AN INTEGRATED TOOLSET FOR THE DESIGN OF PROTOCOLS WITH QoS REQUIREMENTS (ProtCAD/RT)

Gardel Moreira Delfino *José Valentim dos Santos Filho*
Rodrigo de Carvalho Roman *Jorge Lopes de Souza Leão*
Aloysio de C. P. Pedroza
Grupo de Teleinformática e Automação – GTA
Universidade Federal do Rio de Janeiro
COPPE/PEE - Programa de Engenharia Elétrica
C. P. 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brazil
e-mail:<gardel,valentim,roman,leao,aloysio>@gta.ufrj.br

ABSTRACT

The goal of this work is to describe the architecture of an integrated environment for specifying and simulating communication protocols, with timing constraints. The system, under development, is based on the CAD-Estelle toolset, which has been used in many academic works. In the proposed environment, all design steps can be done from the Extended Finite State Machine model of Estelle. The environment also has a visual editing tool based on this model and uses the *Real-Time Estelle* extension to specify distributed systems with timing constraints. This visual tool is being developed in JAVA to allow the WEB access, as well as guaranteeing independence of the system to a specific platform.

Keywords: Communication Protocols, Formal Description Techniques, Estelle, QoS Requirements, Web Interface.

1 INTRODUCTION

In the last decade, we have seen a significant growth on distributed applications where parameters related to the Quality of Service (QoS) offered are fundamental. Some examples that can be mentioned are multimedia applications and real-time control systems.

The entities involved in these systems realize data exchange using communication protocols that, due to the characteristics of these new systems, incorporate QoS restrictions in their specifications and implementations.

Formal Description Techniques (FDTs) assume an important role under this context. They present the possibility of development of an application on a higher level of abstraction than the one obtained with the use of programming languages, as the latter involve highly specific details related to the environment where the system is implemented. Besides, several FDT-based toolsets allow one to validate and test a given architecture before trying to implement it.

This work presents an integrated protocol development system (*ProtCAD/RT-Protocol Computer Aided Design/Real-Time*), that uses the Estelle FDT [1], and which is under development. This new system is based on the experience of using the CAD-Estelle toolset, developed and used in many graduate and under-graduate works at the GTA Computer Networks Group Laboratory (GTA/COPPE-UFRJ) [2]. However, the ISO standard for the Estelle FDT is not able to express real-time requirements, which justifies the use of an extension for this purpose.

The system here presented uses a subset of the *Real-Time Estelle* extension [3]. The main goal of this system is to turn the task of specifying and simulating protocols with QoS restrictions simpler and more intuitive. The examples presented in this paper were extracted from a description in Real-Time Estelle of a CBR transmission system. From this model, we can derive all steps in the project of a protocol with QoS restrictions. The complete specification of this system can be found in Delfino et al. [4].

In Section 2, some important features of the Estelle Language are presented, with emphasis on the real-time extension. In Section 3, the ProtCAD/RT structure will be described, along with its basic modules, their functions and the relations among them. The environment on which the system is being developed will be discussed in Section 4. Finally, in Section 5 we will present some remarks and comments.

2 THE ESTELLE FORMAL DESCRIPTION TECHNIQUE AND ITS REAL-TIME EXTENSION

The Estelle language is a FDT standardized by ISO. It has been successfully used to specify, validate and implement traditional protocols and distributed systems, as we can see by the deployment of some commercial [5] and academic [6, 7, 8] toolsets.

An Estelle specification consists of a set of communicating modules, interacting via message exchange through FIFO (Fist In First Out) queues, using Estelle language-specific primitives, or by exported variables. Module behavior is described by an Extended Finite State Machine (EFSM) model, and data manipulation is written in the Pascal programming language.

Despite its wide use for traditional protocol specification, the Estelle standard is not suitable for the specification of real-time constraints [9]. Even when the *delay* clause is used, this cannot be achieved, as this clause only refers to the firing interval of a specific transition, being unable to describe end-to-end timing requirements. To overcome this restriction, it has been developed an extension to the Estelle FDT called *Real-Time Estelle* [3].

