NESSY: AN OBJECT-ORIENTED NETWORK SIMULATION TOOL

Michel Soto and Pascal Anelli
Laboratoire MASI - CNRS UA 818
University Pierre et Marie Curie
F75252 Paris, FRANCE
E-mail: soto@masi.ibp.fr

ABSTRACT

This paper describes NESSY (NEtwork Simulation SYstem). This software tool aims to deal with the main problems in the area of performance evaluation simulation tool for communication networks. The tool provides an object oriented methodology in order to represent network components. The assembly of these components is then used for building a network model. Furthermore, the tool relies on a formal model to describe these basic components. They are described with a dedicated language derived from the ESTELLE FDT (Formal Description Technique). These descriptions form objects which have good properties to model all kinds of network components, to be reusable and modular. The final simulation model is made by composition of instances of various object classes. One peculiarity of this tool is the full flexibility between the simulation and the model to reduce the prototyping time. This is of great interest for example when transient phenomena are under study.

1. INTRODUCTION

Digital transmission systems and communication networks are complex and varied. It is a long and difficult task to develop a model for performance evaluation purpose of such systems. This task needs a tool that makes easier for the analyst the modeling and the understanding of the system behavior. Such a tool must neither be dedicated to a specialized area nor solve a particular class of problems: this should be opposite to the growing diversity of communication networks. In addition, due to the high cost and long realization time of this tool, it is compulsory to design it on efficient basis with evolution capabilities in middle and long term. A performance evaluation tool for communication network must therefore exhibit general purpose properties. Basically, in simulation, model writing demands a programming expertise. This restricts the population potentially able to lead such a study. Consequently, hiding the writing stage of the models to the final user is a requirement for the tool. 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. Then, performance evaluation stage can be planed in the design cycle because its duration becomes shorter and more easily predictable. However, there is a problem using the digital simulation as a solving method: the model correctness. Without formal proving software, a partial solution is in code reusability. Modularity is the key concept of code reusability. As a consequence, a simulation model must be constructed in modular manner using composition from other models already validated by previous studies. The model reusability is also a strongly recommended feature to satisfy the rapid prototyping constraint.

Dahl and Nygaard designed the Simula language 27 years ago (Dahl and Nygaard 1966). They introduced many concepts taken up again by every contemporary object-based and object-oriented programming language. The aim of Simula language is to describe directly the systems with a simulation point of view. Since the definition of Simula, the link between object oriented techniques and simulation is obvious. Both they derive their principles from the real world. The advantages and drawbacks of Object Oriented Simulation (OOS) are discussed in (Roberts and Heim 1988). Nowadays, it is obvious that a simulation software must be designed with object oriented concepts. In fact, the current simulation tools use, like many other softwares, an Object-Oriented (OO) approach. This process has important advantages. It allows the modeling by hierarchical and modular composition: a model is an object made up by a set of other objects only seen through their interfaces (Zeigler 1987). When hiding internal details of a model, a high level of abstraction is reached. So, this modeling technique helps handling large models. The OO approach increases also re-utilization of models in different studies and makes easier the constitution of "ready to use" library of models. Moreover, the hierarchical structure of real systems can be translated in the model. An object may be composed with simpler objects easier to understand even by a user without any notion in modeling. On the other hand, a skilled user in modeling will access to the methods of the models and will create new components. The limit of utilization of the tool closely depends on the basic object used for building of the network models. This object must allow the modeling of a wide range of real network components and must be used in any combination of objects. In this way, the utilization of the tool is neither limited by the diversity and individual complexity of network components nor limited by the arbitrary composition of objects. Finally, OO approach promotes high quality user interface: objects and methods are well fitted to icon representation and interrogation by dialogue boxes. Thus the network model can be described according a visual representation. In this way, the user has no need to translate from his/her perception of the communication network into a syntax of a textual language. The modeling technique appears therefore more 'natural'.

