.

As in the table in the previous section, the information concerning interfaces is divided between the categories *import* and *export* interfaces, whereby both filters and APIs are mentioned in both categories. As with the other aspects for assessment, the table contains an exhaustive listing of each tool's interfaces, while the descriptions in this chapter only go into greater detail on special features.

The additional category *fileformat* has been included for any disclosed or published information about file formats. This can be important, for example, for the creation of additional filters.by the user.

4.2.6 Documentation/Literature

This heading covers comments on the documentation and literature about the various tools. Our only formal requirement was that any documentation should contain at least a description of the tool's functionality. The assessment therefore only reflects our subjective impressions.

4.3 Tool Portraits

These descriptions contain detailed discussions of the features listed in Section 4.2 of the tools selected according to the criteria given in Section 4.1. When reading the descriptions one should bear in mind that only a small amount of the information given in the table is repeated (e.g. for the description of the net class). For most aspects of functionality (including the quantified assessment of stability and operability), of the required infrastructure, of availability and of the level of information, the relevant details are not repeated.

Cabernet V1.3

Net Class:

The Cab nets used are a typed version of E/R nets (Environment/Relationship). Using C++ syntax, each place is assigned a type (of permitted token), while the predicates and actions (arrow and transition markings also in C++) relate to these types.

E/R nets are hierarchical high-level nets with a very general time concept. (Tokens carry time stamps; firing may only occur between freely definable minimum and maximum delay times.) There are two time semantics. *Weak* semantics: the event in question *may* occur at the specified times. *Strong* semantics: the event in question *must* occur at the specified times, as long as its concession isn't lost beforehand due to the firing of conflicting transitions.

Analysis:

Options:

The analysis is divided into two parts. A net checker (syntactically) checks model properties (e.g. non-allowed identical place names), in particular "timing correctness" (e.g. time functions). In the second part a timed reachability graph can be created, on the basis of which liveness and safeness can be determined within a (global) time interval.

Ergonomic Features:

The graph is displayed in a different window in the editor (buffer mechanism).

Dynamisation:

The token game allows free selection of step lengths and speeds.

Conflict Resolution:

There are two conflict resolution modes:
1. interactive decision by the user
2. automatic selection of one transition according to a uniform probability distribution.

Ergonomic Features:

Once the net has been translated into a C++ file (default with g++, which is relatively complicated) the resulting file is executed. The editor's integrated animator gives a constant indication of the position of the tokens during the token game. A final statistical evaluation is not possible.

Generation:

None

User Interface:

Configurability:

User-oriented configuration is only possible to a certain extent. Colours and fonts can be selected, with the rather unusual property that colours are defined via <u>one</u> scroll bar or via numbers. The system script for menus etc. is very small. Graphical functions are not provided.

Usability:

The character functions have been kept relatively simple (e.g. when moving nodes and drawing arrows, only the start and end points are displayed, but no intermediate states).

Icons are provided for quick access to principle commands and the menus are clearly structured.

There is a status bar, which is rather small.

Ignoring the usual styleguides, "cancel" is used in some places to signal confirmation (e.g. to confirm the new values and close the window after token modification).

Special Features:

At first look, the hierarchy concept is rather unusual (after "level down" out of a node, the whole net is displayed instead of just the section in question).

8 (non-editable) symbols are available for the representation of net elements.

The global time concept is (of course) also applied to the modelling of hierarchies.

Hierarchies are created by refinement, coarsening is not possible.

Interfaces:

Export:

The two provided graphical export filters are fully integrated. The one available API ("Tool Generator") is associated with the statements executed by transitions as they fire. These transitions are part of a net which can be edited and executed within Cabernet. The embedded nets must be safe and untimed.

Import:

As for export API.

Fileformat:

The file format is not described. Although there is a file on the FTP server with the name "fileFormats.ps.gz", it cannot be accessed (no permission).

Documentation/Literature:

The user manual gives clear explanations of all functionality - a little too brief in places. Further documents on the nets and the tool are available via FTP.

Our Experience:

Learnability:

Using the user manual and the help function (same content as the manual) it is easy to learn how to use Cabernet, despite its size. Once successfully implemented, the tutorial would certainly be a great help.

Installation/Operation:

Quick and easy installation is possible, since only the binary file is supplied. Cabernet is available for various computer systems.

Cabernet's operation is not always stable:

The tutorial was not accessible (one of the binary files was not executable).
The help function only works if "tricks" are used.
The editor occasionally crashes, leaving a "core" (e.g. during token modification).

Applicability/Combination Options:

In conceptual terms and according to the examples given, Cabernet is well suited for the modelling and evaluation of sensitive safety applications.

The tool seems to be well suited as an editor with integrated analysis and simulation. Although its openness is limited due to the filters, the "Tool Generator" does allow the integration of other tools (in the form of Cab nets). This means that a form of API is provided, which can be very important for net-based tool kits in particular.

CorMan V1.8.3

Net Class:

FUNSOFT nets are timed hierarchical high-level nets and constitute a generalization of function nets, i.e. the transitions ("instances" or "activities") are linked with "job types" which characterize the firing behaviour and contain an executable description. A distinction is made between "social interaction" and "process programming" job types, modelling human interaction and automatic, locally defined activities respectively. The places ("channels") have additional attributes in the form of scheduling disciplines (LIFO, FIFO, Random) which determine the order in which tokens ("objects") are removed by their post-transitions.

Analysis:

Options:

Reachability analysis, boundedness of places and, to a limited extent, liveness. Net topology can also be examined (critical paths on the basis of simulation results).

Ergonomic Features:

Results are displayed numerically (no graphics).

Dynamisation:

Conflict Resolution:

In simulation mode (jobs not being executed on firing of transitions) the resolution of conflicts works interactively. When the net is being interpreted (jobs being executed on firing of transitions) conflicts can be resolved either interactively or automatically (according to predefined probabilities).

Ergonomic Features:

The token game can be followed on the graphical interface in both simulation and interpretation modes. At the end of each simulation phase, a statistical report is available for the individual nodes.

Generation:

None

User Interface:

Configurability:

It is not possible to configure the user interface.

