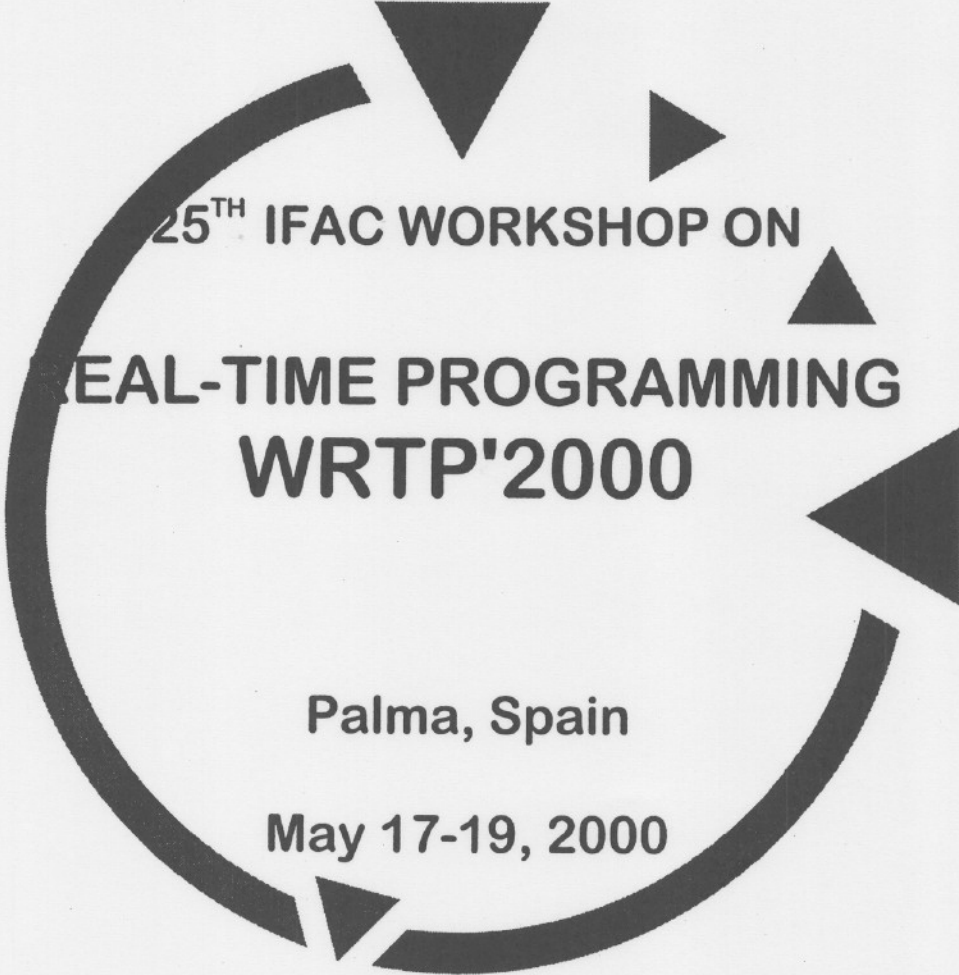




INTERNATIONAL FEDERATION OF AUTOMATIC



25TH IFAC WORKSHOP ON
REAL-TIME PROGRAMMING
WRTP'2000

Palma, Spain

May 17-19, 2000

Preprints edited by:
Alfons Crespo & Joan Vila



Organised by
Universitat Politècnica de València
Universitat de les Illes Balears



Alfons Crespo

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Servicio de Publicaciones

SPUPV-2000.2270

Copyright © 2000 IFAC

All Right Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means: electronic, electrostatic, magnetic tape, mechanical, photocopying, recording or otherwise, without permission in writing from the copyright holders.

First Edition 2000

These preprints were reproduced by means of the photo-offset process using the manuscripts supplied by the authors of the different papers. The manuscripts have been typed using different typewriters and typefaces. The layout, figures and tables of some papers did not agree completely with the standard requirements; consequently the reproduction does not display complete uniformity. To ensure rapid publication this discrepancy could not be changed; nor could the English could be checked completely. Therefore, the readers are asked to excuse any deficiencies of this publication which may be due to the above mentioned reasons.

The Editors.