Tools like OPNET (MIL 3 Inc 1991), RESQME (Gordon *et al.* 1991), TOPNET (Marsan *et al.* 1990) and BONeS (LaRue *et al.* 1990) mainly differ from each other in the formal methods and the way used to describe the internal operations of this basic object. OPNET is based on graphic description of extended finite state machine and C language. RESQME 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, the basic object is

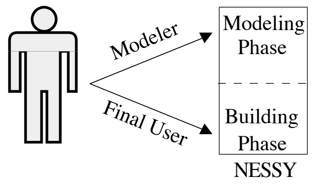

Fig. 1. The two phases of NESSY.

described graphically by a class of timed Petri nets named PROT nets. The script associated with the Petri nets transitions is written in Ada. BONeS 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 at the lowest level of abstraction.

The problem is how to use OO concepts to provide a general purpose tool, a descriptive modeling process and a flexible simulation model? This paper briefly describes the NEtwork Simulation SYstem (NESSY): an environment for building a communication network model solved by simulation. In this paper, we will focus on the modeling methodology and the application of the object oriented techniques used to get a general purpose performance evaluation tool for communication networks. The objectives assigned to NESSY are: (1) to save the final user from the modeling phase; (2) to provide a tool not limited to a specific field of the communication networks; (3) to keep the prototyping time as short as possible; (4) to maintain the correctness of the models.

2. OVERVIEW OF NESSY

The utilization of NESSY is two-phased: the modeling phase where models called *atomic models* are designed and written, and the building phase where simulation model is constructed and resolved. These two phases are clearly separated in time and can be carried out by two distinct people: the *modeler* and the *final user* (only named user in the following), as shown in Fig. 1. The modeler is a specialist in modeling techniques related to computer networks. To improve the writing time, NESSY provides a language specially designed to write atomic model that namely integrates communication capability, extended finite state machine paradigm and time consumption. The user has a good knowledge in computer networks but none in modeling and programming techniques. The absence of programming during the building phase makes easier to the user the utilization of the tool and increases the variety of potential users. This two-phased working permits to save the final user from the modeling phase: the remaining task of the user is to select graphically atomic models and describes the existing links between them. High interactivity is given to the user. During the simulation stage, the user can interfere at any moment to change parameters of the model or to add/remove measurements. No compiling of the models is required after these changes. This capability is useful when the user is working in a learning mode or when he/she is tuning his/her model.

One basic idea of NESSY is to find a unit to construct the model which be stable and highly reusable. Stability means independence between the code of this unit and the conditions of its use (i.e. the specification of the study objective). The experience has shown that the evaluation studies, specially in simulation, are always in evolution by essence. A study can often lead to another study. So, the objective of the study must be separated from the network modeling. To help the user to handle huge network model, the model unit must also integrate abstraction and hierarchical notions in order to map the network model on the user's conceptual model (i.e. the mental representation) of the real network. This unit (e.g. network component like protocol, link and node) is atomic or composable. Thus NESSY offers an atomic unit: the *Elementary Simulation Object* (ESO), and another composable one: the *Composable Simulation Object* (CSO). The detailed description of ESO and CSO is given in next sections.

2.1. Modeling phase

A network component may be viewed at different abstraction levels. For example, a data link may be viewed as an elementary component or may be seen as a more complex component made of one medium and two modems. The modeler designs a model at the lowest useful abstraction level for each component that may be encountered in computer networks. Such a model is call atomic model because it represents the lowest abstraction level of a network component. Once an atomic model is written with the dedicated *LASSY* (LAnguage for the Simulation SYstem) language it becomes in NESSY a *ESO definition*.

It is important to distinguish the ESO definition from the ESO instance. In term OO, one ESO definition describes one ESO class (Khoshafian and Abnous 1990). One ESO instance is an invocation of its corresponding ESO class. Only ESO instances exist in the network model. For example, the ESO definition of "send and wait" protocol describes the ESO class "send and wait". A "send and wait" protocol in the real network is modeled into the network model by an instance with suitable parameters of the ESO class of "send and wait". The approach of NESSY is object-oriented in sense each object definition describes a class. To simplify the presentation, the term class will be omitted. ESO has the property to be a module that may be used in a large number of network model due to a multitude of schemes. It is a shielded and encapsulated entity like a "black box". An ESO is a concurrently executable object into the network model. To respect the principle of the hidden information, ESO has two parts: a public interface part and a private behavior part (i.e. invisible out of the ESO). The interface part specifies the communication features of an ESO. Two types of communication are used: communication with others ESOs and with the "outside" of the network model.