Usability:

For the inexperienced user, the tool requires some familiarization time due to the complexity of the net class and the architecture. The large number of pull-down menus doesn't always make the tool easier to use.

Special Features:

The editor provides graphical functions not included in the normal standard (e.g. rotate).

Interfaces:

Export:

None

Import:

None

Fileformat:

The structure of the file format was not disclosed.

Documentation/Literature:

There are a series of publications and sets of instructions concerning the various components of the system. There is no user manual as such, since the tool wasn't designed for external use.

Our Experience:

Learnability:

Due to the complexity and number of the functions provided, CorMan cannot be learned on-the-fly. The lack of a single user manual is a further reason, since the necessary information has to be gathered from a number of separate documents.

Installation/Operation:

The installation has become more user-friendly, but no less complicated. CorMan is basically stable, but crashes can be provoked. Since we only tested a demo-version, some functions were not available (e.g. critical path calculation).

Applicability/Combination Options:

To date, CorMan has only been used internally or made available as a demo-version. It was not designed for combination with other tools.

DesignCPN

Net Class:

The used nets are hierarchical high-level nets which can be timed. Design/CPN supports Jensen's hierarchical coloured Petri nets (CP nets for short), see [Jensen92].

CP nets use data types, data objects, and variables. The data types are called colour sets, data objects are called tokens. Each place is assigned a colour set, determining the allowed tokens for the place. Transitions can be labelled with a predicate, called guard, to restrict their activation. Each arc is inscribed with an expression, representing the pre- and postconditions for firing a transition. Moreover, some code can be attached to transitions to be executed when firing.

Declarations and net inscriptions have to be formulated in the CPN ML language. CPN ML is based on the functional programming language Standard ML (Meta Language) which has been extended by some syntactic sugar for colour set declarations and for using typed variables as well as reference variables.

When building a timed CP net, tokens may have an associated number, called a time stamp. A timed token is not available for any purpose whatever unless the (internal) clock time is greater than or equal to the token's time stamp.

Analysis:

Options:

There exist two main tools for the purpose of analysis. First of all, a syntax checker is included to
assure that the ML-syntax of the net inscription is correct (the pure net structure is always correct due to the editor). Only syntactically correct nets can be further analysed or interpreted.

The second tool is called OG (occurrence graph) tool. It is used to build and investigate the occurrence graph of a net. The OG tool has a large number of built-in standard queries. They can be used to investigate all the standard properties of a CP-net, such as reachability, boundedness, home properties, liveness and fairness (see [Jensen94]). In addition to the standard queries there are a number of powerful search facilities allowing you to formulate your own, non-standard queries. The standard queries require no programming at all. The non-standard queries usually require that you write 2-5 lines of quite straightforward ML code.

Ergonomic Features:

The OG tool is fully integrated with Design/CPN. This means that you can switch between the
editor/simulator and the OG tool. When an occurrence graph node has been found, it can be inspected in the simulator. This means that you can see the marking directly on the graphical representation of the CPN model. You can see the enabled transition instances, investigate their
bindings and make simulations. Analogously, when a marking has been found in the simulator,
it can be added to the occurrence graph or used as the initial marking for a new occurrence graph.

Dynamisation:

CP nets are actually programs, expressed in a hybrid language of graphics and text. CP net syntax is designed to insure that any legal CP net is completely and unambiguously defined. The nets are not interpreted by themselves but by a run-time package, the simulator. Before starting the simulator a syntax check is necessary.

This CP net simulator that is closely integrated with the editor allows a net to be iteratively constructed and interpreted with a minimum of overhead. The simulator both manages and displays CP net interpretation, and allows its course to be controlled in various ways to facilitate study and debugging. There are many options to for influencing the simulation: the degree of concurrency, conflict resolution, etc.

There are two main interpretation modes:
1. Interactive run: This mode is used to visualize the token game, i.e. this mode acts as an animator. It is allowed to interrupt the run (or to stop it at a breakpoint) and to change marking, simulation options, to resolve a conflict by hand, etc.
2. Automatic run: In this mode the token game is not shown, but the net is simulated. This mode can be used to get simulation statistics, i. e. this mode can act as a simulator. Another possibility is to interpret the net in using the side-effects of code attached to transitions. Since the inscriptions are done in CPN-ML there exists a broad variety of intended side-effects.

Conflict Resolution:

Solving conflicts is done either by the user or randomly. The user can select a binding for an activated transition, i.e. the values under which the transition should fire. When solving randomly the user can influence the conflict resolution with some parameters.

Ergonomic Features:

The token game can be followed on the graphical interface in both simulation and interpretation modes. The setting of breakpoints is possible to suspend the interpretation at various points during the execution cycle. After suspension the user can inspect a marking, binding or other things. Note that breakpoints effect each transition. It is not possible to include or exclude individual transitions.

When interpreting very large nets the interpretation can take a lot of time, especially when doing an interactive run. The token game can be followed on the graphical interface in both simulation and interpretation modes. At the end of each simulation phase, a statistical report is available for the individual nodes.

Generation:

None

User Interface:

Design/CPN is divided into an editor for creating and manipulating CP nets, a syntax checker for validating CP nets, a simulator for executing CP nets and an analyser for calculating, analysing and drawing occurrence graphs (also called state spaces, reachability graphs or reachability trees).

Additionally there exist capabilities for interactive monitoring and debugging, facilities for organizing a net into a hierarchy of modules and animation and charting facilities for displaying simulation results.

Configurability:

The size and graphical attributes of net elements can be freely selected. The appearance of the user interface (i. e. colours and fonts) can be changed by setting some X resources, but the names are not documented.

Usability:

For the inexperienced user, the tool requires a considerable familiarization time. There are no
icons (but key short-cuts) provided for quick access of principle commands. The menus are clearly structured.

A status bar is included for showing information and assistants during complex operations.

Special Features:

The editor provides graphical functions not included in the normal standard (e.g.circular alignment of elements). Some others are not included (e.g. resizing is possible only for single objects).

Interfaces:

Export:

Export to ps format is provided, but only with some tricks.

Import:

None

Fileformat:

The structure of the file format was not disclosed.

Documentation/Literature:

There are five very detailed online manuals available. The theoretical aspects are described in three extra books (the third one is not yet published). The manuals and books contain good examples. Nevertheless, some experience with petri nets is necessary.