2.1 The Real-Time Estelle Extension

The Real-Time Estelle syntax was constructed trying to maintain the same structure of the standard Estelle. This does not happen to other techniques, in which the inclusion of timing constraints leads to structural changes on the basic language. The SDL extension described in Fischer and Leue [10] is an example were the extension to the technique lead to major changes. In Real-Time Estelle, all real-time constraints of a module are put together into a new section in the module body called `TIME CONSTRAINTS`.

The description of the constraints is done by using temporal logic [11] formulas, which created the need for the incorporation of new keywords and operators to the language. In the ProtCAD/RT, we use a Real-Time Estelle subset, whose selected operators and keywords, which are all described in Fischer [3], allow the efficient specification of the most common QoS requirements such as:

1. *Transmission rate*: refers to how much bandwidth the application will need to achieve the desired performance. It can be expressed in number of messages per unit of time.
2. *Delay*: global delay introduced by the processing and transmission of packets.
3. *Jitter*: delay variation on packet transmission.
4. *Error rate*: reliability of the system being specified, in terms of unsuccessfully transmitted packets.

The chosen new elements incorporated to the language are listed in table 1.

Table 1: Additions to the Standard Estelle FDT

Identifier Type	Identifiers
Keywords	*time constraints*
Logical Operators	*and* *or* *otherwise* *eventually* *henceforth* *leadsto* *forbids* *at*
Time Operators	*now*
Simple State Descriptors	*sending of* *receiving of*

The code excerpt listed below presents the specification of timing constraints related to the transmission rate of the CBR system considered in this work. Adjusting the *RATE* constant, we can achieve the desired rate. In this particular model, we used values that lead to 96 Kbps rate, typical for some CBR audio sources [12].

Specification of the Transfer Rate

```
time constraints
  name TransferRate:
    forall t:time;
    forall seqnr:integer;
    henceforth(sending of
    MCEP.DATREQ [seqnr] at t
    leadsto sending of
    MCEP.DATREQ [seqnr +
    1000*RATE/PKTSIZE - 1]
    at (now < t + 1000));
  end;
```

3 THE ProtCAD/RT

The ProtCAD/RT system aims mainly at offering the user enough resources to develop a communication protocol. It must:

1. Allow a visual specification of a protocol and, from this model, offer a textual output that represents the Estelle code of the model graphically described;

2. Compile the Estelle code, verifying the syntax of the formal textual specification and generating an Intermediate Form (IF) which will be used as input to the simulation, implementation and, possibly, to other additional modules to be created in the future;

3. Simulate the protocol in an interactive form, simplifying its debugging process.

The architecture of the system components is shown in figure 1, where we can see the Visual Interface, the Compiler and the Simulator. These modules are to be detailed in the next subsections. This environment differs from other referred toolsets by its explicit support to timing constraints, as a form of describing QoS requirements, and the simplicity of specification by means of a visual interface, using resources that enable the operation of the toolset through the Web.

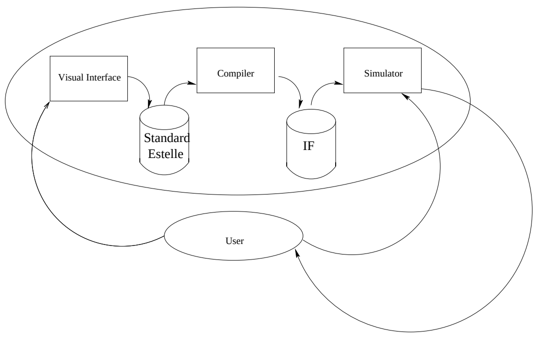

Figure 1: The ProtCAD/RT

3.1 The Visual Interface

The goal of the Visual Interface is to turn the specification of a protocol into a simple task, making it faster and more intuitive than the pure hand coding, trying to achieve the highest possible abstraction level, based on the standard Estelle features.

