A GENERAL PURPOSE NETWORK SIMULATION TOOL

Pascal ANELLI and Michel SOTO
Laboratoire MASI - CNRS UA 818
University Pierre et Marie Curie,
F75252 PARIS, FRANCE.

ABSTRACT:

This paper presents NESSY (NEtwork Simulation SYstem): a whole environment for performance evaluation of communication networks. The modeling methodology proposed by NESSY is mainly based upon the OO (Object-Oriented) techniques and the FDT (Formal Description Technique). With OO approach the network model is a collection of objects. FDT is used to describe the objects that model the network components. These objects are elementary models named ESO (Elementary Simulation Object). These two techniques are used separately through two phases: the modeling phase, where ESOs of a given domain of communication networks are written, and the building phase, where the complete model is built with instances of ESO. The main feature of ESO is to be reusable in different studies. The description of ESO is done according to ECFSM (Extended Communicating Finite State Machine) and with the dedicated LASSY modeling language. This language is derived from the ESTELLE FDT. During the building phase, the model is build by modular and hierarchical composition. The simulation model is not a compiled program but a run-time program. So, full flexibility and interactivity are achieved to control the simulation and to tune the network model.

1. INTRODUCTION

Communication networks significantly increase the processing capacities of computers. Distributed applications, like FTP (File Transfer Protocol), are being used in the administrations of most companies. The main requirement of these applications is that they must be reliable. As a consequence, computer networks must offer the highest QOS (Quality Of Service). To achieve and to maintain the required QOS, the network's managers must attend to the performance of the communication system.

Three general approaches (Kurose and Mouftah 1988) to evaluate the performances of a system are identified: measurement tools, analytical techniques and simulation techniques. Measurement consists of evaluating directly the network behavior, with the help of measurement tools or monitors (Terplan 1987). These tools support the activities of data collection, data reduction, and statistical analysis. The last two techniques are not performed on the real-system but on a model of the system. They are often called performance prediction techniques, as they reveal the performance of a new and non-operational system. Analytical techniques employ a system of equations; the mathematical resolution of these equations yields exact quantitative results. Simulation consists to use a computer to evaluate a model numerically, and data are gathered to estimate the desired true characteristics of the model (Law and Kelton 1991). The analytical techniques cover a very narrow range of utilization. As the model must stay simple, many details of the system are neglected and it becomes no realistic. Simulation allows to deal with a complex model. In communication networks, the models are often complex and require a long and tedious task owing to the size, diversity and complexity of communication systems. To set a new configuration or to detect design errors early, the network manager or system designer need tools for performance prediction that bring them an assistance in the way to master the complexity of his/her task. These tools are based upon the discrete event simulation (Leroudier and Parent 1976).

The constraints that evaluation performance tools must take into account, are classified in several categories. First, due to the high cost and long realization time of such tools, it is compulsory to use them on a wide range of communication networks and a wide range of problems. A performance evaluation tool must therefore exhibit general purpose properties. Second, they must offer a modeling methodology that be simple, natural and rapid. Classically in simulation, writing a model requires a programming expertise. This restricts the population potentially able to lead such a study. Simple modeling means the tool must hide the writing stage of the models to the final user. The modeling appears more "natural", if the network model is described according a visual representation. In this way, the analyst has not to translate from his/her perception of the communication network into a syntax of a textual language. The amount of time needed for development and analysis of a model must also be taken into account. This may be very important given the tight time constraints in many projects. If the duration of the performance evaluation stage is shorter and easy to predict, then it can be planed in the design cycle. Third, the problem of the model correctness increases when the complexity of the model grows. Fourth, the tool must be easy to use for the various steps that compose a typical, sound simulation study (Law and McComas 1991). Lastly, it is desirable the tool provides the analyst with a good interactivity during the simulation execution to make easy the understanding of some phenomena which can occur. Otherwise they would be lost in steady-state statistics.