Much more literature, examples and libraries can be found on the internet.

Our Experience:

Learnability:

Due to the complexity and number of the functions provided, DesignCPN cannot be learned on-the-fly. When building complex nets, you first have to learn CPN-ML, especially when attaching code to transitions.

Installation/Operation:

There were no problems during installation. It is done very fast, since only the binaries are submitted. You must be able to reach the internet, because the program is assessable only via FTP.

Operation is stable, but there is still the possibility of a crash. When problems occur, excellent support is provided via email.

Applicability/Combination Options:

If one wishes to use Jensen's CPNs, this tool is a must. In many publications on Petri nets, these are graphically represented using Design CPN.

Due to missing interfaces, DesignCPN cannot be combined with other tools.

ExSpect V5.0

Net Class:

"ExSpect" nets are special timed hierarchical high-level nets with a large number of underlying predefined (algebraically specified) data types.

Places are generally called *channels*. Special channels called *stores* always contain exactly one token. The transitions are either *processors* (with parameters, function description, precondition and delay) or *systems* (special transition-bordered subnets). Systems also form the basis for a hierarchical module concept ("subnet" = "system").

Due to the time concept, the immediate firing rule is always applied.

Analysis:

Options:

For performance analysis, an ITPN file (interval timed PN) is created and then examined using the integrated tool IAT. Liveness and boundedness properties can be checked, as well as (transition and place) invariants and conflict sets.
Statistics can also be obtained concerning the static net structure (e.g. number of arrows to and from a given place).

The results are displayed and stored in text form.

Dynamisation:

Visualization of the token game is possible. Breakpoints can be selected for predefined interruption of the simulation (but not a number of steps). There are no options for interaction. The simulation results can be displayed online in diagram form using the applications provided (e.g. Exgraph).

The simulation supports multi-user operation.

Generation:

None

User Interface:

Configurability:

The size of the symbols can be freely selected and their form can be chosen from a given set. Colours and grid size can be chosen from a menu.

Additional attributes (e.g. line width, symbol library, printer) can be modified by editing files which are loaded at start.

Usability:

Using ExSpect requires some practise.
The creation of nets with the graphic editor is awkward due to the extensive use of keyboard commands. The complicated menus (many branches, too many in places) have a similar effect.
The availability of a single function (e.g. animation) via various paths (directly or via View) is an unusual feature.

Special Features:

The graphical representation of the net elements (vector graphics?) takes some getting used to.

Interfaces:

Export:

Export is only provided to graphics formats (ps, LaTeX). An external call of these filters is not possible, which means that conversion into these formats always involves calling the whole program package. The filters themselves export the graphics alone - no additional information or graphical elements (e.g. frames).

Import:

Not possible

Fileformat:

ASCII. The format isn't stated explicitly.

Documentation/Literature:

Installation & Customization Guide: the installation procedure is clearly explained and is easy to follow. Several paths have to be defined (we experienced no problems with this).

A Language Tutorial supports the learning-by-doing approach to the description language.

Using an example (the "basic logistic chain"), the Tool Tutorial gives a clear, concise introduction to the use of the relatively complex tool.

The Reference Manual is thorough, although its structure tends to correspond to the structure of the user interface rather than to groups of required functions. The description of mouse clicks and of how to navigate through menus is incomplete (e.g. in Section 4-1, which window contains the menu item "view"? The main menu seems to change from chapter to chapter).

Our Experience:

Learnability:

Learning how to use the tool proved to be harder than expected, even when following the tool tutorial. One reason for this could be the choice of not particularly intuitive names for menu items. There are some inaccuracies, such as menu items which have to be clicked on according to the manual but which are either non-sensitive (greyed out) or not in the menu at all.

Installation/Operation:

There were no problems during installation. Operation was stable.

Applicability/Combination Options:

Combination with a SimCom editor for E/R modelling is possible.

INA V1.7

Net Class:

INA is primarily designed for low-level nets (including timed and time nets). These low-level nets can also be folded to coloured nets to refer to special properties in term of coloured nets. Coloured nets can be unfolded for analysis. Delays can be allocated to all static net elements (places, transitions, arrows).

The firing rule can be selected from "normal" (only the markings of the input places are taken into account), "safe", "with capacities" and Valk-type (for self-modifying nets). It is also possible to define priorities for transitions. The firing strategy can be set to single or maximal step.

There are various time concepts (duration and interval time assignment to places, transitions, and arcs).

Analysis:

Options:

INA offers an unusual range of analysis options, divided into reachability, invariant and structural analysis. The appropriate analysis method for the net type in question is identified by an expert system working on the basis of identifiable properties.
Nets can also be reduced without losing properties (e.g. boundedness, liveness).
The analysis algorithms are adapted automatically to the chosen combination of net features like time concept, firing rule and firing strategy.

Ergonomic Features:

Users accustomed to windows-oriented interfaces will find the text output rather awkward. Results are logged in a file.

Dynamisation:

None

Generation:

None

User Interface:

Configurability:

The only configuration option is the length of line for the output.

Usability:

The limitation to text mode makes usage rather complicated. The fact that Petri nets have the advantage of being able to graphically represent concurrent behaviour cannot be exploited by the text-oriented representation of a stand-alone INA set-up.

Special Features:

The COMMAND file saves all commands issued during each INA session. After a restart, they can be called in the same order to repeat exactly the same analysis procedure for a different net.

Interfaces:

Export:

None

Import:

None

Fileformat:

The file format of the net is described in the user manual, allowing the user to write additional import/export filters to other tools. Other files generated by INA (e.g. configuration files) are not explained in detail. All file formats are readable ASCII files.

Documentation/Literature:

The user manual uses numerous examples to give a good explanation of INA’s functionality. However, in order to understand the content, the corresponding book is required [Starke90]. According to an information from the tool providers (December 1996), a new handbook (in German) will be available in the WWW in 1997.

Our Experience:

Learnability:

The learning process is slowed down by the text mode (which is generally considered obsolete).

Installation/Operation:

No problems during installation, since the binary file only has to be copied. Very stable operation.

The text output of results can produce very long and unhelpful lists.