An ESO communicates with others by data units exchanges as a real component network does. An exchanged data between ESOs is called *Exchanged Simulation Data Unit* (ESDU) and can include some fields of the real data unit that it models. ESDUs can go out and can come in through input-output port named communication gate. Communication gate is the "socket" used to connect the ESO and to enable it to communicate with other ESOs. The definition of a communication gate specifies what types of ESDU are send and received. In the real network, a functional link stands for a flow of data units. This is constructed in the model by a connection of communication

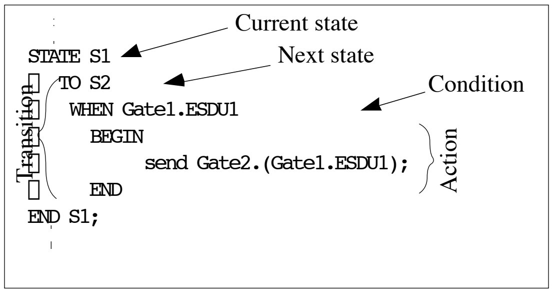

Fig. 2. ECFSM in LASSY.

gate. Thus, the network model is a set of concurrent objects that communicate between them.

The communication with the "outside" defines what it is possible to observe, what information can be get from the ESO and what parameters can be set. This communication is based upon the two types of objects: *action point* and *observation point*. An action point defines external parameter of the ESO. Thus, it permits to the user to customize the ESO and to specify under what conditions he wants to use the model. It may be modified from the outside and consequently modifies the behavior of the ESO. An observation point allows to observe from the outside how the ESO behaves in the network and especially how it reacts under the working conditions it is submitted (i.e. with the chosen parameters). More precisely, the observation point shows a value or the sequence of values which reflects in meaningful way the characteristic behavior or the state of the ESO. This object class is essential to get results. The choice of the number and of the usefulness of the action and observation points is made when the ESO is written during the modeling phase. By separating formally and clearly all the aspects of the ESO communication, the interface part is explicit and well-defined. Basically, an ESO is an integral and autonomous entity. Thanks to that and with the possibility to be parametrized the ESO has the property to be reusable.

The behavior part describes how the ESO reacts when events occur (i.e. when a timer expires or when an ESDU arrives). The description of the behavior is an Extended Communicating Finite State Machine (ECFSM). The choice of ECFSM is two-fold: in a distributed system, components obey the stimuli/reaction principle and communicate between them (e.g. protocol entity or physical unit when receiving stimuli performs one action which may stimulate in turn another entity and so on). The use of ECFSM paradigm leads to homogeneity in the development of different classes of ESO and allows to model any kind of network component.

LASSY modeling language

No existing general purpose or simulation language can be directly applied to the development of an ESO (Vèque 1988). Indeed, the FDT (Formal Description Technique) is most adapted to the description of an ESO, since they contain mechanisms to express communication, and some are based on similar formal models. However, they are not designed for performance evaluation. This is why LASSY is inspired by the FDT and in particular by ESTELLE (ISO 1988). ESTELLE is standardized by ISO and aims at specifying distributed, concurrent information processing systems like communication protocols and services. It also removes ambiguities from ISO protocol specifications in natural language. ESTELLE is a Pascal-like language, with in addition, the notions related to ECFSM (e.g. state, condition and transition).

