Formal Design of Managed Objects

Mirela Sechi Moretti Annoni Notare[1]
mirela@lrg.ufsc.br

Ketter Ohnes Rogerio[2]
ohnes@lrg.ufsc.br

Cristiano Maciel[3]
maciel@lrg.ufsc.br

Bernardo Gonçalves Riso[4]
riso@lrg.ufsc.br

Carlos Becker Westphall[5]
westphal@lrg.ufsc.br

Federal University of Santa Catarina
LRG - Management and Networks Laboratory
P.O. Box 476 Florianopolis - SC Brazil ZIP 88040-970
Phone: +55(48)231-9739 r.235 Fax: +55(48) 231-8770

ABSTRACT: In this work we discuss the use of the Formal Description Technique (FDT) LOTOS on the design of managed objects used in managing systems of computer and telecommunication network. Generally, such managed objects are presented with an utilization of GMDO resorts (Guidelines for the Definition of Managed Objects) defined by means of the ASN.1 language (Abstract Syntax Notation One). Nevertheless, the GDMO and ASN.1 languages besides not having a formally defined semantics do not have the resources for the representation of aspects of systems behaviour. Hence, it is valuable to investigate the possibility of use of the FDT LOTOS for the representation and the design of managed objects in the network management.

KEY-WORDS: network management, formal specification, managed objects, LOTOS.

1. Introduction

One of the most important issues in the design of network management systems is related into the definition of managed objects. Such objects form a representation of network resources (hardware and software) which may be managed in order to mantain a good performance of the network, and moreover, to enable the offer of services demanded by the users [DyAi 94] [AgSo 94].

Generally the design of managed objects includes in its conception an informal representation (in natural language), a formal representation (in GDMO - Guidelines for the Definition of Managed Objects) and its implementation in programming language (of manual or automatic mode). This approach does not allow the complete formal representation of managed objects once that GMDO is not suitable for representing behavioural issues of managed objects [MeNo 96].

This work aims to determine an approach for representing managed objects in an complete manner, that is, presenting the data issues and also the behaviour issues. For this, the LOTOS formal description technique [ISO 8807] is used.

This work is organized as follows: Section 2 states a summary of the principal issues related to the managed object design. In Section 3 a summary of LOTOS is presented. In Section 4 a managed objects is described precisely in LOTOS. Finally, in Section 5 is shown an evaluation of the present work, introducing the conclusion and suggestions for forthcoming works. Section 6 are the acknowledgements and the bibliographic references (Section 7).

---

[1] MSc. in Computer Science (UFSC, 1995). Researcher of CNPq/ProTeM-CC/PLAGERE Project.
[2] PostGraduate Student on Computer Science Course at UFSC.
[3] PostGraduate Student on Computer Science Course at UFSC. CAPES scholarship.
[4] Dr. in Electrical Engineering (UFPB, 1991). Local Coordinator (UFSC) of CNPq/ProTeM-CC/PLAGERE Project.
[5] Dr. in Computer Science (U. Paul Sabatier, 1991). National Coordinator of CNPq/ProTeM-CC/PLAGERE Project.

2. Managed Objects

Managed objects are logical representations of resources and/or available services in a computer or telecommunication network. These objects implement the interface between the network management systems and the actual resource, encapsulating the details of access. In this manner the network manager does not directly interact with the actual resources, once that this communication is intermediated by the managed objects [RaFr 94] [DrGo 94].

For the formal specification of managed objects a definition by ISO (International Organization for Standartization) is presented in the ISO/IS 10165-4 standard, where there is a setting of guidelines for the definition of managed objects. These guidelines received the designation of GDMO and represent the managed objects in terms of attributes, events, operations, and behaviour [ISO 10165-4].

The description of managed objects is performed using the templates shown on standard form. These templates follow the object oriented paradigm, that is, being the definition of managed object class inherits the whole set of the definitions of its superclass [MoSt 94] [RoFr 96].

At the present time the GDMO has its usage widely employed on the specification of managed objects. It can be, in part, credited to the greater utilization by one of GDMO being a specific standard of ISO used in this activity. However, as a formal specification language, GDMO is limited by not having settings that would permit the representation of behavioural issues. Then all the definitions refering to the behaviour of managed objects class are specified in natural language.

