A Formal Description Technology: Graphical E-LOTOS

Li Wen, Ye Xinming*, and Liu Zhiyong
Institute of computing Technology, Chinese Academy of Sciences, Beijing 100080
*(Department of Computer Science, Mogolia University, Hohhot 010021)

Abstract:

In this paper, we propose a method of Formal Description Technology (FDT) GE-LOTOS---a Graphical mode of Enhanced LOTOS (E-LOTOS). Compared with the existing FDT, the GE-LOTOS presented here has greater formal description capability. But GE-LOTOS only has a limited set of simple graphical patterns. Using the hierarchical structures of GE-LOTOS we can describe the structure and hierarchy of protocol more clearly and directly than using the existing E-LOTOS. Meanwhile the GE_LOTOS has another merit over the existing E_LOTOS that it can be executed easily and the system behavior is conveniently visible, so the system can be easily monitored and modified. We have developed a designing system for creating GE-LOTOS specifications under Java Cafe environment, and with further enhancement this system can be used for protocol testing and verification. The GE-LOTOS can be used as a design, execution, verification and testing tool for network communication protocol.

Key words

Testing of protocol, Formal Description Technology (FDT), Graphical E-LOTOS, Behavior expression

1. INTRODUCTION

With the rapid development of computer network, the testing of communication protocol of network has become the extremely important part in the cycle of design and research of network system. The correctness and efficiency of communication are necessary for working smoothly. No matter what we shall do, such as designing or tesing, the first thing is to describe a protocal system. Informal specifications often contain ambiguities and are difficult to check for completeness and correctness. To avoid ambiguty and support powerful logical analysis "Formal Description Techniques" (FDTs) must be used in describing protocols. FDT defines not only a formal syntax for the language, but also a formal semanteme which defines the meaning, in a formal manner, of any valid specification. Many different FDT have been proposed for the protocol engineering cycle, including finite state machines (FSM), Petri nets, formal grammars, high-level programming languages process algebra, abstract data types, and temporal logic. In the late 80s,there are three languages ESTELLE [8], LOTOS[4] and SDL[7], which were developed by ISO and CCITT, and have become the standardization. LOTOS, based upon process algebra, provides facilities for describing the external (observable) and internal (unobservable) behaviours of a system. And it is widely used in specification of networking and communication systems. But in the practical application, these FDTs often need to be extended. [12]

E-LOTOS is the revised version of LOTOS, and was proposed by the WI (Work Item) of ISO/IEC[1]. The purpose and scope of the definition of the new WI on "Enhancement to LOTOS" summarizes the main conclusions obtained from the practical application of LOTOS. E-LOTOS is the result of several years of work in which a large number of enhancement proposals have been analyzed and assessed. The selected enhancements removed the known limitations and extent the expressiveness, abstraction capabilities, structuring capabilities, ... of the preceding version of standard LOTOS. We assume that the readers of this paper are familiar with E-LOTOS and LOTOS.

But the existed E-LOTOS can only describe protocol in static text mode, so it can not directly and vividly describe hierarchy and synchronization of the system structure and the relations between actions. Meanwhile the more expression types there are, the more difficult it is to distinguish the behaviors and understand the protocol system described by E-LOTOS. Worse of all, the system described in E-LOTOS can not be executed. All these will cause of limited application of E-LOTOS. According to the book [11], a direct interactive graphical model has great effects on the application of FST.

For LOTOS, there are at least two graphical versions of it in the literature. The one proposed in WG-LOTOS[10] is not user-friendly. There are a lot of variations, intermingling, and non-uniformly graphical patterns. Especially, the rigidity and crowdness of nested rectangle structure (i.e., rectangles) make it difficult to understand the structure and logic of the designed system. Furthermore it is inconvenient for modification and modular design. The other one UO-GLOTOS [2] is a good graphical version. It has the simple and general graphical pattern and clear control constructs. On the whole, the UO-GLOTOS is a hierarchic model. Therefore it can greatly support the modular design, verification, logical detection, and testing of system specified by LOTOS. Generally, all graphical models are mainly describing the control behaviors of a system. Description of data is mainly based on textual version.

To conquer the limitations of existed E-LOTOS, after we deeply studied E-LOTOS and compared many kind graphical patterns and principles [5,13,14], we develop Graphical E-LOTOS. It can exactly express the semantics of E-LOTOS behavior expressions and has the simple and general patterns. Hereafter, for the convenience of reference, we call Graphical E-LOTOS as GE-LOTOS. Using GE-LOTOS, system is described in the form of a hierarchical structure, so the system structure and the relations between actions can be described more clearly, simply, and easy to understand. Using this feature, system also can be described in various levels of abstraction. Furthermore the GE-LOTOS presented here has many other good properties, such as it is "open" to "semantic" tools, it can be executed easily and the system behavior is conveniently visible, and easy to be monitored and modified.

