ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Κατανεμημένα Συστήματα Διαχείρισης Σφαλμάτων σε δίκτυα ATM

Νικόλαος Κατσάλης

Μεταπτυχιακή Εργασία

Ηράκλειο, Φεβρουάριος 1999

ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΣΦΑΛΜΑΤΩΝ ΣΕ ΔΙΚΤΥΑ ATM

Νικόλαος Κατσάλης

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

ΠΕΡΙΛΗΨΗ

Η διαχείριση σφαλμάτων αποτελεί μια πολύ σημαντική λειτουργία της διαχείρισης δικτύων, η οποία ασχολείται με την επιτήρηση/εποπτεία σφαλμάτων, τον εντοπισμό των πόρων στους οποίους συνέβησαν και την επιδιόρθωσή τους. Ασχολείται επίσης με την εξέταση και ανάλυση της πληροφορίας σφαλμάτων και την εξαγωγή χρήσιμων συμπερασμάτων για την ανοχή του υπό διαχείριση δικτύου, σε περιπτώσεις σφάλματος, με σκοπό την αποδοτικότερη επαναδιαμόρφωση του.

Η εργασία αυτή ασχολείται με την μελέτη κατανεμημένων συστημάτων εντοπισμού και επιδιόρθωσης σφαλμάτων και την σχεδίαση και υλοποίηση ενός πρωτότυπου συστήματος λογισμικού, εντοπισμού και επιδιόρθωσης σφαλμάτων σε εικονικά μονοπάτια συνδέσεων (virtual path connections), στοιχεία (nodes) και φυσικές γραμμές (physical links) ενός δικτύου ασύγχρονου τρόπου μετάδοσης (ATM). Το σύστημα αυτό βρίσκεται από φυσική/μηχανική άποψη στο επίπεδο Διαχείρισης Δικτύου (Network Management Layer - NML) και η λειτουργικότητά του στηρίζεται στο Τμήμα Διαχείρισης Σφαλμάτων της Αρχιτεκτονικής Πόρων Δικτύου της TINA (Fault Management Fragment of TINA NRA), στο σύστημα που σχεδιάστηκε και υλοποιήθηκε στα πλαίσια της ερευνητική εργασίας του ACTS REFORM καθώς και σε επιμέρους απαιτήσεις που έχουν τα συστήματα διαχείρισης σφαλμάτων.

Σκοπός της εργασίας είναι τόσο η μελέτη της ευρύτερης περιοχής διαχείρισης σφαλμάτων όσο και η ειδικότερη μελέτη και επέκταση του συστήματος REFORM, παρέχοντας:

- Δυνατότητα εφαρμογής αλγορίθμων εντοπισμού της πρωταρχικής αιτίας των σφαλμάτων.
- Δυνατότητα, τόσο μέσα από την διατήρηση και εξέταση της πληροφορίας σφαλμάτων όσο και μέσα από την συσχέτισή της, για την εξαγωγή πολύτιμων στατιστικών για τους πόρους του υπό διαχείριση δικτύου.
- Ενσωμάτωση των λειτουργιών της διαχείρισης σφαλμάτων με τις λειτουργίες των άλλων περιοχών διαχείρισης, μέσω καλά ορισμένων διεπαφών και αλληλεπιδράσεων.

Επόπτης: Κώστας Κουρκουμπέτης
Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

DISTRIBUTED FAULT MANAGEMENT SYSTEMS OF ATM NETWORKS

Nikolaos Katsalis

Master of Science Thesis

Department of Computer Science
University of Crete

ABSTRACT

Fault Management is an important functional area of Network Management concerned with fault detection, localization and correction. Fault localization is achieved through the use of correlation algorithms which provide necessary knowledge of the network's fault tolerance aiding in the improved redesign of the managed network.

This work is concerned with the study of distributed fault detection and correction systems and the design and development of a prototype system that detects, surveys and corrects faults occurring on the virtual path connections (VPCs), links and nodes of an Asynchronous Transfer Mode (ATM) network. The proposed system is physical/engineering placed in the Network Management Layer (NML) and its functionality is based on TINA's NRA Fault Management Fragment as well as on the REFORM system and on general requirements that a fault management system could have.

The purpose of this work is to study the area of fault management as well as to study and expand the REFORM system by:

- Providing the framework for the utilization of fault localization algorithms aimed at determining the primary cause of faults.
- Providing the capability of deducing valuable statistics of the managed network's resources through fault event logging and fault correlation algorithms.
- Providing integration of fault management and other fault management functional areas through well defined interfaces and interactions.

Supervisor: Costas Courcoubetis
Professor of Computer Science
University of Crete

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα εργασία δεν θα είχε πραγματοποιηθεί χωρίς την βοήθεια ορισμένων ατόμων στα διάφορα στάδια της πορείας της, τους οποίους και θα ήθελα να ευχαριστήσω.

Αρχικά θα ήθελα να ευχαριστήσω τον επόπτη καθηγητή της εργασίας, κ. Κώστα Κουρκουμπέτη, ο οποίος μου έδωσε τη δυνατότητα υλοποίησης της συγκεκριμένης εργασίας καθώς και την δυνατότητα συνεργασίας μου και με άλλους ερευνητές του τομέα αυτού. Τον κ. Απόστολο Τραγανίτη και την κ. Κατερίνα Χούστη, μέλη της εισηγητικής επιτροπής, για την συμβολή τους στην ολοκλήρωση της εργασίας κάνοντας σχολιασμό της.

Επίσης για την σημαντική τους βοήθεια θα ήθελα να ευχαριστήσω:

Τον Στέλιο Σαρτζετάκη, ερευνητή και συνεργάτη της ομάδας Τηλεπικοινωνιών και Δικτύων του Ινστιτούτου Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας, με τον οποίο συνεργάστηκα στενά στον τομέα της εργασίας μου και η συμβολή του οποίου ήταν καθοριστική στην διεκπεραίωσή της.

Το Φώτη Κίτσο και τη Μάγδα Χατζάκη για την πολύτιμη βοήθειά τους σε διάφορα στάδια της παρούσας εργασίας και ιδιαίτερα κατά την συνεργασία μας στο ερευνητικό πρόγραμμα του ACTS REFORM..

Επίσης όλα τα υπόλοιπα μέλη της ομάδας Τηλεπικοινωνιών και Δικτύων για το καλό κλίμα συνεργασίας που υπήρχε κατά την διάρκεια της παρούσας εργασίας.

Τέλος, το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης και το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την υλικοτεχνική υποστήριξη που μου παρείχαν.

στους αγαπημένους μου γονείς,
Δημήτρη και Θεανώ

Πίνακας Περιεχομένων

Πίνακας Περιεχομένων *7*

Πίνακας Σχημάτων *11*

1 *Εισαγωγή* *12*

1.1 Η Διαχείριση Σφαλμάτων 12

1.2 Αντικείμενο Εργασίας 13

1.3 Οργάνωση Κειμένου 14

2 *Υπόβαθρο* *15*

2.1 Δίκτυο Διαχείρισης Τηλεπικοινωνιών (ITU-T TMN) 15

2.2 Αρχιτεκτονική Τηλεπικοινωνιών και Πληροφορίας Δικτύωσης (Telecommunication Information Networking Architecture - TINA) 17

2.2.1 Συμβατότητα με την Αρχιτεκτονική TINA 18

2.2.2 Η Αρχιτεκτονική Πόρων Δικτύου (Network Resource Architecture - NRA) 18

2.2.3 Η Αρχιτεκτονική Διαχείρισης Σφαλμάτων (Fault Management - FM Architecture) 19

2.3 Η Αρχιτεκτονική CORBA 24

2.3.1 Περίληψη 24

2.3.2 Ονομασία (Naming) 25

2.3.3 Μοντέλο Οντοτήτων (Object Model) 25

2.3.4 Αρχές της CORBA (CORBA Principles) 25

2.3.5 Υλοποιήσεις ORB (ORB Implementations) 26

2.4 Η Υπηρεσία Καταλόγου X.500 27

2.4.1 Το μοντέλο της πληροφορίας του X.500 27

2.5 Το σύστημα REFORM 28

2.5.1 Θέματα Σχεδίασης και Απαιτήσεις 29

2.5.2 Χρησιμοποιούμενες Τεχνολογίες 30

2.5.3 Ο Κόμβος REFORM (REFORM Node) 31

3 *Προδιαγραφή Λειτουργιών Συστήματος* *33*

3.1 Χαρακτηριστικά Διαχείρισης Σφαλμάτων και Πληροφορία Τοπολογίας Πόρων 34

3.1.1 Χαρακτηριστικά Διαχείρισης Σφαλμάτων 34

3.1.2 Πληροφορία Τοπολογίας Πόρων 35

3.1.3 Ανάθεση Χαρακτηριστικών Διαχείρισης και Αρχικοποίηση/Σταμάτημα των διαδικασιών της 35

3.2 Επιτήρηση/Εποπτεία Ειδοποιήσεων Σφαλμάτων 36

3.3 Δοκιμή πόρων 36

3.4 Συσχέτιση Ειδοποιήσεων και Εντοπισμός πρωταρχικής αιτίας σφάλματος (Fault Correlation-Localization) 37

3.5 Δημιουργία και Διατήρηση του χάρτη στατιστικών των πόρων του δικτύου (Network Risk Map) 38

4 *Προδιαγραφή Υπολογιστικής Συστήματος* *41*

4.1 Γραφικοί συμβολισμοί σχημάτων παρουσίασης 41

4.2 Υπολογιστικές Οντότητες Συστήματος 42

4.2.1 Ο Συντονιστής Διαχείρισης Λαθών (FM Coordinator) 42

4.2.2 Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Σφαλμάτων (FM Alarm Manager) 42

4.2.3 Ο Εξυπηρετητής Δοκιμών Διαχείρισης Σφαλμάτων (FM Testing Server) 43
4.2.4 Η Γραφική Διεπαφή Χρήστη Διαχείρισης Σφαλμάτων (FM GUI) 43

4.3 Ο Συντονιστής Διαχείρισης Σφαλμάτων 43
4.3.1 Εσωτερική Λειτουργικότητα 43
4.3.2 Παρεχόμενες Διεπαφές 44

4.4 Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Λαθών 45
4.4.1 Εσωτερική Λειτουργικότητα 45
4.4.2 Παρεχόμενες Διεπαφές 47

4.5 Εξυπηρετητής Δοκιμών Διαχείρισης Σφαλμάτων 48
4.5.1 Εσωτερική Λειτουργικότητα 48
4.5.2 Παρεχόμενες Διεπαφές 48

4.6 Η Γραφική Διεπαφή Χρήστη Διαχείρισης Σφαλμάτων 49
4.6.1 Εσωτερική Λειτουργικότητα 49
4.6.2 Παρεχόμενες Διεπαφές 50

4.7 Συνολικό Σύστημα 50
4.7.1 Περιγραφή Αλληλεπιδράσεων Υπολογιστικών Οντοτήτων 51

5 Προδιαγραφή Μοντέλου Πληροφορίας Συστήματος 56

5.1 Παρουσίαση Εξωτερικής Πληροφορίας και Διεπαφών 56
5.1.1 OMT Διάγραμμα των Οντοτήτων Εξωτερικής Πληροφορίας 56
5.1.2 Ορισμός των Οντοτήτων Εξωτερικής Πληροφορίας 60

5.2 Παρουσίαση Εσωτερικής Πληροφορίας 72
5.2.1 OMT Διάγραμμα του Μοντέλου 72
5.2.2 Ορισμός Οντοτήτων Εσωτερικής Πληροφορίας 75

6 Θέματα Υλοποίησης 86

6.1 Διαφανής Εντοπισμός Οντοτήτων 86

6.2 Οντότητες Εκτός Συστήματος 88
6.2.1 Οντότητες Παρακολούθησης Πόρων 89
6.2.2 Οντότητες Δοκιμής Πόρων 89
6.2.3 Οντότητες Επιδιόρθωσης Πόρων 89
6.2.4 Οντότητα Παροχής Πληροφορίας Τοπολογίας Δικτύου 89
6.2.5 Οντότητα Διαχείρισης Συνδέσεων 90

6.3 Υλοποίηση Οντοτήτων 90
6.3.1 Ο Συντονιστής Διαχείρισης Λαθών (FM Coordinator) 90
6.3.2 Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Λαθών (FM Alarm Manager) 91
6.3.3 Ο Εξυπηρετητής Δοκιμών Διαχείρισης Λαθών (FM Testing Server) 93
6.3.4 Η Γραφική Διεπαφή Χρήστη (FM Graphical User Interface) 93

7 Δοκιμές Συστήματος 95

7.1 Δοκιμή σε Δίκτυο 5 Κόμβων και 8 Φυσικών γραμμών (σφάλμα γραμμής και εικονικού μονοπατιού σύνδεσης) 95

7.2 Δοκιμή σε Δίκτυο 5 Κόμβων και 8 Φυσικών γραμμών (σφάλμα γραμμής και δύο εικονικών μονοπατιών σύνδεσης) 98

7.3 Δοκιμή σε Δίκτυο 4 Κόμβων και 5 Φυσικών γραμμών (σφάλμα κόμβου) 99

7.4 Δοκιμή Στατιστικών σε Δίκτυο 4 Κόμβων και 5 Φυσικών γραμμών 101

8 Συμπεράσματα και Μελλοντικές Επεκτάσεις 105

8.1 Συμπεράσματα 105

8.2 Μελλοντικές επεκτάσεις 106

Παράρτημα Α _____ *108*
Παράρτημα Β _____ *114*
Συντομογραφίες _____ *118*
*Αναφορές*_____ *120*

Πίνακας Σχημάτων

Σχήμα 2-1 Το μοντέλο διαχειριστή/αντιπροσώπου _____ *16*
Σχήμα 2-2 Η TINA διαχωρίζει τις εφαρμογές από τους υπολογιστικούς και δικτυακούς πόρους, χρησιμοποιώντας το DPE _____ *17*
Σχήμα 2-3 TINA επίπεδα _____ *19*
Σχήμα 2-4 Αλληλεπιδράσεις υπολογιστικών οντοτήτων στην Διαχείριση Σφαλμάτων _____ *21*
Σχήμα 2-5 Βασική Υπολογιστική Αρχιτεκτονική Διαχείρισης Σφαλμάτων _____ *21*
Σχήμα 2-6 Αρχιτεκτονική Λειτουργιών Διαχειριστή Ειδοποιήσεων επιπέδου Στοιχείων Δικτύου21
Σχήμα 2-7 Αρχιτεκτονική Λειτουργιών Συντονιστή επιπέδου Στοιχείων Δικτύου _____ *22*
Σχήμα 2-8 Αρχιτεκτονική λειτουργιών Συντονιστή επιπέδου Δικτύου _____ *23*
Σχήμα 2-9 Αρχιτεκτονική λειτουργιών Εξυπηρετητή Δοκιμών/Διάγνωσης επιπέδου Στοιχείων Δικτύου 23
Σχήμα 2-10 Αρχιτεκτονική λειτουργιών Εξυπηρετητή Δοκιμών/Διάγνωσης επιπέδου Δικτύου 24
Σχήμα 2-11 Δομή της CORBA _____ *26*
Σχήμα 2-12 Επικοινωνία CORBA Οντοτήτων _____ *27*
Σχήμα 2-13 Συνολική εικόνα του συστήματος REFORM _____ *29*
Σχήμα 2-14 Αρχιτεκτονική Συστήματος _____ *30*
Σχήμα 2-15 Οι Τεχνολογίες στο σύστημα REFORM _____ *31*
Σχήμα 2-16 Ο Κόμβος REFORM (REFORM Node) _____ *32*
Σχήμα 3-1 Θέση του συστήματος και αλληλεπιδράσεις του με άλλα συστήματα στα δύο επίπεδα (NML και EML) _____ *35*
Σχήμα 3-2 Ακολουθία σφαλμάτων/επιδιορθώσεων πόρων _____ *39*
Σχήμα 4-1 Γραφικοί σχηματισμοί παρουσίασης _____ *42*
Σχήμα 4-2 Ο Συντονιστής Διαχείρισης Σφαλμάτων (FM Coordinator) _____ *45*
Σχήμα 4-3 Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Σφαλμάτων (FM Alarm Manager) __ *47*
Σχήμα 4-4 Ο Εξυπηρετητής Δοκιμών Διαχείρισης Σφαλμάτων (FM Testing Server) _____ *49*
Σχήμα 4-5 Η Γραφική Διεπαφή Χρήστη (FM Graphical User Interface) _____ *50*
Σχήμα 4-6 Συνολικό Προτεινόμενο Σύστημα Διαχείρισης Λαθών (Fault Management System)52
Σχήμα 4-7 Παράδειγμα αλληλεπιδράσεων κατά την λειτουργία του συστήματος _____ *55*
Σχήμα 5-1 OMT Διάγραμμα υποσυνόλου των οντοτήτων εξωτερικής πληροφορίας _____ *58*
Σχήμα 5-2 OMT Διάγραμμα υποσυνόλου των οντοτήτων εξωτερικής πληροφορίας (Χάρτης Στατιστικών και Χάρτης Τοπολογίας) _____ *58*
Σχήμα 5-3 Παράδειγμα διαχειριζόμενου δικτύου _____ *59*
Σχήμα 5-4 Χάρτης Τοπολογίας του δικτύου του Σχήματος 5.3 _____ *60*
Σχήμα 5-5 OMT Διάγραμμα των οντοτήτων εσωτερικής πληροφορίας _____ *74*
Σχήμα 6-1 Τοποθέτηση των οντοτήτων που αποτελούν το προτεινόμενο σύστημα στο TINA DPE 86
Σχήμα 6-2 Σχήμα Ονομασίας που χρησιμοποιήθηκε για τον διαφανή εντοπισμό των οντοτήτων88
Σχήμα 7-1 Δίκτυο πρώτης και δεύτερης δοκιμής (5 κόμβοι, 7 φυσικές γραμμές) _____ *96*
Σχήμα 7-2 Γραφική Διεπαφή αμέσως μετά το διάβασμα της τοπολογίας της πρώτης και δεύτερης δοκιμής 97
Σχήμα 7-3 Παρουσίαση αποτελέσματος τόσο της κατάσταση των πόρων του δικτύου όσο και της πρωταρχικής αιτίας των σφαλμάτων για την πρώτη δοκιμή _____ *97*
Σχήμα 7-4 Παρουσίαση αποτελέσματος τόσο της κατάσταση των πόρων του δικτύου όσο και της πρωταρχικής αιτίας των σφαλμάτων για την δέυτερη δοκιμή _____ *98*
Σχήμα 7-5 Δίκτυο τρίτης και τέταρης δοκιμής (4 κόμβοι, 5 γραμμές 4 VPCs) _____ *100*
Σχήμα 7-6 Γραφική Διεπαφή αμέσως μετά το διάβασμα της τοπολογίας της τρίτης και τέταρτης δοκιμής _____ *100*
Σχήμα 7-7 Παρουσίαση αποτελέσματος τόσο της κατάσταση των πόρων του δικτύου όσο και της πρωταρχικής αιτίας των σφαλμάτων, για την τρίτη δοκιμή _____ *101*
Σχήμα 7-8 Παραγόμενα Στατιστικά της τέταρης δοκιμής _____ *102*

1 *Εισαγωγή*

Η εκτενής έρευνα στον τομέα διαχείρισης δικτύων έχει βοηθήσει στη δημιουργία μεγάλου αριθμού τεχνολογιών με σκοπό την αντιμετώπιση των προκλήσεων των σημερινών δικτύων και υπηρεσιών τηλεπικοινωνίας. Οι τεχνολογίες αυτές διαρκώς εξελίσσονται και δημιουργούνται νέες, με σκοπό την κάλυψη των αναγκών που υπάρχουν αλλά και που δημιουργούνται καθημερινά από την ανάγκη διαχείρισης νέων τηλεπικοινωνιακών υπηρεσιών.

Η Διεθνής Ένωση Τηλεπικοινωνιών *(ITU-T [X700])* έχει ορίσει 5 διαφορετικές περιοχές λειτουργιών διαχείρισης δικτύων. Την Διαχείριση Σφαλμάτων (*Fault Management*), η οποία ασχολείται με την ανίχνευση, απομόνωση και διόρθωση των σφαλμάτων που συμβαίνουν στους πόρους ενός δικτύου. Την Διαχείριση Διαμόρφωσης (*Configuration Management*), η οποία περιλαμβάνει τον ορισμό, έλεγχο και συλλογή δεδομένων για την προετοιμασία, αρχικοποίηση, λειτουργία και κλείσιμο των συνδέσεων σε ένα δίκτυο. Την Διαχείριση Λογαριασμών (*Accounting Management*), η οποία ασχολείται με τρόπους χρέωσης της χρήσης των πόρων ενός δικτύου. Την Διαχείριση Απόδοσης (*Performance Management*), η οποία περιλαμβάνει την αξιολόγηση της απόδοσης των πόρων του δικτύου καθώς και την εφαρμογή μεθόδων βελτίωσή της. Τέλος την περιοχή Διαχείρισης Ασφαλείας (*Security Management*), η οποία περιλαμβάνει την εφαρμογή μέτρων και πολιτικών ασφάλειας, τόσο για τους πόρους ενός δικτύου όσο και για την πληροφορία που διακινείται μέσα σε αυτό.