3. LOTOS

As previously mentioned this work proposes the utilization of LOTOS for the specification of managed objects. This formal description technique presents means for the description of data issues and behaviours. Created with the aim of specifying services and protocols of the OSI model, LOTOS has now been increasingly adopted [NoRi 97]. This evolution occurs due to the greater strength of the representation of the technique, besides the ability of auxiliar tools for the design.

3.1 Notion of Process

For formally describing the behaviour of a process employing the concepts of the FDT LOTOS the general form [ISO 8807] showed bellow must be followed:

```
process <identifier of the process>
                    [<parameters list>]:functionality:=
      <expression of behaviour>
endproc
```

where:

parameters list - includes the connection ports with other processes or with the enviroNment.

functionality - noexit for infinite processes; and exit for processes that makes able other processes;

behaviour expression - sequence of events that determine the behaviour of the process.

3.2 LOTOS Operators

The fdt LOTOS has a set of operators which is relatively small but of great expressivity. Some of these operators are presented as follows.

3.2.1 Nondeterministic Choice

For the representation of the random choice among two or more events of great expressivity, the operator [] is used on the FDT LOTOS. Thus, the expression (a[]b) represents the selection: ocurrence of event a or of event b. An example of utilization of this operator can be reported in [ISO 8807]: two independent buffers regulated by an unique process so that each buffer corresponds to a distinct process.

As the sequence of input and output events can not be determined in advance, we may hereby use the random operator. Following is presented one of the possible specifications of this process using the FDT LOTOS:

```
process two Buffers [in_a, out_a, in_b, out_b]:noexit:=
            in_a; (in_b; (out_a; out_b; stop
                        []out_b; out_a; stop)
                  [] in_b; (in_a; (out_a; stop )
            []in_b; ( in_a; (out_b; out_a; stop
                              []out_a; out_b; stop)
                  []out_b; in_a; out_a; stop)
endproc
```

3.2.2 Recursivity

Some processes as have the feature of recommencing at the end of a given sequence of events. This characteristic can be noticed in several processes that involve computers, as in network management systems.

A failures management system can be described in a simplified manner as a process that expects the occurrence of a failure event, indicates this to the manager, keeps waiting for a response event, and shows (or not) the solution. At the end of this sequence of events the system restarts again in order to detect other failures. For describing these events the FDT LOTOS presents ways of representing the recursivity.

The LOTOS recursivity is hereby exemplified with the process of starting, of running, and of switching off a car. Clearly, when switched off, if it still has fuel, it can be started again. This activity can be performed in LOTOS by:

```
process car[starts, runs, switch_off]:noexit:=
      starts; runs;switch_off;car[starts, runs, switch_off]
endproc
```

The recursivity may be stated by the TDF LOTOS putting the name of the process always after the sequence of events.

3.2.3 Parallelism

The FDT LOTOS has the aim of specificating the distributed systems, and in the design of this sort of system may be necessary to represent the parallelism of processes. The paralell processes can be fully independent of each other, totally dependent or partially dependent. LOTOS supplies operators to these three kinds of parallelism:

Parallelism of independent processes (|||): in this kind of parallelism the processes do not share any port, that is, their events can happen in parallel, but there is not any syncronisation among the processes.

Parallelism of fully dependent processes (||): in this kind of parallelism the processes share events in all ports. Thus, when an event occurs in a port of a given process, it must occurr under all the processes that are dependent in parallel with this one.

Parallelism of partially dependent processes(|[...]|): the operator for representing this sort of parallelism is called general operator of parallelism, and is represented by |[shared ports]|. Thus, the expression Process_1 |[port_1,..., port_n]| Processo_2 describes the situation where the processes Process1, Process2 might be sincronized with respect to the events ocurring at the ports port_1,...,port_n.

3.3 LOTOS Data Types

In this section concepts for data representing using LOTOS are presented. The concepts related here do not describe the totality of concepts, but only those used in this paper. One abstract type of data is specified in LOTOS with a definition of data storage and operators. In the definition of the operators the control and the image of the operation are reported.