3.1.1 *Specifying a Protocol*

The Visual Interface allows the designer to specify the static architecture of modules, communication channels, interaction points and module internal behavior. Parts of a specification associated with data handling (which are basicly Pascal language program excerpts), timing constraints (which are temporal logic expressions) and dynamic behavior of modules are described in an organized textual form, that is enabled by resources given by the interface. The text code must have the standard Estelle format, with the respective timing constraints described formerly.

In the code excerpt shown below we can see part of the specification of the CBR transmission system considered here. This fragment is related to the Sender module. In this subsection, all figures refer to this system.

Code fragment of the Sender Module

```
module SenderType activity;
  ip UCEP:Sys_interface(i_provider);
     MCEP:Medium_interface(i_user);
end;

body SenderBody for SenderType;
  var n:data_type;

  state SenderIdle,
        DtTransfer;

  initialize
    to SenderIdle
    name SenderStart:
      begin
        n:=0;
      end;

  trans

    from SenderIdle
    when UCEP.START
    to DtTransfer
    name StartTransfer:
      begin
      end;

    from DtTransfer
    when UCEP.STOP
    to SenderIdle
    name EndTransfer:
      begin
        n:=0;
      end;
end;
```

Goulart [13] developed a system with a purpose similar to this one. In his work, module behavior could be described in a visual notation derived from SDL. Another related work [14] presents a visual specification syntax of dynamic module behavior, resulting in a rather complex form, as it tried to achieve a complete graphical Estelle syntax.

Protocol specification with ProtCAD/RT starts with its identification, followed by the type of the system specified and some other parameters like timescale, required external modules and declarations of procedures used, for example. Next, the variable types must be specified, using a Pascal language style of declaration. All this information must be available in a textual manner, as they cannot be easily represented in a visual form. For this particular specification stage, the Visual Interface provides a form to be filled, as can be seen in figure 2.

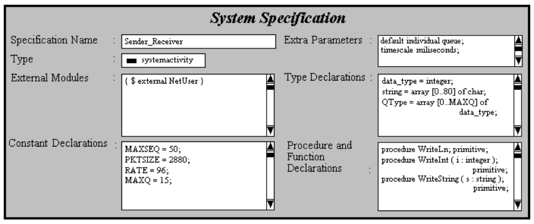

Figure 2: Beginning of a Specification

The communication channels are defined in a very simple way, by the identification of their names and primitives acceptable at each endpoint, which must also be named. Text boxes are provided to be filled by the user in order to define the channel name and endpoints. As shown in figure 3, this specification stage is also considered.

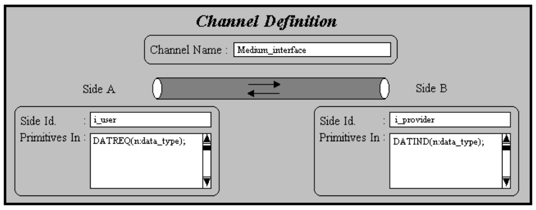

Figure 3: Communication Channel Specification

The definition of existing headers that can be used by module bodies to be defined later on must also be done. In order to accomplish this, it must be provided the identification, module type, starting parameters, exported variables and interaction points, that are associated to channel endpoints defined previously. A module header specification is done as seen in figure 4. Note that, in this example, one of the *ip's* (interaction points) is being edited (the marked one) in such a way that we visualize two *combo-boxes* that contain the name of predefined channels and end points, respectively.

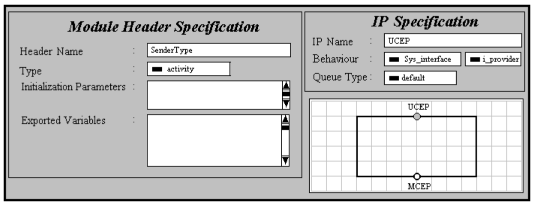

Figure 4: Module Header Specification

The module bodies, which are associated to the headers formerly defined, can be specified as shown in figure 5. At first, constants, variable types, used variables, procedures and functions needed must be declared. The block containing real-time constraints is also specified here, by a text box that must be filled. After that, the EFSM that defines the module behavior must be specified. For each inserted transition, it must be told, by filling a text box, the firing conditions and the actions that will be executed, using Pascal language code.

