A MODULAR ATELIER FOR MODELING, SIMULATING AND PERFORMANCE ANALYSIS OF HETEROGENEOUS COMMUNICATION NETWORKS

Atika COHEN and Radouane MRABET
Université Libre de Bruxelles, Service Télématique et Communication
Bd du Triomphe, CP 230 Brussels, Belgium
cohen@helios.ihe.rtt.be and mrabet@helios.ihe.rtt.be

ABSTRACT
This article describes the structure of AMS, Atelier for Modeling and Simulating communication systems. Setting up a such atelier is the main objective of the OSISIM project. This atelier is meant to be able to model and simulate complex systems, and it is based on an extensible library of basic models. The atelier gives the end-user powerful tools to edit a communication system in a graphical environment. Each model implements uniform and well-defined sets of functions, while having clearly specified interfaces. In addition, the article focuses on the internal structure of basic models to be applied to each component of the library.

1. INTRODUCTION
The availability of good software tools is vital to simulation practitioners in the communication system field. Besides, the system being studied is so complex that it is almost impossible to model it on one shot. An alternate approach, is to divide the system into a collection of less complex subsystems. Each one has to be separately modeled. So, the overall system model will be constructed by linking the concerned subsystems. In order to allow reusability, subsystems corresponding to existing models may be identified, and isolated. The idea is to isolate the maximum of subsystems to be reused, to model them and to make them available. Based on this strategy, AMS (Cohen and Mrabet 1993a, 1994a) the atelier for Modeling and Simulating, has been developed for the OSISIM project (Open System Integrated Simulator). This atelier is built around a library of basic models which includes the most standard networks such as LANs, WANs, and satellite networks. In addition, it integrates facilities and tools to build an architecture of a communication system, and to study its performances.

The basic models are constructed in a very modular way. Each basic model, called Detailed Basic Models (DBM), corresponds to a subsystem, and is detailed so as to reflect its exact behavior. It implements uniform and well-defined sets of functions, while having clearly specified interfaces. To facilitate carrying out the models, their interconnection and their maintenance, an internal structure is proposed, and has to be applied to each model candidate to be in the library.

In addition to the library, the AMS is composed of several modular components which are categorized in four essential phases, namely the edition phase, the ADL phase, the simulation phase, and finally the presentation phase. The role of the ADL phase is to define a clear border between the description and the simulation phases.

To be operational in a reasonable time, the AMS has been designed on the basis of existing packages rather than starting from scratch. QNAP2 (Queueing Network Analysis Package 2), GSS4 (Graphical Support System 4) and MODLINE are the main packages that are used to operate the AMS. Based on those tools, a prototype has been implemented. It is operational on Sun workstations where a specific user interface has been implemented. It is operational on Sun workstations where a specific user interface has been developed which allows the end-user to describe his architecture in a graphical manner.

The description can also be made textually by using the specific language ADL (Architectural Description Language) (Cohen & Mrabet 1994b) designed for this purpose. This can be done in the same machine, or in another one.

Section 2 of this paper focuses on the components of the atelier. The processes behind those components are also described. Section 3 focuses on the internal structure of a basic model. Finally, a conclusion is drawn.

2. STRUCTURE OF AMS
AMS is designed in a modular manner. The main modules constituting the AMS are structured in four phases, each of them is handled by one or several processes. The figure below illustrates this structure. Processes are accessible by the end-user in a graphical and/or a textual manner. The two modes are identified on the figure below by the labels G and T. Besides, the atelier is based on the library of basic models constructed in a very modular fashion to allow more flexibility. This library is the main core of the AMS.

A description of those phases, as well as the processes composing the atelier is given, before describing the library.

The edition phase : In this phase, the end-user edits a communication system. Editing can be done through, either, a textual interface or a graphical interface. The textual interface is any available text-editor; in this case, the end-user has to describe his architecture following ADL syntax. The graphical interface is managed by a dedicated process, the editor. The end-user can create several instances of basic models, connect them by links, and, he can modify the values of some basic model attributes. In this environment, the end-user does not need to know ADL syntax since the editor process generates the ADL code for the edited architecture.
The current version of the atelier is based on the graphical kernel GSS4 which provides generic graph edition facilities to make the development of graphical performance modeling tools easier. Each basic model is represented by a simple icon, i.e. with one display. Besides, the end-user has two facilities to edit the architecture: generic icon display facility and sub-model icon edition facility.

**The AMS Structure**

A generic icon also represents a basic model such as a bridge, but has several displays. The display dynamically changes in the window edition depending on some attributes and/or on which basic model it is connected to. For instance, the display of a bridge is different if it connects two Ethernets, or two token-rings.

The second facility is a powerful mechanism proposed by the atelier. A sub-model icon is a special icon representing a whole sub-model. A sub-model is a part of the system under study and is composed of several basic models. So, it is constructed by using either simple icons or generic ones. For instance, a stack icon is a special icon inside which the end-user can edit a stack of protocols. These special icons allow the end-user to edit a large communication system in a hierarchical manner. In addition, a special icon can enclose one or several other sub-model icons. For instance, when several local area networks are to be interconnected by a satellite backbone, each local area network can be represented by a sub-model icon.

This process internally uses a graphical database, which contains all the information about icons, pop-menu and edition rules, allowing the end-user to edit his communication system graphically.

In order to reduce the number of errors made at this step, the end-user is assisted by a help menu, and some predefined restrictions on the available choices depending on the models used and the way they are linked.