The current simulation tools of communication networks follow, like many other softwares, an OO (Object-Oriented) approach. The model is constructed graphically by an object aggregation as presented by (Zeigler 1987). These objects are pulled out from a library that holds their description. The way used to describe these objects and the formal methods used constitute the main differences from these tools. For example, OPNET (MIL 3 Inc 1991) is based on graphic description of extended finite state machine and C language. RESQME (Gordon *et al.* 1991) has been designed for performance evaluation of resource contention systems such as communication networks. The model is graphically specified according to the queueing network paradigm developed in RESQ. With TOPNET (Marsan *et al.* 1990), the basic object is described graphically by a class of timed Petri nets named PROT nets. The script associated with the Petri nets transitions are written in

Ada. BONeS (LaRue *et al*. 1990) has developed a block oriented paradigm with the aim to be more general than the previous ones. The blocks are graphically assembled and primitives written in C language at the lowest level of abstraction.

This paper presents the simulation tool, NESSY (NEtwork Simulation SYstem), designed for performance evaluation of communication networks that takes into account the previously mentioned problems. This tool provides a modeling methodology so that network manager or system designer does not need to be experts on performance evaluation techniques. NESSY follows the trend taken by the current simulation tools in adopting an OO approach and in combining some concepts directly inspired from FDT (Formal Description Technique). However, NESSY goes further in the use of OO approach. All the aspects of the simulation model are represented by a collection of different object types and these objects do not confine themselves only to the modeling network.

This paper is organized as follows. An overview of NESSY is presented in Section 2. This section describes how the modeling methodology of NESSY is carried into effect. The main concepts of NESSY are discussed. Section 3 reviews the general architecture of NESSY and Section 4 presents a short description of the implementation of the tool.

2. OVERVIEW OF NESSY

2.1. Major Concepts of NESSY

NESSY is designed to analyze the performance of communication networks. It is a prediction tool based on the simulation of the communication system under study. It can be used in the different stages of the life cycle of a network: design, prototyping, enhancement and tuning. NESSY is easy to learn and user-friendly. An elaborate interface permits the control of the main functions of the tool and the network as well as the results are graphically displayed.

The use of NESSY is clearly separated into two phases, namely the modeling and the building phases. These phases may be led by different people. The first user is called the *modeler* and the second the *final user* (afterwards named user). The modeler is assumed to be a specialist both in the network and the performance evaluation techniques. The modeler is able to understand the exact behavior of a network component and is able to isolate the main characteristics of each component for modeling purpose. This empirical process is based on the experience and the practice of the modeler. He/she writes the *atomic models* and stores them in a library the second type of user can include in network models. The second user of NESSY is assumed to be unskilled and to have little knowledge of performance evaluation techniques, he/she is just interested in the program results but not the way they are obtained. This user is typically a network architect, a network manager, or a student. These types of users share one common characteristic: they have a good knowledge of the network. So, NESSY uses the most common concepts in communication networks area to offer well-understood objects to the user and to help him in building the network model. These objects are provided by the library of models written by the modeler. Therefore, a network model is easily constructed without requiring neither detailed modeling nor programming skills. The user of NESSY does not build the library; he/she only focuses on problems.

NESSY uses the network concepts and related functional structures to model the main components of the network. The structure of a network is vertically characterized by its architecture (e.g. OSI layers stacks) and horizontally by its topology (e.g. bus, point-to-point and ring). The topology is the description of elements at the same level of abstraction and the architecture is the hierarchy of the elements (i.e. the aggregation between simple elements to make a more complex element in the topology). The basic components are protocol layers and communication devices or media. These components are featured by exchanging information with each other. Data packets, service primitives, signals or software interrupts are many types of information which are exchanged. In NESSY, all the basic network components become atomic models named *Elementary Simulation Object* (ESO). The definition of an ESO defines a class stored in the aforementioned library. The network model built by the user is a collection of instances of these classes. Providing reusable ESOs permits a great variety of network models to be built with the same set of ESOs.