Edita: Servicio de Publicaciones de la Universitat Politècnica de València
Camino de Vera s/n
46071 València
Tel: +34 963 877 012
Fax: +34 963 877 912

Depósito Legal: V-1768-2000

REAL-TIME PROGRAMMING 2000 (WRTP'2000)

*A Proceedings volume from the 25th IFAC Workshop,
Palma, Spain, 17 - 19 May 2000*

Edited by

A. CRESPO and J. VILA
*Departament d'Informàtica de Sistemes i Computadors,
Universitat Politècnica de València,
Valencia, Spain*

RECEIVED
4-5-01
C1144701
6100042072
CS400 0 IFA REA
18076

Published for the

INTERNATIONAL FEDERATION OF AUTOMATIC CONTROL

by

PERGAMON
An Imprint of Elsevier Science

PERGAMON
INTERNATIONAL FEDERATION OF AUTOMATIC CONTROL

UK
USA
JAPAN

Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington, Oxford, OX5 1GB, UK
Elsevier Science Inc., 660 White Plains Road, Tarrytown, New York 10591-5153, USA
Elsevier Science Japan, Tsunashima Building Annex, 3-20-12 Yushima, Bunkyo-ku, Tokyo 113, Japan

Copyright © 2000 IFAC

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means: electronic, electrostatic, magnetic tape, mechanical, photocopying, recording or otherwise, without permission in writing from the copyright holders.

First edition 2000

Library of Congress Cataloging in Publication Data

A catalogue record for this book is available from the Library of Congress

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 0-08-043686 2

These proceedings were reproduced from manuscripts supplied by the authors, therefore the reproduction is not completely uniform but neither the format nor the language have been changed in the interests of rapid publication. Whilst every effort is made by the publishers to see that no inaccurate or misleading data, opinion or statement appears in this publication, they wish to make it clear that the data and opinions appearing in the articles herein are the sole responsibility of the contributor concerned. Accordingly, the publisher, editors and their respective employers, officers and agents accept no responsibility or liability whatsoever for the consequences of any such inaccurate or misleading data, opinion or statement.

Printed in Great Britain

CONTENTS

INDUSTRIAL PRESENTATIONS

- Applied Research: A Scientist's Perspective 1
U. SCHMID
- Systems Engineering of a Successful Train Control System 9
H.W. LAWSON

EMBEDDED SYSTEMS

- A Contract-Based Language for Embedded Control Systems 17
J. EKER, A. BLOMDELL
- Dynamic Processes in a Static Environment 23
HO. TRUTMANN
- Design and Programming of Peripheral Interfaces for Embedded Real-Time Control Systems 29
M. COLNARIĆ, D. VERBER

REAL-TIME OPERATING SYSTEMS

- Early Experience with an Implementation of the POSIX.13 Minimal Real-Time Operating System for Embedded Applications 35
M. ALDEA RIVAS, M. GONZÁLEZ HARBOUR
- An Approach to Symbolic Worst-Case Execution Time Analysis 43
G. BERNAT, A. BURNS
- Combined Intrinsic-Extrinsic Cache Analysis for Preemptive Real-Time Systems 49
A. MARTÍ, X. MOLERO, A. PERLES, F. RODRÍGUEZ, J.V. BUSQUETS
- Predictable and Efficient Memory Management for Composite Events 57
J. MELLIN

DEPENDABLE SYSTEMS

- Safety Related Real Time Programming 63
W.A. HALANG, G. TSAI
- An Approach to Dependability Modelling of Real Time Systems 69
P. YUSTE, J.C. CAMPELO, P.J. GIL, J.J. SERRANO
- Improving Temporal Behavior with Graphical Method in Real-Time Systems 75
F. COTTET, L. DAVID
- Adaptive Distributed Real-Time Transaction Management in Safety-Critical Systems 81
H.F. WEDDE, S. BÖHM

REAL-TIME SCHEDULING

- Scheduling Real-Time Systems by Means of Petri Nets 89
E. GROLLEAU, A. CHOQUET-GENIET
- Real Time Scheduling Methods Requirements in Distributed Control Systems 95
P. MARTÍ, R. VILLA, J.M. FUERTES, G. FOHLER
- Some Practical Results about Fixed-Priority Scheduling and Offsets 103
J.M. LÓPEZ, J.L. DÍAZ, M. GARCÍA, D.F. GARCÍA

FORMAL METHODS

Formal Specification of a Safety Shell in Real-Time Control Practice A.E.K. SAHRAOUI, E. ANDERSON, J. van KATWIJK, J. ZALEWSKI	109
Schedulability Analysis in Real-Time Embedded Systems Specified in SDL J.M. ÁLVAREZ, M. DÍAZ, L. LLOPIS, E. PIMENTEL, J.M. TROYA	117
Analyzing Temporal Constraints with Binary Decision Diagrams D. DELFIEU, P. MOLINARO, O.H. ROUX	123

OBJECT ORIENTATION IN REAL-TIME SYSTEMS

A Generic Framework for Quantitative Modeling of Real-Time Systems in UML B. SELIC	129
Migrating from a Non-Object-Oriented to a Traceable Object-Oriented Process F. BORDELEAU, M.A. CLOUTIER, D. BIAGÉ	135
A Scheduling Strategy to Preserve Object Autonomy J.L. CONTRERAS, J.L. SOURROUILLE	143
Achieving Hard Real-Time Java M.J. BAXTER, J.M. BASS	151