**The ADL phase**: This phase has recently been introduced in the atelier to fulfill two objectives. The portability of the atelier, which allows its implementation on several machines running different operating systems and graphical kernels. The definition of a clear border between the edition and the simulation phases, which improves the modularity of the atelier. Those two objectives are complementary. This phase is handled by the ADL process. It includes three tasks which are: compiling ADL description, checking the coherence of the architecture and generating the QNAP2 code.

**The simulation phase**: The main objective of this phase is to assess the performance of the system previously edited. This phase includes making scenarios, executing simulations, and post-processing the rough results. It consists of three processes: the experimenter, the simulator, and the post-processor.

Through the first process, the end-user defines the scenarios. A scenario can be edited graphically or textually thanks to the ADL language. A scenario is mainly defined by varying the values of some parameters, called free variables. Each free-variable is assigned either a set of values, or a range defined by the lower boundary, the upper boundary and a step. The end-user can also define a scenario using the notion of parallel step variation, which means that, two or several free-variables will simultaneously change their values by the same step.

The second process, based on QNAP2 simulation solver, compiles and executes the generated code. The execution depends on the scenarios defined by the end-user. At the end of the execution, the desired results are produced.

By means of an existing statistical software, the third process will offer to the end-user powerful primitives to analyze the results produced by the simulator. The candidate software must verify the following criteria: the software must include a high level programming language, preferably an object oriented language. This software has to be able to interact, on-line, with another software in a transparent manner. We are considering integrating the statistical software S-PLUS.

**The presentation phase**: This phase is meant to allow different presentations of the simulation results. It offers the possibility of supervising the simulation by visualizing its execution. In other words, it offers a high level animation of showing, e.g., messages passing through the simulated architecture. This phase is handled by two processes which are the plotter and the animator.

The first process is to represent the results provided by the simulator in different ways depending on the choices of the end-user. The results can be visualized in the form of charts, tables, lists, or a combination of them.

The animation consists of visualizing the execution of the simulation by showing messages passing through the simulated architecture and the behavior of some given measurements. The animator process can be executed in parallel with the simulator process to show on-line the progress of the simulation. In this case, we can speak of a synchronous animation. The animator can also work asynchronously, namely it will be activated, after the simulation is finished. In this case, the process needs a specific file containing the trace of the simulation.

**The library AMS**: The main core of the AMS is the library of basic models. Each basic model corresponds to a communication entity. The library includes the most standard...
networks such as LANs, WANs, satellite and radio networks. Each of them is represented by an icon, the ADL description, as well as a QNAP2 code. So, the library has different views depending on the phase where it is used. For the edition phase, and in the case of a graphical environment, the library is seen as a set of icons. At the ADL phase, the components of the library are described in ADL language; the description is limited to the main attributes of the models. Finally, at the simulation phase, the library is seen as a set of models written in QNAP2.

3. DETAILED BASIC MODELS

The internal structure of a Detailed Basic Model consists of three blocks: Behavior Engine Block (BEB), Interfaces Block (IB) and Measurement Block (MeB). Hereafter, a brief description of these blocks:

**The Behavior Engine Block**: The modeling of the behavior of the DBM is described as an open network of stations. So, each station includes a queue with limited or unlimited capacity, and one or several servers; in addition, an algorithm is associated to each server, describing the provided service.

Besides, the DBM may have a large number of parameters which make it flexible from the end-user's point of view. On the one hand, there are parameters accessible by the end-user of the AMS, which may influence the performance of the system where the DBM is instanced. The default values, specified by the modeler, can be modified by the end-user, according to the range within which the parameters can vary.

Besides, the DBM may have other parameters which are not accessible by the end-user. The values of these parameters (called configuration parameters) depend on the architecture in which the DBM is used, namely on the values of other DBM parameters. For instance, the DBM IP fragments the segments it receives; the size of the fragments depends on the LAN it is connected to: the size for an Ethernet is up to 1500 bytes, while it does not exceed 1024 bytes for a Token-Ring.

**The Interfaces Block**: Interfaces are necessary to harmonize the interconnection between DBMs, and for the BEB to get rid of message exchanges with what is outside the DBM. A DBM cannot be used alone, it must be connected to one or several other DBMs to make up a complex system. This implies having several interfaces. Their number can either be a fixed value known during the DBM modeling, or can vary so that the modeler can fix only minimal and maximal values. The validity of the interconnection between DBMs is not checked by the DBMs themselves but, by the ADL process.

**The Measurement Block**: The measurement block contains two types of measurements, the first type reflects the behavior of the DBM, and thus, it is associated to the BE; the second type is associated to the interfaces and it shows mainly the data flow entering and exiting the DBM.

All the measurements are meaningful for an end-user not specialized in the field of network queueing theory. All the aspects related to this field are transparent. The measurements are associated to metrics used in the field of communication networks. The end-user of the AMS atelier will choose the measurements he wants to have at the end of simulations.

4. CONCLUSION

We have described the design of AMS, an atelier for modeling and simulating communication systems. The atelier has been developed inside the OSISIM project. This atelier is meant to be able to model and simulate complex systems, and it is based on an extensible library of basic models. The atelier is made up of four independent phases, each phase involves one or several processes. The division adopted allows a high degree of modularity and hence the portability. The atelier is operational, and this has been proved by using it to study a concrete architecture consisting of a communication system based on a satellite backbone (Cohen & Mrabet 1993b).

5. ACKNOWLEDGMENT

We wish to express our gratitude to SAIT Systems with whom we have been working on the OSISIM project. Besides, we gratefully acknowledge the financial support granted by IRSIA.

6. REFERENCES