2.2. Modeling objects in NESSY

In this section, we present the basic objects used in our modeling methodology. We firsts focus on the framework of the ESO and then we focus on objects used to specify under what conditions a network model has to be studied.

2.2.1. Key element: ESO

The structure of the ESO has been designed to make ESOs reusable and homogenous. The ESO is composed of two parts: interface and behavior (i.e. code). The whole ESO structure is represented in figure 1. In this section, we only describe the interface part. The behavior part of the ESO is described in Section 2.3.

The interface is the visible part of the ESO (i.e. public to the user or the system). It gathers the communication capabilities of the ESO in direction of the user, from the user and with other ESOs. To communicate with the outside of the ESO, there are three types of objects in the interface: (1) the *observation points* and (2) the *action points* (3) *communication gates*. They are detailed in the following.

- Observation points

Observations of the system play an important role in performance evaluation. In a network model some phenomena represented by quantitative values have to be observed to elaborate performance results. Observation points are associated to typical phenomena which can be exhibited from the ESO. For example, assume an ESO which models the "CSMA/CD access technique" (ISO 1985) used on Ethernet-like networks. Interesting phenomena are number of attempts of transmission, successful transmissions, collisions of transmissions, failed transmissions and delay before a transmission to be successful. Observation points make these phenomena visible. They enable to user to compute results like response time or throughput. The way proposed by NESSY to user for computing performance results from observation points is given in Section 2.2.2.

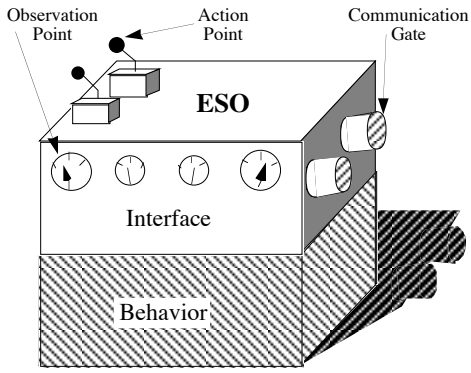

Fig. 1: ESO structure.

• Action points

ESOs have to fit exactly the user's requirements for a given network model. However, to be reusable, the ESO must own the property to be tailored for any network model. Tailoring an ESO is equivalent to set parameters of the ESO to adapt its behavior to a given network model. This is achieved in NESSY by action points which enable the user to set parameters of the ESO. For facilitating understanding, action points may be assimilated to parameters. For example, in the aforementioned ESO modeling the "CSMA/CD access technique", the action points are attempt-limit, backoff-limit, frame size the raw throughput, etc (i.e. all the standard parameters initialized with the default values from the standard). It is also interesting in some studies to modify the parameters' values throughout the simulation (e.g. frame size obeys a general distribution). This is also achieved by using action points.

• communications gates

The major feature of communication networks is information exchange. In the network model, two (or several) ESOs are connected for communication purpose if the corresponding real network components exchange information. In NESSY, the ESOs communicate *Exchanged Simulation Data Units* (ESDUs) to each other. ESDUs are objects that model the exchanged information between network components. Every ESO has at least one *communication gate* and ESOs are connected to each other by way of theirs communication gates. A full duplex *channel of communication* results from the connection of two communication gates. Several channels may exist between two ESOs. Instances of ESDUs are sent and received on a channel through communication gates.

2.2.2. Measurements and Scenarios

Experience in performance evaluation demonstrates the aims of a study are variable by essence. So, meeting the model's reusability requirement is an important matter. The specification of a study is derived from its aims. The questions to answer are: under what conditions the network model has to be simulated, what are the measurements to perform? As mentioned in the precedent section, ESOs can be tailored. This is not sufficient to make ESOs highly reusable. They must also be independent of the specification of any study. The meaning of independence is two-fold: when the objectives of the study change, no modification of the used ESOs is necessary; the way to specify a study is the same whatever the classes of ESO used in the network model. NESSY achieves this independence by clearly separating the network model and the specification of the study. Two classes of objects are involved in this specification: *measurement* and *scenario*. Utilization of these two object classes is given below.