Now many distributed systems need user-friendly and high description capability FDT. Our GE-LOTOS is this kind FDT. We have developed a designing system of GE-LOTOS under Java Cafe environment, and with further enhancement it can be used for protocol testing and verification. The graphical E-LOTOS can be used as a design, execution verification, and testing tool for network communication protocol and distributed systems.

This paper is organized as follows. In section 2, we discuss some considerations and criteria in designing GE-LOTOS. In section 3, we present the basic graphical patterns and the main features of GE-LOTOS. In section 4, we present the enhanced behavior expressions of E-LOTOS and the corresponding graphical mode. In section 5, we give an example of applying GE-LOTOS to describe a part of Alternating Bit Protocol. It includes a comparison with the E-LOTOS specification of the same protocol. Section 6 includes some conclusions and future work of GE-LOTOS.

2. CONSIDERATIONS IN DESIGNING GE-LOTOS

The domain application of FDT is determined by formal description capability. Behavior expressions are the basic component by which the behavior and hierarchy of protocol system can be described, and they directly embody the expression capability of E-LOTOS. Basically, all graphical models are mainly for describing the behavior of a system. In the following, we focus our discussion on the enhanced behavior expressions of E-LOTOS. For well-defining our problem, we highlight some the enhancements in E-LOTOS which have much influence on designing our GE-LOTOS.

In E-LOTOS, some enhancements are introduced by generalizing or enhancing LOTOS operators, others are new operators which have been introduced when absolutely necessary. Sequential composition operation ";" substitutes the operators existing in LOTOS for sequential composition ";", action prefix ";", and exit/enabling ">>" pair. Some notion of quantitative time is introduced, for example "wait(E)", which can be used for precise description of real time systems and allows a precise timed specifications to the execution of actions. Exception mechanisms, "trap" and "raise", permit new ways of describing structure. A more general parallel operator "par" is more readable and flexible, because it clearly identifies the synchronizing gates for each behaviour composed. Other new important constructs are "loop", "rename", "var", and etc. How to represent these new or enhanced constructs in graphical mode is our task.

2.1 Difficulties to design GE-LOTOS

• What kind graphical patterns not only can describe the semantics of enhanced behavior expressions exactly and clearly, but also can be accepted by users?

• How to reduce the numbers of patterns of the enhanced behavior expressions in order to reduce complexity of the whole GE-LOTOS system.

• How to keep the maximum "open" ability, describe the structure of protocol clearly and vividly, and execute system easily.

2.2 Criteria to design Graphical E-LOTOS and advantages of the Graphical E-LOTOS.

• Rather than introducing new graphical patterns, we use the existing graphical patterns of G-LOTOS [2] to represent the generalized and enhanced operators.

• Each construct of G-E-LOTOS is represent independently, and we have avoided the nested structure. For example, the set of operations {**loop**, **forever**, **rename**, **trap**} are all represented by independent graphical element, while they all have the efficient range. Generally efficient range can be described implicitly. When the efficient range are needed, we can use labels.

• Using a versatile hierarchy structure to represent recursive E-LOTOS expressions. For example, different hierarchies are used to describe process definitions and process instantiations.

• The present GE-LOTOS is convenient for manipulation, and convenient to execute the behaviors of the described system.

• For the new operators, we classify the enhanced behavior expressions according to its semantics and function structure. If the enhanced behavior expressions have similar semantics or similar function structure, the same pattern is adopted. The difference between each other can be distinguished by the expression in graphical pattern. For example, the set of E-LOTOS behavior expressions {exit[(RE)], exit(any T), break[X][(E)], raise X[E], stop} all have the similar function structure, and {exit[(RE)], exit(any T)} have the similar semantics. So we use the same pattern -- a stretched oval. Another example is that the set of E-LOTOS behavior operations {par, concurrency, trap, case} use the similar tree-like pattern which shows clearly the operator and it relationship of operands. Their graphical patterns are listed at table 2-1 and table 2-2.