VERIFICATION AND VALIDATION OF REAL-TIME SYSTEMS

Verification of Both Functional and Timing Requirements of Real-Time Systems S. WANG	157
Basic Environment for Real Time Systems Analysis Using CAN Bus M. MARCOS, J. PORTILLO	163
A Tool for Modular Modelling and Verification of Hybrid Systems D. BEYER, H. RUST	169
An Approach to Use System Models to Assist in the Definition of Test Procedures J. GARBAJOSA, H. GARCIA, M. ALANDES, M-A. MAHILLO, M. PIATTINI	175

DISTRIBUTED RT SYSTEMS

A Framework for Developing Distributed Hard Real-Time Applications J.J. GUTIÉRREZ GARCÍA, M. GONZÁLEZ HARBOUR	183
Methodology and Tool Support for Developing Distributed Real-Time Applications C. BRUDNA, C. MITIDIERI, C. PEREIRA, J. KAISER	191

APPLICATIONS

On the Use of a Railroad Model for Real-Time Systems Teaching and Experimenting A. ALONSO, R.F. ALFONSO, J.F. RUÍZ, M. GARCÍA-VALLS	197
Database Model Update Ordering for Fault Handling in Manufacturing Cells S-A. ANDRÉASSON, A. ADLEMO	203

Author Index	211
--------------	-----

AN APPROACH TO USE SYSTEM MODELS TO ASSIST IN THE DEFINITION OF TEST PROCEDURES¹

**Juan Garbajosa, Hector Garcia, Maria
Alandes**

Technical University of Madrid (UPM)
E.U. Informatica. Dpt. OEI
Ctra. Valencia Km. 7, E-28031 Madrid, Spain
Fax: +34-913367520, jgs@eui.upm.es

Maria-Angeles Mahillo

Technical University of Madrid (UPM)
E.U. Informatica. Dpt. LPSI
Ctra. Valencia Km. 7, E-28031 Madrid,
Spain
amahillo@eui.upm.es

Mario Piattini

University of Castilla - La Mancha (UCLM)
E.U. Informatica.
Ronda de Calatrava 5, E-13004 Ciudad Real. Spain
mpiattin@inf-cr.uclm.es

ABSTRACT

Testing is crucial for all kind of systems but for those including real-time features becomes, perhaps, even more crucial. Testing of Complex Systems, at end user level, is more and more often performed in a systematic fashion using test procedures.

The test engineer has usually to produce test procedures in low-level programming languages and with little or no assistance. Within this paper we present an approach in which we outline how a model of the system to be tested can be used to provide assistance to the test engineer.

Copyright © 2000 IFAC

Keywords: system testing, acceptance testing, assistance, validation, tools, environments, model-based testing, test procedures

1. INTRODUCTION

Testing is crucial for all kind of systems but for those including real-time features becomes, perhaps, even more crucial. To perform testing satisfactorily is becoming more and more difficult and costly as long as systems become more and more complex. During the last years a huge effort to improve testing of systems has taken place. Our research effort are focussed on testing from the end-user perspective, known often as acceptance testing,

Among systems to test we can set two groups depending on its behavioural model. The behavioural model must account or not for sequences of operations in which later operations depend on actions taken by earlier operations. For the first group the generation of test cases is still a

subject under research as stated in (Dalal et al., 1999), while in the second group we are starting to have approaches with promising results as we can see in section 2.

Our research work is focused on the first group, for which test procedures (TP) design lays on the knowledge the test engineer. Integration of our developments with some performed for the second group of systems or subsystems is being considered.

This paper presents an approach based on the experience obtained from the work on the subject in the satellite industry (Garbajosa et al., 1997). Satellites require a careful testing before they are launched. Afterwards it is, usually, too late to fix operation mishaps.

¹ This research work was performed within the ARCO project. The Inter-ministerial Commission for Science and Technology of the Government of Spain, CICYT, partially supports the Project ARCO Ref. TIC98-0782.

Our approach is based on providing support to the test engineer in the definition and execution of test procedures by means on an environment specifically designed and built with that objective. At present we are focusing on the telecom domain, specifically network management.

As discussed in (Garbajosa et al., 2000) we claim that our approach is significantly independent of the domain. Correspondence between our present work, and other in the space domain (Valera, 1999) is a confirmation of this statement.

Even when it is not relevant to the main subject presented within this paper, we would like to mention that one of our objectives is to achieve low-cost environments, affordable to industry. A domain independent approach may help to accomplish this objective.

This paper is structured as follows: within this section, Introduction, we present the scope of the article. In Related Research Activities in System Testing we just outline some related work. Then we present a Assisted Test Procedure Definition: Concept and Environment Architecture, where we summarise the concept for Assisted test procedure definition and the Environment architecture to support it.