Για τα σημερινά συστήματα διαχείρισης πρόκληση αποτελεί η ενοποιημένη αντιμετώπιση των αναγκών διαχείρισης δικτύων και τηλεπικοινωνιακών υπηρεσιών μέσω της δια-λειτουργικότητας των διαφορετικών περιοχών της, και της συνύπαρξης συστημάτων διαφορετικών τεχνολογιών.

1.1 Η Διαχείριση Σφαλμάτων

Η διαχείριση σφαλμάτων είναι μία σημαντική λειτουργική περιοχή της διαχείρισης δικτύων. Χωρίς κατάλληλη διαχείριση σφαλμάτων σε ένα δίκτυο, δεν μπορούν να εντοπιστούν και να επιδιορθωθούν σφάλματα τα οποία συνέβησαν σε πόρους αλλά και δεν μπορούν να εντοπιστούν άλλοι πόροι οι οποίοι τυχόν επηρεάζονται από αυτά. Επίσης δεν μπορεί να αποκτηθεί η πληροφορία για τα σφάλματα που έχουν συμβεί κατά μία χρονική περίοδο και η οποία με την κατάλληλη εξέταση και ανάλυση θα μπορούσε να χρησιμοποιηθεί από άλλες περιοχές διαχείρισης για την αποτελεσματικότερη και αποδοτικότερη σχεδίαση/διαμόρφωση του δικτύου.

Η διαχείριση σφαλμάτων καθίσταται ιδιαίτερα σημαντική αλλά και δυσκολότερη για δίκτυα Ασύγχρονου Τρόπου Μετάδοσης (*Asynchronous Transfer Mode - ATM*), που ανήκουν στην γενικότερη κατηγορία των Ευρυζώνιων Ψηφιακών Δικτύων Ενοποιημένων Υπηρεσιών (*BISDN*), λόγω της πολυπλεξίας διαφορετικών υπηρεσιών, της υψηλότερης ταχύτητας και των πολλαπλών συνδέσεων που επηρεάζονται σε περιπτώσεις σφάλματος.

Η εκτενής έρευνα που γίνεται τα τελευταία χρόνια στην περιοχή της διαχείρισης σφαλμάτων έχει ως αποτέλεσμα την ύπαρξη ενός μεγάλου αριθμού γνωστών και αποδεκτών, προτύπων, αρχιτεκτονικών και τεχνολογιών από Διεθνείς Ενώσεις και Οργανισμούς (*ITU, TINA* κλπ). Τα πρότυπα αυτά αφορούν την προδιαγραφή των διαδικασιών που περιλαμβάνει η διαχείριση σφαλμάτων, τον καθορισμό της αρχιτεκτονικής τους και τον ορισμό της πληροφορίας που διαχειρίζονται.

Σύμφωνα με τα πρότυπα αυτά, η διαχείριση σφαλμάτων περιλαμβάνει από λειτουργικής άποψης τριών ειδών διαδικασίες. Αυτές που αναλαμβάνουν την ανίχνευση και αναφορά των σφαλμάτων που θα συμβούν στους πόρους του δικτύου, αυτές που αναλαμβάνουν την ανάλυση των σφαλμάτων αυτών που ανιχνεύονται

και τον προσδιορισμό του πόρου που προκάλεσε εξαρχής το σφάλμα και τέλος αυτές που αναλαμβάνουν την επιδιόρθωση του. Λόγω των διαφορετικών απαιτήσεων που κάθε μία από τις διαδικασίες αυτές έχει, τοποθετούνται, από φυσικής και μηχανικής άποψης, σε δύο διαφορετικά επίπεδα του δικτύου που διαχειρίζονται. Το πρώτο επίπεδο είναι το επίπεδο Διαχείρισης Στοιχείων (*Element Management Layer - EML*) και περιλαμβάνει συνήθως τις διαδικασίες ανίχνευσης και επιδιόρθωσης των σφαλμάτων γιατί αυτές απαιτούν μεγάλη ταχύτητα και δεν απαιτούν πληροφορία από όλο το δίκτυο αλλά μόνο από το συγκεκριμένο στοιχείο του δικτύου του οποίου τους πόρους διαχειρίζονται. Το δεύτερο επίπεδο είναι το επίπεδο Διαχείρισης Δικτύου (*Network Management Layer - NML*) και περιλαμβάνει εκείνες τις διαδικασίες που καταγράφουν και αναλύουν την πληροφορία σφάλματος γιατί αυτές απαιτούν πληροφορία τόσο από τους πόρους όλου του δικτύου όσο και από άλλες περιοχές διαχείρισης.

Στα πλαίσια του ερευνητικού προγράμματος του ACTS REFORM μελετήθηκε, σχεδιάστηκε και υλοποιήθηκε ένα παγκόσμια πρωτότυπο σύστημα, που ονομάστηκε REFORM σύστημα, το οποίο, ακολουθώντας κυρίως την αρχιτεκτονική TINA, παρέχει διαχείριση σφαλμάτων, διαμόρφωσης, σύνδεσης και απόδοσης τόσο σε επίπεδο Στοιχείων Δικτύου όσο και σε επίπεδο Διαχείρισης Δικτύου συνδυάζοντας στο ίδιο σύστημα διαφορετικές αρχιτεκτονικές αλλά και τεχνολογίες. Το σημαντικό πλεονέκτημα που προκύπτει από αυτήν την ενοποιημένη προσέγγιση στη διαχείριση σφαλμάτων είναι ότι παρέχεται εφυής εξισορρόπηση φορτίου, δυναμική δρομολόγηση και διαχείριση των επιπλέων πόρων, ταυτόχρονα και με βέλτιστο τρόπο.

1.2 Αντικείμενο Εργασίας

Η παρούσα εργασία ασχολείται με την μελέτη κατανεμημένων συστημάτων εντοπισμού και επιδιόρθωσης σφαλμάτων και την σχεδίαση και υλοποίηση ενός πρωτότυπου συστήματος διαχείρισης σφαλμάτων για τα εικονικά μονοπάτια συνδέσεων (*virtual path connections*), τα στοιχεία (*nodes*) και τις φυσικές γραμμές (*physical links*) ενός δικτύου *ATM*. Το σύστημα αυτό βρίσκεται, από φυσική-μηχανική άποψη, στο επίπεδο Διαχείρισης Δικτύου και η λειτουργικότητά του στηρίζεται τόσο στο Τμήμα Διαχείρισης Σφαλμάτων της Αρχιτεκτονικής Πόρων Δικτύου της TINA (*TINA NRA*) [NRAv3] και στο ερευνητικό πρόγραμμα του *ACTS REFORM* [D4], όσο και σε επιμέρους απαιτήσεις που έχουν τα συστήματα Διαχείρισης Σφαλμάτων.

Σκοπός της εργασίας είναι τόσο η μελέτη της ευρύτερης περιοχής διαχείρισης σφαλμάτων όσο και η ειδικότερη μελέτη και επέκταση του συστήματος REFORM στον τομέα διαχείρισης σφαλμάτων στο NML επίπεδο, παρέχοντας:

1. Την δυνατότητα εφαρμογής αλγορίθμων εντοπισμού της πρωταρχικής αιτίας σφαλμάτων. Η πρωταρχική αιτία του σφάλματος είναι πολλές φορές διαφορετική από το αναφερόμενο σφάλμα και ο εντοπισμός της αποτελεί χρήσιμη πληροφορία τόσο για την επιδιόρθωση του όσο και για εξαγωγή συμπερασμάτων που αφορούν την σχεδίαση/διαμόρφωση του υπό διαχείριση δικτύου.

2. Την δυνατότητα τόσο μέσα από την διατήρηση της πληροφορίας σφαλμάτων του δικτύου όσο και μέσα από την δυνατότητα συσχέτισής τους για την εξαγωγή πολύτιμων στατιστικών για τους πόρους του υπό διαχείριση δικτύου.

3. Ενσωμάτωση των λειτουργιών της διαχείρισης σφαλμάτων με τις λειτουργίες των άλλων περιοχών διαχείρισης, μέσω καλά ορισμένων διεπαφών και αλληλεπιδράσεων.

Σκοπός της εργασίας, επίσης, είναι να εξετάσει τις μεθόδους (σύγχρονες/ασύγχρονες) συλλογής της πληροφορίας σφαλμάτων καθώς και ποια πληροφορία απαιτείται για την παροχή των αναγκαίων λειτουργιών διαχείρισης σφαλμάτων στο επίπεδο NML.

Ιδιαίτερη προσπάθεια έγινε για την αξιοποίηση και χρήση, τόσο κατά την σχεδίαση της αρχιτεκτονικής του συστήματος όσο και κατά την υλοποίησή του, αρχών, ιδεών και καθορισμένων μοντελοποιήσεων που

προέρχονται από έναν μεγάλο αριθμό ευρέως γνωστών και αποδεκτών προτύπων, αρχιτεκτονικών και τεχνολογιών. Τέτοιες είναι η Αρχιτεκτονική Τηλεπικοινωνιών και Πληροφορίας Δικτύων (*TINA*) [TINA], η Αρχιτεκτονική Ανεύρεσης Κοινών Οντοτήτων της Ομάδας Διαχείρισης Οντοτήτων (*OMG CORBA*) [CORBA], το Δίκτυο Διαχείρισης Τηλεπικοινωνιών της Διεθνούς Ένωσης Τηλεπικοινωνιών (*ITU-T TMN*) [M3400] καθώς και μια σειρά άλλων προτύπων της (*ITU-T Xseries*) [X.721], [X.722], [X.733], [X.735], [X.739].

Το προτεινόμενο από την εργασία σύστημα χρησιμοποιεί για την παροχή των λειτουργιών της διαχείρισης σφαλμάτων στο επίπεδο EML τις διαδικασίες για ανίχνευση και αυτόματη επιδιόρθωση σφαλμάτων, οι οποίες σχεδιάστηκαν και υλοποιήθηκαν στο σύστημα REFORM.

1.3 Οργάνωση Κειμένου

Όπως αναφέρθηκε, η αρχιτεκτονική του προτεινόμενου συστήματος βασίζεται σε ένα μεγάλο αριθμό γνωστών, αποδεκτών και προτύπων αρχιτεκτονικών και τεχνολογιών. Για τον λόγο αυτό, το κεφάλαιο 2 παρουσιάζει σύντομα τις αρχιτεκτονικές και τεχνολογίες στις οποίες βασίστηκαν κατά το μεγαλύτερο μέρος οι επιλογές της σχεδίασης και υλοποίησης.

Το κεφάλαιο 3 παρουσιάζει την προδιαγραφή των λειτουργιών του συστήματος όπου καθορίζονται και περιγράφονται λεπτομερώς οι λειτουργίες του.

Στο κεφάλαιο 4 περιγράφεται η υπολογιστική προδιαγραφή του συστήματος, δηλαδή η τοποθέτηση των λειτουργιών, που αναφέρθηκαν στο κεφάλαιο 3, σε υπολογιστικές οντότητες και ο τρόπος που αυτές οι υπολογιστικές οντότητες επικοινωνούν μεταξύ τους καθώς και με άλλες οντότητες που συνεργάζονται και που βρίσκονται εκτός του συστήματος. Επίσης αναφέρεται λεπτομερώς ο τρόπος διεξαγωγής των λειτουργιών της κάθε υπολογιστικής οντότητας.

Στο κεφάλαιο 5 περιγράφεται το μοντέλο της πληροφορίας, δηλαδή ο ακριβής καθορισμός της πληροφορίας που ανταλλάσσεται και διατηρείται κατά την λειτουργία του προτεινόμενου συστήματος. Το κεφάλαιο αυτό χωρίζεται ουσιαστικά σε δύο μέρη. Στο πρώτο περιγράφεται το μοντέλο της πληροφορίας που ανταλλάσσεται μεταξύ των υπολογιστικών οντοτήτων που περιγράφτηκαν στο κεφάλαιο 4 καθώς και εξωτερικών υπολογιστικών οντοτήτων με τις οποίες το προτεινόμενο σύστημα συνεργάζεται, ενώ στο δεύτερο μέρος περιγράφεται το μοντέλο της πληροφορίας που χρησιμοποιείται για την φύλαξή της πληροφορίας που διατηρείται εσωτερικά σε κάθε υπολογιστική οντότητα.

Στο κεφάλαιο 6 περιγράφεται η φυσική και μηχανική πλευρά του συστήματος, δηλαδή το πως το σύστημα υλοποιήθηκε και οι επιλογές που έγιναν κατά την υλοποίηση και οι οποίες περιλαμβάνουν ουσιαστικά την τοποθέτηση του συστήματος στο περιβάλλον κατανεμημένης επεξεργασίας της TINA (*TINA-DPE*).

Στο κεφάλαιο 7 παρουσιάζονται δοκιμές του συστήματος σε διαφορετικά (όσον αφορά την διαμόρφωση) υπό διαχείριση δίκτυα καθώς και σε διαφορατικά σενάρια σφάλματος και παραθέτονται τα αποτελέσματα του αλγορίθμου εντοπισμού της πρωταρχικής αιτίας σφάλματος καθώς και τα δημιουργούμενα στατιστικά.

Τέλος στο κεφάλαιο 8 παραθέτονται τα συμπεράσματα από την σχεδίαση και υλοποίηση του προτεινόμενου συστήματος καθώς και κάποιες ιδέες και κατευθύνσεις όσον αφορά την επέκταση της παρούσας εργασίας.

Επίσης στο τέλος της εργασίας υπάρχουν δύο παραρτήματα με σκοπό την πληρέστερη κατανόηση της. Το πρώτο αναφέρεται στην υλοποίηση του εσωτερικού μοντέλου πληροφορίας το οποίο χρησιμοποιεί το σύστημα, ενώ το δεύτερο παράρτημα αναφέρεται στις παρεμβάσεις που έγιναν για την χρησιμοποίηση του συστήματος Διαχείρισης Σφαλμάτων του επιπέδου Στοιχείων του Δικτύου της πρώτης φάσης του συστήματος REFORM.

2 Υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζονται σύντομα οι αρχιτεκτονικές και τεχνολογίες στηρίζονται όλα τα μοντέρνα συστήματα διαχείρισης και στις οποίες βασίστηκαν κατά το μεγαλύτερο μέρος οι επιλογές που έγιναν τόσο κατά την σχεδίαση όσο και κατά την υλοποίηση του προτεινόμενου συστήματος διαχείρισης.

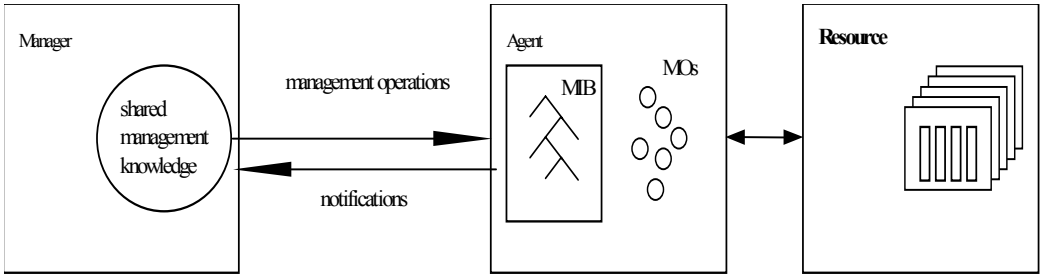
Η οργάνωση του κεφαλαίου έχει ως εξής: πρώτα παρουσιάζεται το Δίκτυο Διαχείρισης Τηλεπικοινωνιών της Διεθνούς Ένωσης Τηλεπικοινωνιών (*ITU-T TMN*). Ακολουθεί η παρουσίαση της Αρχιτεκτονικής Τηλεπικοινωνιών και Πληροφορίας Δικτύωσης (*TINA*) καθώς και της αρχιτεκτονικής *CORBA* της Ομάδας Διαχείρισης Οντοτήτων (*OMG CORBA*). Έπειτα παρουσιάζεται η υπηρεσία καταλόγου X.500 (*Directory Service X.500*) [X.500] και τέλος γίνεται μια σύντομη παρουσίαση της αρχιτεκτονικής και των επιλογών που έγιναν στο σύστημα REFORM.

2.1 Δίκτυο Διαχείρισης Τηλεπικοινωνιών (ITU-T TMN)

Το Δίκτυο Διαχείρισης Τηλεπικοινωνιών έχει προταθεί από την ITU στην σειρά συστάσεων M.30 (*ITU M.30 Recommendations*). Σκοπός του είναι η διαχείριση των δικτύων τηλεπικοινωνιών καθώς και των υπηρεσιών που αυτά προσφέρουν. Η βασική ιδέα πίσω από το δίκτυο τηλεπικοινωνιών είναι η εισαγωγή μιας οργανωμένης αρχιτεκτονικής που θα παρέχει τη δυνατότητα διασύνδεσης διάφορων λειτουργικών συστημάτων και τηλεπικοινωνιακού εξοπλισμού με σκοπό την ανταλλαγή πληροφορίας διαχείρισης μέσω τυποποιημένων πρωτοκόλλων και μηνυμάτων. Γενικά ένα δίκτυο τηλεπικοινωνιών αποτελείται από έναν μεγάλο αριθμό συσκευών όπως μεταγωγείς, πολυπλέκτες, σταθμούς εργασίας και μεγάλα υπολογιστικά συστήματα τα οποία αναφέρονται ως στοιχεία δικτύου (*network elements*) που υπόκεινται σε διαχείριση.

Το δίκτυο τηλεπικοινωνιών είναι εννοιολογικά ένα ξεχωριστό δίκτυο το οποίο διασυνδέεται με το διαχειριζόμενο δίκτυο σε πολλά σημεία του προκειμένου να ανταλλάσσεται μεταξύ τους πληροφορία για τους σκοπούς της διαχείρισης. Η ικανότητα που έχει το δίκτυο τηλεπικοινωνιών να είναι λογικά ξεχωριστό από το φυσικό δίκτυο και τις υπηρεσίες που διαχειρίζεται, του δίνει την δυνατότητα να υλοποιείται σαν ένα κατανεμημένο σύστημα το οποίο εκτελεί πράξεις διαχείρισης σε μεγάλο αριθμό από διάσπαρτα στοιχεία του διαχειριζόμενου δικτύου και να παρέχει υπηρεσίες μεταξύ πολύ απομακρυσμένων συστημάτων. Οι διεργασίες αυτές λειτουργούν σε ρόλο αντιπροσώπων διαχείρισης (*management agents*) και επικοινωνούν με διεργασίες που έχουν το ρόλο του διαχειριστή (*manager*) σύμφωνα με το μοντέλο διαχειριστή/αντιπροσώπου που φαίνεται στο Σχήμα 2-1. Σύμφωνα με αυτό το μοντέλο οι αλληλεπιδράσεις μέσα στο δίκτυο τηλεπικοινωνιών είναι δομημένες σαν 3 οντότητες:

- Τον διαχειριζόμενο πόρο (*managed resource*), που είναι το φυσικό ή υπολογιστικό στοιχείο που υπόκεινται στην διαχείριση.
- Τον αντιπρόσωπο (*agent*) ο οποίος δημιουργεί μια καθορισμένη όψη του διαχειριζόμενου πόρου.
- Τον διαχειριστή (*manager*) ο οποίος αντιπροσωπεύει τον χρήστη.



Σχήμα 2-1 Το μοντέλο διαχειριστή/αντιπροσώπου

Η αρχιτεκτονική του δικτύου διαχείρισης αποτελείται από το δίκτυο διασύνδεσης (*Data Communication Network*), τα συστήματα λειτουργιών (*Operation Systems - OS*), τα στοιχεία δικτύου (*Network Elements - NE*), τα Q adapters (*QA*), τις ενδιάμεσες συσκευές (*Mediation Devices - MD*) και τους σταθμούς εργασίας (*Work Stations - WS*).

Κάθε δομικό στοιχείο της αρχιτεκτονικής μπορεί να είναι είτε αντιπρόσωπος είτε διαχειριστής ανάλογα με την αλληλεπίδραση που έχει. Το σύνολο της πληροφορίας που διαχειρίζεται ένα αντιπρόσωπος λέγεται Διαχειριζόμενη Βάση Πληροφορίας (*Managed Information Base - MIB*) και ορίζεται από ένα σύνολο GDMO και ASN.1 τύπους ορισμού. Αυτοί οι ορισμοί αποτελούν καθορισμένες διεπαφές (*interfaces*) από διαχειριζόμενες οντότητες (*Managed Objects*). Για να έχεις πρόσβαση σε αυτές τις διαχειριζόμενες οντότητες πρέπει να ξέρεις πως θα τις εντοπίσεις και πως θα τις ονομάσεις. Αυτή η γνώση ονομάζεται Κοινή διαχειριζόμενη Γνώση (*Shared Managed Knowledge - SMK*). Για να ενεργήσει σε μια διαχειριζόμενη οντότητα ο διαχειριστής εκτελεί CMISE αιτήσεις.

Οι βασικές λειτουργίες που γίνονται σε ένα OSI σύστημα διαχείρισης είναι:

- Η ανταλλαγή πληροφορίας μεταξύ δύο οντοτήτων και
- Η εκκίνηση λειτουργιών μέσα σε αντιπροσώπους

Η λειτουργικότητα αυτή αναφέρεται σαν CMISE [X.710] και καθορίζεται σε δύο μέρη:

- Την διεπαφή με τον χρήστη που καθορίζει την υπηρεσία που παρέχεται. Αυτό είναι η CMIS (*Common Management Information Service*)
- Το πρωτόκολλο που καθορίζει την φόρμα της μονάδας δεδομένων (*PDU*) και τις σχετικές διαδικασίες.