The name of the signature of the datum type is given to the whole set of carriers and operators. Other features and operators of data type can be described in LOTOS, but are not utilized in this work.

3.3.1 Structured Interactions with LOTOS

In this subsection for specifying the interaction among processes with the data transference some characteristics existing in LOTOS are presented. Some of these systems require that, at the moment of the occurrence of some event, data might be exported or imported from other processes. In this manner, the FDT LOTOS includes means for data storage and transference.

The data storage is carried out in variables. These must belong to a type of data known in the specification, and they can only to storage data of a determined type. The statement of the variable is performed in the expression of behaviour, and has the following syntax:

```
event (! | ?) name_of the_variable: sort_of the_variable
```

In a general form, the symbols ! and ? are used for representing the input and output of data, respectively. In this way, when a given process shares an event with another one, and performs a changing of informations, these symbols are employed for the specification of this change.

In the specification of the output process the event is followed by the ! symbol, and the value to be outputted. The specification of the process that is outputting data in the occurence of the event contains a ? symbol followed by the variable that will storage the data.

4. LOTOS Specification of Managed Objects

The ISO defines for the OSI model of network management a large set of managed objects in order to consider the specifications included in their standards. According with the ISO the managed objects must be related to each other following the three inheritance, containment, and register hierarchies.

The GDMO model is established on the paradigm of objects orientation, and the inheritance hierarchy that is specified in GDMO should follow the demanded hierarchy by ISO. In this manner the managed object chosen for the specification is the root object of the inheritance tree: the Top object (X.721).

The specification in GDMO proposed through the Top object is presented as it follows:

```
top MANAGED OBJECT CLASS
CHARACTERIZED BY
top Packag  PACKAGE
      BEHAVIOUR
      topBehaviour;
      ATTRIBUTES
      objectClass GET,
      nameBinding                   GET ;;;
CONDITIONAL PACKAGES
      packagesPackage         PACKAGE
      ATTRIBUTES  packages          GET;
      REGISTERED AS{smi2Package 16};
      PRESEENT IF "if any registered package,
                  other than this package
            has been instantiated",
      allomorphicPackage            PACKAGE
      ATTRIBUTES
      allomorphs        GET;
      REGISTERED AS{smi2Package 17};
      PRESENT IF "if an object supports allomorphism";
REGISTERED AS           {smi2MObjectClass 14};
topBehaviour            BEHAVIOUR
```

DEFINED AS "This is the top level of managed object class hierarchy and every other managed object class is a specialization of either this generic class (top) or a specialization of subclass of top. The parameter miscellaneousError is to be used when a processing failure has ocurred and the error condition encountered does not match any of object's defined specific error types.";

The specification of the components of a managed object can be described in the body of managed object specification (in-line specification) or after (off-line specification). In the specification of the Top object all the packages were specified in- line, but the attributes specification is done after, and presented below:

```
allomorphs        ATTRIBUTE
      WITH ATTRIBUTE SYNTAX Attribute-ASN1Module.Allomorphs;
MATCHES FOR EQUALITY, SET-COMPARISON, SET-
INTERSECTION
      REGISTERED AS           {smi2AttributeID 50};
      nameBinding       ATTRIBUTE
            WITH ATTRIBUTE SYNTAX Attribute-
      ASN1Module.NameBinding;
            MATCHES FOR EQUALITY;
      REGISTERED AS           {smi2AttributeID 63};
      objectClass ATTRIBUTE
            WITH ATTRIBUTE SYNTAX
                   Attribute-ASN1Module.ObjectClass;
            MATCHES FOR EQUALITY;
      REGISTERED AS           {smi2AttributeID 65};
      packages ATTRIBUTE
            WITH ATTRIBUTE SYNTAX Attribute-ASN1Module.Packages;
            MATCHES FOR EQUALITY, SET-COMPARISON, SET-
INTERSECTION;
      REGISTERED AS           {smi2AttributeID 66};
```

The syntax of the attributes in ASN.1 (Abstract Syntax Notation One) are also presented:

```
Allomorphs ::=SET OF ObjectClass
Packages ::=SET OF OBJECT IDENTIFIER
NameBinding::=OBJECT IDENTIFIER
ObjectClass::=CHOICE {  globalForm  [0] OBJECT IDENTIFIER,
                        localForm   [1]INTEGER
```

This work does not propose the utilization of LOTOS specification of the attributes for the managed objects. In this manner it must obtain the LOTOS specification relating to the Top objects class behaviour, showed on the topBehaviour clause.