• Measurement

A measurement is an object which produces an elementary calculation (e.g. sum, average, maximum, minimum, delay, observed value, count and throughput) from a sequence of sampled values. It enables the user to obtain a performance result. In order to be computed, a measurement must be clearly linked up to one or several observation points. In the same way, several measurements can be linked up to one observation point. This linking is under the control of the user who indicates to NESSY what are the measurements and the observation points to be linked.

In the behavior part of the ESO, every time a value is sampled, it is propagated to all the current measurements linked up to the proper observation point. Through this point, distinctive samples are supplied to the measurement.

• Scenario

A scenario is an object which provides the user with two types of action on the ESO: alteration of parameters' values and generation of ESDUs (e.g. for traffic generation purpose) on communication gates. This object enables the user to adjust precisely the ESO to his/her real needs and to submit it to the environmental conditions required by the study. To be effective, a scenario must be linked up to one action point or communication gate (i.e. communication gates are implicit action points). As for measurements, this linking is under the control of the user who indicates to NESSY what are the scenarios and the action points to be linked. A scenario is triggered on date or on event. So, conditional scenarios can thus be defined with event triggering.

In the previous sections we have defined the modeling objects of NESSY. The next section shows briefly how they are used by the modeler to design the interface of a class of ESO and focuses on description of the behavior part.

2.3. Modeling Phase

The modeler is able to understand the exact behavior of a network component and is able to isolate the main characteristics of each component for modeling purpose. Once, these characteristics are isolated, the modeler expresses them as observation points, action points and communication gates. Then, the behavior of the network component is modeled in the behavior part of the ESO.

We chose Extended Communicating Finite States Machine (ECFSM) to describe the behavior of the ESO. The reason of this choice is two-fold: components of distributed systems obey the stimuli/reactions principle and they communicate between them (i.e. protocol layer or physical unit when receiving external stimuli performs one action which may stimulate in turn another entity, and so on). This kind of model is widely used and it has a wide range of modeling capabilities.

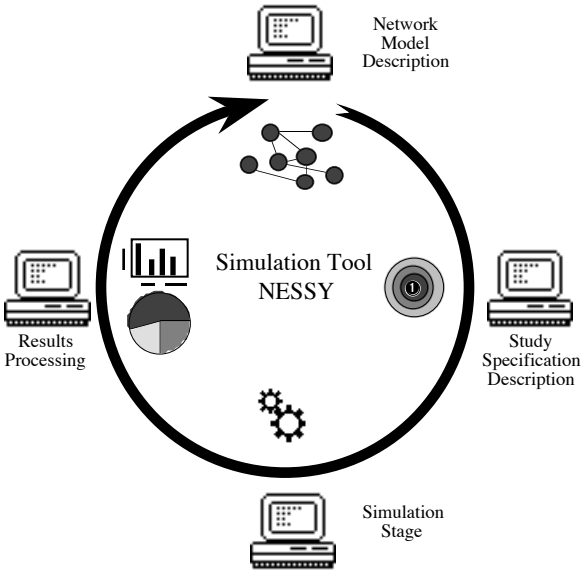

Fig. 2: The building phase.

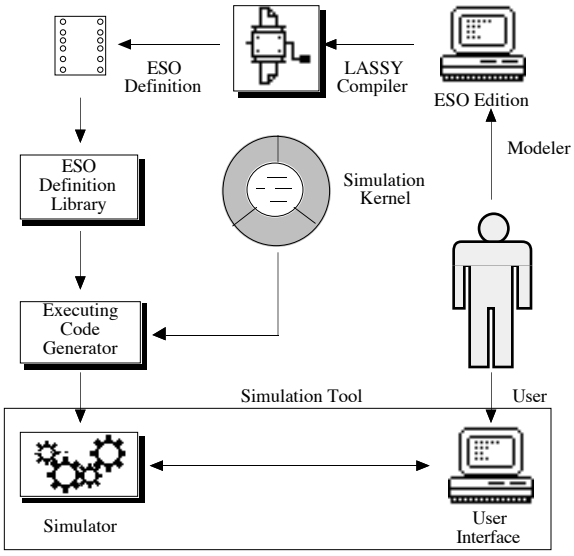

Fig. 3: NESSY environment.