Οι υπηρεσίες CMIS είναι τριών κατηγοριών:

- Υπηρεσίες συνεργασίας (*Association Services*)
- Υπηρεσίες λειτουργιών διαχείρισης (*Management Operation Services*)
- Υπηρεσίες διαχείρισης ειδοποιήσεων (*Management Notification Services*)

2.2 Αρχιτεκτονική Τηλεπικοινωνιών και Πληροφορίας Δικτύωσης (Telecommunication Information Networking Architecture - TINA)

TINA-C είναι μια ομοσπονδία με σκοπό να ορίσει και να επαληθεύσει μια ανοικτή αρχιτεκτονική για κατανεμημένες εφαρμογές τηλεπικοινωνιών. Η λίστα των μελών της περιέχει έναν μεγάλο αριθμό λειτουργών δικτύων, προμηθευτών Η/Υ και κατασκευαστών προϊόντων τηλεπικοινωνιών. Η TINA εφαρμόζει τις αρχές της Ανοικτής Κατανεμημένης Επεξεργασίας (*Open Distributed Processing - ODP*) καθώς και τους καθορισμούς της Ομάδας Διαχείρισης Οντοτήτων (*Object Management Group - OMG*) για τις ανάγκες της βιομηχανίας τηλεπικοινωνιών [TINA]. Αυτό είχε σαν αποτέλεσμα τον ορισμό του Κατανεμημένου Περιβάλλοντος Επεξεργασίας (*Distributed Processing Environment - DPE*) διατηρώντας μέρος των κλασικών αρχών του TMN.

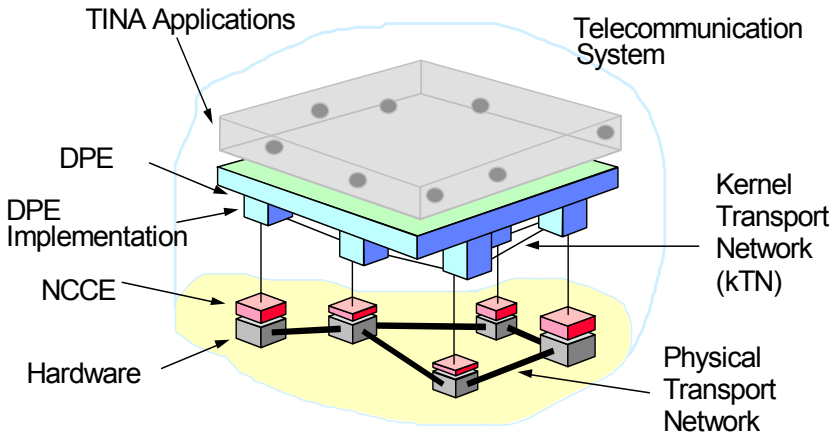
Η βασική ιδέα είναι ότι αντί να περιοριζόμαστε στις ήδη υπάρχουσες υπηρεσίες μπορούμε να δημιουργήσουμε νέες προχωρημένες απευθείας στο DPE, και οι οποίες μπορούν να σχεδιαστούν και να υλοποιηθούν σύμφωνα με οντο-κεντρικές αρχές και τεχνικές κατανεμημένης επεξεργασίας.

Μια από τις βασικές αλλαγές από το TMN είναι ότι η TINA βάζει το λογισμικό να είναι ο πυρήνας της φιλοσοφίας σχεδίασης. Οι βασικές αρχές της αρχιτεκτονικής της TINA φαίνονται στο Σχήμα 2-2 και είναι:

- Το λογισμικό τηλεπικοινωνιών είναι βασικά κατανεμημένο λογισμικό
- Οντο-κεντρικές τεχνικές είναι χρήσιμες
- Διαχωρισμός των εφαρμογών τηλεπικοινωνίας από το λογισμικό υποστήριξης το οποίο είναι το DPE.

Η αρχιτεκτονική της TINA είναι χωρισμένη σε πέντε περιοχές:

- Την Υπολογιστική Αρχιτεκτονική (*Computing Architecture*)
- Την Αρχιτεκτονική Πόρων Δικτύου (*Network Resource Architecture*)
- Την Αρχιτεκτονική Υπηρεσιών (*Service Architecture*)
- Την Αρχιτεκτονική Περιβάλλοντος Κατανεμημένης Επεξεργασίας (*DPE Architecture*)
- Την Επιχειρησιακή Αρχιτεκτονική (Business Architecture)



Σχήμα 2-2 Η TINA διαχωρίζει τις εφαρμογές από τους υπολογιστικούς και δικτυακούς πόρους, χρησιμοποιώντας το DPE

Η TINA αυτήν την στιγμή παρέχει δύο γλώσσες για τον καθορισμό διαχειριζόμενων οντοτήτων: την Γλώσσα Περιγραφής Διεπαφών της OMG (*IDL*) και την Γλώσσα Περιγραφής Οντοτήτων (*ODL*) η οποία επεκτείνει την IDL με επιπλέον δυνατότητες.

Η αρχιτεκτονική TINA βρίσκεται αυτήν την στιγμή στην φάση της επαλήθευσης μέσω χρήσης της σε διάφορες ερευνητικές και βιομηχανικές δραστηριότητες.

2.2.1 Συμβατότητα με την Αρχιτεκτονική TINA

Η διαπίστωση συμβατότητας ενός συστήματος με την TINA είναι μια πολύπλοκη διαδικασία. Δεν υπάρχει αυτήν την στιγμή ακριβής ορισμός της συμβατότητας. Ορισμένες αρχές συμβατότητας από την πιο αδύναμη προς την πιο ισχυρή είναι:

- Να ακολουθεί αρχές: Οντο-κεντρικό, διαφάνεια κατανομής, διαχωρισμός των υπηρεσιών και των πόρων, ορισμός ανοικτών διεπαφών (*open interfaces*)
- Χρήση OMG CORBA
- Μερικές ανοικτές διεπαφές της TINA (*Reference Points*) προς τα έξω

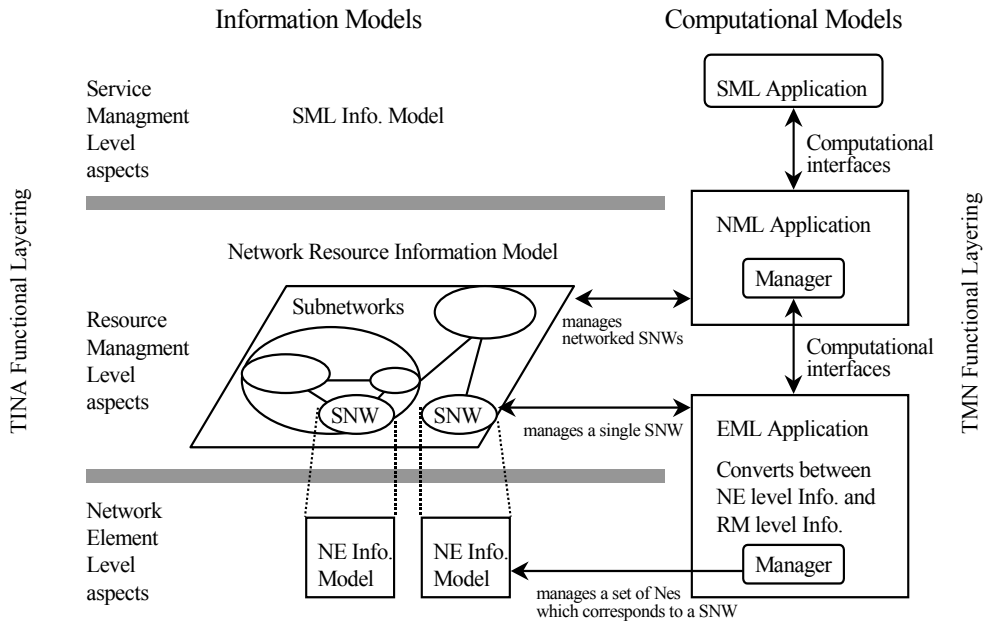
Το πιο σημαντικό πάντως είναι, η αρχιτεκτονική του συστήματος και οι διεπαφές (σε IDL) να συμβαδίζουν όπου είναι δυνατόν με την TINA.

2.2.2 Η Αρχιτεκτονική Πόρων Δικτύου (Network Resource Architecture - NRA)

Η NRA [NRA v3] περιέχει τις αρχές και ιδέες για διαχείριση TINA συστημάτων και δικτύων και στηρίζεται κατά πολύ στην TMN αρχιτεκτονική που είδαμε πιο πάνω. Οι καθορισμοί στην περιοχή της διαχείρισης Διαμόρφωσης (*Configuration Management*) είναι οι πιο αναπτυγμένοι και ιδιαίτερα αυτοί της διαχείρισης των Συνδέσεων (*Connection Management*). Η NRA ασχολείται κυρίως με:

- Το Μοντέλο Πληροφορίας Πόρων Δικτύου (*Network Resource Information Model - NRIM*).
- Την Αρχιτεκτονική Διαχείρισης Συνδέσεων (*Connection Management - CM Architecture*).
- Την Αρχιτεκτονική Διαχείρισης Διαμόρφωσης Τοπολογίας Δικτύου (*Network Topology Configuration Management - NTCM Architecture*).
- Την Αρχιτεκτονική Διαχείρισης Σφαλμάτων και Λογαριασμών (*Fault and Accounting Management Architecture*).

Το Σχήμα 2-3 παρουσιάζει τις ιδέες των λειτουργικών επιπέδων της TINA (TINA functional layering concepts). Υπάρχουν τρία επίπεδα: Το επίπεδο Στοιχείων Δικτύου (Network Element Layer - NEL), το επίπεδο Διαχείρισης Πόρων (Resource Management Layer - RML) και το επίπεδο Διαχείρισης Υπηρεσιών (Service Management Layer - SML). Για τα επίπεδα του υπολογιστικού μοντέλου (Computational Model), χρησιμοποιείται το TMN.



Σχήμα 2-3 TINA επίπεδα

Η Αρχιτεκτονική Πόρων Δικτύου προτείνει ένα σύνολο ανεξάρτητων από την τεχνολογία ιδεών και εργαλείων για την μοντελοποίηση ενός δικτύου το οποίο αναφέρεται ως Μοντέλο Πληροφορίας Πόρων Δικτύου (*NRIM*). Το NRIM τοποθετείται στο NML επίπεδο και μοντελοποιεί το δίκτυο με εντελώς τεχνολογικά ανεξάρτητο τρόπο. Είναι βασισμένο στις ιδέες διαχωρισμού και επιπεδοποίησης του G.803 καθώς και στο γενικό μοντέλο πληροφορίας στοιχείων δικτύου του TMN [M3100] το οποίο αναδρομικά χωρίζει το δίκτυο σε υποδίκτυα και γραμμές όπου το τελευταίο επίπεδο αντιστοιχεί σε ένα στοιχείο του δικτύου. Αντί να παρουσιάζεται ολόκληρο το NRIM, έχουν οριστεί διάφορα τμήματα τα οποία κάνουν το μοντέλο καλύτερα διαχειρίσιμο δίνοντας την δυνατότητα οντότητες οι οποίες ανήκουν σε μια συγκεκριμένη περιοχή να παρουσιάζονται σε μία μόνο σελίδα μαζί με τις σχέσεις που τις ενώνουν.

Το τμήμα του NRIM το οποίο είναι σχετικό με την παρούσα εργασία είναι το τμήμα Διαχείρισης Σφαλμάτων (*NRIM Fault Management Fragment*) [NRIM].

2.2.3 Η Αρχιτεκτονική Διαχείρισης Σφαλμάτων (Fault Management - FM Architecture)

2.2.3.1 Περιγραφή Λειτουργιών

Οι ενέργειες Διαχείρισης Σφαλμάτων γίνονται μέσω αλληλεπιδράσεων μεταξύ του χρήστη υπηρεσιών διαχείρισης σφαλμάτων και των διαδικασιών διαχείρισης. Η παράγραφος αυτή περιγράφει τις ενέργειες Διαχείρισης Σφαλμάτων με όρους λειτουργιών που γίνονται μέσα σε διαδικασίες διαχείρισης. Ανάλογα με την περίπτωση όπου η διαχείριση σφαλμάτων είναι ενεργή, ορίζονται οι ακόλουθες πέντε διαφορετικές ενέργειες:

- Επιτήρηση/Εποπτεία Ειδοποιήσεων (*Alarm Surveillance*),
- Δοκιμή (Testing),
- Εντοπισμός Σφάλματος (*Fault Localisation*),
- Επιδιόρθωση Σφάλματος (*Fault Correction*), και
- Διοίκηση Προβλημάτων (*Trouble Administration*).

Η Επιτήρηση/Εποπτεία Ειδοποιήσεων επιτρέπει την παρακολούθηση πόρων και κάνει την σχετική με κατάσταση σφάλματος πληροφορία του πόρου γνωστή και έξω από τον πόρο.

Ο Εντοπισμός Σφάλματος καθορίζει του ακριβείς πόρους οι οποίοι είναι υπεύθυνοι για ανάρμοστη συμπεριφορά στο διαχειριζόμενο δίκτυο. Η διαδικασίες διαχείρισης σφαλμάτων κάνουν εντοπισμό μέσω εξέτασης εγγραφών προηγούμενων ειδοποιήσεων και μέσω δοκιμών ενός συνόλου πόρων οι οποίοι έχουν πιθανόν σχέση. Επίσης ο χρήστης διαχείρισης σφαλμάτων μπορεί να κάνει εντοπισμό μέσω δοκιμών σε κάποιο σύνολο πόρων.

Η Επιδιόρθωση Σφάλματος αναφέρεται σε εκείνες τις ενέργειες που επιτρέπουν την επιδιόρθωση των λειτουργιών των πόρων που την στιγμή εκείνη βρίσκονται σε κατάσταση σφάλματος. Γενικά ένα σφάλμα σε ένα δίκτυο μπορεί να αντιμετωπιστεί σε τρία διαφορετικά επίπεδα: στο επίπεδο των πόρων, στο επίπεδο διαχείρισης και στο επίπεδο του χρήστη υπηρεσιών διαχείρισης. Στο επίπεδο των πόρων η επιδιόρθωση γίνεται μέσω αυτόματων μηχανισμών προστασίας και επιδιόρθωσης για εσφαλμένους πόρους. Στο επίπεδο του χρήστη υπηρεσιών διαχείρισης η επιδιόρθωση αναφέρεται στις ενέργειες που ο χρήστης κάνει όπως για παράδειγμα την αίτηση για δέσμευση νέων πόρων στο δίκτυο. Τέλος στο επίπεδο διαχείρισης η επιδιόρθωση γίνεται με αντικατάσταση των πόρων από άλλους (αναπληρωματικούς) πόρους οι οποίοι υπάρχουν.

Η Δοκιμή/Διάγνωση αναφέρεται στην δοκιμή και ανάλυση των αποτελεσμάτων δοκιμής, κυκλωμάτων, μονοπατιών, εξοπλισμού μετάδοσης και πόρων υποστήριξης των στοιχείων του δικτύου.

Η διοίκηση προβλημάτων αναφέρεται στην ενέργεια μεταξύ εξυπηρετητή και εξυπηρετούμενου η οποία επιτρέπει να αναφερθούν τα προβλήματα και να βρεθεί η κατάστασή τους.

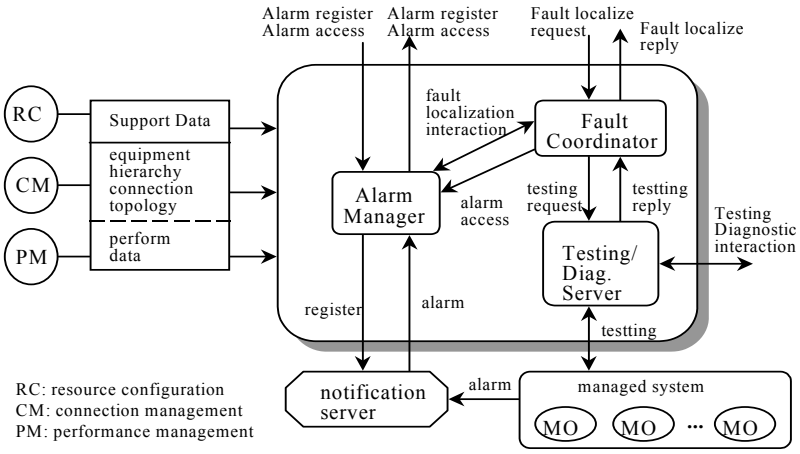
2.2.3.2 Περιγραφή Υπολογιστικών Οντοτήτων

Ο βασικός σκοπός αυτής της παραγράφου είναι να ορίσει τις υπολογιστικές οντότητες καθώς και τις αλληλεπιδράσεις μεταξύ τους (οι διεπαφές δεν περιγράφονται).

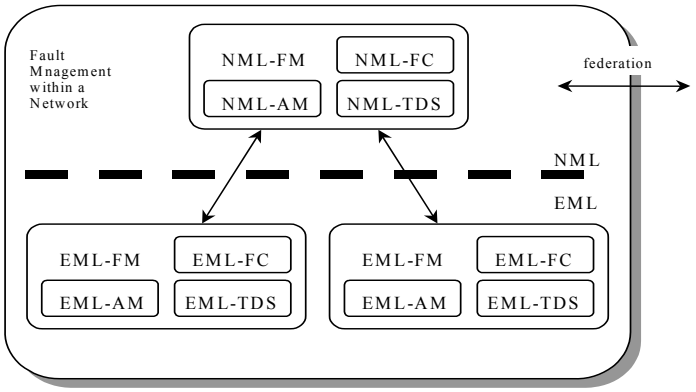
Οι υπηρεσίες διαχείρισης σφαλμάτων των πόρων του δικτύου παρέχονται μέσω αλληλεπιδράσεων των υπολογιστικών οντοτήτων μέσα και έξω από την περιοχή διαχείρισης. Οι υπολογιστικές οντότητες που ορίζονται είναι οι εξής:

- Διαχειριστής Ειδοποιήσεων (*Alarm Manager - AM*)
- Συντονιστής Σφαλμάτων (*Fault Coordinator - FC*)
- Εξυπηρετητής Δοκιμών/Διαγνωστικών *(Testing/Diagnostic Server - TDS)*

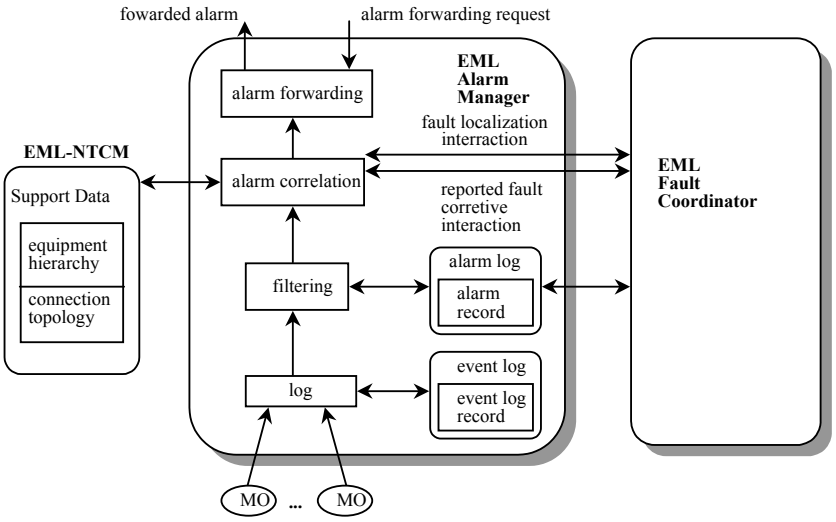
Στο Σχήμα 2-4 φαίνονται οι αλληλεπιδράσεις μεταξύ των υπολογιστικών οντοτήτων για την υλοποίηση των διαδικασιών διαχείρισης λαθών. Οι αλληλεπιδράσεις αυτές γίνονται μεταξύ υπολογιστικών οντοτήτων οι οποίες μπορεί να ανήκουν και στο επίπεδο Δικτύου (*NML*) αλλά και στο επίπεδο Στοιχείων Δικτύου (*EML*), όπως φαίνεται και από το Σχήμα 2-5.