Next, we describe the Model Representation of our system, in the telecom domain. It is actually a subschema of our model, but it will facilitate the understanding of how assistance is supported. Then, in Implementing Assistance, we present some mechanisms we use to support assistance based on the system model, and finally, some Conclusions are drawn, outlining future work.

2. RELATED RESEARCH ACTIVITIES IN SYSTEM TESTING

Approaches available in literature are mostly applicable to systems for which a data model is sufficient to capture the system's behaviour, that is, control information is not required in the model. As stated in (Dalal et al, 1999) the complexity of the system under test's response to a stimulus must be relatively low.

Model-based testing, can be found in (Dalal et al, 1999). Testers using this approach concentrate on a data model and generation infrastructure instead of hand-crafting individual tests.

In (Cunning et al. 1999) an algorithm to automatically generate event scenarios is presented. This algorithm extracts the needed information from the requirements forms and produces a set of scenarios that can be used to test transaction oriented systems. Related approaches can be found in (Lamsweerde et al.1998) and (Sutcliffe et al. 1998).

In (Benjamin et al. 1999), it is possible to bridge the gap between formal verification and simulation by using hybrid techniques. Traditional techniques based on simulation can only verify operation over a small subset of the state space and it is often hard to generate tests to hit interesting corner cases.

A discussion and a comparison of these approaches and that described within this paper can be found In (Garbajosa et al, 2000).

3. ASSISTED TEST PROCEDURES

DEFINITION: CONCEPT AND ENVIRONMENT ARCHITECTURE

To produce TP the user (test engineer or end user) must be acquainted with a lot of information about the system to test and programming languages. He must be able to fill a lot of gaps about the system. He also needs to have a wide knowledge on computing issues such as configuration, and distribution of systems, and so on. In most of cases it should be desirable to free control and test engineers from knowing the entire structure of those complex systems.

3.1. ASSISTED TEST PROCEDURE CONCEPT

Measurement and action concepts are the base of test definitions. Test activities involve the generation of actions and the visualisation and analysis of measurements. Actually, measurements verify that the actions performed –by means of the test engineer commands– get the desired effect. In object oriented terms, actions may be associated to object operations and measurement to values returned. In the satellite application domain, these are called Telemetry & Telecommand, usually known as *TM/TC* to collect under the same term a widest set of forms to represent communication between the test system and the Man Machine Interface. We shall designate the system to be tested as *Unit Under Test*. Our developments are based on this scheme.

As we already mentioned system test procedure definition is a complex activity that requires the test engineer to be fully acquainted with several areas such as the physical systems for which the tests are being defined, and programming techniques.

Testing of these systems requires the production of series of commands that validate the system operation, in a repetitive fashion. These series of procedures are often called *test procedures*. Test procedures are produced using programming languages.

The gap between the way the definition is performed and what the engineer should understand as a natural way of working is quite significant. Thinking in terms of test engineers, that might be not so familiarised with computer concepts as computer scientists are, the use of these programming languages requires to work at low level of abstraction, farther away from what the real problem is. Programming languages used (C, Atlas, Tcl/Tk ETOL) are frequently not high level ones. But even high level programming languages, such as object oriented, are not yet so close to real life systems as we would like to be.

Therefore it makes a lot of sense to assist the test engineer in his work. This scheme is valid not only for the satellite domain but also for most of the application domains with systems to be tested. Figure 1 shows a comparison between a traditional environment, to define and execute test procedures, and those under development. A further discussion and a comparison of traditional versus assisted test procedure definition can be found in (Garbajosa et al. 1999) and (Garbajosa et al. 2000).

Therefore the test engineer faces three problems:

- To design—define—the test procedures that will validate the system.
- To translate these test procedures into a programming language.
- To execute these tests procedures with little or no help of the test environment.

Derived problems to cope with are the following:

- To maintain tests procedures. Maintainability will also get an advantage from the approach we propose.
- Guarantee full test coverage

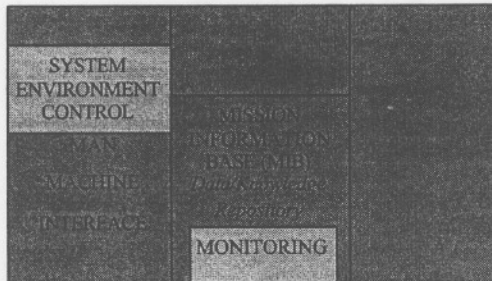


Figure 1: TOPEN Environment Architecture

3.2. TEST AND OPERATION ENVIRONMENT ARCHITECTURE

In [Garbajosa et al., 2000] the basic components of the Test and Operation Environment Architecture (TOPEN) are described. Here we shall just mention them.