Applicability/Combination Options:

The strengths of INA are its capabilities for analysis and reduction. As far as analysis is concerned, it constitutes an exception and concerning this aspect it is the strongest of the tools we have examined.
The Petri net editors PED and MoPEd can be used as a graphical fronts end to INA. It is also planned to integrate a library version of INA into PEP.

As a recent development, a tcl interface (called inatcl) is being created which can be used autonomously and as part of the PEP tool. More information on this topic is available from Lutz Pogrell of Humboldt University.

LOOPN V2.003

Net Class:

"Object Petri Nets" are used to attempt "full object orientation" of Petri nets. Unlike other object net classes, inheritance is therefore allowed. As in function nets, executable C-routines can be attached to the transitions. However, the version of the tool we examined does not yet match this complicated but internally consistent net class (cf. "Documentation").

Analysis:

None

Dynamisation:

Conflict Resolution:

No details.

Ergonomic Features:

In the net description grammar, it is possible to insert C-output routines (e.g. printf). These commands are executed during simulation. This is the only output option during simulation. After simulation, the user can request a status message (numerical output of statistics).

Generation:

None

User Interface:

Configurability:

None

Usability:

The tool consists of a collection of 5 binary files. There is no uniform user-guidance - each program is called separately via parameters. An editor is not included (still in planning phase).

Special Features:

None

Interfaces:

Export:
None

Import:
None

Fileformat:
A grammar is defined.

Documentation/Literature:

The user manual is not easy to understand, despite the numerous examples. There are many other publications on the FTP server. "From Coloured Petri Nets to Object Petri Nets" (tr94-9.ps) provides a good explanation of the current state of development.

Our Experience:

Learnability:
The effort required to learn how to use LOOPN is fairly large because of the lack of graphical user guidance and the apparent complexity of the manual.

Installation/Operation:
Installation is easy since source code and an easily adaptable make-file are provided. The user can also extend LOOPN (by changing the source code). The tool's operation was found to be stable, except that some of the options described in the manual were executed wrongly or not at all.

Applicability/Combination Options:

To a certain extent, the net class is revolutionary, so that its integration into a net-based toolkit is not yet possible and doesn't even seem necessary.

MoPEd V3.2

Net Class:

Hierarchical low-level nets with labelling for all net elements. A special extension consists of the modelling of modular nets (with port concept) and modular process nets (with event concept) [Wik96].

Analysis:

Options:
Apart from the representation of conflict clusters, no analysis options are available.

Dynamisation:

None

Generation:

None

User Interface:

Configurability:

The representation within the character table can be finely adjusted using internal options, i.e. optional display of the various individual information items concerning each net element.

No configuration options (fonts etc.) apart from the usual X-options.

Usability:

The tool is designed for intuitive use. The status is displayed via the form of the cursor. Short, manageable (non-context-sensitive) menus make the editor easy to handle. When an graphical object is double-clicked, the editor automatically selects the correct entry mask for details about the corresponding net element.

Special Features:

Special graphical functions are provided (rotate, flip). It is also possible to draw on more than one A4 page. Nets from a library can be introduced via an include function. The hierarchical concept supports both top-down (refinement) and bottom-up (coarsening) work methods. There is a broad range of options for the graphical representation of transitions.

Interfaces:

Export:

The export filters are all integrated in the graphical interface.

Import:

The import filters exist as separate files which have to be called externally. As well as acting as translators, they also have a placement function.

Fileformat:

The net's file format will be described in the planned user manual. It is a readable ASCII file.

Documentation/Literature:

Internal documents only.

Our Experience:

Learnability:

MoPEd is intuitive and easy to learn/use.

Installation/Operation:

Due to the makefile, in which only the paths have to be adjusted, installation is no problem and suitable for a large number of systems.

Very stable operation.

Applicability/Combination Options:

Due to its outstanding graphical capabilities, the tool is best suited as an editor. MoPEd can be combined with other tools easily due to the existence of filters to various net and graphics formats.

PAPETRI V4

Net Class:

Processes place/transition nets, timed, coloured and algebraic nets.

Analysis:

Options:

The choice of analysis options varies from net class to net class. For place/transition nets, algorithms are provided to determine the overlap and reachability graphs, to calculate invariants and for decomposition. For timed nets there are two procedures for the calculation of reachability graphs. For algebraic nets, P-invariants and a number of special structures can be identified.

Ergonomic Features:

Although the editor follows a consistent graphically-oriented concept, the analysis results are only available in text form.

Dynamisation:

Conflict Resolution:

Three modes available: "manual" (interactive conflict resolution); "automatic" (random resolution of conflicts according to uniform probability distribution); and "semi-automatic" (mixed mode).

Ergonomic Features:

A tracefile can be created after every simulation.

Generation:

None

User Interface:

Configurability:

The user interface can be finely adjusted to user requirements for node, script and grid size. Above-average degree of configurability for printer output (or PS export).

Usability:

The tool is relatively easy to operate although the button functions are unusually distributed (usual left mouse-button functions on middle button).

The menus show only those options which are possible at a given time (e.g. the analysis menu changes according to the net class).

Special Features:

Graphical functions (e.g."rotate") and drawing on more than one A4 sheet. Also a "preview" function, making repeated tests and printouts unnecessary.

Interfaces:

Export:

A filter for graphical formats is integrated in the user interface.

Import:

None

Fileformat:

The file format and the format of other generated files is not documented.

Documentation/Literature:

A user manual is provided (in ASCII file form) but it is very brief and gives no information on the supported net classes.

Our Experience:

Learnability:

The user interface is very well designed, making it easy to learn the wide range of functions. The context-sensitive menus are particularly helpful.

Installation/Operation:

As a binary file is provided, only three environment variables have to be set. Operation was stable during the test phase.

Applicability/Combination Options:

Well suited for use as an editing and analysis tool due to the variety of net classes it can process.

PED V4.3

Net Class:

PED is basically dedicated for hierarchical place/transition nets. An assignment of additional attributes like priorities, capacities, time delays and time intervals to places, transitions, and arcs is possible. By this way, most of types of Petri nets can be analysed using INA (with the exception of coloured Petri nets). This includes net classes with non-stochastic time assignments like time nets, timed nets, and duration interval nets.