Σχήμα 2-4 Αλληλεπιδράσεις υπολογιστικών οντοτήτων στην Διαχείριση Σφαλμάτων



Σχήμα 2-5 Βασική Υπολογιστική Αρχιτεκτονική Διαχείρισης Σφαλμάτων



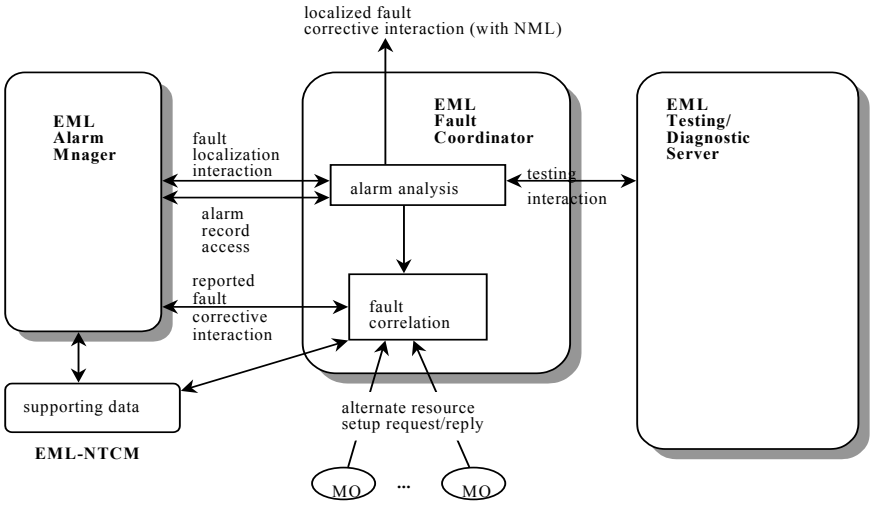
Σχήμα 2-6 Αρχιτεκτονική Λειτουργιών Διαχειριστή Ειδοποιήσεων επιπέδου Στοιχείων Δικτύου

Οι υπολογιστικές οντότητες περιγράφονται ως εξής:

Ο διαχειριστής Ειδοποιήσεων (*AM*) δέχεται όλες τις ειδοποιήσεις που έχουν σχέση με σφάλματα και εκτελεί ενέργειες όπως συσχέτιση ειδοποιήσεων (*alarm correlation*), φιλτράρισμα ειδοποιήσεων (*alarm filtering*), προώθηση ειδοποιήσεων στον Συντονιστή ή στον χρήστη υπηρεσιών διαχείρισης σφαλμάτων (*forwarding the alarm to the FC or fault management service user*), και διαχείριση των εγγραφών ειδοποιήσεων (*alarm record management*). Ο κάθε Διαχειριστής Ειδοποιήσεων έχει τα δικά του κριτήρια με τα οποία αποφασίζει ποιες ειδοποιήσεις θα αποθηκεύσει και θα προωθήσει. Μία ειδοποίηση πρώτα λαμβάνεται από τον Διαχειριστή Ειδοποιήσεων του επιπέδου Στοιχείων Δικτύου (*ELM-AM*) όπου γίνονται οι απαραίτητες ενέργειες (Σχήμα 2-6) και μετά προωθείται στον Διαχειριστή Ειδοποιήσεων του επιπέδου Διαχείρισης Δικτύου (*NML-AM*).

Η δομή και οι λειτουργίες του NML-AM είναι παρόμοιες με αυτές του EML-AM εκτός από το ότι δεν περιλαμβάνει την διαδικασία αποθήκευσης των ειδοποιήσεων των σφαλμάτων.

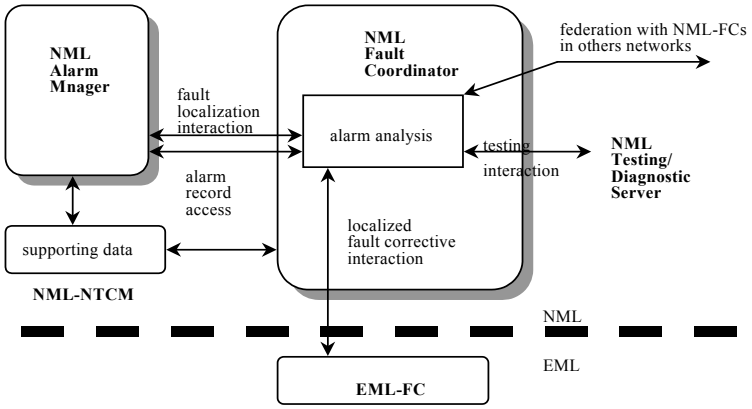
Ο Συντονιστής Σφαλμάτων περιλαμβάνει ικανότητες για ανάλυση των ειδοποιήσεων από πολλαπλούς πόρους για να αποφασίσει το επόμενο βήμα για τον εντοπισμό/επιδιόρθωση. Για αυτόν τον λόγο ο Συντονιστής συσχετίζει και αναλύει όλη την διαθέσιμη πληροφορία για την ανίχνευση της πρωταρχικής αιτίας του σφάλματος. Κατά την διάρκεια της ανάλυσης ο Συντονιστής καλεί τον Εξυπηρετητή Δοκιμών για να δοκιμάσει πόρους που χρειάζεται. Οι διαδικασίες του Συντονιστή επιπέδου Στοιχείων Δικτύου (*EML-FC*) είναι ανάλυση και επιδιόρθωση. Η ανάλυση γίνεται πάνω σε ειδοποιήσεις που έρχονται καθώς και σε παλαιότερες που είναι αποθηκευμένες και κατά την διάρκειά της έχουμε αλληλεπίδραση και με τον Εξυπηρετητή Δοκιμών/Διάγνωσης. Η επιδιόρθωση γίνεται μέσω ενεργοποίησης άλλου πόρου που παίρνει την θέση αυτού που έσφαλε. Το Σχήμα 2-7 δείχνει την αρχιτεκτονική λειτουργιών του EML-FC.



Σχήμα 2-7 Αρχιτεκτονική Λειτουργιών Συντονιστή επιπέδου Στοιχείων Δικτύου

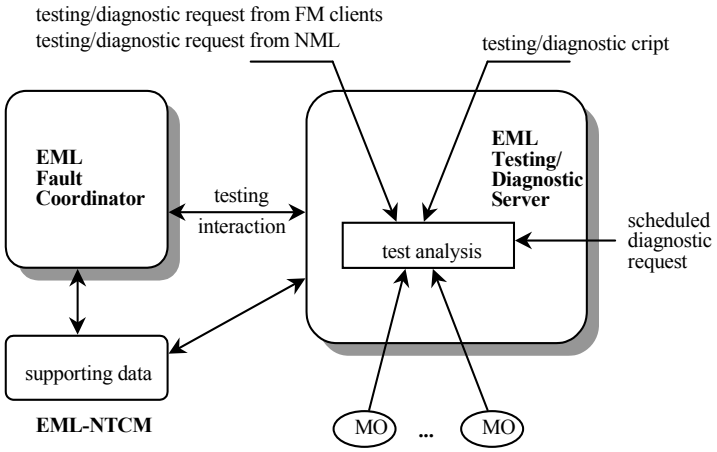
Ο Συντονιστής επιπέδου Δικτύου (*NML-FC*) κάνει ανάλυση για μια καινούργια ειδοποίηση και αλληλεπιδρά με τον Εξυπηρετητή για να μπορέσει να εντοπίσει την κύρια αιτία ενός συνόλου σφαλμάτων. Για τον εντοπισμό σφαλμάτων που αναφέρονται σε διαφορετικά δίκτυα υπάρχει επικοινωνία μεταξύ πολλών NML-FCs από όλα αυτά τα δίκτυα μέσω συνομοσπονδίας (federation). Στο Σχήμα 2-8 φαίνεται η αρχιτεκτονική λειτουργιών του NML-FC.

Ο εξυπηρετητής Δοκιμών/Διάγνωσης ασχολείται με την δοκιμή πόρων με σκοπό την επιδιόρθωση συντήρησή τους. Από την πλευρά της διαχείρισης ο Εξυπηρετητής Δοκιμών/Διάγνωσης καλείται είτε από τον Συντονιστή είτε από τον χρήστη υπηρεσιών διαχείρισης (φυσικά υπάρχει η δυνατότητα για κλήση του και από άλλες εξωτερικές υπολογιστικές οντότητες).

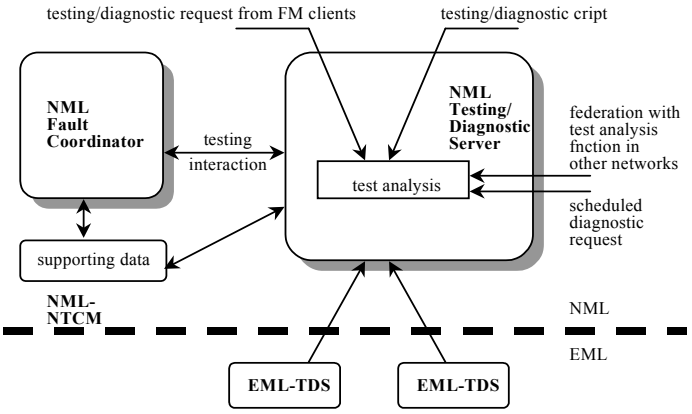


Σχήμα 2-8 Αρχιτεκτονική λειτουργιών Συντονιστή επιπέδου Δικτύου

Ο Εξυπηρετητής Δοκιμών/Διάγνωσης του επιπέδου στοιχείων (*EML-TDS*) παρέχει λειτουργίες δοκιμής και διάγνωσης σε ένα σύνολο πόρων ενώ ο Εξυπηρετητής Δοκιμών/Διάγνωσης του επιπέδου Δικτύου παρέχει λειτουργίες δοκιμής και διάγνωσης πόρων από διαφορετικά υποδίκτυα. Η αρχιτεκτονική λειτουργιών του Εξυπηρετητή για τα δύο επίπεδα φαίνεται στα Σχήμα 2-9 και Σχήμα 2-10. Η διαδικασία διάγνωσης περιλαμβάνει μια περίπλοκη διαδικασία ανάλυσης των αποτελεσμάτων των δοκιμών με σκοπό την ανίχνευση της πρωταρχικής αιτίας του σφάλματος (αν αυτή υπάρχει).



Σχήμα 2-9 Αρχιτεκτονική λειτουργιών Εξυπηρετητή Δοκιμών/Διάγνωσης επιπέδου Στοιχείων Δικτύου



Σχήμα 2-10 Αρχιτεκτονική λειτουργιών Εξυπηρετητή Δοκιμών/Διάγνωσης επιπέδου Δικτύου

2.3 Η Αρχιτεκτονική CORBA

2.3.1 Περίληψη

Η Ομάδα Διαχείρισης Οντοτήτων (*Object Management Group - OMG*) σκοπεύει στην σχεδίαση μίας αρχιτεκτονικής που θα υποστηρίζει λεπτομερή ορισμό διεπαφών και θα οδηγήσει την βιομηχανία σε δια-λειτουργικά, επαναχρησιμοποιούμενα, συμβατά κατανεμημένα κομμάτια λογισμικού βασιζόμενα σε πρότυπες οντοκεντρικές διεπαφές. Αυτό είχε σαν αποτέλεσμα τον ορισμό ενός αφαιρετικού μοντέλου αναφοράς το οποίο ονομάζεται Αρχιτεκτονική Διαχείρισης Οντοτήτων (*Object Management Architecture - OMA)* καθώς και στην τυποποίηση μίας πλατφόρμας η οποία ονομάζεται Αρχιτεκτονική Αίτησης Κοινών Οντοτήτων *(Common Object Request Broker Architecture - CORBA*).Η CORBA σχεδιάστηκε από την OMG με σκοπό να παρέχει μια πραγματικά "ανοιχτή" κατανεμημένη αρχιτεκτονική.

Η αρχιτεκτονική CORBA αποτελείται από ένα δίαυλο λογισμικού, τον Object Request Broker (*ORB*), όπου βρίσκονται οι πελάτες και οι εξυπηρετητές. Και οι δύο μοιράζονται έναν μοναδικό ορισμό των διεπαφών τους σε μία συγκεκριμένη γλώσσα που λέγεται Γλώσσα Ορισμού Διεπαφών (*Interface Definition Language - IDL*). Η γλώσσα αυτή είναι ανεξάρτητη από τις γλώσσες υλοποίησης το οποίο δίνει την δυνατότητα στον κατασκευαστή να επιλέξει από έναν αριθμό γλωσσών για την υλοποίηση της εφαρμογής (*C++, SmallTalk, ADA, Java*). Επιπλέον η CORBA προσφέρει:

1. Βασικές υπηρεσίες όπως: ονομασίας (*naming*), διατήρησης (*persistence*), συμβάντων (*events*) κλπ οι οποίες θεωρείται ότι διευρύνουν την λειτουργικότητα του ORB.
2. Κοινά εργαλεία (*Common Facilities*) όπως Διαχείρισης Πληροφορίας (*Information Management*), Διαχείρισης Εργασιών (*Task Management*) κλπ.
3. Οντότητες Εργασίας (*Business Objects*) οι οποίες βρίσκονται στο επίπεδο εφαρμογής και παρέχουν διαδικασίες συγκεκριμένες για εργασία.
4. Αλλαγή διαχείρισης των οντοτήτων μέσω μετάφρασης των διεπαφών σε κώδικα στην γλώσσα υλοποίησης.
5. Αλλαγή της θέσης του πρωτοκόλλου το οποίο τώρα είναι κρυμμένο από τον χρήστη.

Όλα αυτά τα χαρακτηριστικά δίνουν μία αφαίρεση υψηλού επιπέδου στον κατανεμημένο προγραμματισμό. Παρ' όλα αυτά πρέπει να εξεταστεί κατά πόσο αυτή η αφαίρεση δημιουργεί προβλήματα απόδοσης καθώς και προβλήματα επέκτασης στις εφαρμογές.

2.3.2 Ονομασία (Naming)

Στην CORBA η μονάδα διευθυνσιοδότησης είναι ο εξυπηρετητής ο οποίος μπορεί να χειρίζεται μία ή περισσότερες οντότητες. Η ονομασία αναφέρεται στο όνομα του μηχανήματος και στην πόρτα στην οποία βρίσκεται ο CORBA εξυπηρετητής ενώ η διευθυνσιοδότηση αναφέρεται στην οντότητα αναφοράς και δεν εξαρτάται από την υλοποίηση. Επειδή κάθε οντότητα εξαρτάται από τον εξυπηρετητή στον οποίο βρίσκεται και επειδή όπως είπαμε αυτός εξαρτάται από το όνομα και την πόρτα του μηχανήματος το να μεταφέρεις μια οντότητα κάπου αλλού δεν είναι καθόλου διαφανές.

Ο ρόλος της υπηρεσίας Ονομασίας (*Naming Service*) είναι να κρατάει μία αντιστοίχηση μεταξύ ενός λογικού ονόματος και της οντότητας στην οποία αναφέρεται. Ο ρόλος της λοιπόν είναι να επιτρέπει ένα όνομα να αντιστοιχιστεί με μία οντότητα και να μπορεί κάποιος να βρει αυτήν την οντότητα χρησιμοποιώντας το όνομα χωρίς να χρειάζεται να γνωρίζει λεπτομέρειες για τον εξυπηρετητή, με αποτέλεσμα να μπορούμε να μετακινήσουμε μία οντότητα με διαφανή τρόπο. Περισσότερες λεπτομέρειες για το πως ακριβώς γίνεται η αντιστοίχηση και η επίλυση ενός ονόματος σε μία οντότητα μπορούν να βρεθούν στο [CORBA].

2.3.3 Μοντέλο Οντοτήτων (Object Model)

Το μοντέλο οντοτήτων που ορίζεται από την OMG είναι πρωταρχικής σημασίας. Χρησιμοποιεί τις περισσότερες ιδέες από τα άλλα μοντέλα οντοτήτων και ειδικεύει σε μερικές από αυτές. Οι ποιο κοινές ιδέες που χρησιμοποιεί είναι:

1. Ένα σύστημα οντοτήτων αποτελείται από μονάδες οι οποίες ονομάζονται οντότητες. Μία οντότητα είναι μία μονάδα που έχει ταυτότητα και μπορεί να παρέχει μία ή περισσότερες υπηρεσίες σε έναν πελάτη.
2. Μία αναφορά οντότητας είναι ένα όνομα το οποίο μπορεί μοναδικά να σηματοδοτεί μία οντότητα. Συγκεκριμένα ένα όνομα θα αναφέρεται στην ίδια οντότητα όσες φορές και να χρησιμοποιηθεί.
3. Μία αίτηση είναι ένα συμβάν το οποίο περιέχει πληροφορία για: μία λειτουργία, μία οντότητα στόχο, μηδέν ή περισσότερες παραμέτρους και ένα προαιρετικό περιβάλλον αίτησης. Μία αίτηση μπορεί να προκαλέσει μία υπηρεσία να εκτελεστεί για λογαριασμό του πελάτη. Αν κάτι πάει λάθος κατά την διάρκεια μιας αίτησης δημιουργείται μία εξαίρεση (exception).
4. Μία διεπαφή είναι η περιγραφή ενός συνόλου πιθανών λειτουργιών τις οποίες μπορεί να χρησιμοποιήσει ο πελάτης από μία οντότητα. Ο τύπος μίας διεπαφής είναι ένας τύπος που ικανοποιείται από οποιαδήποτε οντότητα ικανοποιεί μία συγκεκριμένη διεπαφή. Οι τύποι διεπαφών μπορεί να έχουν μία ή περισσότερες υλοποιήσεις.

Ο διεπαφές ορίζονται σε IDL.

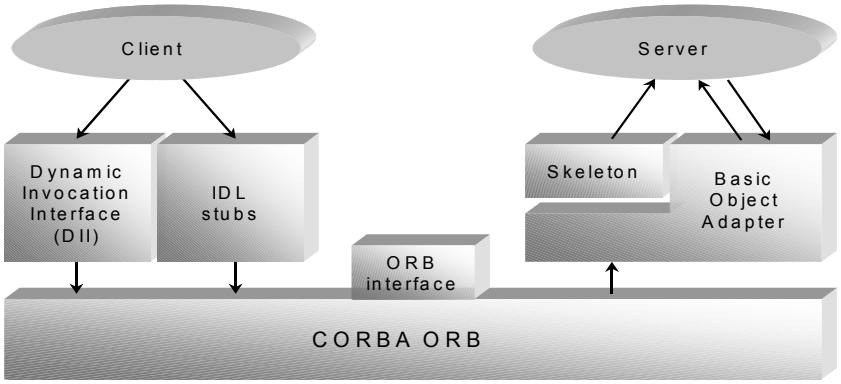
2.3.4 Αρχές της CORBA (CORBA Principles)

Το ORB (Σχήμα 2-11) παρέχει μία υποδομή που επιτρέπει στις οντότητες να επικοινωνούν μεταξύ τους σε ένα κατανεμημένο περιβάλλον, ανεξαρτήτως από την συγκεκριμένη πλατφόρμα στην οποία βρίσκονται καθώς και από την συγκεκριμένη τεχνική υλοποίησης. Εγγυάται φορητότητα και δια-λειτουργία οντοτήτων σε ένα ανομοιογενές δίκτυο. Είναι υπεύθυνο για τους μηχανισμούς ανεύρεσης της οντότητας για την αίτηση, για την προετοιμασία της να δεχτεί την αίτηση (ενεργοποίηση) καθώς και για την επικοινωνία των δεδομένων της αίτησης από και προς την οντότητα.

Ένας πελάτης κάνει μία αίτηση βρίσκοντας την Αναφορά Οντότητας (*Object Reference*) για μια οντότητα και γνωρίζοντας τον τύπο της οντότητας και την συγκεκριμένη λειτουργία που χρειάζεται. Η αίτηση αρχικοποιείται από τον πελάτη καλώντας stub routines οι οποίες είναι συγκεκριμένες για την οντότητα ή

κατασκευάζοντας την αίτηση δυναμικά. Η οντότητα δεν γνωρίζει αν καλέστηκε δυναμικά ή στατικά. Η δυναμική κλήση επιτρέπει να ζητηθεί μία υπηρεσία η οποία δεν ήταν γνωστή κατά την υλοποίηση του πελάτη.

Η έκδοση 2 της CORBA όρισε ένα πρωτόκολλο που χρησιμοποιεί το IP και το οποίο είναι γνωστό ως Internet Inter-ORB Protocol [IIOP] (*IIOP*). Επίσης περιγράφει το Dynamic Skeleton Interface.



Σχήμα 2-11 Δομή της CORBA

2.3.5 Υλοποιήσεις ORB (ORB Implementations)

Για να μπορέσει να γίνει η επικοινωνία μεταξύ ενός εξυπηρετητή και ενός πελάτη με αποδοτικό τρόπο ανεξαρτήτως της διεπαφής που χρησιμοποιείται, οι υλοποιήσεις του ORB παρακολουθούν την τοποθεσία του πελάτη και του εξυπηρετητή και χρησιμοποιούν το ανάλογο πρωτόκολλο επικοινωνίας για κάθε αίτηση.

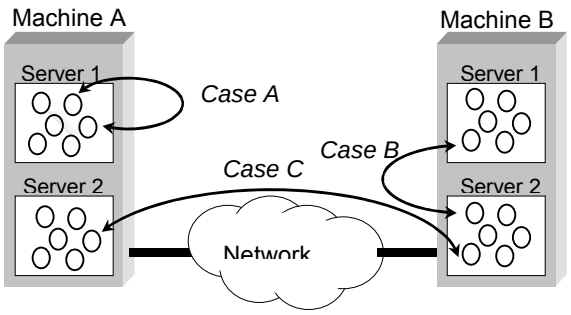
Ανάλογα με την τοποθεσία της οντότητας του εξυπηρετητή και του πελάτη διακρίνονται τρεις περιπτώσεις:

- **Περίπτωση Α:** Οι δύο οντότητες βρίσκονται στον ίδιο εξυπηρετητή CORBA. Σε αυτήν την περίπτωση και οι δύο οντότητες βρίσκονται στον ίδιο χώρο διευθύνσεων και επομένως η υλοποίηση του εξυπηρετητή είναι απευθείας προσβάσιμη από τον πελάτη. Οι αιτήσεις γίνονται σαν συνηθισμένες κλήσης συναρτήσεων. Η λειτουργία _bind που καλεί ο πελάτης επιστρέφει έναν δείκτη στην υλοποίηση του εξυπηρετητή.