Though LASSY takes from ESTELLE the structure to describe a ECFSM as shown in Fig. 2, it is both a simplification and an extension. LASSY does not need the semantics of parallel execution (asynchronous, synchronous). Conversely, the language must permit to specify quantitative time description as, for instance, time consumption. LASSY also adopts the syntax of the Modula-2 programming language (Wirth 1985) (successor of Pascal programming language) in order to possess features of a high-level programming language: structured types of data, control structures, functions and procedures. The LASSY language is used to facilitate and quicken the ESO's and ESDU's definition. The advantages brought by LASSY in the modeling process fits into its adaptation to describe the network component. Its structure expresses the encapsulation and its syntax is easy to master because it is well-known.

To write an ESO definition, the modeler has to study the accurate behavior of the corresponding network component. For example, for a communication protocol, it consists in studying the standard and then, to read the papers which describe performance studies on this particular protocol. It allows especially the definition of the interesting action points, and what kinds of results are to be computed from this ESO. Then the modeler has to describe the model in the form of an ECFSM and to translate it into the LASSY language. The next step consists in compiling the ESO. When the ESO is completely tested, the modeler stores it in the ESO definition library.

2.2. Building phase

The building phase enables the user to construct the simulation model from network model and to specify the study. Then this model must be solved by simulation. The modeling methodology strongly separates the construction of the network model from the specification of the study objective. This point is essential to make the network model reusable in different studies.

The modeling of network is performed by the user in a descriptive way. The user does not need to program. He/she does a simple stocktaking of network components and relationships between them. This stage comes to draw the graph symbolizing the organization of the studied network. The network model is build by invoking ESO definition from the ESO definition library and by connecting the ESO instances. In fact, ESO definition is a template which must be instantiated. Thus, there is as many instances as invocations. As in a real network, a network model must cover two sights, the topology and the architecture. The network topology represents the nodes and the communication links. The network architecture describes the nodes structure such as protocol stacks. This latter sight permits to take into account the notion of the hierarchy. In order to let the user construct his/her network model hierarchically. NESSY offers a second unit: the CSO. This object represents a network component which is not elementary for a given level of abstraction. The CSO is mainly used to describe the architecture of a network component which the model is not in the library. It is composed by aggregation of ESO and others CSO with a lower level of abstraction. Actually, this form of aggregation is made by the connection of communication gates. The result gives a new object enables to be handled as one entity. As ESO, the CSO has an interface part makes up by the set of the interfaces of the aggregated objects and a hidden part but which the structure is visible only when the user decreases of one level of abstraction. Finally, this unit allows to help the user to manage the complexity of the model of large network by changing the level of abstraction and creating new network component model with those which already exist. A model of network component can be either an ESO or a CSO. In the latter case, it is build up from one or more simulation object (i.e. ESO or CSO). This modeling form is called hierarchical, modular composition whose concepts are described in (Zeigler 1987).

In order to observe the behavior of the simulation network during the simulation stage, some results have to be computed. This computing mechanism both uses a build-in object class named *measurement* and the observation point. A measurement is an elementary computation like, for instance, sum, average, delay, observed value, throughput, etc. To be active, a measurement must be linked with one or several observation points. The values appearing on the observation point are samples feeding the measurement computation. Sampling is asynchronous; indeed, it is done every time state of the observed object changes.

The observation points are located on the ESO, as previously stated, on the ESDU and on the measurement. The ESDUs model information which travels into the network, they are information source of first importance and it is for that, which ESDUs have build-in observation points already defined by NESSY. A measurement can be associated with a single instance of ESDU or with all the instances of a class of ESDU (i.e. the measurement is related to one class of ESDU). Thus, the measurement allows to study the ESDU individually and collectively. The measurement's observation point gives it the property of composability. With elementary measurement provided by NESSY, by composition, it is possible to make up it more complex. The measurement composable takes again the idea of the modular composability applied at the network modeling. This idea consists of reusing objects to form other ones objects more complex and adapted to a particular need.