The first sentence contained the topBehaviour clause actually does not express behaviour, it is only a remark describing the object classes position on the inheritance hierarchy. Nevertheless, the second sentence expresses the behaviour of these object classes. This behaviour can be seen in the Figure 1. The TOP class receives an error message associated with a condition not recognized by its subclasses and the miscellaneousError parameters then used.

Figure 1 - TOP class behaviour.

The specification in LOTOS of the TOP object behaviour is presented as it follows:

```
specification TOP_Class [failure, parameter]: noexit
behaviour
      TOP[failure,parameter]
where
process TOP[failure,parameter]: noexit:=
      failure? error:errorCondition;
      parameter!miscellaneousError;
      TOP[failure,parameter]
endproc
endspec
```

5. Using LOTOS Tools for validation

The Eucalyptus toolbox [Gara 96] is its a graphical user-interface (GUI) based on X-Windows. Although the Eucalyptus toolbox groups different tools developed by different partners, extensive efforts have been done to achieve a smooth integration, by making tools compatible with each other, by developing gateways that allow different tools to interoperate, and by providing a unified user-interface. The funcionalities of the Eucalyptus toolbox include tools for:

- analysis (contains frontend tools performing lexical, sintactic, and static semantics analysis);
- simulation (the toolbox supports various forms of simulation, such as interactive simulation (step-by-step execution with backtracking), symbolic expansion (in which input values are handled symbolically), goal-oriented simulation, and random execution);
- exhaustive verification (the toolbox allows to generate the LTS corresponding to a LOTOS description. LTSs with millions of states and transitions can be generated, within the limits of memory available. These LTSs can be analyzed and verified in several ways. They can be minimized and compared modulo various equivalence (bissimulations) and preorder relations);
- compositional verification (due to the well-known state explision problem, exhaustive generation of LTSs is not always possible. The Eucalyptus toolbox allows to divide a LOTOS description into parallel

processes, to generate the LTSs corresponding to these processes, to minimize these LTSs modulo a bissimulation relation, and to build the LTS for the whole system by recombining these reduced LTSs);

- "on the fly" verification (as an alternative approach to avoid the state explosion problem, the Eucalyptus toolbox allows certain properties to be verified without generating the whole range from simple properties, such as deadlock detection and search of particular execution sequences, up to more elaborated properties such as "on the fly" comparisons of LTSs modulo bissimulation relations);
- graph drawing (the toolbox contains several tools to display the LTSs generated from LOTOS descriptions. For small LTSs (e.g., with less than one hundred states), these tools generate automatically a PostScript representation);
- test generation (from the LOTOS descriptions, one can automatically generate test sequences, which will be used to assess the conformity of real implementations with respect to the original descriptions);
- trace analysis (the Eucalyptus toolbox allows to verify whether a given trace (execution sequence) can be obtained from a LOTOS description); and finally,
- code generation (there are compilers to translate LOTOS types and process definitions into C code that can be executed and/or embedded in application programs).

These funcionalities are contained in the following tools:

- CAESAR: is a compiler that translates LOTOS descriptions into LTSs. These LTSs can be generated either in the BCG format or in other formats used by various verification tools.
- CAESAR.ADT: is a compiler that translates the data part of LOTOS descriptions into libraries of C types and functions. Each LOTOS sort is translated into an equivalenet C type and each LOTOS operation is translated into an equivalent C function (or macro-definition). The user can also decide to provide hand-written C code for some LOTOS sorts and/or operations.
- ALDEBARAN: is a tool for comparing and reducing LTSs (or networks of communicationg LTSs) modulo various equivalenece relatios (such as strong bisimulation, observational equivalenece, delay bisimulation, and safety equivalenece) and preorder relations (such as simulation preorder and safety preorder). For instance, one can check that the LTS of a protocol (generated using Caesar) is equivalent (modulo various abstraction criteria) to the LTS representing the service of this protocol.
- OPEN/CAESAR: is an extensible environment for designing programs performing simulation, execution, verification, and test generation. Several application programs are currently available within the Open/Caesar framewok, including:
    - Executor: is a random execution tool; (See the Figure 5.4)

