IDEAS: an Integrated Design Environment for Assessment of Computer Systems and Communication Networks[*]

R. Fricks[†] C. Hirel[‡] S. Wells C.W. Ro[§] X. Zang K. 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 arcaic textual interfaces. There is now substantial literature on **human-computer interaction** *(HCI), a research subject widely reconized as a vital component of successful computer applications. But, when we evaluate HCIs currently available on analytical modeling packages of computer systems we realize the enormous gap perceived between their interfaces and modern HCI trends. A possible reason may be the extensive attention given to the numerical intrinsics related to the modeling problem. Now, however, the current practice on designing HCIs to modeling packages has to evolve. 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 and without giving up the mathematical engines of old tools. The approach is presented*

[*]This work was supported by CACC IDEAS core project, funded in part by NASA Lewis Research Center

[†]Supported in part by a GTE Fellowship.

[‡]Supported by grants from the Lord Foundation and a Duke University Research Instrumentation award.

[§]Supported by grants from Korea Science and Engineering Foundation (KOSEF), the Lord Foundation and a Duke University Research Instrumentation award.

through the description of an integrated environment for modeling computer systems and communication networks named IDEAS, which is currently under development at Duke University. The key elements considered are **integration** *and* **user-centred design**. *A powerful set of selected modeling tools provides a strong foundation for IDEAS, while the two referred elements provide guidelines throughout the project.*

1 Introduction

Many powerful modeling tools have been proposed or developed, yet, their use in practice has been very limited. Despite their inherent merits, performance modeling tools (e.g., QNAP2 [1], RESQME [2], and PEPS [3]), reliability modeling tools (e.g., SURF-2 [4], HARP [5], and SAVE [6]) and performability modeling tools (e.g., SHARPE [7], SPNP [8], UltraSAN [9], DSPNexpress [10], and TimeNET [11]) still have limited penetration in industrial environments[1].

One possible method of encouraging, indeed coaxing, potential users into utilizing powerful tools is to provide an integrated modeling environment which offers a uniform and easy access to many existing and future tools. Thus, a range of capabilities will be available to a product designer without the need for learning many cumbersome interface languages and output formats. The integration will allow the user to carry out performance, reliability and performability analysis using the same toolkit.

Whenever integration is sought in the computer modeling context, the first idea that usually comes to mind is the combination of different analytic-numeric engines or of analytic and simulation engines. An initial success at such an integration in the domain of reliability modeling has been achieved jointly by Boeing, University of Washington and Duke University [15]. To carry the idea of integration several steps further we created the **Integrated Design Environment for Assessment of Computer Systems and Communication Networks** (IDEAS) project. In the IDEAS, we are also exploring other facets of integration:

[1]For a survey on modeling tools, the reader is referred to [12]

1. Integration of modeling packages under a common user friendly environment geared towards the assessment of complex computer and communication systems.

2. Possibility of integration of automatic data collection procedures coordinated by the IDEAS environment, coupled with the possibility of interconnection of statistical analysis of measured data.

3. Integration of off-the-shelf software components (e.g., data base manager, graphical interface builder, etc.) to speed-up the development process and to enhance the quality of the final product.

4. Aggregation of different system design factors such as performance, reliability, cost, and weight to be modeled in the same framework.

5. Software/hardware co-design: since the overall system performance is dependent on both hardware and software, an integrated approach to system design and evaluation is required.

6. Development of a common specification environment for design and mathematical modeling of computer systems, allowing for the integration of qualitative and quantitative analyses.

7. Development of a single intermediate language that supports several distinct application domain oriented languages.

Apart from the integrated workbench under implementation, IDEAS also offers several other significant advantages: (i) From the user perspective: it helps minimizing problems associated with the selection of modeling techniques most appropriate for a specific problem; through its higher level interface, users can access a multiplicity of modeling engines (solvers) without the associated costs of learning each of these tools; reports can be automatically produced in this environment better documenting the design/modeling process. (ii) From the modeling researcher perspective: it speeds-up the development and distribution of new modeling tools since their interface may be the one provided by the IDEAS environment.