It is up to the user to add, to delete, or to modify the measurements wherever he wishes in the network model. The user does not explicitly code the measurements, but just chooses one measurement class among those proposed by NESSY in order to create one instance by invocation. The modeler specifies in the ESO code the sampling times of the observation point within a call to a sampling primitive. This primitive permits to apply a value to the sample. The sampling time for observation point on the ESDU and measurement are defined by NESSY. In short, the modeler has the role in deciding the observation points of ESO and theirs sampling times. The end user just has to choose and link measurements with observation points.

A network model and the description of the results extracted are not enough to lead a performance evaluation, the working constraints must also be set. This assumes two sights: (1) modeling or, more exactly, characterizing environment of network (e.g. the bursty traffic, rate of transfer errors.) and (2) acting on the ESO behavior during the simulation stage (e. g. size of anticipating window flux control, execution time and throughput) or, simply, adapting the ESO working parameters compared to the network component modeled. This work is carried out by the *scenario*. A scenario is an object which specifies an action on an ESO. Two kinds of actions are possible: setting parameters, and creating ESDU. The first action allows to adjust the parameter values of ESO. The second action is used to generate traffic. Scenario interventions are triggered off on date or on event. In this latter case, the scenario becomes conditional in the sense its action is synchronized with an event captured on an observation point. Actions triggered off on date permits to schedule the scenario interventions during the simulation unfolding. The time Δ, between each intervention, can be a constant or can follow a specific probability distribution. The scenario can intervene at a given instant (one-off) or periodically. Like the measurement, the scenario acts on the ESO's interface. A setting parameter action act on the action point and creating ESDU action sends ESDU to the communication gate. To act, a scenario must be clearly attached, by the user, with ESO interface part. In the simulation, scenarios are concurrent with each other and with ESO. The scenario is a built-in class, which one of its instance is created by invocation. The role of the user is to specify the working condition with scenarios, to attach them on the ESO interfaces and to put their suitable parameters.

The set of all the measurements and scenarios are named *study objective*. The simulation model is both the network model and the study objective. A complete example of use of NESSY can be found in (Vèque *et al.* 1991).

3. TOOL STRUCTURE

NESSY is made up of several modules which appear in Fig 3. An ESO definition is written with any text editor, compiled by the LASSY compiler and the result is stored in the ESO definition library. This module is the link between the modeling and building phase. Without it, NESSY is like an "empty shell". The separation between the library and the simulation kernel give a high flexibility at each other. The library enables NESSY to be customized for domain-specific modeling without any changes of the simulation kernel. The executing code generator makes a simulation program from the ESO definition library and from the simulation kernel. This module includes the measurement and scenario classes and necessary structures to lead a simulation phase. Then the building phase can be carried out. The user performs the different stages of his evaluation performance task through a graphical interface in order to make NESSY an easy to use tool. The detailed interface specification and simulator architecture are describes in (Anelli 1992).

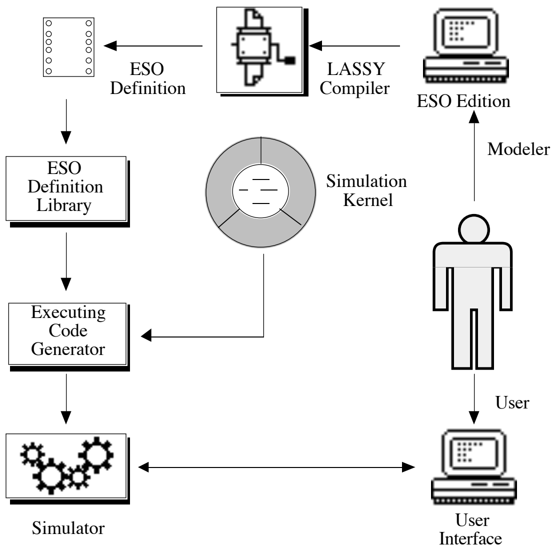

Fig. 3. Environment of NESSY.