3. THE BASIC GRAPHICAL PATTERNS AND MAIN FEATURES OF GE-LOTOS

Depended on above-mentioned criteria, our GE-LOTOS only has six different basic graphical patterns, which can represent 31 kinds of behavior expressions of E-LOTOS. There is corresponding expression inside the graphical pattern. The six different basic graphical patterns are:

• Stretched oval denotes termination, raise exception and internal constructor. They are distinguished by the expression inside the stretched oval.

• Hexagon denotes some declaration or control constructs---**var**, **rename**, **loop**.

• Tree-like pattern denotes the behaviours which have parralel, disable, trap, or case logical relationship. These relationships are distinguished by the key expression of left root.

• Rectangle with reference denotes process definition and process instantiations. A reference number at the upper-left corner of the rectangle represents a definition, whereas the same reference number at lower-right corner of the rectangle represents its instantiation. This construct is a very important pattern which can make GE-LOTOS to describe various levels of abstraction of system and to design large-scaled specification.

• strip shape denotes "assign" or "delay" behavior expressions.

• strip shape with disjunctive line denotes "action" behavior expressions.

Now we can know the present GE-LOTOS have the features below:

• A limited set of simple and elegant graphical patters, and versatile capability of description.

GE-LOTOS only has six different basic graphical patterns. But using these graphical patterns we can clearly and vividly describe the hierarchy, sequence, concurrency, synchronism, and all of the behaviors in the protocol system.

• Hierarchical and open structure.

The system described in GE-LOTOS is hierarchical in nature. So we describe the structure of protocol system clearly and exactly. Using process definition and process instantiations graphical patterns, we can describe various levels of abstraction of system and design large-scaled specification. This feature also makes it convenient for supporting logical exploration, modular design, verification, and testing.

• The system described in GE-LOTOS is executable.

This feature makes it easy to graphically simulate the behaviours of the system and generate executable paths and test case.

The Graphical E-LOTOS which has versatile expression capability can be used as a design, execution verification and testing tool for network communication protocol and distributed systems.

4. THE GRAPHICAL PATTERNS OF GE-LOTOS

4.1 Symbol specification used in behavior expressions(Table 1)

Meaning	abbreviation	Meaning	abbreviation
Natural number	i, j, k, l, m, n	Expression	E
Behavior expression	B	Record expression	RE
Exception identifier	X	Local variable	LV
Gate identifier	G	Rename parameter	NP
Type expression	T	Behavior match	BM
Pattern	P	List	N(length|N|)

Table 1 Symbol specification used in behavior expressions

4.2 A graphical syntax of enhanced behavior expressions of Graphical E-LOTOS

In table 2-1 and 2-2, we present the enhanced behavior expressions of E-LOTOS and the corresponding graphical pattern. Column 2 lists our graphical syntax. Column 1 contains the corresponding enhanced behavior expressions. We have not presented the behavior expressions of LOTOS and corresponding graphical pattern [2].

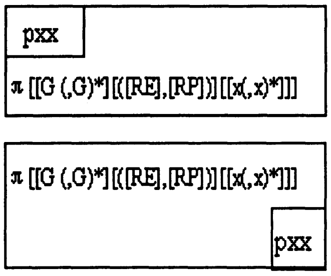
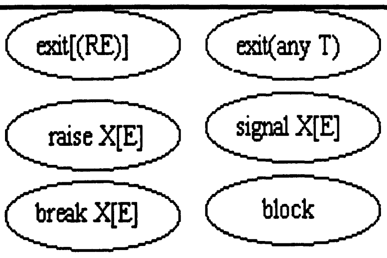
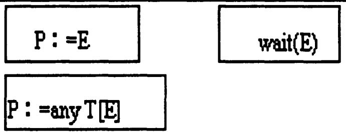
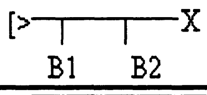
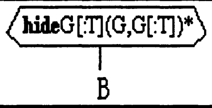
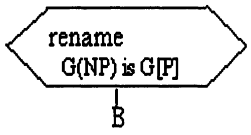
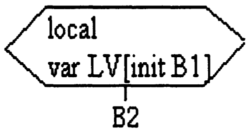
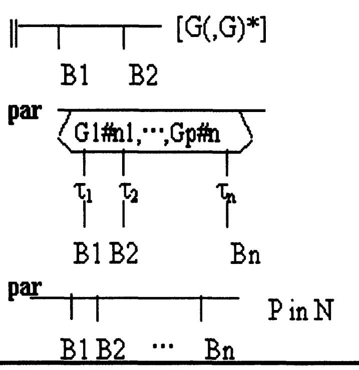
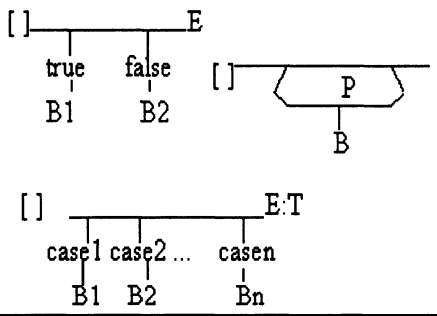
Behavior Expressions of E-LOTOS	Graphic Expression of E-LOTOS
process-define and process-instance **π [[G (,G) *]] [([RE,RP])] [[(x(,x)*]]** (pxx' at top-left-corner represent process-define, 'pxx' at bottom-right-corner represent process-instance.)	 pxx π [[G (,G)*] [([RE],[RP])] [[x(,x)*]]] π [[G (,G)*] [([RE],[RP])] [[x(,x)*]]] pxx
successful termination **exit[(RE)]**, nondeterministic termination **exit[any T]**, raising exception **raise X[E]**, **signal X[E]**, breaking iteration **break [X][(E)]**, time block **block**	 exit[(RE)] exit(any T) raise X[E] signal X[E] break X[E] block
Assign **P: =E**, Nondeterministic assign **P=any T[[E]]**, delay **wait(E)**	
suspend/resume **B1 [X>B2**	
Hiding **hide** G[:T](G,G[:T])* in B **endhide**	
Rename **Rename** (G[(NP)] is G[P] | X (NP) is X[E])* in B **endren**	
local specification **local** var LV[init B1] in B2 **endloc**	