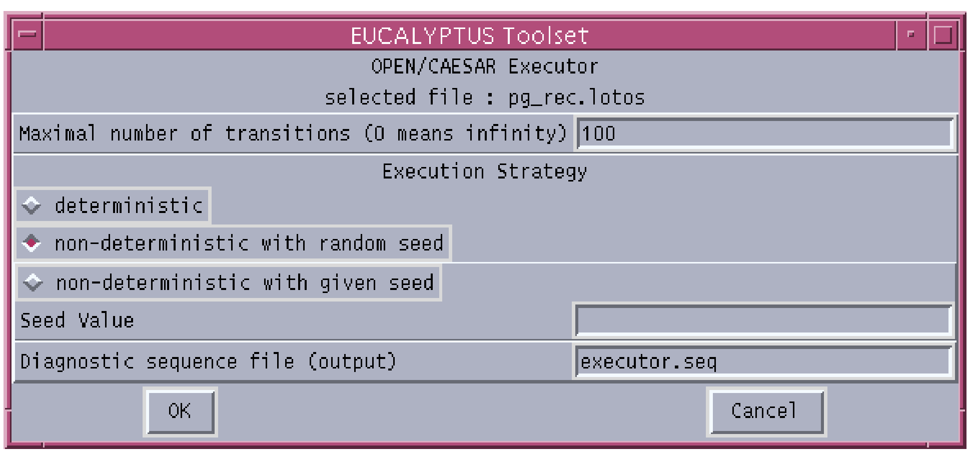

Figure 5.4 - Chosing execution strategy.

    - Exhibitor: seraches for execution sequences mathching a given pattern defined by a regular expression; (See the Figure 5.5)