The IDEAS project shall provide a high level environment for the design and analysis of computer systems and communication networks, allowing a multiplicity of modeling tools to be used for evaluation of alternative designs. The undergoing project includes the development of an environment for performance, dependability and performability modeling powered by successful analytic modeling packages such as SPNP and SHARPE, and research tools such as Distributed SPNP (D-SPNP) [16] and the Fluid Stochastic Petri Net (FSPN) simulator [17].

The rest of the paper is organized as follows. Section 2 introduces the design philosophy behind the IDEAS project. Section 3 details the first phase of the IDEAS project. Section 4 presents the graphical user interface developed for SPNP, one of the highligths of the first phase of the project. Section 5 concludes the paper with the current status of the project.

2 Design Philosophy of the IDEAS Project

The interface metaphor adopted by the IDEAS project is of a file cabinet (see Figure 1) available at the desktop level of the designer/modeler. Each modeling engine and specialized interface integrated in the IDEAS project has its own drawer in the file cabinet. This structure allows IDEAS to accomodate several distinct types of users: network designers can use the specialized interface (e.g., stSDL) to input theirs modeling problems without the burden of learning the intrinsics of stochastic Petri nets, Markov chains or other modeling paradigms. On the other hand, modelers can access directly those familiar structures through the available *graphical user interface* (GUI) of any of the modeling engines integrated to IDEAS. The virtual file cabinet even has a big advantage over the real one, since it allows the continuous evolution of IDEAS functionality by adding an unlimited number of new drawers as necessary.

Whenever a drawer is opened, a uniform structure is presented to the user in a similar way to the action of opening drawers in real file cabinets. The graphical interface offered by all drawers is consistent, minimizing the time and other resources necessary to train new

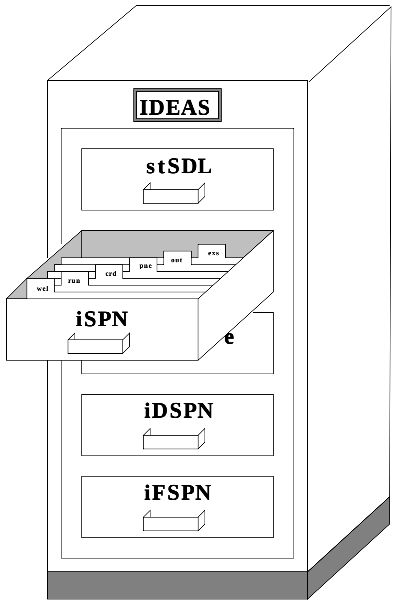

Figure 1: File cabinet metaphor.

users of IDEAS. Once a new user becomes familiar with a particular interface (drawer) he should be able to use proficiently all the others with minimal training.

On the computer display, the virtual action of openning a drawer corresponds to the activation of one HCI program. This program mimics the looks and feels of real file cabinet drawers (compare the sketch of Figure 1 and the snapshot of iSPN reproduced in Figure 5) by using *windows with title tabs* to ease the sorting of overlapping windows [18]. The HCI design model of each GUI offered in the IDEAS project is based on modern premises [19] 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.

Making the HCI of the IDEAS project following the established premisses was ensured by the appropriate selection of design model. We use one proposed by Hix and Harston [20, 21], known as the *star life cycle* (see Figure 2). This life cycle was 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

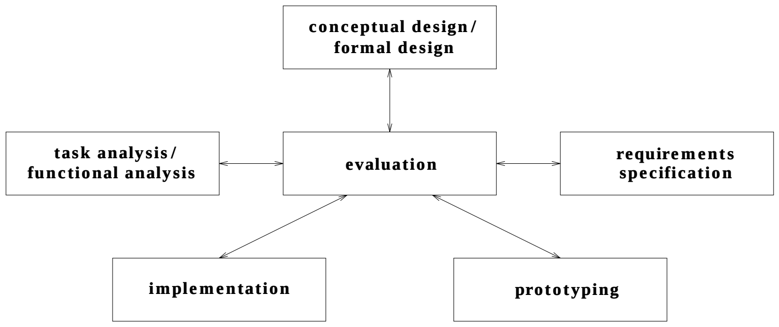

Figure 2: The star life cycle.

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 of its natural orientation on being supportive of both top-down and bottom-up development. A feature extremely important in the IDEAS project since the development of some of its components starts from the underlying modeling tools (such as SPNP and SHARPE) and progresses upward towards the user, while the specialized interfaces (e.g., for the network application) follow exactly the opposite direction. Yet, another contributing reason is that the star life cycle also stresses rapid prototyping and an incremental approach of the final product.

