iSPN: an Integrated Environment for modeling using Stochastic Petri Nets

C. Hirel* S. Wells R. Fricks† K.S. Trivedi
Center for Advanced Computing and Communication
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0291

Abstract

Interaction with computers has come a long way since the archaic textual interfaces. There is now substantial literature on -computer interaction (HCI) [2, 3], a research subject widely recognized as a vital component of successful computer applications. But, when we evaluate HCIs currently available on analytical modeling packages, we realize the enormous gap perceived between their interfaces and modern HCI trends. The developers of analytic modeling packages need to deliver beneficial services to the user, and deliver them in a usable way. This paper suggests an approach of delivering this next generation of modeling tools with improved HCI. The approach is followed in the development of an integrated environment for modeling using Stochastic Petri Nets, named iSPN. Careful consideration was given to the design and implementation of iSPN to facilitate the creation of SPN models. iSPN increases the power of SPNP (the Stochastic Petri Net Package)[1] by providing a means of rapidly developing stochastic reward nets (SRNs); the model type used for input. Input to SPNP is specified using CSPL (C based SPN Language), but iSPN removes this burden from the user by providing an interface for graphical representation of the model. The use of Tcl/Tk [5] in designing iSPN makes this application portable to all platforms.

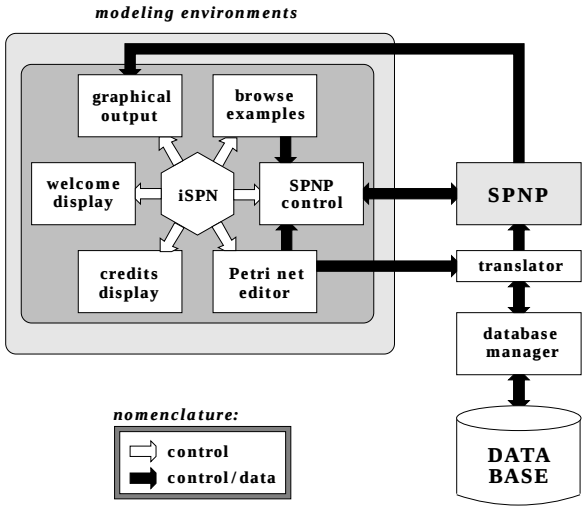

Figure 1: iSPN main software modules.

1 Development Strategy

iSPN mimics the look and feel of real file cabinet drawers by using *windows with title tabs* to ease the sorting of overlapping windows [2]. The HCI design model of iSPN is based on modern premises [3] that design should

: (i) be *user-centred* and involve users as much as possible so that they can influence it; (ii) *integrate* knowledge and expertise from the different disciplines that contribute to HCI design; and (iii) be highly *iterative* so that testing

can be done to check that the design does indeed meet users' requirement.

Designing the HCI of iSPN based on the established premises is ensured by the appropriate selection of design model. We use the one proposed by Hix and

*Supported by grants from the Lord Foundation, by a Duke University Research Instrumentation award, and by a core project of the Center for Advanced Computing and Communication.
†Supported in part by a GTE Fellowship.

Harston [4], known as the *star life cycle* (see Figure 2). This life cycle w

as selected particularly because it encourages iteration. The central point of the star design cycle is evaluation, which is viewed as relevant at all stages of the life cycle and not just at the end of the HCI development as traditional software product

developments suggest. All aspects of systems development are subjected to constant evaluation by users and by experts. Another reason for the selection of the star life cycle is its natural orientation on being supportive of both top-down and bottom-up de

velopment. A feature extremely important in iSPN since the development of some of its components starts from the underlying modeling tools (such as SPNP) and progresses upward towards the user, while the specialized interfaces (e.g., for the network appli

cation) follow exactly the opposite direction. Yet, another contributing reason is that the star life cycle also stresses rapid prototyping and an incremental approach towards the final product.

The *rapid prototyping* approach adopted in iSPN deals directly with the problem of needing to check that the users' requirements are being met by the design at different stages. New design ideas can be exercised and commented on by users before a gre

at deal of expensive development work is completed. Besides, the prototyping approach helps the HCI designers to cope with the problem of understanding requirements.

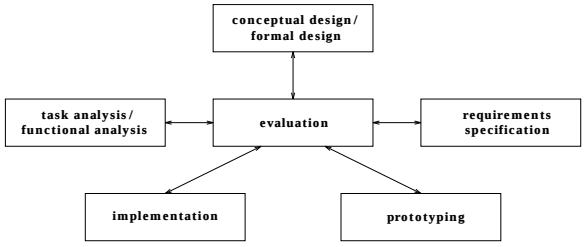

Figure 2: The star life cycle.

2 Software organization

The major components of the iSPN interface (see Figure 1) are a Petri net editor which allows graphical input of the stochastic Petri nets and an extensive collection of visualization routines to analyze output results of SPNP and aid fo

r debugging. Each module corresponds to a page in the software.

- **Input Data** :

  iSPN provides a higher level input format to CSPL which provides great flexibility to users. iSPN is capable of executing SPNP with two different file formats: (i) files created in the CSPL (C-based Stochastic Petri net Language) ; and (ii) files created using iSPN's Petri Net editor.