4. OTHERS ASPECTS OF NESSY FRAMEWORK

The tool is written in the Modula-2 programming language. The advantages of this language for process-oriented discrete event simulation can be found in (L'Ecuyer and Giroux 1987) (e.g the concurrent programming mechanism based on the concept of coroutine). The simulation kernel is constructed in a message passing style as shown by (Ullrich and Cummins 1990). This style is well adapted for process-oriented simulation. Each active object in the network (i.e. ESO) is represented by a process and implemented as coroutine. A process holds its private data and describes the sequence of actions it experiences throughout its life. The process approach of simulation is recognized as a natural way to model a system (Law and Kelton 1991). Moreover, the tool is developed with a style based on objects (Wegner 1989).

The user handles objects to describe his simulation model. The modeler represents a component network by an object. The tool is programmed in an object approach. From utilization to programming, NESSY uses the object techniques and a symbiosis is get. In this way, the simulation model becomes highly interactive and flexible. Once the simulation model (i.e. both the network model and the study objective) definition is complete. The model is ready for the simulation. All the components are created and consistency checking are made on-line. Indeed, 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 which dynamically creates the needed objects. Thus, the user can stop the simulation, to change his simulation model and to execute it again immediately without losing the current state of its model.

5. SUMMARY

NESSY is a tool for modeling and simulation of communications networks. It allows to design easily and to handle large and complex network models. ESO is the basic object of NESSY for building a network model. It is written with to dedicated language LASSY. This language, derived from ESTELLE FDT, gives to the ESO a wide range modeling capability in communication networks and distributed systems. The proposed object-oriented and hierarchical modeling methodology plus the wide range modeling capability of the ESO provide a extensible and powerful tool not limited to any specific field of the computer communication networks.

The two separated parts inside the ESO: interface and hidden behavior, make ESO reusable, easier to understand and to set up. Moreover, NESSY reinforces ESO reusability as flexibility by fully separating measurements, scenario and ESO definition. The reusability of ESO and the high interactivity provided to the user during the simulation improve the prototyping time. The reuse of ESOs already validated by previous studies is also a partial solution to maintain the correctness network models.

The library of "ready to use" models saves the final user from the modeling phase and the proposed natural way of building a network model makes NESSY usable by non-specialist users in modeling.

Nevertheless the power of NESSY has some drawbacks because network models are solved by simulation. The main drawback is the user can easily build too large and too detailed network models which take a very long time to simulate. As ESOs are autonomous process, parallel or distributed simulation may help to overcome this drawback. However, overhead due to communication between ESOs may occur. So, the full benefit of this solution relies on the assumption that network models are build with loosly-coupled ESOs. This is a very stringent assumption in computer networks. The currently planed solution in NESSY to overcome this problem is to encourage global modeling. The library will provide the user with ESOs modeling as a whole set of different network components. For example, one ESO will model a data link (i.e. the two modems and the medium) as single unit. For further study, a complementary solution is to provide several ESOs modeling the same network component but with different levels of detail. In turn, the problem is to help the user to choose the appropriate ESOs.

6. ACKNOWLEDGMENTS

We would like to thank Eric Horlait for his help and Veronique Vèque for her design work of LASSI.

7. 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.).

Dahl, O.; and K. Nygaard. 1966. "SIMULA - An Algol-based Simulation Language." *Communications of the ACM* 9, no. 9 (Sept.): 671-678.

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.

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

Khoshafian, S.; and R. Abnous. 1990. *Object Orientation: Concepts, Languages, Databases, User Interfaces.* John Wiley, New York, NY.

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, NJ, 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, USA, 30 Sept.-3 Oct. 1990). IEEE; Armed Forces Commun. Electron. Assoc.; U.S. Dept. Defense, 689-695.

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

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.

Roberts, S.D.; and J. Heim. 1988. "A Perspective on Object-Oriented Simulation." In *Proceedings of the 1988 Winter Simulation Conference* (San Diego, CA, Dec. 12-14). IEEE, Piscataway, NJ, 277-281.

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, CA, 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.).

Vèque, V.; M. Soto; and E. Horlait. 1991. "NESSY: NEtwork Simulation SYstem." In *International Teletraffic Congress* (Copenhague, Danemark, Jun.). 109-115.

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

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

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 Associate 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 Associate 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.