The *rapid prototyping* approach adopted in the IDEAS project deals directly with the problem of needing to check that users' requirements really are being met by the design at different stages. New design ideas can be exercised and commented on by users before a great deal of expensive development work has been completed. Besides, the prototyping approach helps the HCI designers to cope with the problem of understanding requirements.

3 IDEAS-kernel Project: The First Phase of IDEAS

The first phase of IDEAS implements a complete modeling environment (performance, dependability, and performability) for communication networks, and establishes the foundation for the other phases. At first, the kernel only allows dependability analysis, but later performance and performability analyses will also be available through the common environment. The output of the first phase is named **IDEAS-kernel** because everything else shall be developed around it. The IDEAS-kernel is an additional software level that is inserted between the user (a computer systems modeler or a network designer) and the modeling engines. Because of its position on top of the modeling tools, the package allows the insertion of higher level abstraction than the underlying engines. In the IDEAS context, this new abstraction level is at the description level of communication networks, completely hiding the underlying modeling paradigms such as Markov chains and Petri nets, to name a few. In the first stage, IDEAS will only provide interface with human operators, although automatic data acquisition is under consideration for implementation in later versions of the environment.

The functionality of the IDEAS-kernel is made available through the assistance of three software modules and four independent tools (termed *auxiliary tools*). The software modules are:

- The *User Interface Module* is composed of two major components: a *textual interface* (TI) and a GUI. Both components allow users to access the modeling engines either through specialized interfaces (the network modeling interface in the first phase) or the primitive resources provides by the integrated modeling packages. The GUI implements all the expected functionality of a computer-aided design interface for two-dimensional designs, including features like levels of zooming, panning, scrolling, macro functions, extensive report capabilities, etc.
- The *Environment Management Module* acts as a general service module for the other two. Some of the features available at the user interface level need strong support from the data management level, which is not only responsible for providing an uniform interface with the modeling engines, but also to keep a data base (actual and historical) of designs and of additional information necessary to smooth the operation of the whole environment. The integration of the modeling engines is provided through an interpreter that maps one intermediate language implemented by the user interface into the input languages of the auxiliary tools (e.g., CSPL for ths SPNP package). Additional automation implemented in the interpretation

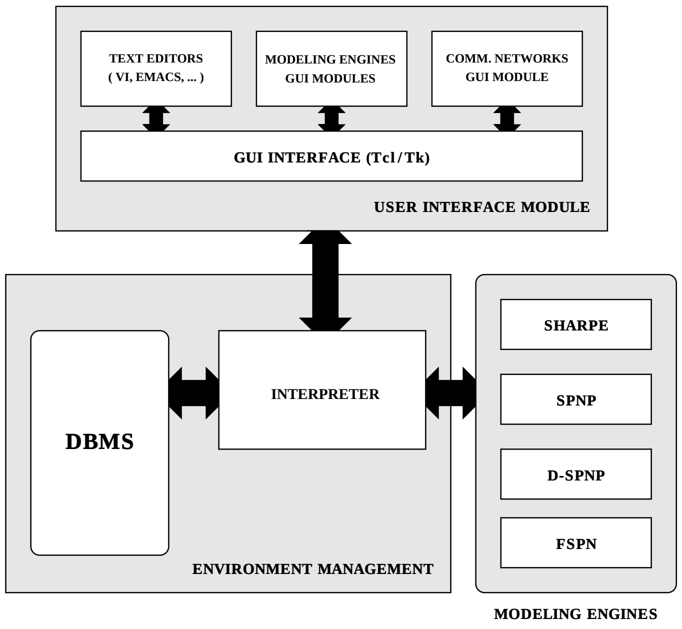

Figure 3: Functional Specification.

process is the automatic selection of the auxiliary package(s) most suitable for a particular problem. The selection process can be overridden by the user. This is an additional feature implemented to allow IDEAS to accommodate several distinct types of users with different modeling skills. If the user has a strong modeling background he or she will be able to select and work directly with his or her choice of auxiliary tool, but even in this case IDEAS provides a superior interface than the individual auxiliary packages themselves. The interpreter also provides a bi-directional interface with the auxiliary tools. Not only mapping the intermediate language into the auxiliary packages' input languages, but also it takes results from several packages and maps them into a common interface to present to the user through tables and appropriate plots.