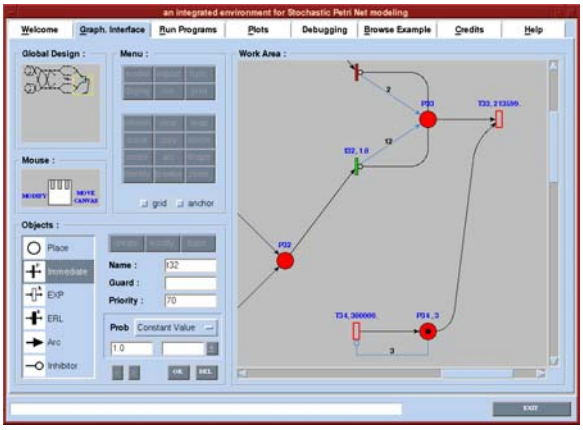

Figure 3: The petri net editor.

The Petri Net editor, which is the software module of iSPN that allows users to graphically design the input models, introduces a new way of programming SPNP: the user can draw the SRN model and establish all the necessary additional functions (i.e., rewa

rds rates, guard function, etc.) through a common environment. The Petri Net editor provides several characteristics normally available only in sophisticated two-dimensional graphical editors and a lot of new features designed specifically for the SPNP en

vironment.

iSPN also provides a textual interface, which is necessary if we wish to accommodate several categories of users. Beginners may feel more comfortable using the Petri Net editor whereas experienced SPNP users may wish to input their models using CSPL directly. Even if the textual input is the option of choice, a lot of new facilities are offered through the integrated

environment. In both cases, the "SPNP control" module provides everything a user needs to run and control the execution of SPNP without having to switch back-and-forth among distinct environments (i.e., the UNIX command or a text editor).

- **Output Data** :

  The main goal of most GUIs is only to facilitate the creation of input data for its underlying engine. Usually, the written communication between the software's output and the GUI is neglected. One of the advantages of iSPN is the incorporation of display

  ing SPNP's results in the GUI application. iSPN's own graphing capability allows the results of experiments to be graphically displayed in the same environment. Different combinations of input data may be compared against each other on one plot or viewed

  simultaneously. The graphical output format is created in such a way that it may be viewed by other vizualization packages, such as gnuplot or xvgr.

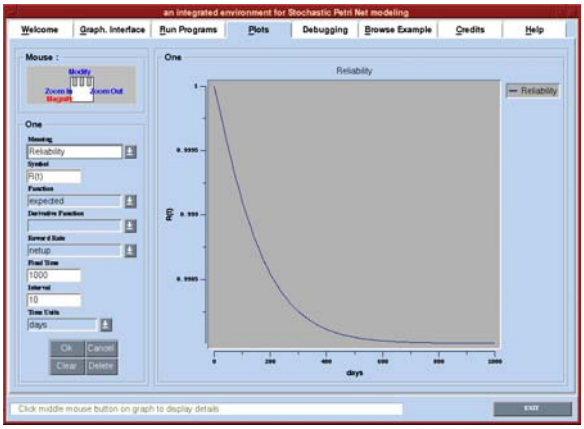

Figure 4: The output page.

- **Library and Examples and Debugger** :

  Previously, debugging was a difficult task due to the textual environment. Opening many intermediate files created during the compilation of SPNP was necessary in order to know the validity of the CSPL file. One of these intermediate files, ".rg", is im

  portant for finding bugs in the model description, if they exist. This file is a description of the reachibility graph and is displayed in a tree format as a part of iSPN. The debugging feature is based on navigation on the reachability graph resulting in

  an innovative function of iSPN that should provide improved efficiency in the development of stochastic Petri net models. iSPN also offers the user the unique ability to efficiently and selectively browse a database of old SPNP models. Users can now easily benefit from the multitude of modeling examples developed in more than a decade of academic and professional use of SPNP

  . This specialized browser provides an easy way of organizing SPNP programs, which can be very beneficial, even for the more seasoned users.

Contact : chirel@ee.duke.edu, wells@ee.duke.edu; fricks@ee.duke.edu, kst@ee.duke.edu.

References

[1] G. Ciardo, A. Blakemore, P.F.J. Chimento, J.K. Muppala, K.S. Trivedi, "Automated Generation and Analysis of Markov Reward Models using Stochastic Reward Nets", in: *Linear Algebra, Markov Chains, and Queuing Models*, Editors: C. Meyer and R. J. Plemmons, Vol.48 of *IMA Volumes in Mathematics and its Applications*, Springer-Verlag, 1992.

[2] B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Wokingham, England, 1992.

[3] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S.Holland, and T.Carey, *Human-Computer Interaction*. Addison-Wesley, Wokingham, England, 1994.

[4] D. Hix and H.R. Harston, *Developing User Interfaces: Ensuring Usability Through Product and Process*. Wiley, New York, USA, 1993.

[5] Brent B. Welch, *Practical Programming in Tcl and Tk*, Prentice Hall, Upper Saddle River, NJ, USA, 1995.