LAnguage for Simulation SYstem (LASSY) has been developed for NESSY to implement ECFSM. LASSY is derived of the basic principles of ESTELLE, a FDT standardized by ISO based on ECFSM (ISO 1988). ESTELLE aims at specifying distributed, concurrent information processing systems like communication protocols and services, and removing ambiguities from ISO protocol specifications in natural language. LASSY is both a simplification and an extension of ESTELLE. The reasons for designing a specific language rather than using ESTELLE are developed in (Vèque 1988). For performance evaluation purpose, the language enables the modeler to specify quantitative description. For example, it provides primitives for time consumption and primitives for defining service time (e.g. constant or based on a pseudo-random number generator). LASSY has also the features of every high-level programming language. When the writing of the ESO is completed, it is compiled and store in the library of ESO classes. The next section presents the process followed by the user to carry out a study

2.4. Building Phase - How to Carry out a Study.

A network model has first to be constructed by the user with instances of the ESO classes. An instance is created by invocation of an ESO definition contained in the library. The user manipulates and assembles these instances to specify the network model. To achieve this task, the user's guidelines are the topology and the architecture of the network to be modeled. So, the description of the network model aims to copy the studied network. Such a description can be compared to the drawing process performed with a drawing software. Nevertheless, this drawing process may be inadequate for very large networks. A hierarchical approach is proposed by NESSY with the *Composable Simulation Object* (CSO). The CSO is mainly used to describe non elementary network components at a chosen level of abstraction. A CSO is an aggregation of ESOs and CSOs with a lower level of abstraction. This aggregation is then handled as a single entity.

When the network model is complete, the user specifies the objectives of the study with measurements and scenarios linked, respectively, to observation points and action points. The network model with the study specification is called the *simulation model*. Figure 2 illustrates the different stages of the building phase.

Finally, the user starts the simulation stage. This stage is not a '"tunnel process" where the model is put at one extremity and get back, solved, at the other extremity: it is an interactive process. At any time, the user can change, add or remove any measurement or scenario. In the same way, current values of measurement can be displayed at any moment. This is useful to feel how the model behaves before solving it in the steady state or when studying transient phenomena. Then, the performance results can be processed to satisfy different presentations (e.g. histogram and graph).

We have presented the modeling methodology provided by NESSY. In the next section, we present the architecture of NESSY to support this methodology.

3. NESSY ARCHITECTURE

NESSY is a complete environment dedicated to the performance evaluation. It is composed of several functional modules: the ESO definition library, the LASSY compiler, the simulation kernel, the executing code generator, the simulator and the user-interface. The figure 3 shows the NESSY environment with the different modules and the user duality. The composition of the simulation tool for the user also appears on this figure.

- ESO Definition Library

The library of ESO definitions is one essential module of the tool. It forms the database of NESSY to construct a network model. It gathers all the models written by the modeler and used by the user. It is the transcription of the modeler's knowledge about the modeling of network components. The library is the link between the modeler and the user. Thanks to this module the aim, to provide a tool for an unskilled user in modeling, is achieved. Without it, the simulation tool is as an "empty shell" and the modeling phase consists in filling up that shell. The library contents will influence directly the modeling capabilities of tool. So, a particular attention must be given it to its accomplishment.

• LASSY Compiler

A peculiar module realizes the compilation of the ESO written in LASSY. In the code generation phase, the compiler produces an object code in Modula-2 programing language. The LASSY compiler is completely separate from the simulation tool. It is not used by the same type of user; the compiler is used by the modeler whereas the rest of NESSY is for the user.