- The *Modeling Module* organizes all modeling engines integrated through IDEAS. The interpreter concept allows for continuous growth of the modeling engine module, either in the number of supported tools or their characteristics. New modeling packages shall be easily integrated, providing freedom for each individual installation of the environment to select tools more appropriate for their particular use.

Due to its flexibility, availability and quality, the initial auxiliary tools to be integrated are:

- **SPNP (Stochastic Petri Net Package)**: This package was developed by Ciardo et.al. [8]. The model type used for input is a *stochastic reward net* (SRN). SRNs incorporate several structural extensions to GSPNs [22] such as marking dependencies (marking dependent arc cardinalities, guards, etc.) and allow reward rates to be associated with each marking. The reward function can be marking dependent as well. They are specified using CSPL (C based SRN Language) which is an extension of the C programing language with additional constructs for describing the SRN models. SRN specifications are automatically converted into a Markov reward model which is then solved to compute a variety of transient, steady-state, cumulative, and sensitivity measures. For SRNs with absorbing markings, mean time to absorption and expected accumulated reward until absorption can be computed.

- **SHARPE (Symbolic Hierarchical Automated Reliability/Performance Evaluator)**: A modeling package originally developed in 1986 by Sahner and Trivedi. The main feature of SHARPE is that hybrid and hierarchical models can be easily constructed [7]. The overall system model may consist of several submodels of possibly different types. The model types allowed are fault-trees, reliability block diagrams, reliability graphs, Markov chains, semi-Markov chains, single and multi-chain product form queuing networks, GSPNs, and series-parallel task graphs. For example, the reliability of a system can be modeled by a reliability block diagram where reliability of each block is computed by solving a Markov chain. If a single Markov model for such a system was constructed, then it could have a state space of exponential size. Hierarchical modeling alleviates the problems of model largeness and stiffness to a large extent. SHARPE can compute distribution function (symbolic in t), mean, and variance of time to failure of the system.

- **Distributed SPNP (D-SPNP)**: One of the key difficulties in using state space based models like SRNs is the size of the underlying state space. As a single workstation tool, SPNP, SHARPE etc. are limited to small problems whose state space can be stored within the workstation's memory. One possible solution to the problem is to parallelize the generation of the underlying Markov chain. Ciardo et. al. from College of William and Mary have developed an efficient algorithm for distributed state space generation. A prototype distributed engine has also been developed for generating the underlying state space associated with SPNP models. This prototype is referred to as the D-SPNP. The current implementation of the engine can be run on either a set of networked workstations or on a SP/2 multiprocessor. For details on D-SPNP, please see [16].

- **Fluid Stochastic Petri Net (FSPN)**: Another approach to tackling the state space explosion is to use models based on continuous flow rather than discrete movement of tokens. By incorporating this idea into the SPNs, Kulkarni et. al. [17] have developed a modeling approach using Fluid Stochastic Petri Nets (FSPN). FSPNs and corresponding solution engine will enable the analyst to easily specify and automatically generate/solve fluid models. For details on how such models are specified and solved for desired measures, the reader is referred to [23, 24].

The development strategy for the IDEAS-kernel can be summarized in three major activities (see Figure 4): (i) development of GUIs for SPNP, SHARPE, D-SPNP, and FSPN; (ii) development of a graphical environment for dependability analysis of communication networks (stSDL and associated interpreter); and (iii) development of a backbone for the integration of all the tools using a DBMS.

The implementation of the IDEAS-kernel starts with the development of GUI for the modeling tools we selected. These GUIs are important in the integrated environment because they allow great flexibility to its users: if the user feels more confortable dealing with analytic modeling concepts such as reliability block diagrams or Markov chains, he will be able to use then directly via the SHARPE GUI (iSHARPE). However, users oriented exclusively towards network design will also be able to use the environment through the specialized interface stSDL.