It must also be provided the corresponding names of the main states. Still in figure 5, we show an example of a module body specification, where we emphasize transition editing (the one in evidence). Here, we also have two *combos* that will be used to select the transition firing condition for those transitions enabled by the reception of messages (which are associated to the primitives defined in the header chosen for this module).

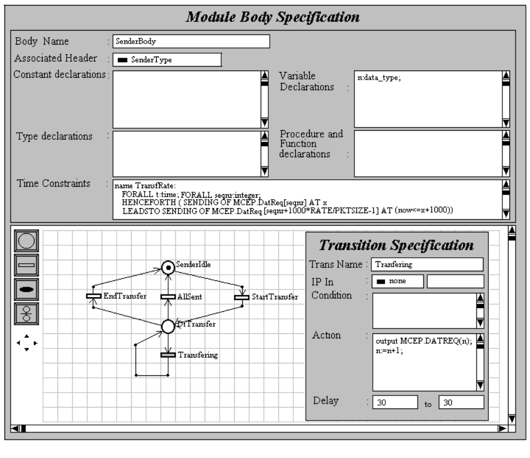

Figure 5: Module Body Specification

At last, after defining all modules and channels that take part in the system, we have to specify the static architecture, defining the modules to be launched and the connections between them. Figure 6 shows how this can be done in a visual manner.

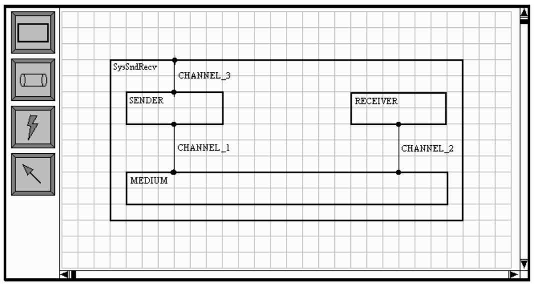

Figure 6: Module Architecture Specification

3.2 The Real-Time Estelle Compiler

The purpose of the Estelle Compiler for protocols with timing constraints is to generate the Intermediate Forms that serve as input files for the Simulator and, possibly, for other system modules that may be developed. These forms follow the design pattern described in Oliveira [15]. In this context, the Estelle specification of a certain protocol generates two IFs that are described below:

1. *Static IF*: contains all the data related to the transition system of the protocol, its modules, transition-enabling conditions, and so on. Summarizing, it is a textual formalization of the protocol transition system and of its hierarchical structure.

2. *Dynamic IF*: contains the information about the protocol at the firing of its transitions, like the actions executed, for example. The Pascal functions and procedures related to the data handling in systems and subsystems of the specification are also included. All the dynamic objects of the system are described in this form.

Based on this description we can conclude that the static IF defines how the protocol runs, and the dynamic, what it does while running. The static IF is, in fact, a text file with a certain syntax, that shows the relations among Estelle objects (states, interaction points, channels,and others) of a specification. The dynamic IF would be a translation to an existing programming language of the actions executed by Estelle modules, which allows the generation of executable files.

There are other Estelle compilers which have a different approach, for their purpose is somewhat diverse from ours. The PET (Portable Estelle Translator) compiler, for example, aims at generating a portable IF that enables the development of other Estelle-based tools using it as a starting point. One of these tools, the XEC (eXperimental Estelle Compiler), uses a modified version of PET, and its goal is to generate highly efficient distributed implementations, making it possible to realize implementation experiences with the Estelle language. This system also supports the specification of open distributed systems using another extension to the Estelle FDT (Open Estelle [16]).

The goal of the ProtCAD/RT compiler is to generate code that allows the simulation of protocols with timing constraints. For this reason, it was decided to develop an implementation of the compiler based on the original structure of the one present in the CAD-Estelle, which has the same approach, only without the support to QoS requirements. As we can notice, it must be included the timing constraint treatment in the compiler which implies some modifications in the code generating part. In turn, this will modify the IFs generated. Due to the characteristics of the syntax and semantics of timing constraints, part of them will be mapped directly to the static IF, as they do not relate to dynamic objects of the transition system. However, all components of the constraints which are changeable by the system under execution, such comparisons as involving time variables, should be mapped to the dynamic IF.