Table 2-1 Behavior Expressions and Graphical Expression

Behavior Expressions of E-LOTOS	Graphic Expression of E-LOTOS
concurrency **B1 |[G(,G)*]| B2** general parallel **par** G1#n1,…,Gp#n_p [τ_1] for B1 ||[τ_2] for B2 ||…… ||[τ_n] for Bn **endpar** parallel over values **par** P in N ||| B **endpar**	 || [G(,G)*] B1 B2 par G1#n1,…,Gp#n τ1 τ2 τn B1 B2 Bn par P in N B1 B2 … Bn

choice-expressions
if E **then** B1 [else B2]
endif
case [E:T]is BM **endcase**
 	
choice P[]B **endch**
	 [ ] E true false B1 B2 [ ] P B [ ] E:T case1 case2 … casen B1 B2 Bn

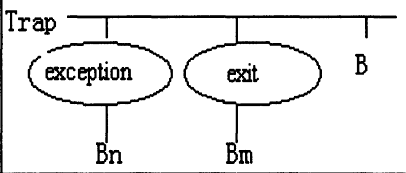
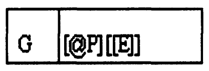
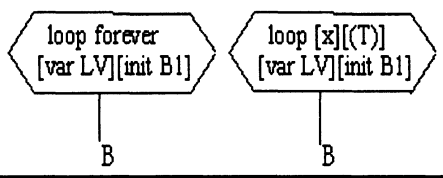
Trap (**exception** X[(LV)] is B)* **endexn** [**exit** [P] is B **endexit**] **in** B **endtrap**	
action **G[P][@P][[E]]**	
iteration **loop** forever **loop** [x][(T)] [var LV][init B1] [var LV][init B] **in** B **in** B **endloop** **endloop**	 loop forever [var LV][init B1] B loop [x][(T)] [var LV][init B1] B

Table 2-2 Behavior Expressions and Graphical Expression

5. A PROTOCOL EXAMPLE DESCRIBED IN GRAPHICAL E-LOTOS

Communication protocols are the rules that govern the communication between the different components within a distributed computer system. Here we select a part of the Alternating Bit Protocol which is often used as example. This part process is:

Before send message, start up chronoscope.
2) Wait for feedback message.
3) If the right received message comes back in the set time, the process is over,
otherwise repeats above-mentioned process until correctly received.

For comparison purpose, we first describe this part process in E-LOTOS.

```
process safemesser[sendpdu:pdurec,receiveack:pdurec]
            (datamsg:data; seqsender:bit):exit(none) is
  local   var timestart,reset,expired:time;  in
  hide tm:time in
   (tm(!timestart);sendpdu(!makepdu(datamsg,seqsender));
   ((messer[receiveack](seqsender);tm(!reset);exit
    [>
    (tm(!expired); signal "timeout";
  safemesser[sendpdu,receiveack](datamsg,seqsender))))
  |[tm]| timer[tm]
endloc  endproc

process messer[receiveack:pdurec](seqsender:bit):exit(none) is
  loop forever  in  var ack:bit
  receiveack(?ack);
  if (ack==seqsender)
    break
endloop   endproc

process timer[tm:time]:exit(none) is
 local   var timestart,reset,expired:time;   in
  tm(!timestart);
  (i;tm(!expired) ;exit
   [> tm(!reset));exit
 endloc   endproc
```

The above-mentioned process is shown in Graphical E-LOTOS in figure 1-1 and figure 1-2.