All GUI dedicated to the off-the-shelf tools will be developed according to a bottom-up approach, since most of the tools are already available. Extra effort is devoted to ensuing

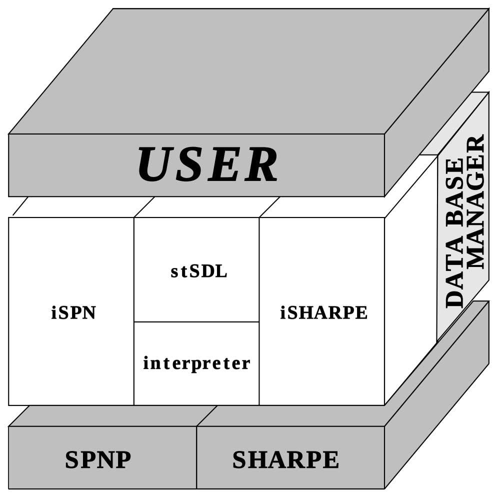

Figure 4: Software organization.

a uniform user interface in spite of intrinsic differences among the underlying tools. Most functions will present and behave the same way across the plethora of modeling tools. A property that should significantly reduce the training effort of new users.

We started the kernel implementation by designing a GUI for the SPNP package. The GUI is the key element to the success of the project, hence we decided to focus our efforts initially on this activity. The reason SPNP was first selected is because the mathematical engines of two of the modeling packages (D-SPNP and FSPN) are still under development, and we judged the design of a graphical environment for Petri nets more demanding than the one necessary for the SHARPE package.

The development of the specific modeling interface for communications network is being conducted in parallel to the development of the SPNP GUI. The design is being carried out according to a top-down approach, starting by comprehensive analysis of dependability modeling of networks. The analysis aims to identify the design procedures and elements to be incorporated in the project. After completing the initial analysis, an interpreter should be developed to examine problem characteristics of each project and assist the user (automatically or under user intervention) to select the modeling tool most suitable for each problem instance. After identifying the modeling tool, translation from the abstract network model into the selected tools will be automatic.

The DBMS will be the last piece to be incorporated into the kernel and should assist the interpreter as the intercommunication medium for various tools of the environment, besides being the information source for all of them. The DBMS that shall be incorporated into the IDEAS-kernel project is an off-the-shelf product. Working with a commercial DBMS should allow good performance, excellent interface and report capabilities to the IDEAS project.

4 iSPN - A Graphical User Interface for SPNP

According to the development strategy of the IDEAS-kernel project, a subproject named "iSPN: an integrated environment for SPNP" was started a few months ago. The purpose of this subproject is to develop a prototype GUI to SPNP. The development uses the scripting language Tcl (Tool Command Language), developed by John Ousterhout, and extension Tk, a toolkit for X windows [25, 26]. The selection of the script-based approach brings to the project three major benefits: (i) Tcl/Tk provides a higher-level interface to X than most standard C library toolkits; (ii) development of the IDEAS environment will be fast because of fast turnaround, aiding the debbuging process and refinement of the interface; and (iii) the user interface is clearly isolated from the rest of the application, making the overall design easy to maintain and expand.

The role of this first GUI is very important for the overall IDEAS project since it will establish a role model to be followed in all the subsequent GUI developments. Because of that, extra time was spent in the careful design of this interface where we incorporated some novel concepts in human interaction made available through Tcl/Tk languages.

The prototype, when completed, will be fully functional and presented to SPNP users for evaluation. After the evaluation process and consequent tune-up of the design/ implementation, the process of converting the prototype into an engineering product will start. Although we selected Tcl/Tk environment for the prototype development, we are also considering other graphical environments for the final implementation of iSPN. Careful planning led to the implementation of a graphical user interface for the SPNP package,