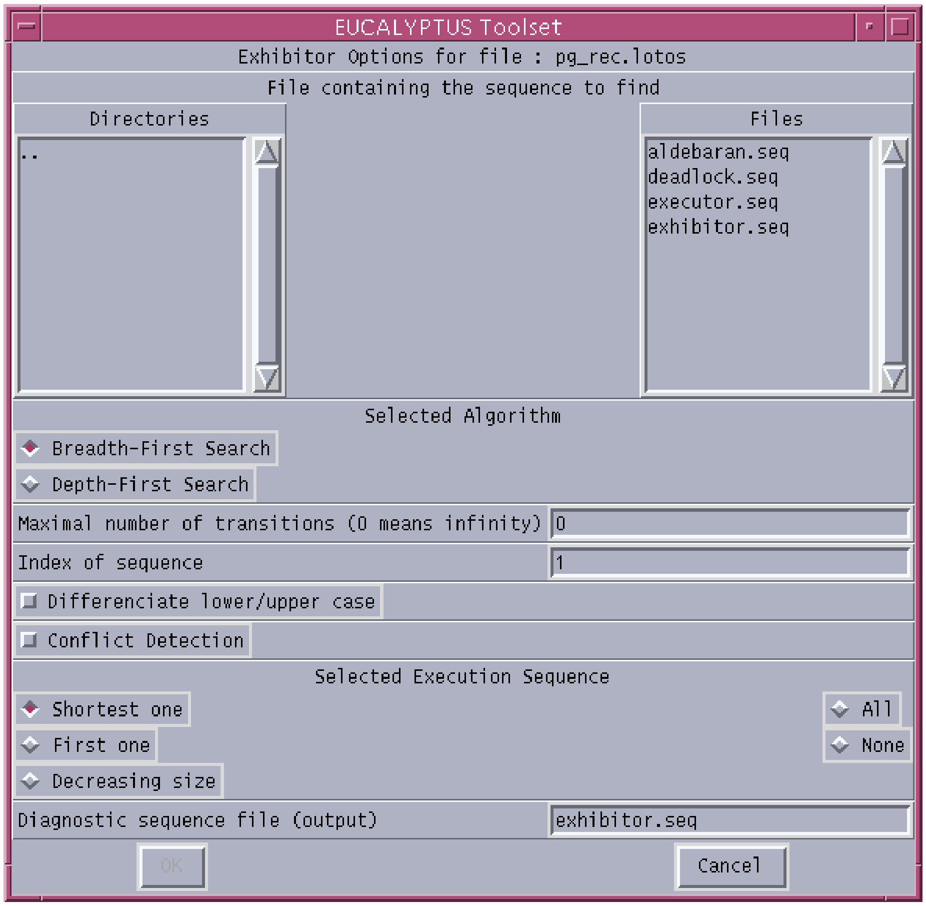

Figure 5.5 - Search for Execution Sequences.

- Generator: performs reachability analysis and generates an LTS;
- Simulator: is an interactive simulator with a command-line interface;
- Terminator: is a deadlock detection tool; (See the Figure 5.6)

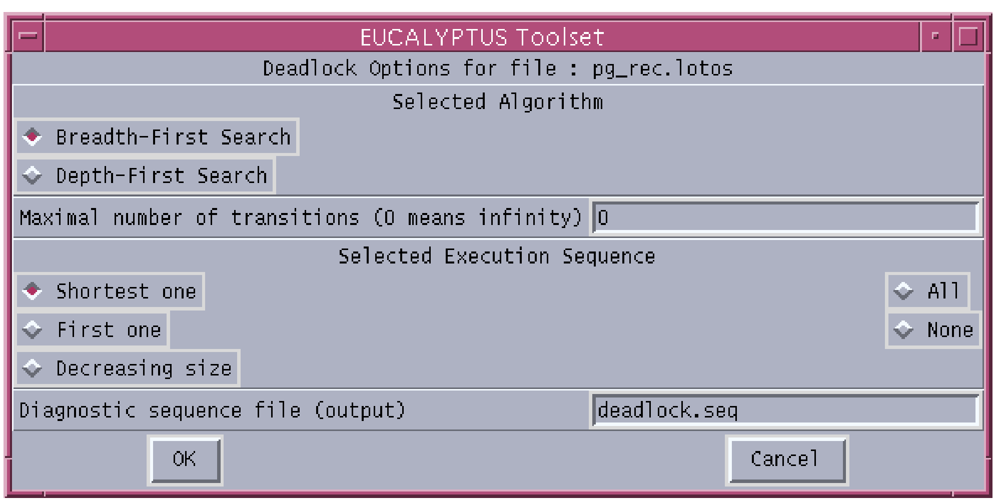

Figure 5.6 - Detecção de deadlock.

    - Xsimulator: is another interactive simulator eith a graphical user interface based on X-Windows.
- BCG (Binary-Coded Graphs): is both a format for the representation of explicit LTSs and a collection of libraries and programs dealing with this format. Compared do ASCII-based formats for LTSs, the BCG format uses a binary representation with compression techniques resulting in much smaller (up to 20 times) files.
- ISLA: provides a step-by-step execution mode wich alllows to simulate the sequence of possible actions that are permitted by a LOTOS description. The execution of a LOTOS description can be represented as a tre, where the root of the tree is the description itself, the intermediate nodes are behaviour expressions and the branches of the tree represent LOTOS actions. The user may choose to simulate the whole description at once, or only parts of it (certain processes). At each step, during simulation, the user is prompted with a menu of possible next actions. The user chooses the next action to be executed and, if the selected action requires data to be supplied by the environment (the user plays the role of the environment), then data must be entered for the simulation to continue. It is possible to save the sequence of actions, executed up to some point in the tree, in the memory or in an external file, thereby gaining the possibility of continuing the simulation, at a later time, from where it was left off.
- TESTGEN: is a tool for generating optimal test sequences from the LOTOS description of a protocol, in order to check the conformance of a protocol implementation to its formal description. This approach , combined with powerful optimization techniques produces optimal test sequences, which check the conformance of a protocol implementation by performing a minimal cost tour of the reference LTS. This method overcomes the limited controllability and observability of the protocol implementation by an external tester.
- TETRA (Test and Trace Analyzer): compares a given trace of interactions with a reference description written in LOTOS, checking whether an execution history of the LOTOS description could produce the given trace. The tool allows for two modes of analysis:
    - off-line trace analysis, where the reference description is compiled together with the traces to be analyzed, wich are written in the form of LOTOS process;
    - on-line trace analysis, wich compiles the reference description alone and analyses the interactions of a trace one after the other as they are received from another site executing/simulating the implementation under test. The result of a trace analysis is either “valid trace” or “invalid trace”.
- VISCOPE: is a tool for automatic display (in 2- or 3-dimensions) of LTSs. It generates a PostScript representation with a layout of states and transitions that make concurrent actions visible.