Analysis:

None

Interpretation:

None

Generation:

None. Generation of executable code is in preparation.

User Interface:

Configurability:

Colours, fonts, text labels, and function keys can be configured by means of a resource file.

Usability:

PED is very easy to use. A broad offer of editing functionalities allows a comfortable design of very large nets. This includes rotation of net parts and alignment functionalities, adjustment of net element sizes and using of a grid.

Special Features:

Splitting and Merging of nodes is supported. A search function can be used to look up and edit net elements.

The integrated hierarchical browser provides many editing and navigation features. It is possible to save parts of the nets.

Interfaces:

Export:

Graphical export to PostScript, FrameMaker Intermediate Format and the XFig file format are supported. Additionally, export functions to the net analysis tools INA, PEP, and PROD are provided. The export functionality is fully integrated into the interface. An export to TimeNet is in preparation.

Import:

None

Fileformat:

The file format (ASCII) will be described in the forthcoming user manual.

Documentation/Literature:

A user manual is in preparation.

Our Experience:

Learnability:

PED is very intuitive. A competent handling can be captured in very short time.

Installation/Operation:

Due to a make file, where only paths have to be adjusted, installation is problem free. Operation is stable.

Applicability/Combination Options:

Well suited for the design of large, structured nets. Due to the variety of export functions, PED is suitable as a graphical front end to several net analysis tools.

PEP V1.4a

Net Class:

PEP supports several description languages for non-sequential processes: $B(PN)^2$, a concurrent programming language called $B(PN)^2$, PBC, a process algebraic formalism, a class of high-level nets called M-nets, and non-hierarchical place/transition nets without arc weights. Due to the lack of experiences with the former three formalism we will discuss only to the program parts dealing with place/transition nets.

Analysis:

Options:

Reachability analysis, boundedness, and freedom of deadlock are supported. It should be mentioned that the integration of a library version of INA is in preparation.

PEP provides also a model checker for a rather restricted version of Computational Tree Logic (CTL). Instead of computing a complete or reduced version of the reachability graph like, for instance, the model checking component of PROD, the model checking component of the PEP is based on the construction of a finite prefix of an unfolding of a net, which does not contain interleaving concurrent transition occurrences. However, this method works only for safe nets. PEP provides model checking for a rather restricted version of CTL.

Ergonomic Features:

The formula editor is not very comfortable. In the case that a checked CTL formula is not valid for the net under consideration, the model checker gives no information where the required property is violated.

Interpretation:

The basic token game is supported.

Conflict Resolution:

There are two conflict resolution modes:

1. interactive decision by the user
2. automatic selection of a randomly chosen transition.

Ergonomic Features:

The net animation is very rudimentary.

Generation:

None

User Interface:

Configurability:

Several analysis options can be chosen. Colours and fonts are not configurable.

Usability:

The net editor is very rudimentary and suitable only for small examples.

Special Features:
None

Interfaces:

Export:
PostScript

Import:
None

Fileformat:
The file format (ASCII) is documented in an additional paper.

Documentation/Literature:

A user manual and an on-line help function are available.

Our Experience:

Learnability:
PEP is very easy to handle. This is partly because of the lack of sophisticated features of the user interface.

Installation/Operation:
PEP requires a specific version and a certain path level of the TCL/TK-libraries. This makes the installation difficult. Once installed, operation is stable.

Applicability/Combination Options:

Due to the rudimentary interface, the most useful part is the model checker. The performance is extraordinary fast. The provided temporal logic is too restricted to express more complex system properties.

A useful combination with other tools is the usage of the Petri net editor PED as a graphical front end.

PNS V1.0

Net Class:

Untimed place/transition nets. Immediate firing rule with transition priorities and weighted arrows.

Analysis:

None

Dynamisation:

Conflict Resolution:

Simulation can be carried out in various modes: single step, defined (freely specifiable) number of steps, or undefined number of steps. (Firing is discontinued if the net dies, if a break condition is fulfilled or if a user interrupts it.) Sequential and parallel operation are possible. In the former, only one of the activated transitions fires, whereas in the latter, a maximum number of activated transitions fire. In all cases, conflicts are resolved either interactively or at random.

Ergonomic Features:

The animator gives a view of the current token positions during enactment. The positions aren't recorded, so that subsequent evaluation is not possible.

Generation:

None

User Interface:

Configurability:

A resource file loaded at start enables a wide range of colour, button and size adjustments to the X-resources. Further options are not provided.

Usability:

PNS is very easy to use.

Special Features:

None

Interfaces:

Export:

None

Import:

None

Fileformat:

The file format (ASCII) is explained in the user manual.

Documentation/Literature:

The user manual contains a description of the program and some introductory theoretical details about Petri nets. Most of the algorithms used to run the program are also explained. Further publications are not known to us.

Our Experience:

Learnability:

The small number of command elements and the fact that they are represented by icons make the functionality easy and quick to learn. As there is also a (small) help function, the user manual is hardly necessary.

Installation/Operation:

Simple installation involving the adjustment of three environment variables, since the binary file for Sun OS is provided. If a different configuration is installed, it can easily be recompiled. Operation is exceptionally stable.

Applicability/Combination Options:

PNS can be used without extras as a small, simple editor and animator. Its simplicity makes it robust and stable. Although PNS offers no interfaces, it is easy be combine with other tools on account of openness (source text available).

PROD (June'96)

Net Class:

Predicate/Transition Nets. Token types, markings, and transition functions are specified as an extension of the C-language.

Analysis:

Options:

PROD consists of several independent components linked by a batch program. The net description is translated into a reachability graph generator. Additionally to the construction of the full state space of the net, it is possible to apply (combinations of) several algorithms to obtain a reduced version of the state space. This includes stubborn set reduction, sleep set reduction and the computation of symmetries. The stubborn set method is suitable for deadlock analysis.

There is also a program component to compute the strongly connected components of the reachability graph.

PROD provides a query language called PROBE which allows the examination of the generated reachability graph. PROBE is strong enough to express (by means of the definition of macros) a full version of propositional Computational Tree Logic (CTL). Unfortunately, the evaluation of CTL formulae requires the previous construction of the complete state space of a Petri net which is unacceptable at least for medium sized systems (state explosion problem).