The conceptual architecture can be implemented, in terms of components, as follows: Man Machine Interface, Data/knowledge Repository and code generator. The test engineer handles the Man

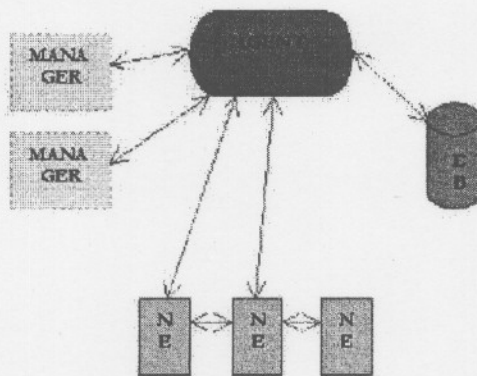


Figure 2: Network Management System

Machine interface. The System Environment Control subsystem allows the test engineer to modify the system state so that situation that represent incidents can be supported.

A data and knowledge repository is used to support a model of the system. Other components are a code generator module, which generates the code that will get into the industrial system and, finally, a low-level interface between the industrial system and environment would be needed. As we shall see below these basic

components can be refined into a more detailed architecture. Monitoring can be performed working with MIB data.

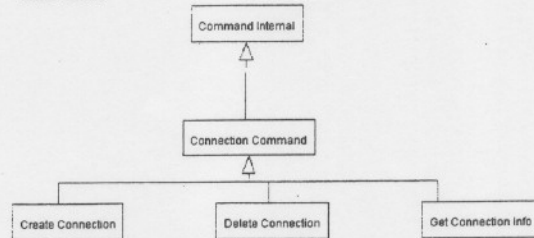


Figure 3. Command data Model

4. MODEL REPRESENTATION

As we outlined in section 3 UUT subsystems and components can be represented by objects, and actions by operations. Measurements are the result or output of applying an action on an object.

In figure 2 we can see a Network Management System schema. An Agent receives commands from a number of Managers that represent operators. Those commands handle a number of network Elements (NEs). Information needed to handle this is in a database. The TOPEN man machine interface implements the Managers man machine interface. Actually TOPEN interacts directly with the Agent.

In figure 3 we can see a subschema for the Command data model. It shows the class hierarchy for the connection-related commands. We have that a Connection command is a subclass of Commands. A connection command can be to create, delete or get connection. These commands would operate the UUT, that is the Network management system.

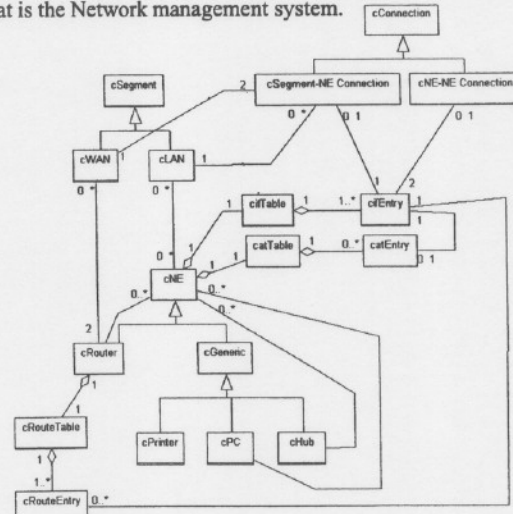


Figure 4. Network Model (UUT).

Now we can analyse the UUT model. This model, another subschema shown in figure 4, has LAN and WANs as central components. For our example we shall focus our attention in WANs, though we kept LANs for easier understanding. Connection relates network elements, as segments. The information of the different connections is described in the route tables, with its route entries. We see that a network element can be either a router or a generic. In the example of next section all the NE will be routers, that are the nodes of specific LANs.

But only the routers will be of interest to us for our purposes.

5. IMPLEMENTING ASSISTANCE

We can use two different situations to show examples of how assistance can be provided. The first one, *providing assistance to set connections*, will describe a situation in which the test engineer gets assistance based on the structural characteristics of the model. It is a rather static view of the model.

In the second case, *creating new route tables*, there exist a response to an unexpected system state after an incident has taken place. It is a more dynamic view of the model even when it does not imply real simulation.

5.1. PROVIDING ASSISTANCE TO SET NE CONNECTIONS

In the case of network management systems the test engineer may have problems to know, at a given moment, which elements are currently installed, their physical and logical configuration, etc. However, this information is stored in the TOPEN MIB, as long as all the network elements (NEs) are stored there together with their state.