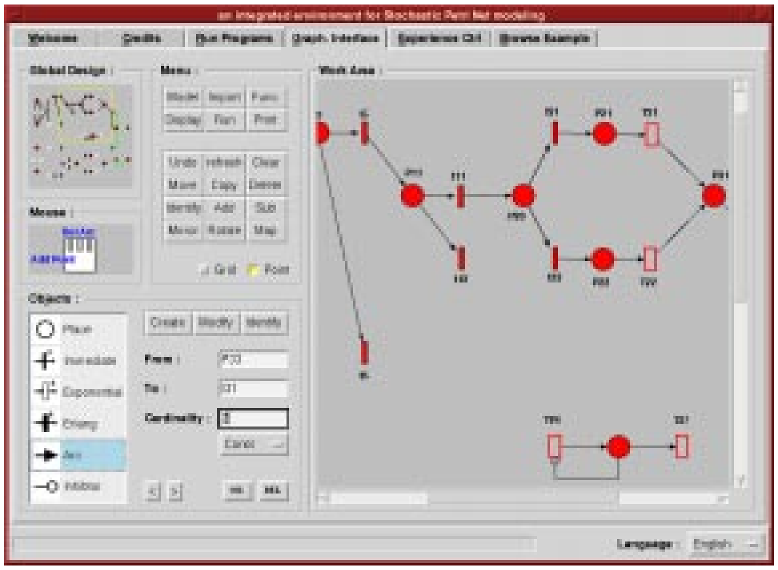

Figure 5: Snapshot of the Petri net editor.

destined to be the template for all GUI to be developed in the future for IDEAS. Extra effort shall be put into offering the user the most uniform interface possible in spite of the intrinsic characteristics of underlying tools. Most functions are displayed in a similar format and have the same behavior for all modeling tools. A property that should significantly reduce the training effort of new users as one learns each modeling language.

The iSPN prototype is almost complete (a snapshot of one of its displays is shown in Figure 5) and allows the possibility to execute SPNP with two different file formats: (i) files created in the CSPL (C-based Stochastic Petri net Language) language directly; and (ii) files created using iSPN's Petri Net editor. The software organization of the prototype (see Figure 6) should be reproduced in all the other GUIs to be developed for the IDEAS environment.

The major components of the iSPN interface are a Petri net editor (a program that allows the graphical input of stochastic Petri nets) and an extensive collection of visualization routines to analyze output results of SPNP. These software modules and the others shown in Figure 6 implement an HCI module to SPNP with the following properties:

- **Improved Input Capabilities**:

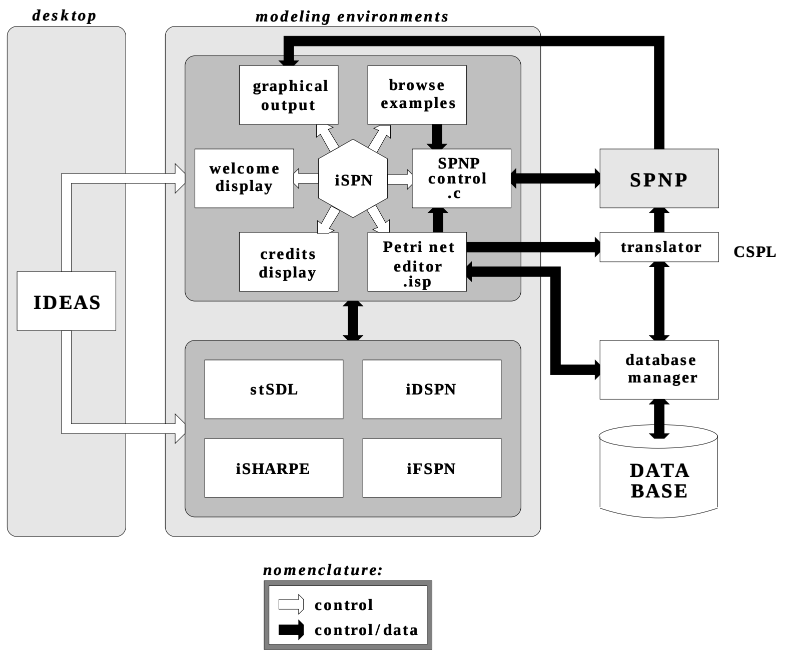

Figure 6: iSPN main software modules.

- *Improved Speed of Learning CSPL*: iSPN implements a higher level input format to CSPL. This first design is important in the integrated environment because it provides great flexibility to users. If the user feels more comfortable dealing with analytic modeling concepts such as Petri Nets for iSPN, he will be able to design the Petri Net directly rather than writing code in the specification language.
- *Improved Speed of Creating Petri Net*: The *Petri Net editor*, which is the window in iSPN where the user designs the input data, introduces a new way of programing SPNP: the user can draw the SRN model and establish all the necessary additional functions (like rewards rates and specialized output functions) through a common environment. The Petri Net editor provides several characteristics normally available only in good bi-dimensional graphical editors and a lot of new features designed specifically for the SPNP environment.
- *Graphical Database*: Another part of iSPN that helps improve the speed of creating Petri nets is a graphical database. This gives the user the unique ability to efficiently and selectively import a subnet from a database of existing

iSPN models into a Petri net which is being created.