Another type of temporal logics provided by PROD is a version of Linear Time Temporal Logic (LTL). LTL formulae not containing the nexttime operator can be checked very efficiently by construction of a reduced state space which is in so-called CFFD-equivalence to the complete state space using the stubborn set method. Model checking for LTL formulae is done on-the-fly, i.e. while state space generation.

Ergonomic Features:

The query language PROBE allows the formulation of very sophisticated analysis questions. The
usage of the on-the-fly verification method requires that the formula to verify is directly included in the net description file.

Dynamisation:

None

Generation:

None

User Interface:

Configurability:

PROD is fed by a batch file in which options of the several program components can be set.

Usability:

PROD is purely text based, a graphical interface is not provided.
The generation of the stubborn-set-reduced reachability graph runs very fast and the obtained state space is considerably smaller than the original one. This applies also to the combination with the on-the-fly verification.

Interfaces:

Export:

None

Import:

None

Fileformat:

Nets are specified in an extension of the C-language. The file format is documented in the user
manual.

Documentation/Literature:

A detailed user manual is available. The documentation of the query language PROBE is very brief and hard to understand. No information on the theoretical background is given.

Our Experience:

Learnability:

The handling of PROD requires a deeper knowledge in several fields, at least in temporal logics and graph theory.

Installation/Operation:

The installation is problem free due to a make file. Only paths have to be adjusted.

PROD runs very stable. A very good performance is obtained by the combination of the on-the-fly model checker and the stubborn set reduction method.

Applicability/Combination Options:

PROD is very suitable for the verification of system properties expressible by LTL. If the supported Predicate/Transition nets are used to model place transition nets, the Petri net editor PED can be used as a graphical front end.

SystemSpecs V2.1

Net Class:

Specs nets are hierarchical high-level nets which are especially suited to the modelling, analysis and interpretation of real-time systems.

All net elements can be inscribed using the net description language SpecsLingua. A list of attribute types or (initial) attribute values is allocated to places, conditions and actions (resulting in executable code) are attributed to the transitions, and the arrows can include attribute names.

Tokens are linked to data types (standard, predefined and user-defined).

The safe firing rule always applies, i.e. a maximum of one token per place is allowed.

A global time concept is provided by the option of allocating *delays* and *activation times* **to the transitions** for the purposes of timed simulation.

The I/O elements are a special feature enabling the explicit specification of the interaction with display or output data during simulation.

Analysis:

Conflicts and invariants are displayed in text form. The form of the output appears unusual at first due to the inclusion of the internal hash numbers. Selecting an element in the display table highlights the corresponding place in the graphical net representation. Unfortunately, no user action is provided for showing all places belonging to a given conflict/invariant set.

Dynamisation:

Simulation with and without animation, with adjustable step lengths.

Number of transitions fired and average firing rates available as simulation results. Predefined I/O elements also allow graphical display in diagram form.

It is also possible to obtain traces (sequences of fired transitions, I/O elements and place values) and "token histories" for any given place. Traces are also available in Excel format.

Generation:

C++, ANSI-C, VHDL and coprocessor code can be generated from nets. Since the test version did not include this functionality, tests were not possible.

User Interface:

Configurability:

Node symbols (in bitmap form) can be created and altered. Element and grid size, element colours, and language (e.g. English/German) can be adjusted.

Usability:

Work is supported by well-structured, context-sensitive menus throughout. The tool is very stable. Individual errors are filtered out "gently" as they occur and passed on semi-automatically to system support.

The graphical representation of nets is near perfect.

Special Features:

Analysis results (e.g. invariants) can be graphically displayed in the editor. The node elements can also be in editable bitmap form. Additional quantitative information can be embedded in the I/O elements in convenient graphical forms (e.g. curves, dials, meters, etc.).

Interfaces:

Export:

The export interface can only be called from within the tool and is suitable for the automatic generation of documentation.

Import:

None

Fileformat:

Data is stored in ASCII - the format is not explained.

Documentation/Literature:

The reference handbook (incl. net description) is well laid out and easy to understand.

The "Getting Started" tutorial offers a comprehensible overview, instructions for installation and configuration, and a very instructive example.

There is a series of publications about additional functions (e.g. generation) and modifications for special applications (StructWare).

Our Experience:

Learnability:

Quick familiarization with the tutorial and examples provided.

Installation/Operation:

Easy installation, although problems were experienced with the files ".Xauthority" and Magic Cookies".

With its fault tolerance and security, the operation standards of this tool are pointing the way for this software category.

Applicability/Combination Options:

System Specs is the only one of the tools tested which can be combined with the standard evaluation programs (e.g. Excel) without additional effort.

TimeNET V1.4

Net Class:

Many generalizations of stochastic Petri nets (GSPN and DSPN) can be examined using TimeNET. It supports timed and untimed transitions (deterministic, exponentially and randomly distributed), weights (firing probabilities) and priorities (in the case of immediate transitions).

Analysis:

Options:

Structural analysis: calculation of place invariants, extended conflict sets and confusions.

Analysis of time behaviour: For GSPNs DSPNs and extended DSPNs, both stationary and instationary characteristics of the corresponding (semi-)Markov processes can be calculated. For generalized and "concurrent" DSPNs, stationary characteristics can be approximated. There is also a simulation component which can be used to obtain transient as well as stationary measures of SPNs.

Ergonomic Features:

The results are presented in numerical form only.

Dynamisation:

Conflict Resolution:

During the token game, conflicts are resolved manually without taking firing times into consideration. If the net is being evaluated, the priorities for the conflict sets of untimed transitions decide.

Ergonomic Features:

Net interpretation is accompanied by a very clear graphic display. After a net has been evaluated, the time-related characteristics of interest (which can be defined individually) are displayed in numerical form in a window. The results are also logged in a RESULT file. These values can either be calculated or obtained by simulation.

Generation:

None

User Interface:

Configurability:

There are no configuration options apart from the usual X-options. Minimal changes can be made to the view by setting a small number of internal options.

Usability:

Quick familiarization helped by short, clear menus. The two part menu "change/marking" for the modification of places is unusual, but can also be mastered without difficulty.