• Simulation Kernel

The simulation kernel performs functions very similar to those offered by a scheduler in an operating system; that is, process activation, processor sharing, context switching, interfacing with other modules of the system. In addition to these functions, specific problems of simulation and of performance evaluation are added: virtual time management, event scheduling, measurements computation. And more, the simulation kernel has to implement the modeling object of NESSY and to make the communications of ESO with the outside possible. The separation between the ESO definition library and the simulation kernel give a high flexibility at each other. The library is a module that can be change according to a domain-specific modeling whereas the simulation kernel is more stable and changes only for new releases.

• Executing Code Generator - Simulator

The executing code generator has got job of generating the simulator from a given library and from the simulation kernel. This module does just a computer processing. It compiles the library by the Modula-2 compiler and then links it with the simulation kernel. The result is an executable program named simulator. It is the actual application. After the user has constructed his/her model, the simulator does not make a file because the model is constructed dynamically on-line. Like this, the simulation model is not frozen during its simulation. It can be still modified.

• User Interface

The interface is used for performing the different tasks of a study performance. It provides the user the means to interact with the simulator, offers features which make the description of simulation model easy and bring some "comfort" in the utilization of NESSY. During the specification of the simulation network, the user interface ensures the dialogue with the user and the graphic displays of simulation model abstractions. This dialog is made as easy as possible and user-friendly. The interaction style is based upon the direct manipulation (Shneiderman 1983) with the metaphor of the communication networks area. Thanks to the interface, the user is able to describe, in a graphic way, his/her real network by a collection of icons representing the network components. Like this, the visual representation of the user is mapped on his/her mental representation. The scenarios and the measurements are also displayed by icons. They are linked by the users with the icons of the different network components. The interface allows the user to control interactively the progress of the simulation. However, its role does not stop here, it also presents to the user the simulation results under a comprehensible form.

The separation between specific functions about one domain (i.e. application) and the mechanics of display (i.e. interface) is a fundamental concept now accepted by everybody. This separation is a criterion of a well-structured simulation tool architecture. Each module can evolve to be adapted to new needs without rethinking logic of the other module.

4. IMPLEMENTATION ISSUE

NESSY is implemented in a UNIX environment. The user interface is based upon the X-Window system and has been developed with the interface generator Aïda-Masaï (Ing 1991). The compiler was implemented with the help of the YACC tool (Johnson 1978) which permits the generation of a compiler from a description of a LR1 grammar. The simulation kernel was developed with the Modula-2 programming language (Wirth 1985). The aim of this choice is two-fold: the kernel of NESSY must control the parallel execution of the ESOs (thus structure of control for parallelism is needed) and the kernel must be portable. This advanced language permits to achieve this double aim. Transporting the kernel of NESSY is minimized because it is sufficient to compile it on the target machine. The Modula-2 language allows to create and to manage co-routines in such a manner that we can control the execution in a pseudo-parallel way. This choice was reinforced by (Ullrich and Cummins 1990) who demonstrate coroutines are well adapted to develop discrete-event simulation software. Moreover in simulation, execution time is essential. Modula-2, to a simulation language such as SIMSCRIPT, allows to decrease the simulation time from 20 to 60 % depending on the models (L'Ecuyer and Giroux 1987). Finally, the Modula-2 language is a good support to apply the main notion of software engineering: the modularity. This language also provides a facility to program in an object-based style (Wegner 1989). This style fits well to the kernel because it was designed with an OO technique. The details of the implementation of the tool are described in (Anelli 1992).

5. SUMMARY

NESSY is a new tool for modeling and simulating communication networks. Due its architecture and the proposed modeling methodology, this tool is general (i.e. modeling a large range of communications networks), easy to use and interactive. NESSY provides a modeling language especially designed for the communication networks derived from the ESTELLE FDT and a simulation model based on OO technique. The modeling methodology is two-phased: modeling network components and building simulation models. These two phases are done to permit to have user of different level. The modeler accesses methods for writing new models of network

components. The final user just uses these models to build a complete simulation model.