  - *Display Existing CSPL as Petri Net*: Old files written in just CSPL can also be imported into iSPN and its graphical model will be created. The use of editors will diminish as the use of iSPN's graphical input increases because all files will be represented in an "isp" format.

- **Improved Graphical Output**: The main goal of most GUIs is only to facilitate the creation of input data for another software. Usually, the written communication between the software's output and the GUI is neglected. One of the advantages of IDEAS is the incorporation of displaying the software's results in the GUI application.

- **Graphing Capability**: 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 graphing applications, such as gnuplot or xvgr.

- **Easy Debugging**: 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 important in finding bugs, if they exist. This file is a reachibility graph and is displayed in a tree format as a part of iSPN. The debugging feature based on navigation on the reachability graph is an inovative function of iSPN that should provide improved efficiency in the development of stochastic Petri net models.

- **User Friendly**:

  - *Notebook*: The NoteBook widget, the primary interface of iSPN, is used to display many windows in a limited space. The notebook is divided into a stack of pages (windows). At one time, only one of these pages can be shown. The user can navigate through these pages by choosing the visual "tabs" at the top of the NoteBook widget. By using a Notebook, the number of pop-up menus are limited and the user is able to see all major windows easily.

  - *Different Editors*: iSPN allows the user to choose his/her own editor. In contrast, most other software impose a particular editor on the user, regardless of preference.
  - *Multi-lingual*: The user can dynamically choose the language (e.g., French, English, etc.) used for all text used in the application (i.e., labels, titles, menus).
  - *Library of Example Files*: 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.

- **Extensive Help Setup**:
  - Statusbar which always contains text relative to the operation currently being performed. Three different messages are displayed here with different colors: errors, warnings, and help messages.
  - Balloon help is used to show popped-up messages that describe the functions of the widgets in the application. When the user moves the cursor inside a widget to which Balloon help is available, a small pop-up window with a descriptive message is shown on the screen.
  - Mouse help is available as part of the Petri Net editor to help the user quickly create petri net designs and become familiar with the actions of each button of the mouse.
  - Hypertext help explains in detail each part and action of the software. Detailed help for an object can be automically displayed to the user in a seperate window.

5 Conclusions

Using off-the-shelf modeling tools, a GUI based environment is being developed which eases the specification of the modeling and analysis problem and uses one or more of the tools provided to solve the model. In our current prototype, using the philosophy of incremental design and implementation, we have restricted ourselves to integrating SPNP, SHARPE, D-SPNP and FSPN in a single environment and provide easy to use GUI based support to do model specification, editing, debugging, execution and result generation for dependability modeling.

The IDEAS-kernel project has met its initial objectives. The possibility of integrating diverse modeling tools through a common interpreter has been successfully demonstrated in the SDM environment [15]. A spin-off project of IDEAS named "iSPN: an integrated environment for SPNP" is nearly complete. The purpose of iSPN is the careful design and implementation of a GUI for SPNP, crafting recent advances in HCI into the design. The role of this project is central to IDEAS since it establishes a template to be followed in all subsequent GUI developments. The implementation of the IDEAS-kernel is now entering a new phase where the designed GUI will be adapted for other modeling tools and the integration experiment will be extended. The next main stage of the IDEAS-kernel project shall start in parallel with the evaluation of the iSPN prototype. In this second phase we will also start developing a GUI to the stochastic SDL language.

References

[1] M. Veran, D. Potier, "QNAP2: A Portable Environment for Queuing System Modeling", in: *Modeling Techniques and Tools for Computer Performance Evaluation*, Editor: D. Potier, North-Holland, pp.25–63, 1985.

[2] Chang, K.C., Gordon, R.F., Loewner, P.G., and MacNair, E.A., "The RESearch Queueing Package Modeling Environment (RESQME)," IBM Research Report RC-18687, Yorktown Heights, New York, February 1993.