Let us suppose that the test engineer wishes to make a number of connections among several NEs to enable several new routes. Even when he may have a clear idea of the initial and final nodes, there exists an additional problem. Each NE supports a number of protocol interfaces, and he will be able to connect only those NE interfaces that are compatible, and not yet used in a connection. This fact can be considered as a structural constraint: connections can be established between compatible interfaces. The data model supports it, and its implementation would not allow the database to not to respect it. This is easily achievable, with a relational database implementation, such the one we are producing.

Therefore the test engineer, once he would select the *connect* command, he would be requested to indicate the first NE from the entire list of NEs and then the second, and if would press *OK*. This is shown in figure 5. If he failed to hit the NE with the right available interface, the test engineer could just receive a message such as *incompatible NE interfaces*. It may be worse: *data base operation rejected*. Or worse: *internal error xx*. Or even worse such as receiving a message from the Agent explaining that the operation failed, what additionally would overload the net. But for the moment little assistance has been provided.

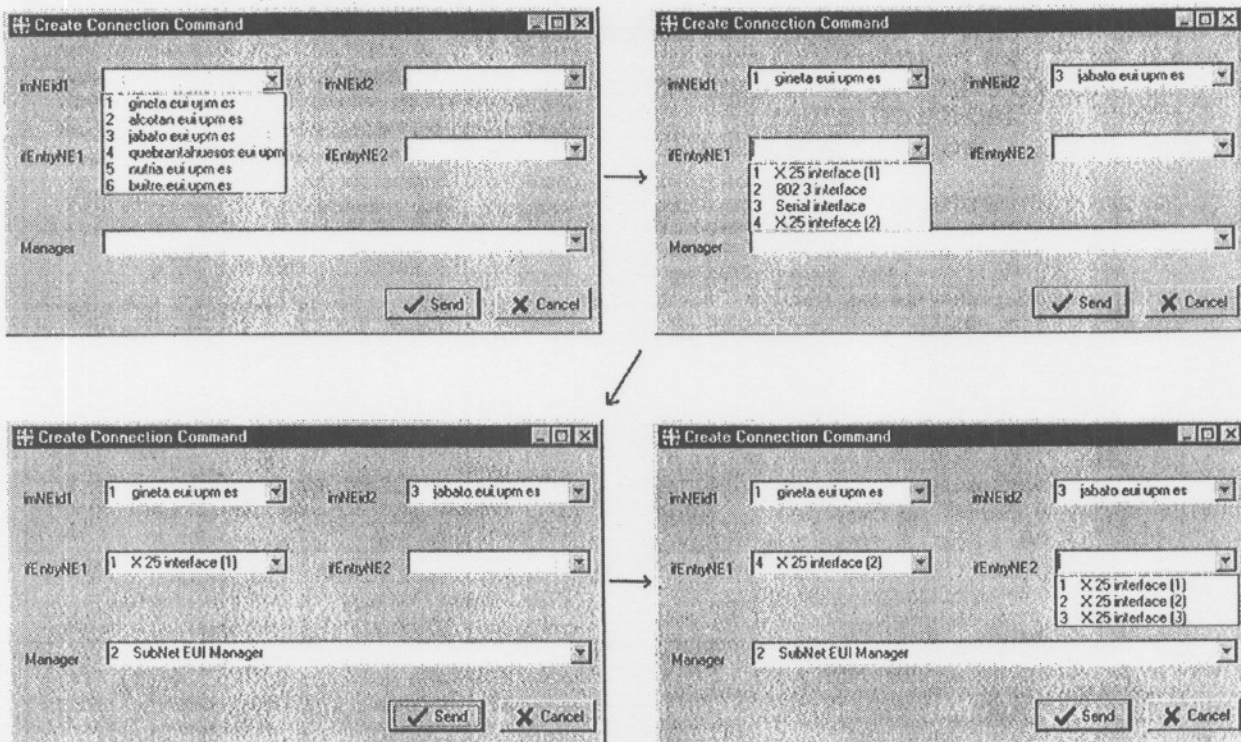


Figure 5 – Generation of connection command

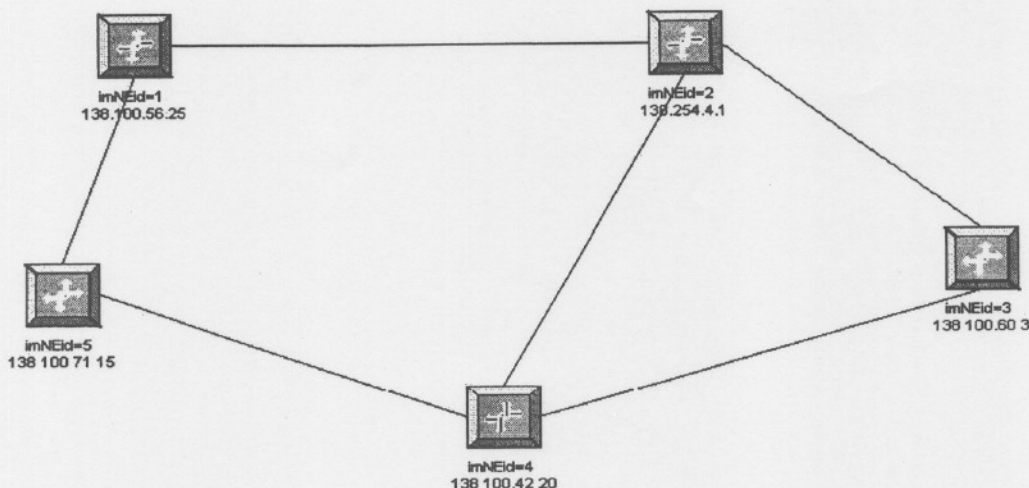


Figure 6. Router level configuration

Yet we can positively use the information stored in the MIB. Once introduced the first NE the TOPEN can show *only* those NEs with available interfaces that are compatible with those not yet used interfaces of the selected NE. Furthermore, when two instances of NE are selected, the system will allow the user to connect only those NE interfaces compatible and not connected.

In this way the TOPEN provides two desired functionalities:

- 1) Assistance to the user at the time to define commands or TP
- 2) We are avoiding overloading communication lines. As the TOPEN does not allow to define wrong commands, we are eluding those command packets that could not be executed, and, furthermore, both replies and notifications.

5.2. CREATING NEW ROUTE TABLES

If we are testing a Network Management System the Test and Operation Environment (TOPEN) will be able to detect a number of problematic situations, for which assistance provision may be useful.