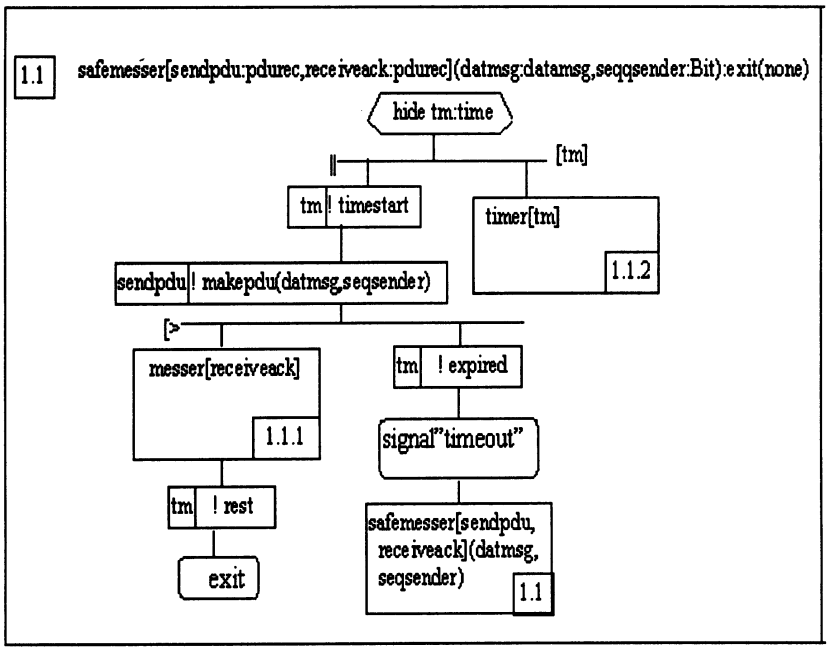

1.1
safemesser[sendpdu:pdurec,receiveack:pdurec](datmsg:datamsg,seqqsender:Bit):exit(none)
hide tm:time
[tm]
tm ! timestart
timer[tm]
1.1.2
sendpdu ! makepdu(datmsg,seqsender)
[>
messer[receiveack]
1.1.1
tm ! expired
signal"timeout"
tm ! rest
exit
safemesser[sendpdu, receiveack](datmsg, seqsender)
1.1

Figure 1-1 Part of the Alternating Bit Protocol in GE-LOTOS

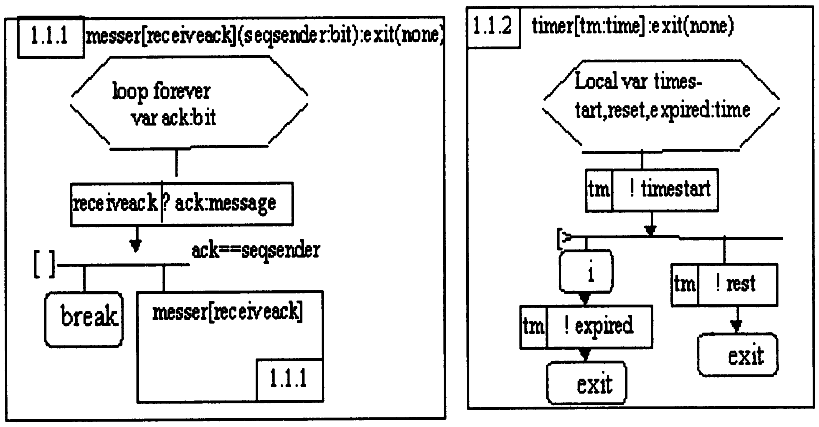


Figure 1-2 Part of the Alternating Bit Protocol in GE-LOTOS

The protocol in GE-LOTOS is described with two levels. "messer" and "timer" in first level are described abstractly. In the second level they are detailed respectively. After some comparison, we can know that the

Graphical E-LOTOS specification can show the hierarchy structure of protocol system and the relationship between behavior more clearly and straightly than the existed E-LOTOS does, such as the parallel relationship between “sendpdu” and “timer”. The simple and "open" properties can also be seen in this example. It is easier to understand the above-mentioned process in Graphical E-LOTOS than in the existed E-LOTOS. The Graphical E-LOTOS specification can show the logical flow of the protocol much more clearly.