[3] B. Plateau, J.-M. Fourneau, K.-H. Lee, “PEPS: A Package for Solving Complex Markov Models of Parallel Systems”, in: *Modeling Techniques and Tools for Computer Performance Evaluation*, Editors: D. Potier, R. Puigjaner, pp.291–305, 1990.

[4] C. Béounes et al., “SURF-2:A Program for Dependability Evaluation of Complex Hardware and Software Systems,” in *23rd IEEE Int. Symp. Fault-Tolerant Computing*, Toulouse, France, pp. 668–673, 1993.

[5] S.J. Bavuso, J. Bechta Dugan, K.S. Trivedi, E.M. Rothmann, W.E. Smith, “Analysis of Typical Fault-Tolerant Architectures using HARP”, *IEEE Transactions on Reliability* **36**(2), pp.176–185, 1987.

[6] A. Goyal, W.C. Carter, E. de Souza e Silva, S.S. Lavenberg, K.S. Trivedi, “The System Availability Estimator”, *Proceedings FTCS 16*, IEEE Computer Society Press, pp.84–89, 1986.

[7] R. Sahner, K. S. Trivedi and A. Puliafito, *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*, Kluwer Academic Publishers, 1995.

[8] 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.

[9] J.A.Couvillion, R. Freire, R. Johnson, W.D. Obal II, A. Qureshi, M. Rai, W.H. Sanders, J.E. Tvedt, “Performability Modeling with UltraSAN”, *IEEE Software*, pp.69–80, September 1991.

[10] C. Lindemann, R. German, “DSPNexpress: A software package for efficiently solving deterministic and stochastic petri nets”, in *Performance Tools 1992*, Eds. R. Pooley, J. Hillston, Edinburgh University Press Ltd., forthcoming, 1992.

[11] R. German, Ch. Kelling, A. Zimmermann, G. Hommel, “TimeNET: A Toolkit for Evaluating Non-Markovian Stochastic Petri-Nets,” *Perf. Eval.*, 24:69-87, 1995.

[12] K. S. Trivedi, B. Haverkort, A. Rindos and V. Mainkar, "Techniques and tools for reliability and performance evaluation: problems and perspectives", *Computer Performance Evaluation: Modeling Techniques and Tools,* (invited) Lecture Notes in Computer Science 794, G. Haring and G. Kotsis (eds.), Springer Verlag, pp. 1-24, 1994.

[13] G. Chiola, "A Graphical Petri Net Tool for Performance Analysis", in: *Modeling Techniques and Performance Evaluation*, Editors: S. Fdida, G. Pujolle, North-Holland, pp.323–333, 1987.

[14] S.C. Johnson, R.W. Butler, "Automated Generation of Reliability Models", *Proceedings of the 1988 Annual Reliability and Maintainability Symposium*, pp.17–22, 1988.

[15] A. V. Ramesh, K. S. Trivedi, A. K. Somani, D. W. Twigg, U. R. Sandadi and T. C. Sharma, "Integrated reliability modeling environment," *submitted for publication*, 1996.

[16] G. Ciardo, J. Gluckman and D. Nicol, "Distributed state space generation of discrete-state stochastic models", to appear in *ORSA Journal on Computing.*

[17] K. Trivedi and V. Kulkarni, "FSPNs: Fluid Stochastic Petri Nets," *Proc. 14th International Conference on Application and Theory of Petri Nets*, Chicago, Jun. 21-25 1993.

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

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

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

[21] H.R. Harston and D. Hix, "Towards empirically derived methodologies and tools for HCI development," *International Journal of Man-Machine Studies*, 31:477–494, 1989.

[22] M. Ajmone-Marsan, G. Conte, and G. Balbo. "A class of Generalized Stochastic Petri Nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems*, Vol. 2, No. 2, pp. 93-122, 1984.

[23] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi, "Fluid stochastic Petri nets: theory, application, and solution", Technical Report 96-5, *Institute for Computer Applications in Science and Engineering*, 1996.

[24] G. Ciardo, D. Nicol, and K. Trivedi, "Discrete Event Simulation of Fluid Stochastic Petri Nets," in preparation.

[25] John K. Ousterhout. *Tcl and the Tk Toolkit*, Addison-Wesley, Workingham, England, 1994.

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