To provide this assistance TOPEN needs to store every Command/Test Procedure that has been executed and every Reply or Notification received from the NMS. Knowing commands and responses it is possible to identify the current state of the elements of the network to be tested.

Let us assume a NMS, with routers as nodes, with the following forms route tables defined as follows:

ipRouteDest: The destination IP address of this route.

ipRouteIfIndex: The index value which uniquely identifies the local interface through which the next hop of this route should be reached.

IpRouteNextHop: The IP address of the next hop of this route. (In the case of a route bound to an interface which is realized via a broadcast media, the value of this field is the agent's IP address on that interface.)"

IpRouteMask: Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field.

ImNEid: The NE identifier in the network. This identifier must be unique for each NE instance.

Let us assume that we have a network as defined in figure 6. Let us also assume the following values for route table fields:

ipRouteDest	ipRouteIfIndex	ipRouteNextHop	IpRouteMask
138.254.4.0	2	138.254.4.1	255.255.4.0
138.100.71.0	1	138.100.71.15	255.255.71.0
138.100.42.0	1	138.100.71.15	255.255.42.0
138.100.60.0	2	138.254.4.1	255.255.60.0

Table 1 – imNEid = 1 ipRouteTable

ipRouteDest	ipRouteIfIndex	ipRouteNextHop	ipRouteMask
138.100.56.0	1	138.100.56.25	255.255.56.0
138.100.71.0	1	138.100.56.25	255.255.71.0
138.100.42.0	1	138.100.42.20	255.255.42.0
138.100.60.0	2	138.100.60.3	255.255.60.0

Table 2 – imNEid = 2 ipRouteTable

ipRouteDest	ipRouteIfIndex	ipRouteNextHop	IpRouteMask
138.100.56.0	2	138.254.4.1	255.255.56.0
138.100.71.0	2	138.100.42.20	255.255.71.0
138.100.42.0	2	138.100.42.20	255.255.42.0
138.254.4.0	1	138.254.4.1	255.255.4.0

Table 3 – imNEid = 3 ipRouteTable

IpRouteDest	IpRouteIfIndex	IpRouteNextHop	IpRouteMask
138.100.56.0	1	138.100.71.15	255.255.56.0
138.100.71.0	1	138.100.71.15	255.255.71.0
138.100.60.0	3	138.100.60.3	255.255.60.0
138.254.4.0	2	138.254.4.1	255.255.4.0

Table 4 – imNEid = 4 ipRouteTable

IpRouteDest	IpRouteIfIndex	IpRouteNextHop	IpRouteMask
138.100.56.0	1	138.100.56.25	255.255.56.0
138.100.42.0	2	138.100.42.20	255.255.42.0
138.100.60.0	2	138.100.42.20	255.255.60.0
138.254.4.0	1	138.100.56.25	255.255.4.0

Table 5 – imNEid = 5 ipRouteTable

If the NE with IP address 138.254.4.1 goes down, that is our NE with imNEid = 2, the Agent shall send an *equipmentAlarm* notification that will be received by the TOPEN. Though this incident might take place spontaneously, for testing purposes, the test engineer may induce it through the System Environment Control interface.