The modeling phase is used to write the Elementary Simulation Objects (ESO). These objects communicate by messages and can model all kinds of network components. They have the properties to be reusable and modular. The reusability property is needed to use models that have produced correct results in previous simulation studies. The ESO is described with the ECFSM abstract model and written in LASSY. With a such language the modeling work can be significantly accelerated by deriving models from existing ESTELLE specification. LASSY adopts the syntax of Modula-2 and includes special primitives for simulation (i.e. quantitative description of time and random generation number). Finally, the ESO is compiled and stored in a library.

The building phase enables the user to construct the simulation model by a simple and natural modeling methodology that consists to describe the elements in the real-world and to assemble them in a modular way. This methodology appears natural because the user draws a model corresponding to his/her mental representation of the real-world. This drawing technique integrates the notion of abstraction with respect to the human way of thinking. The CSO provides abstraction levels in the network model to handle many ESOs as single element. All the aspects of the simulation model are separated. Each aspect is represented by an object and not confined to the network model. The network model is a collection of ESO and CSO and the study specification is set of measurements and scenarios. The measurements are the object classes for the computation of performance results. The scenarios are object classes for the description of the conditions of the study (e.g. traffic load) of network model. These two kinds of object classes allow to perform any type of study. The distinction between network model and the specification of the study increases reusability of the network model. The same network model may be used with different objectives. The separation between measurement, scenario and ESO is a very important feature. It makes the ESO independent of a particular study and, thus, reusable. As an additional advantage, the measurement and the scenario definition can be changed interactively without modifying the ESO definition.

The result of the building phase (i.e. the final simulation model) is not a compiled program as in the other tools but a run-time program that dynamically creates the needed objects. This feature gives to the model the capability to be flexible; that is, it can still be modified during all the phases of evaluation performance study. This feature increases also the feed-back speed between the simulation and the description of the model and reduces the tuning time.

ACKNOWLEDGEMENTS

The authors would like to thank the peoples that have worked in the design of the tool especially Eric Horlait and Veronique Vèque.

REFERENCES

Anelli, P. 1992. "Computer Network Performance: Toward an User Interface for NESSY Simulator into an Object-Oriented Environment." Ph.D. Paris VI (In French). 4, Place Jussieu. 75252 Paris Cedex 05. FRANCE. (Jun.).

Gordon, K.J.; J.F. Kurose; R.F. Gordon; and E.A. MacNair. 1991. "An Extensible Visual Environnement for Construction and Analysis of Hierarchically-Structured Models of Resource Contention Systems." *Management Science* 37, no. 6 (Jan.): 714-732.

Ing, B. 1991. "Aïda et Masaï: Deux Outils de Génie Logiciel pour les Interfaces Graphiques." *Génie Logiciel & Systèmes Experts*, no. 24 (Sept.): 82-88.

ISO. 1988. *ESTELLE: a Formal Description Technique Based on an Extended State Transition Model*. IS 9074. ISO/TC97/SC21/WG16-1, (Nov.).

ISO. 1985. *Carrier Sense Multiple Access With Collision Detection*. DIS 8802.3.

Johnson, S.C. 1978. "Yacc: Yet Another Compiler-Compiler." Bell Laboratories, Murray Hill, New Jersey 07974.

Kurose, J.F.; and H.T. Mouftah. 1988. "Computer-Aided Modeling, Analysis, and Design of Communication Networks." *Journal on Selected Areas in Communications* 6, no. 1 (Jan.): 130-145.

L'Ecuyer, P.; and N. Giroux. 1987. "A Process-Oriented Simulation Package Based on Modula-2." In *Proceedings of the 1987 Winter Simulation Conference* (Atlanta, GA, Dec. 14-16). IEEE, Piscataway, N.J., 165-173.