- **Περίπτωση Β:** Οι δύο οντότητες βρίσκονται στο ίδιο μηχάνημα αλλά σε διαφορετικό εξυπηρετητή CORBA. Σε αυτήν την περίπτωση ενώ και οι δύο οντότητες βρίσκονται στο ίδιο μηχάνημα δεν βρίσκονται στον ίδιο χώρο διευθύνσεων. Για την βελτιστοποίηση της απόδοσής του το ORB χρησιμοποιεί τον μηχανισμό της επικοινωνίας δια-διεργασιών (*IPC*) του λειτουργικού συστήματος. Η ουρά μηνυμάτων IPC δημιουργείται με την κλήση της λειτουργίας _bind από τον πελάτη.

- **Περίπτωση Γ:** Οι δύο οντότητες βρίσκονται σε διαφορετικούς εξυπηρετητές CORBA οι οποίοι τρέχουν σε διαφορετικά μηχανήματα. Σε αυτήν την περίπτωση χρησιμοποιείται το δίκτυο. Η σύνδεση δικτύου που θα χρησιμοποιηθεί για την αίτηση δημιουργείται με την κλήση της _bind από τον πελάτη.

Και οι τρεις περιπτώσεις φαίνονται στο Σχήμα 2-12.



Σχήμα 2-12 Επικοινωνία CORBA Οντοτήτων

2.4 Η Υπηρεσία Καταλόγου X.500

Η Υπηρεσία Καταλόγου (*Directory Service - X.500*) που έχει προταθεί από την ITU και τον ISO περιγράφει μια κατανεμημένη βάση δεδομένων (*Directory*) καθώς και μια αρχιτεκτονική και ένα σύνολο πρωτοκόλλων τα οποία παρέχουν τους μηχανισμούς για πρόσβαση στη βάση αυτή. Βασικός στόχος της υπηρεσίας είναι η παροχή ενός παγκόσμιου καταλόγου ο οποίος αποτελείται από μεγάλο αριθμό εγγραφών και περιέχει γενικές πληροφορίες για πολλών ειδών οντότητες (για παράδειγμα ανθρώπους, διεργασίες, συστήματα και υπηρεσίες του δικτύου).

Η πληροφορία του καταλόγου είναι κατανεμημένη σε ένα αριθμό από ειδικές διεργασίες οι οποίες ονομάζονται DSAs (*Directory Service Agents*). Κάθε χρήστης του καταλόγου (ουσιαστικά κάθε διεργασία εφαρμογής που χρησιμοποιεί τον κατάλογο) αντιπροσωπεύεται από ένα DUA (*Directory User Agent*) ο οποίος παρέχει μηχανισμούς προσπέλασης, αναζήτησης, ενημέρωσης και τροποποίησης της πληροφορίας του καταλόγου μέσω του πρωτοκόλλου DAP (*Directory Access Protocol*), το οποίο χρησιμοποιείται για την επικοινωνία μεταξύ DUA και DSA. Αξίζει να σημειώσουμε ότι ο χρήστης καταλόγου σχετίζεται με έναν μόνο DSA στέλνοντας σε αυτόν αίτηση για κάποιο κομμάτι πληροφορίας. Αν αυτή η πληροφορία δεν βρίσκεται στο τμήμα καταλόγου που ο DSA διατηρεί, τότε ο DSA επικοινωνεί με κάποιον άλλο (ή άλλους) DSAs και προωθεί σε αυτούς την αίτηση.

2.4.1 Το μοντέλο της πληροφορίας του X.500

Η πληροφορία που αποθηκεύεται στο X.500 αποτελείται από εγγραφές οι οποίες αντιπροσωπεύουν οντότητες και ονομάζονται οντότητες καταλόγου (*directory objects*).

Το σύνολο των οντοτήτων καταλόγου απαρτίζει την Βάση Πληροφορίας του Καταλόγου (*Directory Information Base - DIB*), η οποία είναι οργανωμένη ιεραρχικά σε μία δενδροειδή δομή.

Κάθε οντότητα καταλόγου χαρακτηρίζεται από τις κλάσεις (*directory object class*) στις οποίες ανήκει. Οι κλάσεις αυτές καθορίζουν ένα σύνολο από υποχρεωτικά και προαιρετικά χαρακτηριστικά (*attributes*) τα οποία μπορεί να έχει η οντότητα. Κάθε χαρακτηριστικό μπορεί να έχει περισσότερες από μία τιμές σχηματίζοντας με αυτόν τον τρόπο ζευγάρια χαρακτηριστικού-τιμής. (Για παράδειγμα, το χαρακτηριστικό της υλικής υποδομής - hardware - μπορεί να δηλώνει περισσότερες από μία πλατφόρμες οι οποίες υποστηρίζονται). Οπως έχουμε ήδη προαναφέρει, για κάθε οντότητα καταλόγου ένα χαρακτηριστικό, το οποίο καλείται χαρακτηριστικό ονομασίας (*naming attribute*), χρησιμοποιείται μαζί με την τιμή του για τον σχηματισμό του ονόματος (σχετικά διακεκριμένο όνομα (*Relative Distinguished Name - RDN*) της οντότητας καταλόγου.

Η συνένωση όλων των *RDN* των κόμβων στο μονοπάτι από τη ρίζα του δέντρου της *DIB* μέχρι τον κόμβο που αντιστοιχεί σε μια συγκεκριμένη οντότητα καταλόγου αποτελεί το
διακεκριμένο όνομα (*Distinguished Name - DN*) της οντότητας.

Σύμφωνα με τις τυποποιήσεις του X.500 που έχουν προταθεί, όσον αφορά τη λογική οργάνωση της DIB, στα ψηλότερα επίπεδα εγγράφονται οντότητες που αντιπροσωπεύουν χώρες, γεωγραφικές περιοχές και οργανισμούς, ενώ στα χαμηλότερα επίπεδα βρίσκονται οντότητες που αντιπροσωπεύουν ανθρώπους, συστήματα και διεργασίες. Επίσης πρέπει να επισημάνουμε ότι λόγω των πολλών ειδών πληροφορίας που αποθηκεύονται στον κατάλογο, μερικές οντότητες έχουν κυρίως ομαδοποιητικό ρόλο, δηλαδή, περιλαμβάνουν
εγγραφές που έχουν κοινά χαρακτηριστικά.

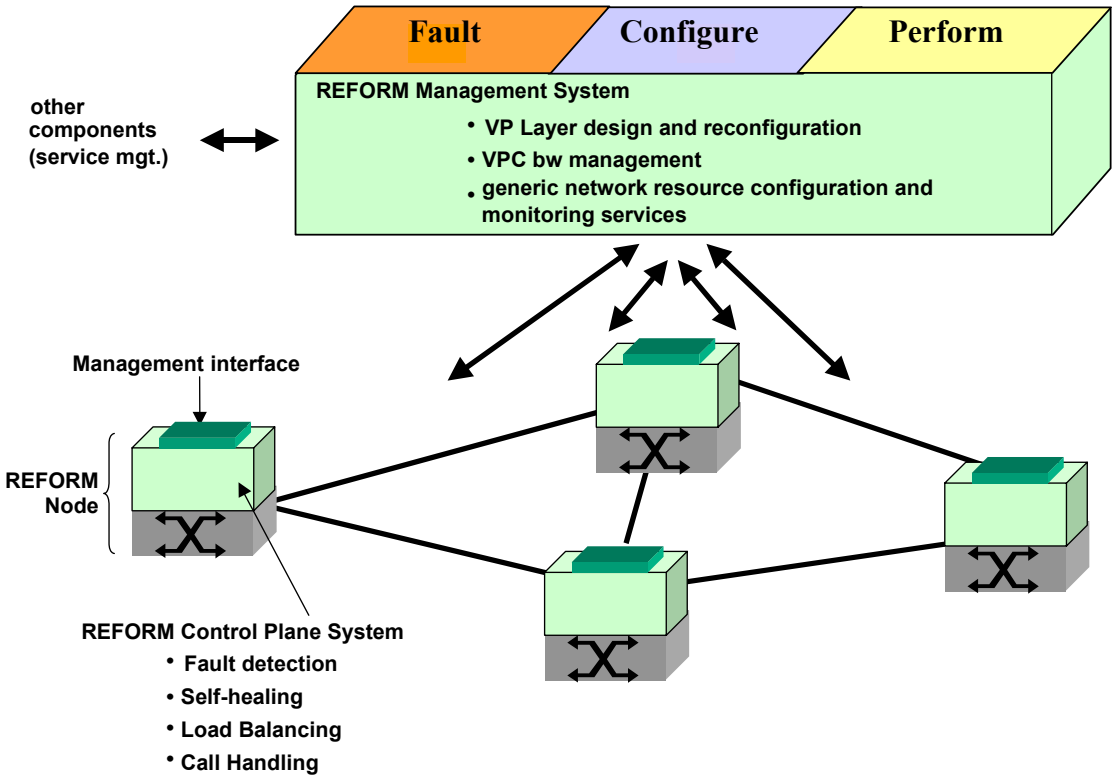
2.5 Το σύστημα REFORM

Υιοθετώντας την οπτική γωνία ενός παροχέα ATM υπηρεσιών ο οποίος παρέχει διαφορετικής ποιότητας ανά απαίτηση υπηρεσίες σύνδεσης η ερευνητική εργασία του ACTS REFORM σκοπεύει στην σχεδίαση, υλοποίηση και δοκιμή ενός πρωτότυπου συστήματος το οποίο παρέχει τα απαραίτητα μέσα και την απαραίτητη λειτουργικότητα για την εξασφάλιση της βιωσιμότητας του δικτύου σε αποδεκτά επίπεδα και με χαμηλό κόστος. Με τον όρο βιωσιμότητα εννοούνται δύο πράγματα: η διαθεσιμότητα του δικτύου για την αποδοχή νέων συνδέσεων και η δυνατότητά του για αποδοτική επαναφορά των ήδη υπαρχόντων υπηρεσιών του δικτύου σε συνθήκες σφάλματος. Το REFORM σύστημα, το οποίο φαίνεται στο Σχήμα 2-13, περιλαμβάνει της περιοχές διαχείρισης της διαμόρφωσης, της απόδοσης και των σφαλμάτων και συνδυάζει δύο διαφορετικά κομμάτια/μέρη:

Το control plane σύστημα (στο επίπεδο Στοιχείων του Δικτύου) το οποίο περιέχει την απαιτούμενη λειτουργικότητα διασύνδεσης στο δίκτυο μαζί με την ανίχνευση λαθών, την αυτόματη επιδιόρθωση και τη βασισμένη στην ποιότητα υπηρεσίας δυναμική λειτουργικότητα της δρομολόγησης. Αυτό το κομμάτι του REFORM έχει ενοποιηθεί μαζί με τα υπάρχοντα στοιχεία του δικτύου και αποτελεί τα στοιχεία του REFORM (REFORM Nodes).

Το management plane σύστημα (στο επίπεδο Διαχείρισης του Δικτύου) το οποίο ασχολείται με την αρχική διαμόρφωση και, κατά την λειτουργία, την δυναμική επαναδιαμόρφωση του επιπέδου εικονικών μονοπατιών σύνδεσης (VPCs) του δικτύου. Το κομμάτι αυτό περιλαμβάνει δυναμική διαχείριση του εύρους ζώνης των VPCs, την σχεδίαση του επιπέδου των VPCs καθώς και την δυναμική επανασχεδίασή του, την διαχείριση σφαλμάτων καθώς και γενικές λειτουργίες διαμόρφωσης και παρακολούθησης των πόρων.

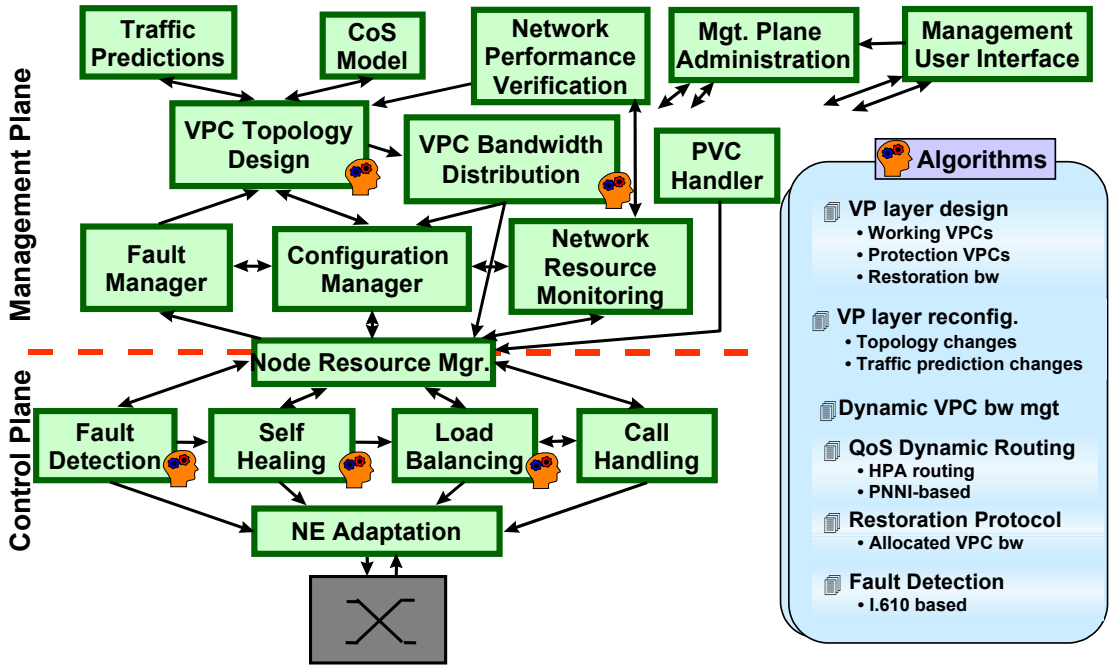
Το συνολικό σύστημα λειτουργεί με βάση ένα ιεραρχικό μοντέλο σε διαφορετικά επίπεδα τόσο όσον αφορά την ταχύτητα λειτουργίας όσο και την αφαίρεση. Στο Σχήμα 2-14 φαίνονται τα διαφορετικά μέρη (η αρχιτεκτονική) του συστήματος τόσο στο control όσο και στο management επίπεδο.



Σχήμα 2-13 Συνολική εικόνα του συστήματος REFORM

2.5.1 Θέματα Σχεδίασης και Απαιτήσεις

Εκτός από τις γενικές απαιτήσεις σχεδίασης (επεκτασιμότητα, φορητότητα, συμβατότητα, διαλειτουργικότητα, απόδοση) τα ακόλουθα θέματα ληφθήκανε υπόψη τόσο κατά την σχεδίαση όσο και κατά την υλοποίηση.



Σχήμα 2-14 Αρχιτεκτονική Συστήματος

2.5.1.1 Σχεδίαση των Στοιχείων του Δικτύου

Καθώς εξελίσσονται οι τεχνολογίες και οι ανάγκες γίνονται ολοένα πιο απαιτητικές, τα στοιχεία του δικτύου και οι κόμβοι μετατρέπονται από υλικό τερματισμού συνδέσεων σε πολύπλοκα συστήματα τα οποία περιλαμβάνουν ολοένα και περισσότερη λειτουργικότητα. Καθώς αυτή η λειτουργικότητα μετακινείται από το υλικό στο λογισμικό, η διατήρηση και η χρησιμοποίηση εξαρτάται ολοένα και περισσότερο από την διαμόρφωση του λογισμικού των στοιχείων. Γι' αυτόν το λόγο τα στοιχεία του δικτύου πρέπει να σχεδιαστούν ώστε νέες δυνατότητες να μπορούν να προστεθούν και οι ήδη υπάρχουσες να μπορούν να μεταβληθούν με ελάχιστο κόστος και σε όσο το δυνατόν λιγότερο χρόνο.

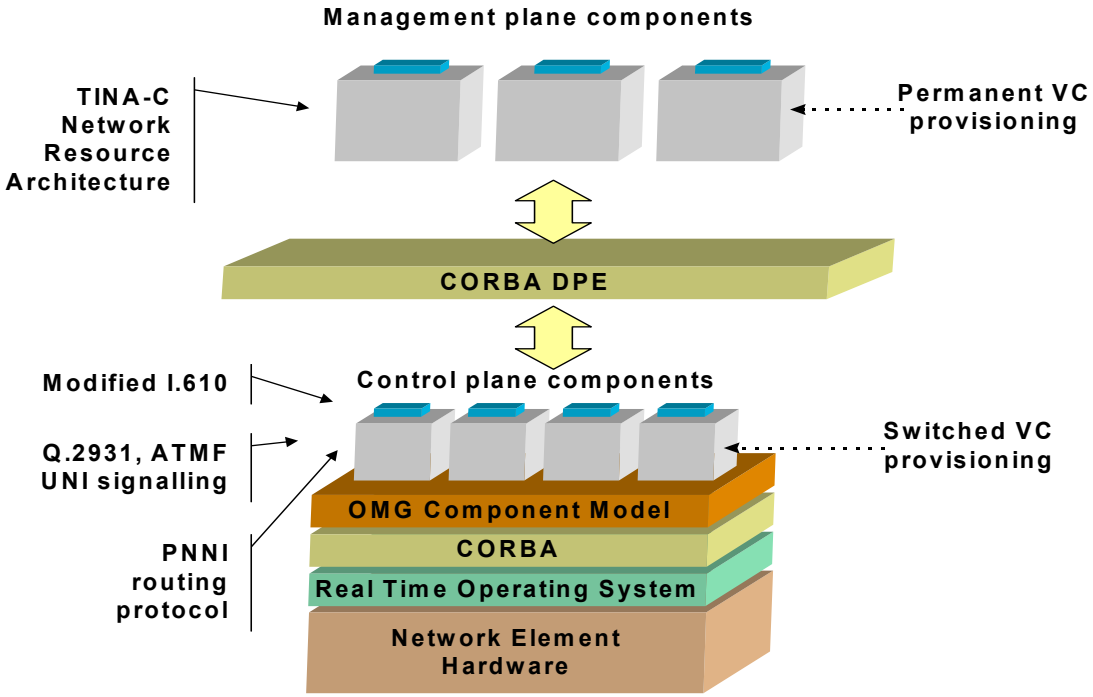
2.5.1.2 Ολοκλήρωση και Δια-λειτουργία των διαδικασιών Διαχείρισης του Δικτύου

Το management σύστημα του REFORM ενοποιεί την λειτουργία του σχεδιασμού του δικτύου με την δυναμική διαμόρφωση και την διαχείριση σφαλμάτων και απόδοσης. Η σχεδίαση του δικτύου αποτελεί την καρδιά του συστήματος παρέχοντας χαμηλού κόστους λειτουργία του δικτύου και δυνατότητα εφαρμογής πολιτικών που αφορούν την παροχή υπηρεσιών. Η δυναμική διαμόρφωση καθώς και η διαχείριση σφαλμάτων και απόδοσης απαιτούνται για την διαρκή βελτίωση της απόδοσης του δικτύου σύμφωνα με την πραγματική χρήση του

2.5.2 Χρησιμοποιούμενες Τεχνολογίες

Με βάση τις απαιτήσεις λειτουργικότητας το σύστημα REFORM σχεδιάστηκε και υλοποιήθηκε με την χρήση πολλών διαφορετικών τεχνολογιών (βλέπε Σχήμα 2-15). Η συνολική σχεδίαση βασίστηκε σε μια ανεξάρτητη από τον κατασκευαστή προδιαγραφή - το μοντέλο διαδικασιών του REFORM - παρά στην τυφλή υιοθέτηση μιας συγκεκριμένης τεχνολογίας. Συγκεκριμένα οι διαφορετικές τεχνολογίες που χρησιμοποιήθηκαν είναι:

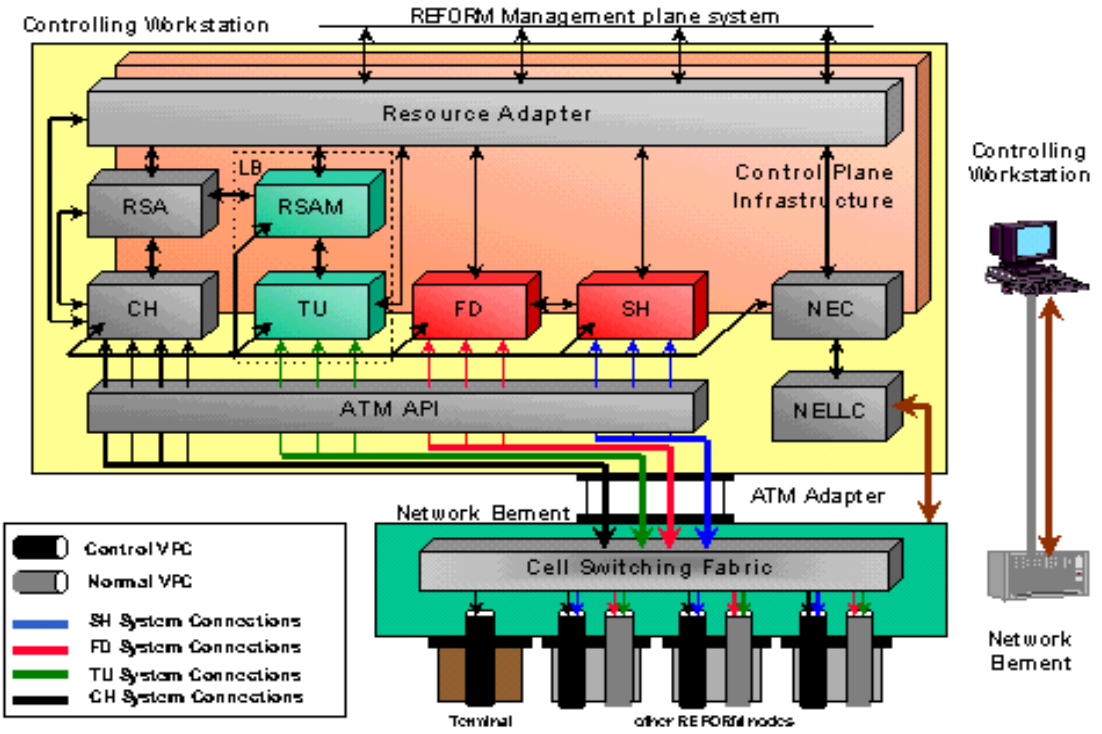
1. Λειτουργίες Λειτουργίας και Συντήρησης (Operation and Maintenance functions - OAM) [I.610]
2. ATM UNI 3.0 και ITU-T Q.2931
3. PNNI
4. OMG Component Model
5. OMG CORBA
6. TINA
7. TMN