Special Features:

Above all the option of creating "rounded arrows". Using Splines, TimeNET can make curves out of the corners in the connections (arrows) between places and transitions. A second interesting feature is the "rescale" function which allows nets to be made larger or smaller.

Interfaces:

Export:

The export functionality is fully integrated into the interface and cannot be used externally.

Import:

The filter is activated automatically when a DSPN express file is identified. It can also be called externally.

Fileformat:

The structure of the net and the graphical data are described in an ASCII file. One noteworthy feature is the comments option. An explanation of the entire file format is given in the user handbook.

Documentation/Literature:

The user manual gives a good explanation of the installation and how to use the tool, although it is sometimes rather brief. The net examples given are not supplied as models. There are a number of publications about the evaluation procedures which have been implemented.

Our Experience:

Learnability:

The user interface is easy to use and the handbook is well written so that it doesn't take long to learn how to design nets. Further assistance is given by the status bar showing the operations available via mouse. The performance evaluation function is more difficult to master (lack of examples, see above).

Installation/Operation:

In the latest version, user-driven compilation of the user interface is possible. The necessary configuration file is easy to create and once several paths and variables have been set, the tool runs smoothly. One exception is that it crashes during "load net" if access is denied for the file in question.

Applicability/Combination Options:

TimeNET offers an acceptable editor and standard analysis options for untimed nets. Its strength when compared with other tools lies in the large number of qualified analysis algorithms for timed nets.

Combination with other tools would be no problem since the file format is stated.

WAM-Subset (August'95)

Net Class:

Modular process nets [Wik95].

Analysis:

None

Dynamisation:

Conflict Resolution:

Conflicts are solved interactively by the user with controlled support from other elements of the WAM system.

Ergonomic Features:

There is no actual evaluation. However, the firing sequence is observable in the progress of the workflow system and on the PP's graphical display.

Generation:

None

User Interface:

Configurability:

The Subset consists of the WAM components PP (process net presenter) and PI (process net interpreter). Configuration is not possible beyond the X-options.

Usability:

Can only be used as part of the WAM system.

Special Features:

As a graphical component, the PP can only display a subset of MoPEd's graphical symbols (WAMSubset). The PP's implementation ensures usability for users at b/w terminals.

Interfaces:

Export:

None

Import:

The PI only understands MoPEd files - since it only knows one format, there can be no "filter" in the usual sense of the word.

Fileformat:

See MoPEd

Documentation/Literature:

Documentation only available on request (research project funded by external contractor).

Our Experience:

Learnability:

See Usability

Installation/Operation:

In the current version, the effort for installation and operation is relatively high (due to being embedded in the overall WAM system). The operation of Subset is stable.

Applicability/Combination Options:

At the moment, these WAM components can only function as part of the WAM system. It can be combined with other software by Using the Corba implementation Orbix, combination with other software components should be possible.

YANED V1.0

Net Class:

The tool is open to all net classes, the idea being that it should be possible to model any net class using a model description.

Analysis:

None

Dynamisation:

None

Generation:

None

User Interface:

Configurability:

No configuration beyond the X-options. When the options are evaluated on call, errors occur in the distinction between X-options and Yaned's own parameters. It is also not possible to adjust the graphical symbols (despite of the range of net classes the tool can handle).

Usability:

The tool is easy to use.

Special Features:

The implemented hierarchy concept seems to be inconsistent: the deletion of interface nodes in a net refinement does not lead to the removal of the corresponding nodes at the next highest level.

A positive aspect is the graphical representation of INA analysis results (which is unique among the tools examined). The tool also includes palette menus based on a mechanism otherwise only found in Open Look applications (Push-Pins).

Interfaces:

Export:

The export functionality is integrated in the user interface and cannot be used separately.

Import:

The import filter is a separate file which must be called externally. It also serves as a placement tool.

Fileformat:

Two ASCII files are used to store the information, one for net description, the other for the model description of the net class. Both formats are explained in the dissertation ([Heg95]).

Documentation/Literature:

The tool was created as a dissertation project. One section of the dissertation text deals with the tool's operation. However, it is not long enough to be called a user manual. Further publications are not known.

Our Experience:

Learnability:

YANED is relatively easy both to learn and to use. Anyone who is familiar with MoPEd will have few difficulties, since the concept is borrowed from MoPEd.

Installation/Operation:

Problem-free installation since a fully compiled binary file for SunOS is available. Easy installation by means of source code is also possible for other systems. The editor runs without serious problems. Calling an INA analysis when there was no INA resulted in a core of the child process.

Applicability/Combination Options:

YANED is an editor with an open-class concept. This is an important aspect (in the context of a net-based toolkit). A further advantage is its interoperability with INA, which makes it unique among the tools examined here.

5 Final Remarks

As expected, each of the tools examined has its specific strengths and weaknesses, thus confirming the need for a combination of various tools within a net-based toolkit. The creation of such a toolkit would make it possible to isolate individual features of different tools and to bundle them in a common environment. The resulting synergetic effects should lead to further improvements in functionality and user comfort.

We feel that the primary aim of this study - to provide a basis for a user-driven design of adequate net-based toolkits - has been well achieved. Hopefully, it will also serve as a guideline for improvement the quality of the tools for their developers.

It remains to hope that one of the essential problems in the easy realization of such toolkits - the lack of one dominating or even standard description format - will be solved at least partly by the current developments towards a recognized ISO standard for Petri nets.