LaRue, W.W.; E. Komp; S. Schaffer; V.S. Frost; K.S. Shanmugan; and D. Reznik. 1990. "A Block Oriented Paradigm for Modeling Communications Networks." In *MILCOM'90. A New Era. IEEE Military Communications Conference*. (Monterey, CA, 30 Sept.-3 Oct.). IEEE, New York, N.Y.; Armed Forces Commun. Electron. Assoc.; U.S. Dept. Defence, 689-695.

Law, A.M.; and W.D. Kelton. 1991. *Simulation Modeling and Analysis*. McGraw Hill Book Company, New York, N.Y.

Law, A.M.; and M.G. McComas. 1991. "Secrets of Successful Simulation Studies." In *Proceedings of the 1991 Winter Simulation Conference* (Phoenix, Arizona, Dec. 8-11). IEEE, Piscataway, N.J., 21-27.

Marsan, A.M.; G. Balbo; G. Bruno; and F. Neri. 1990. "TOPNET: A tool for the Visual Simulation of Communication Networks." *IEEE Journal on Selected Areas in Communications* 8, no. 9 (Dec.): 1735-1747.

MIL 3 Inc. 1991. "OPNET: OPtimised Network Engineering Tool." The INTELSAT Building. 3400 International Drive, N.W., Washington, D.C. 20008.

Shneiderman, B. 1983. "Direct Manipulation: a Step Beyond Programming Languages." *Computer* 16, no. 8 (Aug.): 57-69.

Terplan, K. 1987. *Communication Networks Management*. Prentice Hall International, Englewood Cliffs, N.J.

Ullrich, J.R.; and D.E. Cummins. 1990. "Message passing, Discrete Event Simulation Using Modula-2 Processes." In *Proceedings of the SCS Multiconference on Object Oriented Simulation* (San Diego, CA, Jan. 17-19). SCS, San Diego, C.A., 109-112.

Vèque, V. 1988. "Networks Performance: a Tool Based on the Simulation." Ph.D. Paris VI (In French). 4, Place Jussieu. 75252 Paris Cedex 05. France. (Dec.).

Wegner, P. 1989. "Learning the language." *Byte* 14, no. 3 (Mar.): 245-253.

Wirth, N. 1985. *Programming in Modula-2*. Springer-Verlag, New York, N.Y.

Zeigler, B.P. 1987. "Hierarchical Modular Discret-Event Modeling in an Object-Oriented Environment." *Simulation* 49, no. 5 (Nov.): 219-230.

AUTHORS, BIOGRAPHY

Pascal ANELLI was born in Paris, France in 1963. He received the Ph.D specializing in computer science from the University Pierre et Marie Curie (Paris 6), France in 1992. He is currently Assistant Professor at University of Pierre et Marie Curie. Since 1989, he leads his research into the MASI Laboratory (CNRS-UA 818). His research interests are in the area of networks simulator for performance evaluation and the modeling techniques.

Michel SOTO was born in Montpellier, France in 1962. He received the Doctorat from Pierre et Marie Curie University (PhD), Paris in 1990. He is currently Associte Professor at University of Pierre et Marie Curie. His research is in the area of networks, high speed protocols, and tool development for performance evaluation of computer networks.

AUTHORS, ADDRESS

Pascal Anelli

Laboratoire MASI
Université Pierre & Marie Curie
4, place Jussieu
75252 PARIS Cedex 05
FRANCE

Tel. : 33-1-44-27-71-29
Fax : 33-1-44-27-62-86
E-mail: `anelli@masi.ibp.fr`

Michel Soto

Laboratoire MASI
Université Pierre & Marie Curie
4, place Jussieu
75252 PARIS Cedex 05
FRANCE

Tel. : 33-1-44-27-71-29
Fax : 33-1-44-27-62-86
E-mail: `soto@masi.ibp.fr`