Σχήμα 2-15 Οι Τεχνολογίες στο σύστημα REFORM

2.5.3 Ο Κόμβος REFORM (REFORM Node)

Στο σύστημα REFORM χρησιμοποιήθηκαν δίκτυα με μεταγωγείς που προέρχονται από διαφορετικούς κατασκευαστές με αποτέλεσμα να δημιουργούνται προβλήματα αφού οι σημερινές υλοποιήσεις του ίδιου πρωτόκολου σηματοδότησης (signaling protocol), από διαφορετικούς κατασκευαστές, παρουσιάζουν ακόμα προβλήματα διαλειτουργικότητας. Γιαυτό το λόγο αλλά και επειδή τα υπόλοιπα μέρη του control plane του συστήματος χρειαζόταν να συν-λειτουργούν με τα κομμάτια της σηματοδότησης τόσο για την συλλογή στατιστικών όσο και για την παροχή της λειτουργίας της δρομολόγησης, κάτι που δεν είναι δυνατόν με τα σημερινά στοιχεία που παρέχουν οι κατασκευαστές, το REFORM εισήγαγε την έννοια του Κόμβου REFORM (REFORM Node). Ο Κόμβος REFORM (Σχήμα 2-16) περιλαμβάνει τις λειτουργίες του control plane του συστήματος επιτρέποντας αλληλεπίδραση μεταξύ των στοιχείων του δικτύου καθώς και ομοιόμορφη σηματοδότηση και δρομολόγηση, ανεξαρτήτως του κατασκευαστή. Αυτό επιτυγχάνεται με την απομόνωση της συγκεκριμένης από τον κατασκευαστή τεχνολογίας από τα κομμάτια του συστήματος παρέχοντας ένα ενδιάμεσο επίπεδο προσαρμογής (adaptation layer) σε κάθε στοιχείο και κάνοντας τις κατάλληλες αλλαγές στα υπάρχοντα πρωτόκολλα.



Σχήμα 2-16 Ο Κόμβος REFORM (REFORM Node)

3 *Προδιαγραφή Λειτουργιών Συστήματος*

Στο κεφάλαιο αυτό παρουσιάζονται λεπτομερώς οι λειτουργίες του προτεινόμενου συστήματος. Όπως ήδη αναφέρθηκε στην εισαγωγή, οι λειτουργίες που παρέχει το σύστημα στηρίζονται κατά ένα μεγάλο βαθμό στις απαιτήσεις διαχείρισης σφαλμάτων του ερευνητικού προγράμματος του ACTS REFORM καθώς και στον καθορισμό λειτουργιών της διαχείρισης σφαλμάτων της Αρχιτεκτονικής Πόρων Δικτύου της TINA [NRAv3] (*Fault Management Fragment of TINA NRA*).

Τόσο στο τμήμα Διαχείρισης Σφαλμάτων της Αρχιτεκτονικής Πόρων Δικτύου της TINA, όσο και στην ερευνητική εργασία του ACTS REFORM, περιλαμβάνονται λειτουργίες και σε επίπεδο Διαχείρισης Στοιχείων (*EML*) και σε EML και σε NML επίπεδο. Επειδή, όπως αναφέρθηκε στην εισαγωγή, το προτεινόμενο σύστημα τοποθετείται στο NML επίπεδο, παρέχει εκείνες μόνο τις λειτουργίες που αναφέρονται σε αυτό. Στο Σχήμα 3-1 φαίνεται η θέση του συστήματος στο NML επίπεδο καθώς και οι αλληλεπιδράσεις με τα άλλα συστήματα διαχείρισης τόσο στο ίδιο επίπεδο (NTCM και CM) όσο και στο EML (REFORM Nodes ή άλλα EML συστήματα διαχείρισης σφαλμάτων).

Στο τμήμα Διαχείρισης Σφαλμάτων της TINA-NRA περιγράφονται συνολικά πέντε διαφορετικές λειτουργίες για ένα σύστημα διαχείρισης σφαλμάτων. Οι λειτουργίες αυτές είναι, η Επιτήρηση Ειδοποιήσεων (Alarm Surveillance), η Δοκιμή Πόρων (Testing), ο Εντοπισμός Σφαλμάτων (Fault Localization), η Επιδιόρθωση Λαθών (Fault Correction) και η Διοίκηση Προβλημάτων (Trouble Administration). Στο προτεινόμενο σύστημα περιλαμβάνονται μόνο τρεις, η Επιτήρηση Ειδοποιήσεων, η Δοκιμή Πόρων (Testing) και ο Εντοπισμός Λαθών (Fault Localization). Όπως αναφέρθηκε ήδη, το προτεινόμενο σύστημα θεωρεί ότι λειτουργία της επιδιόρθωσης σφαλμάτων (Fault Correction):

- είτε παρέχεται από άλλο σύστημα που παίρνει την γνώση της κύριας αιτίας (ή/και τις ειδοποιήσεις σφάλματος) από αυτό το σύστημα και εφαρμόζει κάποια πολιτική επιδιόρθωσης,
- είτε παρέχεται μέσω αυτόματων μηχανισμών επιδιόρθωσης στο EML επίπεδο όπως σχεδιάστηκε και υλοποιήθηκε στο REFORM σύστημα (SH component of REFORM Node)

Σε κάθε περίπτωση όμως παρέχεται η γνώση της πρωταρχικής αιτίας σφάλματος προς άλλα συστήματα και λαμβάνονται οι ειδοποιήσεις για τις επιδιορθώσεις για να είναι εφικτή η παροχή των υπολοίπων λειτουργιών του συστήματος.

Όσον αφορά τη Διοίκηση Προβλημάτων (Trouble Administration), θεωρείται ότι παρέχεται από άλλο σύστημα ή τον χρήστη διαχείρισης σφαλμάτων και δεν περιλαμβάνεται στις λειτουργίες του συστήματος.

Επομένως οι λειτουργίες που περιλαμβάνονται και που θα δούμε αναλυτικά σε αυτό το κεφάλαιο είναι:

1. Ανάκτηση και διατήρηση της πληροφορίας που αφορά την τοπολογία του φυσικού επιπέδου του δικτύου.
2. Ανάκτηση της πληροφορίας δέσμευσης/αποδεύσμεσης των πόρων του δικτύου καθώς και χαρακτηριστικά παροχής ποιότητας υπηρεσιών αυτών.
3. Ανάθεση χαρακτηριστικών διαχείρισης σφαλμάτων σε κάθε πόρο, ανάλογα με τα χαρακτηριστικά παροχής ποιότητας υπηρεσιών του.
4. Αρχικοποίηση/σταμάτημα των διαδικασιών παρακολούθησης πόρων ανάλογα με την πληροφορία δέσμευσης/αποδέσμευσής τους και τα χαρακτηριστικά διαχείρισης σφαλμάτων που τους έχουν ανατεθεί.
5. Επιτήρηση/Εποπτεία των ειδοποιήσεων σφαλμάτων (Fault *Alarm Surveillance*) τόσο από τις διαδικασίες παρακολούθησης όσο και από τις διαδικασίες επιδόρθωσης.
6. Δυνατότητα δοκιμής των πόρων (*Resource Testing*).

7. Συσχέτιση των ειδοποιήσεων και εντοπισμός της πρωταρχικής αιτίας σφάλματος (*Fault Correlation-Localization*).
8. Κατασκευή και διατήρηση ενός χάρτη με στατιστικά στοιχεία για τα σφάλματα των πόρων του δικτύου (*Network Risk Map*).

Το πως οι περιγραφόμενες στο κεφάλαιο αυτό λειτουργίες χωρίζονται στα διάφορα υπολογιστικά μέρη του συστήματος καθώς και η ακολουθία πραγματοποίησής τους και η συσχέτισή τους με άλλα συμβάντα (events) στο υπό διαχείριση δίκτυο δίνεται στο κεφάλαιο 4 (Προδιαγραφή Υπολογιστικής Συστήματος). Επίσης αναλυτική περιγραφή της πληροφορίας που διατηρείται για τις ειδοποιήσεις σφάλματος, τις δοκιμές, την τοπολογία του δικτύου και τα στατιστικά καθώς και των δομών που χρησιμοποιούνται για την διατήρησή της δίνεται στο κεφάλαιο 5 (Προδιαγραφή Μοντέλου Πληροφορίας Συστήματος).

3.1 Χαρακτηριστικά Διαχείρισης Σφαλμάτων και Πληροφορία Τοπολογίας Πόρων

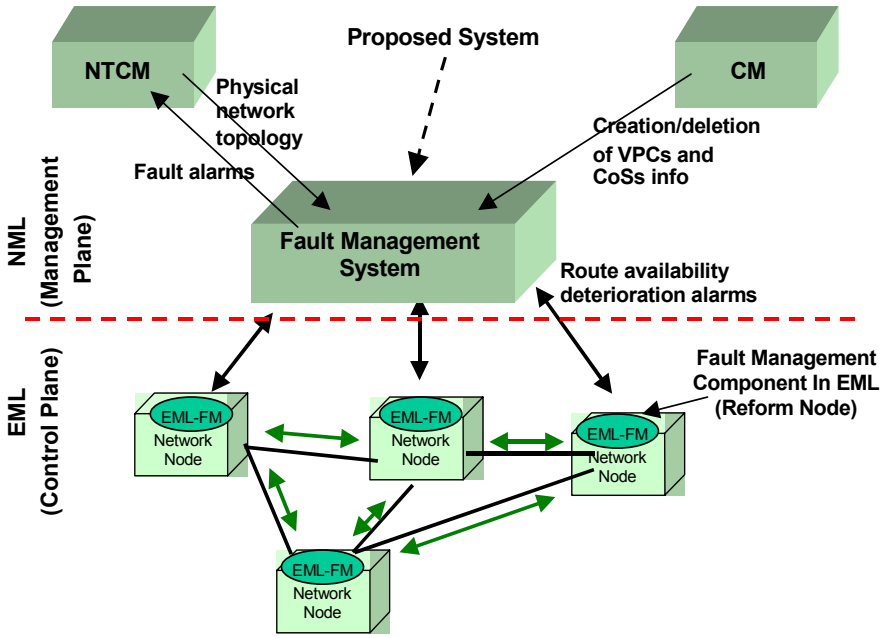
3.1.1 Χαρακτηριστικά Διαχείρισης Σφαλμάτων

Για να μπορεί ένα σύστημα διαχείρισης σφαλμάτων να εκτελεί ενέργειες σχετικές με διαχείριση σφαλμάτων (παρακολούθηση/επιδιόρθωση πόρων) σε ένα δίκτυο το οποίο παρέχει διαφορετικής ποιότητας υπηρεσίες όπως είναι ένα δίκτυο *ATM*, πρέπει να γνωρίζει κάποια χαρακτηριστικά διαχείρισης σφαλμάτων για κάθε πόρο. Αυτά τα χαρακτηριστικά περιλαμβάνουν:

1. Τις παραμέτρους παρακολούθησης των πόρων (**Monitor Class**), που αντιστοιχούν στο μέγιστο χρονικό διάστημα που μπορεί να περάσει αφού συμβεί ένα σφάλμα και πριν αυτό ανιχνευτεί από τις διαδικασίες παρακολούθησης. Η παράμετρος αυτή είναι χρήσιμη γιατί καθορίζει την συχνότητα με την οποία ένας πόρος εξετάζεται για την κατάσταση σφάλματος του, καθώς και την προτεραιότητα που θα δοθεί από την διαδικασία παρακολούθησης στην αναφορά του σφάλματος αν ανιχνευθούν πολλά σφάλματα μαζί με διαφορετικές παραμέτρους παρακολούθησης.

2. Τις παραμέτρους επιδιόρθωσης των πόρων (**Restoration Class**), που αντιστοιχούν στο μέγιστο χρονικό διάστημα που μπορεί να περάσει αφού συμβεί ένα σφάλμα σε έναν πόρο και πριν αυτό επιδιορθωθεί. Η παράμετρος αυτή είναι χρήσιμη γιατί καθορίζει την προτεραιότητα που θα δοθεί από την διαδικασία επιδιόρθωσης αν περισσότεροι από έναν πόρο είναι να επιδιορθωθούν κάποια δεδομένη στιγμή και αυτό δεν μπορεί να γίνει παράλληλα.

Φυσικά υπάρχει άμεση συσχέτιση μεταξύ των δύο αυτών παραμέτρων αφού δεν είναι δυνατόν οποιαδήποτε κλάση παρακολούθησης να μπορεί να χρησιμοποιηθεί για έναν πόρο ο οποίος έχει συγκεκριμένη κλάση επιδιόρθωσης. Έτσι στο παρόν σύστημα επιλέχθηκε, η κλάση επιδιόρθωσης να περιλαμβάνει και την κλάση παρακολούθησης στην οποία αντιστοιχεί (Κεφάλαιο 5, Προδιαγραφή Μοντέλου Πληροφορίας Συστήματος).



Σχήμα 3-1 Θέση του συστήματος και αλληλεπιδράσεις του με άλλα συστήματα στα δύο επίπεδα (NML και EML)

3.1.2 Πληροφορία Τοπολογίας Πόρων

Η πληροφορία τοπολογίας πόρων του δικτύου χρειάζεται σε ένα σύστημα διαχείρισης σφαλμάτων τόσο για να γνωρίζει το σύστημα αυτό που βρίσκονται οι πόροι στους οποίους θα αρχίσουν διαδικασίες παρακολούθησης (και επιδιόρθωσης αν χρειαστεί), αλλά και για να μπορεί να συσχετίζει τα σφάλματα που συμβαίνουν σε διαφορετικούς πόρους. Μα αυτόν το τρόπο είναι δυνατή η απόκτηση της πληροφορίας της επιρροής του σφάλματος ενός πόρου στην κατάσταση σφάλματος άλλων πόρων.

3.1.2.1 Πληροφορία Τοπολογίας Φυσικού επιπέδου

Η πληροφορία τοπολογίας φυσικού επιπέδου, δηλαδή των γραμμών και των κόμβων, του υπό διαχείριση δικτύου λαμβάνεται από την Διαχείριση Διαμόρφωσης Τοπολογίας Δικτύου (*NTCM*). Η αλληλεπίδραση για την απόκτηση της φαίνεται στο Σχήμα 3-1.

3.1.2.2 Πληροφορία Τοπολογίας επιπέδου εικονικών μονοπατιών σύνδεσης

Η πληροφορία τοπολογίας επιπέδου εικονικών μονοπατιών σύνδεσης (*VPCs*) λαμβάνεται από την Διαχείριση Συνδέσεων (CM). Συγκεκριμένα λαμβάνεται η πληροφορία της δέσμευσης (δημιουργίας) ενός εικονικού μονοπατιού σύνδεσης, που περιλαμβάνει την τοπολογία του και την ποιότητα υπηρεσίας του, και η πληροφορία αποδέσμευσης (σβησίματος) του. Σε κάθε ενημέρωση για δέσμευση/αποδέσμευση, ενημερώνεται ανάλογα και η τοπολογία του δικτύου που διατηρείται από το σύστημα. Καθώς και οι υπεύθυνες για τον πόρο διαδικασίες παρακολούθησης/επιδιόρθωσης. Η αλληλεπίδραση για την απόκτηση της πληροφορίας αυτής φαίνεται επίσης στο Σχήμα 3-1.

3.1.3 Ανάθεση Χαρακτηριστικών Διαχείρισης και Αρχικοποίηση/Σταμάτημα των διαδικασιών της

Όπως αναφέρθηκε στην προηγούμενη παράγραφο το σύστημα λαμβάνει την πληροφορία για την δέσμευση/αποδέσμευση ενός πόρου στο υπό διαχείριση δίκτυο καθώς και τα χαρακτηριστικά ποιότητας υπηρεσίας του. Ανάλογα με τα χαρακτηριστικά αυτά αναθέτει την κατάλληλη κλάση επιδιόρθωσης στον πόρο αυτόν. Συγκεκριμένα το σύστημα διατηρεί τρεις κλάσεις επιδιόρθωσης (A, B, C) και τρεις παρακολούθησης (A, B, C) και όσο ποιο υψηλή ποιότητα υπηρεσίας παρέχει ένας πόρος τόσο ποιο υψηλή κλάση επιδιόρθωσης του αναθέτει (καθώς και την αντίστοιχη κλάση παρακολούθησης).

Αμέσως μετά την πληροφορία δέσμευσης ενός πόρου και την ανάθεση των χαρακτηριστικών διαχείρισης σε αυτόν, το σύστημα επικοινωνεί με τις υπεύθυνες για τον πόρο αυτό διαδικασίες παρακολούθησης και τις αρχικοποιεί παρέχοντάς τους την πληροφορία του πόρου καθώς και την πληροφορία της κλάσης επιδιόρθωσης του. Οι διαδικασίες παρακολούθησης εξάγουν την πληροφορία της κλάσης παρακολούθησης από την κλάση επιδιόρθωσης και αρχίζουν την παρακολούθηση με βάση την κλάση αυτή.

Αν ληφθεί ενημέρωση για αποδέσμευση κάποιου πόρου, το σύστημα επικοινωνεί πάλι με τις υπεύθυνες διαδικασίες παρακολούθησης και τις σταματάει.

3.2 Επιτήρηση/Εποπτεία Ειδοποιήσεων Σφαλμάτων

Η επιτήρηση/εποπτεία ειδοποιήσεων σφαλμάτων (Fault Alarm Surveillance) είναι μια από τις πιο σημαντικές λειτουργίες της διαχείρισης σφαλμάτων αφού επιτρέπει την παρακολούθηση των πόρων και την απόκτηση και διατήρηση της πληροφορίας σφάλματός τους έξω από αυτούς.

Στο σύστημά η λειτουργία αυτή περιλαμβάνει τις παρακάτω υπολειτουργίες:

1. Ανίχνευση του σφάλματος ενός πόρου μέσα σε ένα δίκτυο με την λήψη ειδοποιήσεων σφάλματος από διαδικασίες παρακολούθησης του πόρου αυτού.
2. Φιλτράρισμα των ειδοποιήσεων σφάλματος, τόσο για να μην έχουμε περιττές ειδοποιήσεις οι οποίες οδηγούν σε άσκοπες και χρονοβόρες ενέργειες (ξεκίνημα διαδικασιών εντοπισμού της πρωταρχικής αιτίας σφάλματος ενώ αυτή έχει ήδη εντοπιστεί κλπ) όσο και για την καλύτερη προώθησή τους (Forwarding) στον ενδιαφερόμενο.
3. Προώθησή των ειδοποιήσεων σφάλματος (Fault Alarm Forwarding) στον ενδιαφερόμενο. Η λειτουργία αυτή περιλαμβάνει την αποθήκευση της πληροφορίας του ενδιαφέροντος για συγκεκριμένες ειδοποιήσεις που αφορούν συγκεκριμένους πόρους του δικτύου, έλεγχο της πληροφορίας αυτής κάθε φορά που κάποια ειδοποίηση σφάλματος λαμβάνεται, ώστε να εντοπισθούν οι ενδιαφερόμενοι παραλήπτες, και αποστολή της ειδοποίησης σε αυτούς.
4. Δημιουργία εγγραφών (records) των ειδοποιήσεων σφαλμάτων και δυνατότητα απόκτησης τους από οποιονδήποτε ενδιαφερόμενο. Οι εγγραφές αυτές αποτελούν την ιστορία σφάλματος του πόρου και μπορούν να χρησιμοποιηθούν είτε από το ίδιο το σύστημα είτε από κάποιο άλλο για την παροχή άλλων σύνθετων λειτουργιών (εντοπισμός πρωταρχικής αιτίας σφάλματος, αναφορά συνολικών σφαλμάτων, δημιουργία στατιστικών κλπ).

3.3 Δοκιμή πόρων

Η Δοκιμή των πόρων είναι μια απλή λειτουργία ανάμεσα στις λειτουργίες διαχείρισης σφαλμάτων που σκοπεύει στην επιβεβαίωση της ορθής ή μη λειτουργίας ενός πόρου του δικτύου.