When using our GE-LOTOS tools, description of protocols will become more easy and need less work. In this designing system, user can conveniently choose elements through the corresponding icon buttons. The modifications about the elements can be done through the “move”, “delete”, and “contents”, which are used to change content of elements. Besides these, GE-LOTOS files can be “open”, “save”, or “save as” in the standard dialog frame.

GE-LOTOS has the powerful description capabilities. It can be used to describe varieties of complex and large systems at various levels of abstraction. For example, it has successfully described the ODP trader computational viewpoint [9]. The ODP trader is an object which enables software components to find appropriate services providers within an open and dynamically changing distributed system. Because of the space limitation of the paper, this example is not included here.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we present a method of Formal Description Technology (FDT) GE-LOTOS. GE-LOTOS have many good properties, such as simple and open properties, and capability for supporting logical exploration, modular design, execution, verification and testing. The simple and open properties of Graphical E-LOTOS are very good properties to design large-scale system. Compared with the existing FDT, the graphical E-LOTOS presented here not only has greater formal description capability, but also can describe the parallel, temporal order, structure and hierarchy of protocol more clearly and directly than the existing E-LOTOS.

We have developed a designing system for creating GE-LOTOS specifications under Java Cafe environment. In our future work, we will provide the theory and algorithms for systems verification and testing. Meanwhile we will develop the designing system into a system which can provide a graphical environment for carrying out many research activities, such as control and data flow tracing, graphical execution and verification of systems described in GE-LOTOS. The GE-LOTOS can become a very useful method which can be used in design, execution, verification, and testing for network communication protocol and distributed system.

REFERENCES

[1] ISO/IEC JTC1/SC21 WG7. Enhencements to LOTOS. May 1998.

[2]T.Y.Cheung,Y.C.Ye, G.Q.wang. UO-GLOTOS: A Syntax/System for Representing, Editing and Translating Graphical LOTOS. In Proc of 2nd International Conference on Formal Description Techniques for Distributed Systems and communications Protocols.Vancouver.pages 33-48,1989.

[3]G.v.Bochmann, A.Petrenko. Review of Methods and Relevance for Software Testing. Montreal Canada Publication, #923,1994,6.

[4]T.Bolognesi,E.Brinksma. Introduction to the ISO Specification Language LOTOS. Computer Networks and ISDN Systems, vol.14,No.1,1987.

[5]Luciano Baresi,Alessandro Orso,and Mauro Pezze. Introducing Formal Specification Methods in Industrial Practice. In Proc. Of the 19th International Conference on software Engineering. Massacbusetts,pages 56-64,1997.

[6]L.Baresi. Formal Customization of Graphical Notations. PhD thesis, Dipartimento di Milano, Italian, 1997.

[7]F.Belina and D.Hogrefe. The CCITT-Specification and Description Language SDL. Computer Networks and ISDN Systems. Vol.16.1989.

[8]S.Budkowski and P.Dembinski,"An introduction to Estellle:a Specification Language for Distributed Systems". Computer Networks and ISDN Systems. vol.14.No.1.1987.

[9]Giovanny Lucero and Juan Quemada. An E-LOTOS specificaton of the ODP Trader. Input document(GRI) to the ISO/IEC JTCI/SC21/WG7/E-LOTOS meeting in Grenoble,December 1996.

[10]ISO/IEC JTC1/SC21,N3253. G_LOTOS: a graphical syntax for LOTOS. Jan 1989.

[11]Willian R.Mallgren. Formal Specification of Interactive Graphics Programming Languages.The MIT Press Cambridge, Massachusetts London, England. 1982.

[12]Luc Leonard and Guy Leduc. An introduction to ET-LOTOS for the description of time-sensitive systems. Computer Networks and ISDN systems, 29(3):271-292,1997.

[13]Enoch Y. Wang, Heather A. Richter and el at.. Formalizing and Integrating the Dynamic Model within OMT*. In Proc. of the 19th International Conference on software Engineering. Massacbusetts,pages 45-55,1997.

[14]J-P.Jacquot and D. Quesnot. Early Specification of User-Interfaces: Toward a Formal Approach. In Proc. of the 19th International Conference on software Engineering. Massacbusetts,pages 150-160,1997.