Using TOPO tool (www.dit.upm.es) the LOTOS specifications is translated to C code automatically.

6. Evaluation and Conclusion

During the development of this work it was observed that the utilization of a formal technique for the description of managed objects behaviour presents benefits in relation to the use of the natural language. Among these benefits the non existence of ambiguous interpretations for the same text can be quoted. The use of LOTOS for describing behavioural issues may demand more knowledge than the simple specification in ordinary text, however the obtained specification provides a more reliable implementation.

The Eucalyptus toolset was very efficient in the validation processes. It is includes a nice graphical generation of LTSs, important proofs of absence of deadlocks, good performance in simulations (exaustive and interative) and even so in the task to find sequences os events occourence.

As future works, it is intended to apply the concepts presented here for the specification of issues relating to the behaviour of other objects and their implementation.

7. Bibliographic Reference

[AgSo94] AGOULMINE, N.; SOUZA, J.N.; PAVLOU, G.: "**Design and implementation of Telecommunication management network: system and information viewpoints"**. In: Proceedings of the ITS'94, SBT/IEEE International Telecommunications Symposium. Rio de Janeiro, August 22-26th, pp. 342-346.

[DrGo94] DRUMMOND, R.; GONÇALVES JUNIOR, C.: **"Distributed Objects in MC"**. 12th SBRC Curitiba (PR), pp.188-201,1994.

[DyAi 94] DYSART, H.; AIDAROUS, S.: "**Subnetwork management and TMN in evolving network"**. In: Proceedings of the ITS'94, SBT/IEEE International Telecommunications Symposium. Rio de Janeiro, August 22-26th, pp 337-341.

[Gara 96] GARAVEL, H. Cæsar Aldébaran Distributed Package. Version Z. Grenoble - France, 06/12/1996.

[ISO 10165-4] ISO/IS 10165-4: Information Technology - Open Systems Interconnection Structure of Management Information - Part 4: **Guidelines for Definition of Managed Objects**. 08/1991.

[ISO 8807] Information Processing System-Open Systems Interconnection - LOTOS - **A formal description technique based on the temporal ordering of observational behaviour**, 1988.

[MeNo96] MELLO, B. A.; NOTARE, M. S. M. A.; RISO, B. G.: **"Using LOTOS on the design of portable agents for networks management"** (in Portuguese). Proceedings of the VI INFOWEEK- Computation Week of UFBA, pp.313-323. 1996.

[MoSt94] MOUTINHO, C. M. P.;STANTON, M. A.: **"Applications on inteligent networks management"** . 12th SBRC, Curitiba (PR), pp.154-172.1994.

[NoRi 97] NOTARE, M. S. M. A; RISO, B. G.; LORENA, P. S.; PENNA, M. C. de O.; WESTPHALL, C. B. Formal Design of a Telecommunications Networks Management System. AT&T, IEEE ISCC'97 - International Symposium on Computers and Communications, Alexandria, Egito, 07/1997. (Trabalho aceito).

[RaFr94] RAY, P.; FRY, M.: "**Interpreted service management within heterogeneous environments"**. In: Proceedings of the ITS'94, SBT/IEEE International Telecommunications Symposium. Rio de Janeiro, August 22-26th,, pp.347-351.

[RoFr96] RODRIGUES, R.; FREIRE, P.; OLIVEIRA, M.: **"A methodology for the development of Applications on the OSIMIS platform based on Configuration paradigm"** . 14th SBRC, Fortaleza (CE), pp.411-432, 1996.