References

[Bau93] Bause, Falko:
Queueing Petri Nets - a formalism for the combined qualitative and quantitative analysis of systems.
In: Proceedings of 5th International Workshop on Petri Nets and Performance Models, Toulouse, 1993
S. 14-23

[Bau94] Bause, Falko; Kemper, Peter; Kritzinger, Pieter:
Abstract Petri Net Notation.
Dortmund: Lehrstuhl Informatik IV, University Dortmund
Research Report, 1994

[ChS96] Christensen, Søren; Mortensen, Kjeld Høyer:
Petri Nets Web
Aarhus: Computer Science Department, University of Aarhus
http://www.daimi.aau.dk/PetriNets/maintainers.html

[Den93] Dunn, Robert H.:
Software-Qualität: Konzepte und Pläne
(Software Quality: Concepts and Plans. In German)
München; Wien: Hanser
London: Prentice Hall, 1993

[Fel85] Feldbrugge, Frits:
Petri Net Tools.
In: Advances in Petri Nets 1985 / Rozenberg, G. (Ed.).
Berlin: Springer, 1985 (Lecture Notes in Computer Science 222)
S. 203-223

[Fel92] Feldbrugge, Frits:
Petri Net Tool Overview.
In: Petri Net Newsletter 41, Gesellschaft für Informatik,
Bonn: Gesellschaft für Informatik, Fachgruppe 0.0.1, 1992
S. 2 - 42

[Ger94] German, Reinhard et al:
TimeNET - A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets.
Berlin: Fachbereich Informatik der Technischen Universität,
Research Report, 1994

[Gou95] Goutal, Jean-Michel:
Petri Net Tools Description List.
now at: http://www.daimi.aau.dk/PetriNets/tools/db.html

[Gru91] Gruhn, Volker:
Validation and Verification of Software Process Models.
In: Proceedings of the European Symposium on Software Development Environments and CASE Technology, June 1991 in Königswinter / Endres, A.; Weber, H. (Eds.).
Berlin: Springer, 1991 (Lecture Notes in Computer Science 509)

[HaW95] Hahmann, André; Wikarski, Dietmar:
Petrinetz-Werkzeuge - eine vergleichende Studie.
Berlin: Fraunhofer ISST (Interne ISST-Berichte 6/95) December 1995

[Heg95] Hegner, Dirk:
Konzept und Implementierung des graphischen Editors YANED für Petrinetze verschiedener Netzklassen.
(Concept and Implementation of the Y ANED graphical editor for Petri nets of dif ferent net classses. In German)
Berlin: Institut für Informatik der Humboldt-Universität, Diploma Thesis, 1995

[Hei94] Heiner, Monika; Wikarski, Dietmar:
An Approach to Petri Net Based Integration of Qualitative and Quantitative Analysis of Parallel Systems.
Cottbus: Fakultät für Mathematik, Naturwissenschaften und Informatik der Technischen Universität,
Research Report, 1994

[Hub90] Huber, P.; Jensen, K.; Shapiro, R.M.:
Hierarchies in Coloured Petri Nets.
In: Advances in Petri Nets 1990 / G. Rozenberg (Hrsg.)
Berlin: Springer, 1990 (Lecture Notes in Computer Science 483)

[ISO95] Draft International Standard ISO/DIS 9241-11
Ergonomic requirements for office work with visual display terminal (VDTs) - Part 11: Guidance on usability

[ISO96] ISO/IEC JTC1/SC7/WG11:
Project 7.19.3 (Petri Net Standardisation), Version1.1
www.itr.unisa.edu.au/tsec/pn_standard/
March 1996

[Lak95] Lakos, Charles:
From Coloured Petri Nets to Object Petri Nets.
In: De Michelis, G.; Diaz, M. (Eds.): Proceedings of 16th International Conference on the Application and Theory of Petri Nets, Torino, June 1995.
Berlin: Springer, 1995 (Lecture Notes in Computer Science 935)

[Laz84] Lazowska, Edward D.; u.a.:
Quantitative System Performance.
Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1984

[Les89] Leszak, Marek; Eggert, Horst:
Petri-Netz-Methoden und -W erkzeuge.
(Petri Net Methods and T ools. In German)
Berlin: Springer, 1989
(Informatik-Fachberichte; 197)

[Löw95] Löwe, Michael; Wikarski, Dietmar; Han, Yanbo:
High-order Object Nets and Their Application to Workflow Modelling.
Berlin: Fachbereich Informatik der Technischen Universität,
Technical Report, 1995

[Mar84] Marsan, Ajmone M.; Balbo G.; Conte G.:
A class of Generalised Stochastic Petri Nets for the performance evaluation of multiprocessor systems.
In: ACM Transactions on Computer Systems (1984) Vol. 2,
S. 93-122

[Mar87] Marsan, Ajmone M.; Chiola G.:
On Petri Nets with deterministic and exponentially distributed firing times.
In: Advances in Petri Nets 1986 / Rozenberg, G. (Hrsg.).
Berlin: Springer, 1987 (Lecture Notes in Computer Science 266)
S. 132-145

[Mur89] Murata, Tadao:
Petri Nets: Properties, Analysis and Applications.
In: Proceedings of the IEEE **77** (1989) No. 4,
S. 541-580

[OSF91] Open Software Foundation:
OSF/Motif Style Guide.
Revision 1.1
Englewood Cliffs, New Jersey: Prentice Hall, 1991

[Rei85] Reisig, Wolfgang:
Systementwurf mit Netzen.
(System Design with Nets. In German)
Berlin: u.a.: Springer, 1985
(Springer Compass)

[Ros91] Rosenstengel, Bernd; Winand, Udo:
Petri-Netze: eine anwendungsorientierte Einführung.
(Petri nets: an application-oriented introduction. In German)
4th improved and enlarged edition,
Braunschweig: Vieweg, 1991

[ShR91] Shackel, Brian; Richardson, S.J. (Eds.):
Human Factors for Informatics Usability.
Cambridge: Cambridge University Press, 1991

[Sta90] Starke, Peter H.:
Analyse von Petri-Netz-Modellen.
(Analysis of Petri Net Models. In German)
Stuttgart: Teubner, 1990
(Leitfäden und Monographien der Informatik)

[Wik88] Wikarski, Dietmar:
A Proposal for an Integrated System Evaluation Concept.
in: Modelling and Performance Evaluation of Distributed Systems. Proc. Intl. Seminar, Berlin: Institut für Informatik und Rechentechnik, 1988 (Informatik - Inf, Rep. 17/88)

[Wik96] Wikarski, Dietmar:
An Introduction to Modular Process Nets.
Technical Report TR-96-019,
Berkeley: International Computer Science Institute (ICSI), April 1996

[Wik95a] Wikarski, Dietmar; Heiner, Monika
On the Application of Markovian Object Nets to Integrated Qualitative and Quantitative Software Analysis.
Berlin: Fraunhofer ISST, 1995 (ISST-Berichte 29/95).