3.3 Real-Time Protocol Simulator

The Simulator offers the designer a support to testing protocols with timing constraints. The system considered is stimulated by a test sequence automatically generated or chosen by the user. This way, it is possible to realize a logical analysis of a given protocol before entering the implementation phase on a specific host system.

During simulation, module instances are created. These instances may be destroyed and created again. A module instance creation corresponds to the creation of an object instance set associated to the given module, just like the queues corresponding to the ip's, variables and modules declared in it.

The IFs described in the latter section serve as input to the Simulator, which has its architecture split into a set of modules inter-related by a hierarchical rule [17]. Quantitative timing requirements should be evaluated when modules are communicating, making it necessary the implementation of an algorithm that enables end-to-end monitoring of a certain message. Thus, a header is added to each message that carries the sending time and present simulation time. Moreover, the total time spent by the message to be sent can be computed, making it possible to evaluate if timing constraints were satisfied.

The Simulator engine is responsible for the execution of the model based on the simulation environment. The Simulator creates queues that interconnect several EFSMs. The basic Simulator algorithm is described as follows:

1. Obtain a user interaction (this phase is executed by the user interface);
2. Create a list including all enabled transitions;
3. Choose from this list a transition to be fired (this task may be done by the user or in a random manner);
4. Execute the corresponding action; messages are monitored during simulation allowing to evaluate timing constraints; the model then changes to the next global state;
5. Test breakpoint. If test is true, then the user gets back the control, else, it goes back to item 2.

The time spent during the execution of an action related to a given transition depends on the target system processing ability. The user must give an estimate on such times using Estelle delay clauses, telling the time interval in which a certain machine executes such task. In this interval, transition firing times are chosen randomly.

Another relevant point is concerned with the delay introduced by the communication medium. Here, the user must estimate the medium delay and use it as a parameter to a delay clause in the medium specification.

The simulator provides the user, among others, with these features:

- Breakpoint insertion and simulation traces;
- Step-by-step execution, making it possible for the user to choose the transition to be fired, turning the protocol debugging process easier;
- Introduction of test sequences during a programmed simulation stop;
- Access to values of real-time requirements: delay, transmission rate, jitter and error rate, providing a comparative analysis between the specified requirements and those obtained by simulation.

4 IMPLEMENTATION ENVIRONMENT

The Linux Operating System was chosen to implement our toolset as it is a free distribution platform with an open architecture (*freeware* philosophy). There are versions that can be used under Intel-compatible computers (since the 386). There is also an extensive support documented even in the Internet. Development of this system under a free and open distribution environment can minimize the risks of the release of versions incompatible with the previous ones, making the system obsolete. It is also notorious the diffusion of Linux in academic and commercial environments.

In order to create a platform-independent system and to diffuse it we choose the Internet as the access and running environment for the toolset. Any visual browser will interact to the project environment, as it can provide enough textual and visual resources for this purpose, besides being used for Web navigation.

In order to generate the forms and visual manipulations needed, obtain a high level of interaction between user and project environment generating an application that could be executed independent of the platform, we used the Java language. This language permits also the execution of code into the client machine and supports more interactive applications.

5 COMMENTS

This work presented a protocol development system which supports a real-time extension to the Estelle FDT. The core of this system is based on three modules: a Visual Interface, a Compiler and a Real-Time Simulator.

The Visual Interface will generate a textual Estelle specification, that may be compiled and simulated with the aid of the other two modules.

The ProtCAD/RT visual interface is being developed to enable its use through the **Web**, offering high interactivity and platform-independence, due to the choice of the **Java** language for its development. This architecture allows remote execution, making it useful as an academic tool and enabling its use diffusion through the Internet.

The ProtCAD/RT system, which is under development, takes into account the specification of timing constraints. Also, its conception under a more open philosophy may guarantee an extended lifetime compared to the present systems in terms of implementation environment.