Η λειτουργία δοκιμής των πόρων στο προτεινόμενο σύστημα χωρίζεται στις εξής δύο υπολειτουργίες:

1. Συστηματική δοκιμή των τμημάτων (segments) μιας σύνδεσης (κόμβοι, γραμμές και εικονικά μονοπάτια σύνδεσης), και αποθήκευση των αποτελεσμάτων των δοκιμών αυτών σε εγγραφές (records).
2. Παροχή συγκεκριμένων διεπαφών μέσω των οποίας κάποιο άλλο σύστημα μπορεί τόσο να αρχίσει δοκιμή σε έναν πόρο και να λάβει το αποτέλεσμά της όσο και να λάβει τις εγγραφές αποτελεσμάτων παλαιότερων δοκιμών.

3.4 Συσχέτιση Ειδοποιήσεων και Εντοπισμός πρωταρχικής αιτίας σφάλματος (Fault Correlation-Localization)

Ένα σφάλμα στο δίκτυο μπορεί να προκαλέσει τη δημιουργία, ενός μεγάλου αριθμού ειδοποιήσεων σφάλματος από διάφορους πόρους οι οποίοι είναι άμεσα ή έμμεσα συσχετισμένοι με αυτό (π.χ. ένα σφάλμα σε μια γραμμή θα δημιουργήσει ειδοποιήσεις σφαλμάτων σε όλα τα εικονικά μονοπάτια τα οποία παρακολουθούνται και χρησιμοποιούν την γραμμή αυτή).

Η γνώση της πρωταρχικής αιτίας του σφάλματος οδηγεί τόσο σε καλύτερα συμπεράσματα για την σχεδίαση και ανοχή του δικτύου σε περιπτώσεις σφαλμάτων (αφού βγαίνουν πολύτιμα συμπεράσματα με την μορφή στατιστικών για την ανοχή των πόρων) όσο και στην αποφυγή περιττών επιδιορθώσεων πόρων (αφού η επιδιόρθωση αυτού που δημιούργησε τα σφάλματα οδηγεί σε άμεση επαναλειτουργία των υπολοίπων). Για τον λόγο αυτό μία από τις λειτουργίες του προτεινόμενου συστήματος είναι και η δυνατότητά του να εντοπίζει την πρωταρχική αυτή αιτία του σφάλματος στο υπό διαχείριση δίκτυο.

Ο συσχετισμός των ειδοποιήσεων σφαλμάτων (alarm correlation) είναι μία πολύπλοκη διαδικασία η οποία αποτελεί ξεχωριστή περιοχή μελέτης και η οποία πολλές φορές αναφέρεται και ως συσχετισμός συμβάντων (event correlation) και φιλτράρισμα ειδοποιήσεων (alarm filtering). Για τον συσχετισμό χρησιμοποιούνται μοντέλα συμπεριφοράς (behaviour models), μηχανές κατάστασης (state machines), κανόνες (rule-based), γράφοι (graphs) κλπ. Επειδή σκοπός της εργασίας δεν είναι η εφαρμογή πολύπλοκων αλγορίθμων συσχέτισης αλλά η παροχή των απαραίτητων λειτουργιών και πληροφορίας για να είναι δυνατή η λειτουργία τους, υλοποιήθηκε κυρίως για λόγους επίδειξης ένας απλός, αλλά αποτελεσματικός όπως φαίνεται από τις δοκιμές (κεφάλαιο 7), αλγόριθμος εντοπισμού, ο οποίος βασίζεται στο γράφο (graph-based) που αντικατοπτρίζει την τοπολογική σχέση των πόρων του δικτύου. Σύμφωνα με τον αλγόριθμο αυτό:

1. Για κάθε ειδοποίηση σφάλαμτος που λαμβάνεται, και αφού αυτή φιλτραριστεί, και περάσει κάποιο χρονικό διάστημα για να ληφθούν ειδοποιήσεις που τυχόν σχετίζονται, ξεκινάει την διαδικασία εντοπισμού, η οποία περιλαμβάνει:

   - Εξέταση της τοπολογίας του δικτύου για εντοπισμό των πόρων στους οποίους θα μπορούσε να οφείλεται εξαρχής το σφάλμα.
   - Έλεγχο, για τους πόρους αυτούς, των ειδοποιήσεων σφάλματος που έχουν ληφθεί ώστε να βγουν συμπεράσματα για την πρωταρχική αιτία.
   - Αρχικοποίηση μιας ακολουθίας δοκιμών στους πόρους εκείνους όπου δεν υπάρχει κάποια ειδοποίηση σφάλματος που να υποδηλώνει την μη λειτουργία τους, και στους οποίους μπορεί να οφείλεται το σφάλμα.
   - Και τέλος, με βάση τις ειδοποιήσεις που έχουν ληφθεί, την τοπολογία του δικτύου και τα αποτελέσματα των δοκιμών, αναφορά της πρωταρχικής αιτίας σφάλματος τόσο στον χρήστη διαχείρισης όσο και σε άλλα συστήματα που ενδιαφέρονται.

2. Αφού βγει αποτέλεσμα πρωταρχικής αιτίας σφάλματος, και εφόσον δεν έχει ληφθεί ειδοποίηση για επιδιόρθωση του, κάθε άλλο σφάλμα το οποίο ανιχνεύεται και εξαρτάται από ταυτό που δεν έχει επιδιορθωθεί, θεωρείται ότι οφείλεται σε αυτό και δεν αρχίζει καινούργιος αλγόριθμος εντοπισμού (αλλά μόνο εφόσον περάσει αρκετός χρόνος και δεν έχει ληφθεί ακόμα ειδοποίηση επιδιόρθωσης).

3.5Δημιουργία και Διατήρηση του χάρτη στατιστικών των πόρων του δικτύου (Network Risk Map)

Για την καλύτερη/αποδοτικότερη σχεδίαση/διαμόρφωση των συνδέσεων του δικτύου (επιλογή μονοπατιών πάνω από τα οποία τα εικονικά μονοπάτια σύνδεσης θα δημιουργηθούν, αντικατάσταση προβληματικών γραμμών ή/και κόμβων) είναι σημαντικό να γνωρίζουμε την ιστορία σφαλμάτων των διαφόρων πόρων με την μορφή στατιστικών για αυτούς. Επίσης η πληροφορία αυτή είναι χρήσιμη και για να μπορούμε να γνωρίζουμε την ανοχή/συμπεριφορά του δικτύου μας σε περιπτώσεις σφάλματος και γενικότερα κακής λειτουργίας.

Επειδή μόνο το σύστημα διαχείρισης σφαλμάτων στο NML επίπεδο μπορεί να έχει την πληροφορία αυτή (αφού προέρχεται από όλο το δίκτυο), γι' αυτό και αυτό πρέπει να την δημιουργελι και παρέχει προς άλλα συστήματα που τα ενδιαφέρει (πχ NTCM).

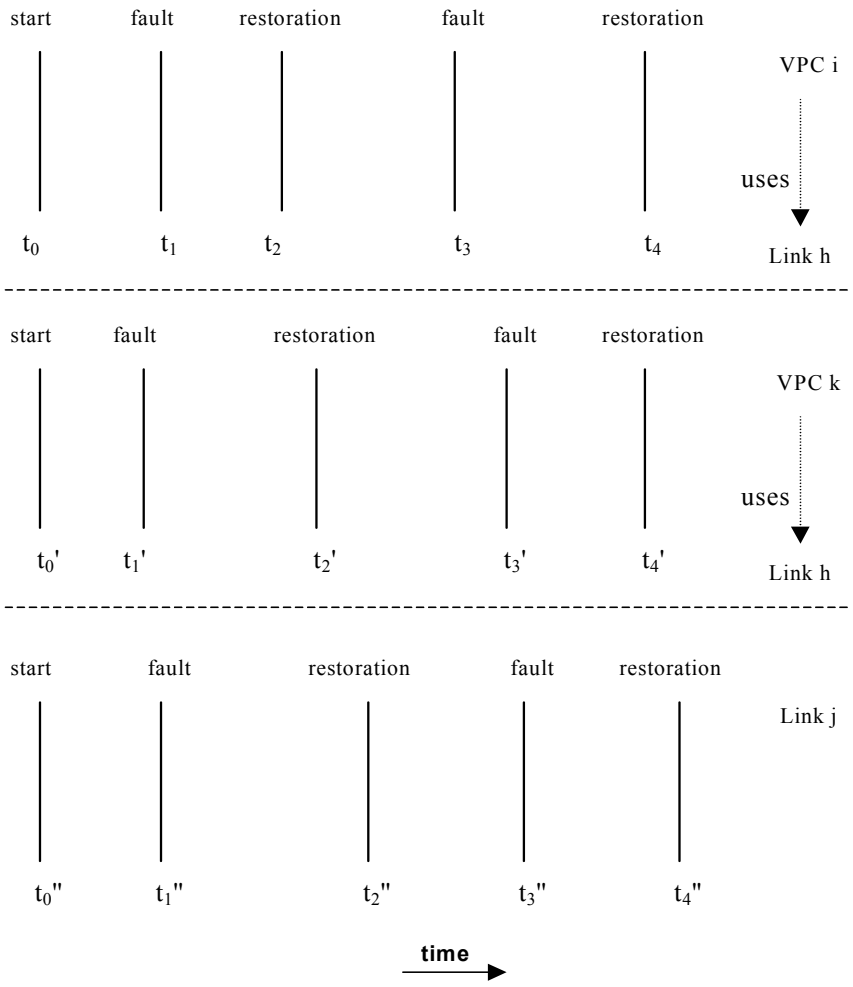
Επειδή σκοπός της εργασίας δεν είναι η παροχή κάποιων συγκεκριμένων στατιστικών αλλά η παροχή της δυνατότητας δημιουργίας τους μέσα από την διατήρηση της ιστορίας κατάστασης σφάλματος των πόρων και την συσχέτιση των σφαλμάτων τους, για σκοπούς κυρίως επίδειξης της δυνατότητας αυτής το σύστημα υπολογίζει και παρέχει τα εξής στατιστικά:

1. Το μέσο χρόνο για τον οποίο ο πόρος ήταν εκτός λειτουργίας (*MTTR - Mean Time To Repair*).
2. Το μέσο χρόνο μεταξύ δύο διαδοχικών σφαλμάτων σε έναν πόρο (*MTBF - Mean Time Between Failures*).
3. Την αξιοπιστία του πόρου ως προς την χρήση του από άλλους πόρους (RUR - Resource Usage Reliability). Το στατιστικό στοιχείο αυτό αναφέρεται σε γραμμές και κόμβους του δικτύου και υπολογίζεται για επίδειξη της δυνατότητας παραγωγής πολύπλοκων στατιστικών μέσα από την συσχέτιση των σφαλμάτων. Η τιμή του υπολογίζεται ως ο μέσος όρος του μέσου όρου των MTBF των άλλων πόρων που χρησιμοποιούν τον πόρο αυτό. Δηλαδή αν n πόροι χρησιμοποιούν έναν άλλο πόρο, τότε το *RUR* του είναι:

$$RUR = \frac{\sum_{i=1}^{n} MTBF_i}{n}$$

Στην περίπτωση που κάποιος πόρος δεν έχει παρουσιάσει σφάλμα (MTBF=0) το MTBF του στον υπολογισμό τίθεται ίσο με την χρονική διάρκεια ύπαρξής του (life time). Αυτή η συνάρτηση υπολογισμού της τιμής επιλέχτηκε τόσο γιατί χρησιμοποιεί τις τιμές χαρακτηρισμού σφάλματος των πόρων που χρησιμοποιούν τον πόρο αυτόν όσο και γιατί σκοπός μας είναι να βαθμολογήσουμε τα συμβάντα (events) των σφαλμάτων (και αυτό το πετυχαίνουμε με χρήση του MTBF) και όχι τόσο τις συνέπειες σε κάθε συμβάν.

Στο Σχήμα 3-2 φαίνεται μία περίπτωση ακολουθίας σφαλμάτων μίας γραμμής (*Link j*) και δύο εικονικών μονοπατιών σύνδεσης (*VPC i, k*) τα οποία χρησιμοποιούν μία κοινή γραμμή (*Link h*) και τα δύο. Με βάση αυτήν την ακολουθία η πληροφορία ενημερώνεται ως εξής:



Σχήμα 3-2 Ακολουθία σφαλμάτων/επιδιορθώσεων πόρων

Γραμμή j (Link j):

$$MTTR_j = \frac{(t_2'' - t_1'') + (t_4'' - t_3'')}{2}$$

$$MTBF_j = \frac{(t_1'' - t_0'') + (t_3'' - t_1'')}{2}$$

Γραμμή h (Link h):

$$RUR_h = \frac{MTBF_i + MTBF_k}{2}$$

Εικονικό Μονοπάτι Σύνδεσης i (VPC i):

$$MTBF_i = \frac{(t_1 - t_0) + (t_3 - t_1)}{2}$$

$$MTTR_i = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}$$

Εικονικό Μονοπάτι Σύνδεσης k (VPC k):

$$MTBF_k = \frac{(t_1' - t_0') + (t_3' - t_1')}{2}$$

$$MTTR_k = \frac{(t_2' - t_1') + (t_4' - t_3')}{2}$$

Το προτεινόμενο λοιπόν σύστημα κατασκευάζει, με βάση τις ειδοποιήσεις σφάλματος που λαμβάνει και της συσχέτισής τους, έναν χάρτη στατιστικών σφάλματος του δικτύου (*Network Risk Map*) και παρέχει την κατάλληλη διεπαφή ώστε κάθε άλλο ενδιαφερόμενο σύστημα να μπορεί να αποκτήσει.

Τα στατιστικά αυτά επιλέχτηκαν τόσο για επίδειξη των δυνατοτήτων του παρόντος συστήματος, όπως προαναφέρθηκε, αλλά και για χρήση από τους αλγορίθμους επαναδιαμόρφωσης του υπό διαχείριση δικτύου που εφαρμόζονται στο σύστημα REFORM ACTS.

4 *Προδιαγραφή Υπολογιστικής Συστήματος*

Στο κεφάλαιο αυτό περιγράφεται η εφαρμογή που υλοποιεί το προτεινόμενο σύστημα διαχείρισης σφαλμάτων με όρους υπολογιστικών οντοτήτων (ή κομματιών προγραμμάτων) οι οποίες περικλείουν μέρος των παρεχόμενων λειτουργιών, που περιγράφηκαν στο κεφάλαιο 3, και αλληλεπιδρούν μεταξύ τους για να πετύχουν το συνολικό αποτέλεσμα του συστήματος.

Ο τρόπος με τον οποίο διαχωρίζονται οι λειτουργίες στις διάφορες υπολογιστικές οντότητες καθώς και οι διεπαφές που αυτές παρέχουν για επικοινωνία με άλλες υπολογιστικές οντότητες και συστήματα βασίζεται τόσο στην υπολογιστική προδιαγραφή του τμήματος Διαχείρισης Σφαλμάτων της TINA-NRA όσο και σε γενικότερες αρχές κατανεμημένων συστημάτων (ανεξαρτησία υπολογιστικών οντοτήτων, κατανομή φόρτου λειτουργιών κλπ).

Η παρουσίαση ξεκινάει με την περιγραφή των γραφικών συμβολισμών σχημάτων που θα χρησιμοποιηθεί στο κεφάλαιο αυτό για να μπορεί ο αναγνώστης να κατανοήσει τα σχήματα. Στην συνέχεια περιγράφονται ποιες είναι οι τέσσερις υπολογιστικές οντότητες που αποτελούν το σύστημα και ακολουθεί η αναλυτική προδιαγραφή της εσωτερικής λειτουργικότητας τους καθώς και των διεπαφών για αλληλεπίδραση που κάθε μια από αυτές έχει. Τέλος παρουσιάζονται και οι τέσσερις οντότητες με τον ακριβή τρόπο που αλληλεπιδρούν δίνοντας την συνολική εικόνα του συστήματος.

Ο αναλυτικός ορισμός των διεπαφών που παρέχει κάθε υπολογιστική οντότητα γίνεται σε Γλώσσα Ορισμού Διεπαφών [IDL] (*Interface Definition Language - IDL*) στο κεφάλαιο 5 (Προδιαγραφή Μοντέλου Πληροφορίας Συστήματος).

4.1 Γραφικοί συμβολισμοί σχημάτων παρουσίασης

Για την παρουσίαση της προδιαγραφής της υπολογιστικής του συστήματος χρησιμοποιούνται τα παρακάτω σύμβολα στα σχήματα (Σχήμα 4-1):

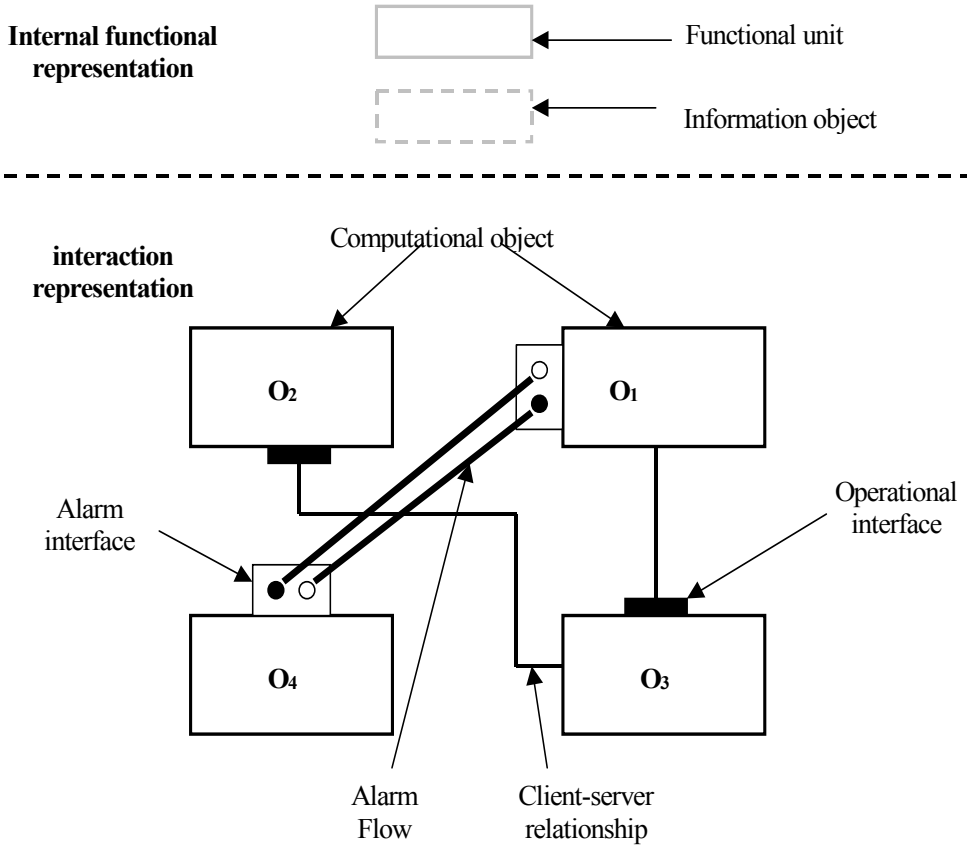
Για την εσωτερική λειτουργικότητα:

1. Μικρά άσπρα ορθογώνια παραλληλόγραμμα με αχνό περίγραμμα παριστάνουν μονάδες λειτουργιών (*functional units*), δηλαδή μια συγκεκριμένη λειτουργία που έχει αναφερθεί στο προηγούμενο κεφάλαιο και η οποία αναλύεται στο κείμενο.
2. Μικρά άσπρα ορθογώνια παραλληλόγραμμα με αχνό και διακεκομμένο περίγραμμα παριστάνουν οντότητες πληροφορίας (*information object*) (περιγράφονται στο κεφάλαιο 5).

Για την αναπαράσταση των αλληλεπιδράσεων:

1. Μεγάλα άσπρα ορθογώνια παραλληλόγραμμα με έντονο περίγραμμα παριστάνουν υπολογιστικές οντότητες (*computational objects*).
2. Μικρά μαυρισμένα ορθογώνια παραλληλόγραμμα παριστάνουν λειτουργικές διεπαφές (*operational interfaces*).
3. Μικρά άσπρα ορθογώνια παραλληλόγραμμα παριστάνουν διεπαφές ειδοποιήσεων (*alarm interfaces*), και μικροί άσπροι κύκλοι μέσα σ' αυτά παριστάνουν πηγές ροής ειδοποιήσεων ενώ μικροί μαύροι κύκλοι παριστάνουν προορισμούς ροής ειδοποιήσεων.
4. Μια λεπτή γραμμή από μια υπολογιστική οντότητα προς μία λειτουργική διεπαφή μιας άλλης υπολογιστικής οντότητας παριστάνει ότι η πρώτη υπολογιστική οντότητα είναι πελάτης στην λειτουργική διεπαφή της δεύτερης υπολογιστικής οντότητας που παριστάνει τον εξυπηρετητή (*client-server relationship*).

5. Μια παχιά γραμμή μεταξύ μιας πηγής ροής ειδοποιήσεων και ενός προορισμού ροής ειδοποιήσεων παριστάνει μια ροή ειδοποιήσεων (*alarm flow*).