It may help to understand the example to mention that an address such as 138.100.56.0 for ipRouteDest means that destination may be any 138.100.56 router.

In this case assistance may be structured as follows:

1. Detect the scope of the incident
2. Suggest *what to do* to the test engineer
3. Provide a number of feasible choices if that is case

Once an *equipmentAlarm* notification has been received and processed the TOPEN will check if the system NEs configuration let the system remain operational. The decision will be taken according to the information stored within the MIB. If the answer is negative, as in this case, TOPEN will suggest the test engineer to include within the TP a command *set ipRouteTables* to modify the route tables for NEs with imNEid equal to 1 and 3 as shown below. Those are the tables affected by the faulty router. It will return the system to an operational situation even being the router faulty. Obviously those packets sent to the subnetwork controlled by 138.254.4.1 will be lost, but not those sent to those subnetworks that are routed through the router that has gone down, because an alternative route could be generated, as shown in the route tables shown below.

In this case TOPEN will be suggesting only one choice as alternative route table. This should be properly understood. Actually the example is very simple. In a real situation we might have several routes to choose and it should be the test engineer who decides.

IpRouteDest	IpRouteIfIndex	IpRouteNextHop	IpRouteMask
138.254.4.0	2	138.254.4.1	255.255.4.0
138.100.71.0	1	138.100.71.15	255.255.71.0
138.100.42.0	1	138.100.71.15	255.255.42.0
138.100.60.0	1	138.100.71.15	255.255.60.0

Table 6 – imNEid = 1 new ipRouteTable

IpRouteDest	IpRouteIfIndex	IpRouteNextHop	IpRouteMask
138.100.56.0	2	138.100.42.20	255.255.56.0
138.100.71.0	2	138.100.42.20	255.255.71.0
138.100.42.0	2	138.100.42.20	255.255.42.0
138.254.4.0	1	138.254.4.1	255.255.4.0

Table 7 – imNEid = 3 new ipRouteTable

In the case of network management systems a lot of problems could be handled according to this approach, such as those related to quality of service, processing errors, or congestion of the traffic

6. CONCLUSIONS

We have shown how, taking a system model as a basis, assistance can be provided to a test engineer as long as a specific environment is used. This approach is especially useful for systems whose behaviour depends on the result of former operations.

This approach is significantly domain independent. We are applying it to network management system and former work with satellite applications respects the same basic architecture. It will be necessary to systematise its development. We are working on this issue. This let us think of being able to build low-cost environments

Another issue of interest is that within this environment can also integrate tools that are used to support model-based testing as described in (Dalal et al., 1999). Our second assistance example could be understood as such integration.

Acknowledgements

Authors are indebted to Bryan Melton and Serge Valera for those discussions that, later, became fundamental to obtain some of the results. Authors are also indebted to F. Sanchis, C. Cuvillo, F. Arizmendi and E. Marcos for their useful comments. Authors are grateful to the students who have taken part in the implementation of the environment: Angelica Gonzalez, Esther Alonso, Antonio García y Ahmad Marcie

Generation from a Structured Requirements Specification. IEEE ECBS'99. S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton B. M. Horowitz.

Lamsweerde, A. van, and Willemet, L (1998). Inferring Declarative Requirements Specifications from Operational Scenarios. IEEE Transactions on Software Engineering. Vol. 24, No. 12: December 1998, pp. 1089-1114

Requirements Engineering. IEEE Transactions on Software Engineering. Vol. 24, No. 12: December 1998, pp. 1072-1088.

Dalal, S. R., Jain, A., Karunanithi, J. M. Leaton, J.M., Lott, C.M., Patton, G.C. and Horowitz B.M.(1999). Model-Based Testing in Practice. IEEE Int. Conf. on Software Engineering. ICSE '99 Los Angeles CA.

Benjamin, M., Geist, D., Hartman, A., Wolfsthal, Y., Mas, G., Smeets, R.(1999). A Study in Coverage-Driven Test Generation. ACM DACS'99.

Garbajosa, J. and Wolff M.(1997). Automatic Generation of Satellite Test Procedures. Proc. International Symposium Computing Tools, Systems Engineering and Competitiveness. Association Aeronautique et Astronautique de France (AAAF) Paris, March.

Garbajosa, J., Piattini, M., Mahillo, M.A., Garcia, H., (1999). A Development Environment to Support Assisted Definition of Test-Procedures for Complex Systems. Proc. ICSSEA Conf. Paris.

Garbajosa, J., Piattini, M., Mahillo, M.A., Alandes, M. (2000). Assisting the Definition and Execution of Test Suites for Complex Systems. Proc. IEEE ECBS'2000. Edinburgh, April.

Valera, S. (1999). EGSE and Mission Control System. The PROBA EGSE/SCOS2 Experience. ESTEC. European Space Agency. June.