At the present time, there is a semi-automatic code generator, which is in a final development stage, that supports real-time constraints and that can be integrated to the new toolset. Based on this system, other additional modules can be developed in the future. As an example, we may mention a performance analyzer for protocols with time constraints and a QoS properties verifier.

References

[1] M. Diaz, *Formal Description Technique ESTELLE*. North-Holland, 1989.

[2] A. C. P. Pedroza, C. C. Goulart, P. R. O. Valim, and R. de Oliveira, "Um Sistema de Auxílio ao Projeto de Protocolos de Comunicação para Redes de Computadores," in *Seminário Franco-Brasileiro de Sistemas de Informação Distribuídos (SFBSID'89)*, (Florianópolis, SC, Brazil), 1989.

[3] S. Fischer, "Real-Time ESTELLE," tech. rep., Universität Manheim, Germany, 1996.

[4] G. M. Delfino, J. V. dos Santos Filho, R. C. Roman, J. L. S. Leão, and A. C. P. Pedroza, "Especificação em Estelle Tempo-Real de um Sistema de Transmissão CBR," tech. rep., GTA-UFRJ, Brazil, 2000.

[5] Estelle Development Toolset. Institut National des Telecommunications `http://www-lor.int-evry.fr/lor/edt/`.

[6] J. Thees and R. Gotzhein, "The eXperimental Estelle Compiler - Automatic Generation of Implementations from Formal Specifications," in *The Second Workshop of Formal Methods in Software Practice (FMSP'98)*, (Clearwater Beach, Florida, USA), 1998.

[7] R. Sijelmassi and B. Strauser, "The Portable Estelle Translator: an Overview and User Guide," tech. rep., National Institute of Standards and Technology, System and Network Architecture Division, Gaithersburg, USA, 1991.

[8] R. Sijelmassi and B. Strauser, "The Distributed Implementation Generator: an Overview and User Guide," tech. rep., National Institute of Standards and Technology, System and Network Architecture Division, Gaithersburg, USA, 1991.

[9] S. Fischer, "On the Suitability of ESTELLE for Multimedia Systems," tech. rep., Universität Manheim, Germany, 1995.

[10] S. Leue and S. Fischer, "Formal Methods for Broadband and Multimedia Systems," *Computer Networks and ISDN Systems*, vol. 30, pp. 865–899, 1998.

[11] R. Gotzhein, "Temporal Logic and Applications - A Tutorial," *Computer Networks and ISDN Systems*, vol. 24, pp. 203–218, 1992.

[12] J.-P. Courtiat and R. C. de Oliveira, "On RT-LOTOS and its application to multimedia protocol specification and validation," in *Proc. Int. Conf. on Multimedia Networking (MmNet'95)*, (Aizu, Japan), 1992.

[13] C. C. Goulart and A. C. P. Pedroza, "Proposta de uma Forma Gráfica para a Linguagem Estelle," in *7º Simpósio Brasileiro de Telecomunicações, 7º SBT*, (Fortaleza, Ceará, Brazil), 1991.

[14] J. Templemore-Finlayson, J.-L. Raffy, P. Kritziger, and S. Budkowski, "A Graphical Representation and Prototype Editor for the Formal Description Technique Estelle," in *Formal Description Techniques and Protocol Specification, Testing and Verification (FORTE/PSTV '98)*, (France), 1998.

[15] R. C. de Oliveira Junior, "Um Compilador para a Linguagem Estelle," Master's thesis, Programa de Engenharia Elétrica - COPPE/UFRJ, 1991.

[16] J. Thees and R. Gotzhein, "Open Estelle - An FDT for Open Distributed Systems," in *Formal Description Techniques and Protocol Specification, Testing and Verification (FORTE/PSTV '98)*, (France), 1998.

[17] P. R. de Oliveira Valim, "Um Simulador de Protocolos para a Linguagem Estelle," Master's thesis, Programa de Engenharia Elétrica - COPPE/UFRJ, 1990.