Σχήμα 4-1 Γραφικοί σχηματισμοί παρουσίασης

4.2 Υπολογιστικές Οντότητες Συστήματος

Οι υπολογιστικές οντότητες από τις οποίες αποτελείται το σύστημα είναι τέσσερις:

1. Ο Συντονιστής Διαχείρισης Σφαλμάτων (**FM Coordinator**)
2. Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Σφαλμάτων (**FM Alarm Manager**)
3. Ο Εξυπηρετητής Δοκιμών Διαχείρισης Σφαλμάτων (**FM Testing Server**)
4. Η Γραφική Διεπαφή Χρήστη Διαχείρισης Σφαλμάτων (**FM GUI**)

4.2.1 Ο Συντονιστής Διαχείρισης Λαθών (FM Coordinator)

Ο Συντονιστής είναι η κύρια υπολογιστική οντότητα του συστήματος και είναι υπεύθυνος τόσο για την παροχή των διεπαφών και απαραίτητων λειτουργιών για επικοινωνία με τις άλλες περιοχές διαχείρισης (*CM και NTCM*) όσο και για τον εντοπισμό (Localization) της πρωταρχικής αιτίας σφάλματος, εφαρμόζοντας αλγόριθμους συσχέτισης/εντοπισμού.

4.2.2 Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Σφαλμάτων (FM Alarm Manager)

Ο Διαχειριστής Ειδοποιήσεων είναι η υπολογιστική εκείνη οντότητα η οποία παρέχει την δυνατότητα για αρχικοποίηση και σταμάτημα των διαδικασιών παρακολούθησης, την λειτουργία της επιτήρησης/εποπτείας ειδοποιήσεων (Alarm Surveillance) και την δημιουργία και παροχή, μέσω συγκεκριμένης διεπαφής, του χάρτη (ένα μέρος του) στατιστικών σφαλμάτων.

4.2.3 Ο Εξυπηρετητής Δοκιμών Διαχείρισης Σφαλμάτων (FM Testing Server)

Ο Εξυπηρετητής Δοκιμών είναι η υπολογιστική οντότητα που παρέχει την δυνατότητα για την δοκιμή ενός πόρου του δικτύου και την επιστροφή του αποτελέσματος σε αυτόν που την ζήτησε.

4.2.4 Η Γραφική Διεπαφή Χρήστη Διαχείρισης Σφαλμάτων (FM GUI)

Η Γραφική Διεπαφή Χρήστη είναι η υπολογιστική οντότητα η οποία χρησιμοποιείται από τον χρήστη διαχείρισης σφαλμάτων για την γραφική παρουσίαση της πληροφορίας σφάλματος στο υπό διαχείριση δίκτυο. Επίσης δίνει την δυνατότητα, μέσω της κατάλληλης διεπαφής χρήστη (User Interface) και της επικοινωνίας με τις άλλες οντότητες, για το ξεκίνημα/σταμάτημα παρακολούθησης, δοκιμής και επιδιόρθωσης ενός πόρου.

4.3 Ο Συντονιστής Διαχείρισης Σφαλμάτων

Στο Σχήμα 4-2 φαίνονται οι λειτουργίες του Συντονιστή καθώς και οι διεπαφές λειτουργιών και ειδοποιήσεων που παρέχει για να μπορούν άλλες υπολογιστικές οντότητες να τις χρησιμοποιήσουν.

4.3.1 Εσωτερική Λειτουργικότητα

Όπως φαίνεται και από το σχήμα ο Συντονιστής Διαχείρισης Λαθών είναι υπεύθυνος:

1. Για την ανάθεση των χαρακτηριστικών διαχείρισης σφαλμάτων στους πόρους (Start/Stop Managing και Initialize). Η ανάθεση γίνεται όταν καλείται η διεπαφή αρχικοποίησης και παίρνει την φυσική τοπολογία από το σύστημα Διαχείρισης Διαμόρφωσης όπου σε κάθε γραμμή αναθέτει μία κλάση επιδιόρθωσης (την μέγιστη, Α). Επίσης η ανάθεση γίνεται και όταν λαμβάνει, από το σύστημα Διαχείρισης Συνδέσεων, την πληροφορία δημιουργίας ενός καινούργιου εικονικού μονοπατιού σύνδεσης μαζί με την κλάση υπηρεσίας που έχει.

2. Για το ξεκίνημα και σταμάτημα των διαδικασιών παρακολούθησης (Start/Stop Managing) και το ξεκίνημα των δοκιμών των πόρων (Test Request). Το ξεκίνημα παρακολούθησης γίνεται ως απόρροια της απόκτησης της πληροφορίας της τοπολογίας του φυσικού επιπέδου και του επιπέδου εικονικών μονοπατιών σύνδεσης. Η παρακολούθηση κάθε πόρου γίνεται με την κλήση της ανάλογης διεπαφής του Διαχειριστή Ειδοποιήσεων στην οποία περνάει σαν παράμετρο, τον πόρο, την κλάση επιδιόρθωσης που του έχει αναθέσει και την διεπαφή ειδοποιήσεών του για να του επιστρέψει τις ειδοποιήσεις που αφορούν τον πόρο αυτό. Το σταμάτημα της παρακολούθησης γίνεται όταν έχουμε επανεκκίνηση του Συντονιστή (δηλαδή κλήση στην διεπαφή επανεκκίνησής του) ή όταν δίνεται από την Διαχείριση Συνδέσεων η πληροφορία για κάποιο εικονικό μονοπάτι που αποδεσμεύτηκε. Το σταμάτημα γίνεται και αυτό με την κλήση και πάλι της κατάλληλης διεπαφής του Διαχειριστή Ειδοποιήσεων. Το ξεκίνημα (και σταμάτημα, αφού η δοκιμή είναι μια σύγχρονη διαδικασία) των δοκιμών γίνεται ως απόρροια της διαδικασίας εντοπισμού της κύριας αιτίας ενός σφάλματος (Fault Localization). Η δοκιμή κάθε πόρου γίνεται με την κλήση της ανάλογης διεπαφής του Εξυπηρετητή Δοκιμών.

3. Για την ενημέρωση της Γραφικής Διεπαφής Χρήστη (alarm source to FM GUI) για τους πόρους που διαχειρίζεται ώστε να παρουσιάζονται γραφικά. Η ενημέρωση γίνεται τόσο κατά την αρχικοποίηση όσο και όταν λαμβάνεται η πληροφορία της δέσμευσης/αποδέσμευσης ενός καινούργιου εικονικού μονοπατιού σύνδεσης. Η ενημέρωση αυτή γίνεται μέσω της κατάλληλης διεπαφής ειδοποιήσεων της Γραφικής Διεπαφής Χρήστη.

4. Την διατήρηση και χρησιμοποίηση, κατά την διαδικασία εντοπισμού της πρωταρχικής αιτίας σφάλματος, της πληροφορίας που αφορά της τοπολογίας του δικτύου (Network Topology Information).

5. Τον εντοπισμό της πρωταρχικής αιτίας σφάλματος (Fault Localization) στο δίκτυο χρησιμοποιώντας την πληροφορία της τοπολογίας, των ειδοποιήσεων και των αποτελεσμάτων των δοκιμών. Το ξεκίνημα του εντοπισμού της πρωταρχικής αιτίας σφάλματος είναι απόρροια της λήψης ειδοποίησης σφάλματος.

6. Την επικοινωνία με τις διαδικασίες επιδιόρθωσης (To Restoration Processes) για την επιδιόρθωση της πρωταρχικής αιτίας σφάλματος (εφόσον κάτι τέτοιο δεν έχει ήδη γίνει).

7. Την αναφορά των ειδοποιήσεων που λαμβάνει καθώς και της πρωταρχικής αιτίας του σφάλματος στο σύστημα Διαχείρισης Διαμόρφωσης (Report alarms to NTCM).

8. Την ανάκτηση του χάρτη των στατιστικών από τον Διαχειριστή των ειδοποιήσεων, την ενημέρωση της πληροφορίας του χάρτη που είναι αποτέλεσμα της συσχέτισης των πόρων (*RUR*) και την παροχή του προς τους ενδιαφερόμενους (Get Network Risk Map). Η διαδικασία αυτή είναι αποτέλεσμα της κλήσης της κατάλληλης διεπαφής του.

4.3.2 Παρεχόμενες Διεπαφές

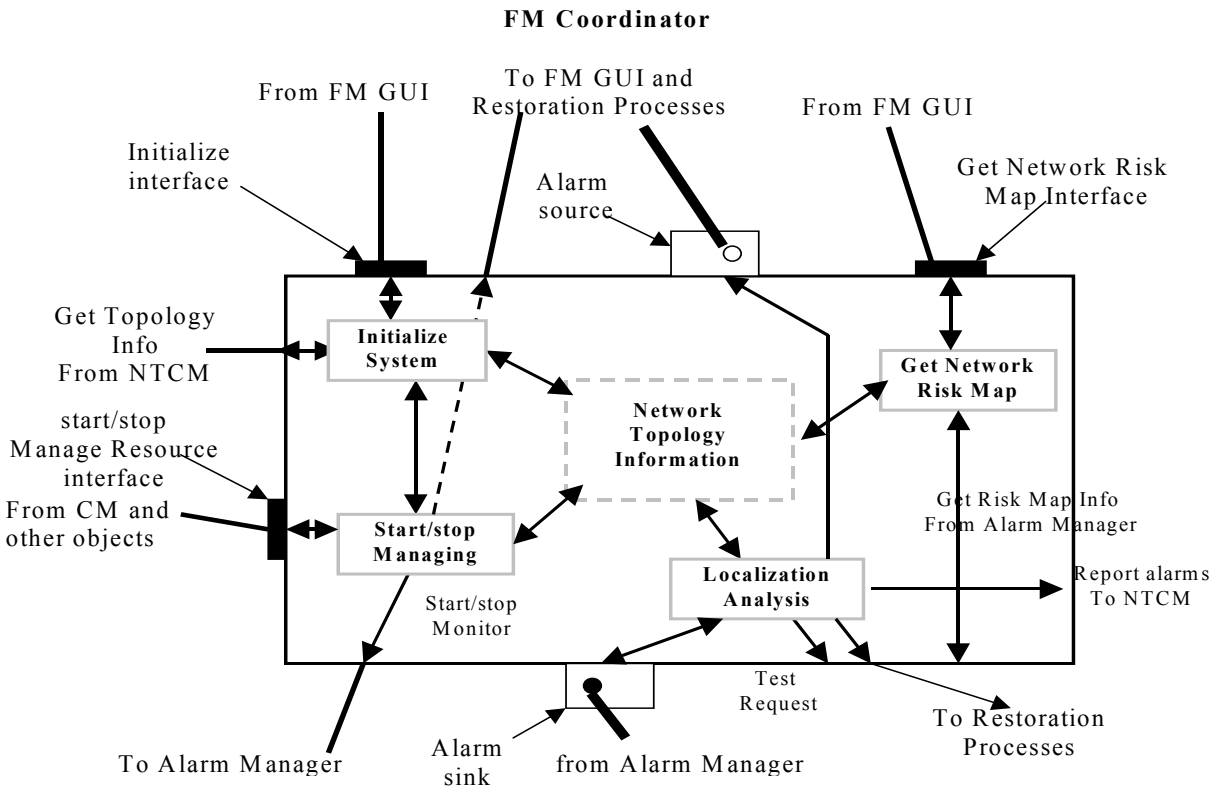
Ο Συντονιστής παρέχει (δηλαδή έχει τον ρόλο του εξυπηρετητή) τις εξής διεπαφές:

4.3.2.1 Λειτουργικές Διεπαφές

1. Διεπαφή **Αρχικοποίησης/Επανεκκίνησης** (**Initialize** Interface): Χρησιμοποιείται για να αρχικοποιήσει το σύστημα. Η κλήση της προκαλεί το σύστημα να σταματήσει την διαχείριση των πόρων, να ξαναπάρει την πληροφορία της τοπολογίας του φυσικού επιπέδου, να αρχίσει την διαχείριση των πόρων που αυτή περιλαμβάνει και να περιμένει κλήση της διεπαφής **Διαχείρισης Πόρου** από την Διαχείριση Συνδέσεων για την διαχείριση νέων εικονικών μονοπατιών σύνδεσης. Η διεπαφή αυτή καλείται από την Γραφική Διεπαφή Χρήστη.

2. Διεπαφή **Διαχείρισης Πόρου** (**startFM/stopFM Resource** Interface): Χρησιμοποιείται για να προκαλέσει κάποιος το ξεκίνημα ή το σταμάτημά της διαχείρισης ενός συγκεκριμένου πόρου του δικτύου από το σύστημα. Σαν παράμετρο δίνεται η πληροφορία της τοπολογίας του πόρου (εικονικό μονοπάτι σύνδεσης) καθώς και της κλάσης υπηρεσίας την οποία προσφέρει. Καλείται από το σύστημα Διαχείρισης Συνδέσεων ή κάποια άλλη εξωτερική υπολογιστική οντότητα.

3. Διεπαφή **Παροχής Χάρτη Στατιστικών των Πόρων** (**Network Risk Map** Interface): Χρησιμοποιείται για να πάρει κάποιος τον χάρτη των στατιστικών σφαλμάτων που έχουν συμβεί στους πόρους του δικτύου. Καλείται από την Διαχείριση Διαμόρφωσης ή άλλη εξωτερική υπολογιστική οντότητα.



Σχήμα 4-2 Ο Συντονιστής Διαχείρισης Σφαλμάτων (FM Coordinator)

4.3.2.2 Διεπαφές Ειδοποιήσεων

1. Διεπαφή **Λήψης Ειδοποιήσεων** (**Alarm Sink** Interface): Χρησιμοποιείται για την λήψη ειδοποιήσεων σφαλμάτων και επιδιορθώσεων πόρων του δικτύου. Η λήψη μιας ειδοποιήσεως σφάλματος ενδέχεται να προκαλέσει το ξεκίνημα της διαδικασίας εντοπισμού της πρωταρχικής αιτίας σφάλματος (όπως αυτή περιγράφηκε στο προηγούμενο κεφάλαιο). Καλείται από τον Διαχειριστή Ειδοποιήσεων όταν υπάρχει ειδοποίηση για πόρο του οποίου η παρακολούθηση έχει ξεκινήσει από κλήση του Συντονιστή (στην οποία κλήση του είχε δώσει ως παράμετρο την διεπαφή αυτή).

4.4 Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Λαθών

Στο Σχήμα 4-3 φαίνονται οι λειτουργίες του Διαχειριστή Ειδοποιήσεων καθώς και οι διεπαφές λειτουργιών και ειδοποιήσεων που παρέχει για να μπορούν άλλες υπολογιστικές οντότητες να τις χρησιμοποιήσουν.

4.4.1 Εσωτερική Λειτουργικότητα

Όπως φαίνεται και από το σχήμα ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Λαθών είναι υπεύθυνος:

1. Για την αρχικοποίηση και σταμάτημα της παρακολούθησης ενός πόρου στο δίκτυο (Initiate/Stop Monitor). Το ξεκίνημα γίνεται όταν κάποια υπολογιστική οντότητα καλέσει την κατάλληλη διεπαφή του. Στην διεπαφή αυτή περνάει σαν παράμετρο τον πόρο που θέλει να παρακολουθήσει, το είδος της παρακολούθησης καθώς και μια διεπαφή ειδοποιήσεων στην οποία ο Διαχειριστής Ειδοποιήσεων θα στείλει την ειδοποίηση όταν λάβει κάποια για τον πόρο αυτό (κοίτα προώθηση ειδοποιήσεων). Από την

πλευρά του ο Διαχειριστής Σφαλμάτων από την πληροφορία του πόρου εντοπίζει (ο τρόπος περιγράφεται στο κεφάλαιο 6 - Θέματα Υλοποίησης) την υπολογιστική οντότητα, στο EML επίπεδο, που είναι υπεύθυνη για την παρακολούθηση του και καλεί την διεπαφή της στην οποία περνάει ως παράμετρο τον πόρο, την κλάση επιδιόρθωσης και την διεπαφή ειδοποιήσεων του για να μπορεί να του στείλει τυχόν ειδοποιήσεις σφάλματος για τον συγκεκριμένο πόρο. Το σταμάτημα της παρακολούθησης γίνεται όταν κάποια υπολογιστική οντότητα καλέσει την κατάλληλη διεπαφή του στην οποία περνάει ως παράμετρο και την διεπαφή ειδοποιήσεων που είχε δώσει κατά το ξεκίνημα για να βεβαιωθεί ο Διαχειριστής Σφαλμάτων για την ταυτότητα του (ότι πρόκειται για την ίδια οντότητα).

2. Για την λήψη ειδοποιήσεων που αφορούν σφάλματα πόρων καθώς και επιδιορθώσεις πόρων. Η λήψη γίνεται μέσω της διεπαφής ειδοποιήσεων που διαθέτει (alarm sink).

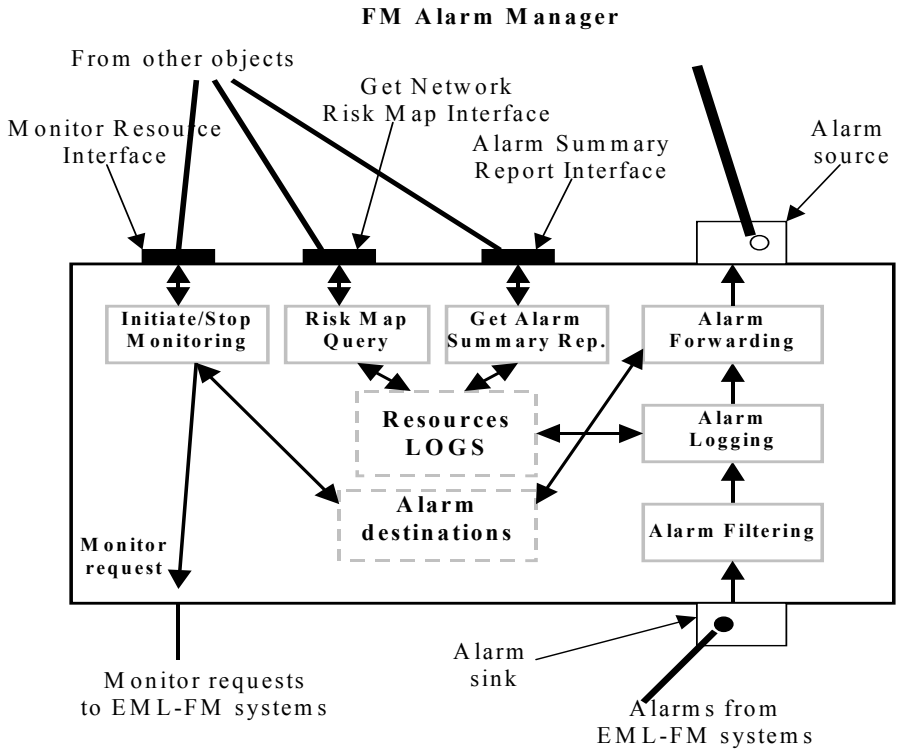
3. Για το φιλτράρισμα (Alarm Filtering) των ειδοποιήσεων ώστε να μην αποθηκεύονται ή προωθούνται περιττές ειδοποιήσεις. Το φιλτράρισμα γίνεται απορρίπτοντας τις ειδοποιήσεις σφάλματος των πόρων για τους οποίους έχει ήδη λάβει ειδοποίηση σφάλματος, δεν έχει λάβει ειδοποίηση επιδιόρθωσης και δεν έχει περάσει μεγάλο χρονικό διάστημα.

4. Για την αποθήκευση των ειδοποιήσεων (Alarm Logging), μετά την διαδικασία του φιλτραρίσματός τους, σε ειδικές οντότητες πληροφορίας (LOGS) που υπάρχουν για κάθε πόρο του δικτύου. Οι οντότητες αυτές πληροφορίας περιγράφονται στο κεφάλαιο 5.

5. Για την προώθηση των ειδοποιήσεων, μετά την διαδικασία φιλτραρίσματος και αποθήκευσης, σύμφωνα με πληροφορία που έχει αποθηκευτεί κατά την εκκίνηση της παρακολούθησης για τον προορισμό των ειδοποιήσεων (Alarm Destinations). Η πληροφορία αυτή για τον προορισμό αποκτάται σαν όρισμα στην διεπαφή εκκίνησης παρακολούθησης, περιλαμβάνει τον προορισμό και το είδος της ειδοποιήσεως και αποθηκεύεται σε ειδική οντότητα πληροφορίας για κάθε πόρο (κεφάλαιο 5).

6. Για την ενημέρωση της πληροφορίας που περιέχεται στον χάρτη στατιστικών των πόρων του δικτύου (Alarm Logging) και για την παροχή του χάρτη αυτού όταν ζητηθεί. Η ενημέρωση γίνεται αυτόματα με την λήψη κάθε ειδοποίησης και αποθηκεύεται σε ειδική οντότητα πληροφορίας (κεφάλαιο 5).

7. Για την παροχή αναφοράς ειδοποιήσεων (Get Alarm Summary Report) που έχουν ληφθεί για κάποιον πόρο όταν αυτό ζητηθεί. Η παροχή αναφοράς γίνεται με την κλήση της κατάλληλης λειτουργικής διεπαφής και προκαλεί την ανάκτηση της πληροφορίας από την οντότητα πληροφορίας που έχει αποθηκευτεί (Resources LOGS) και την μετατροπή της σε κατάλληλη μορφή (κεφάλαιο 5).



Σχήμα 4-3 Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Σφαλμάτων (FM Alarm Manager)

4.4.2 Παρεχόμενες Διεπαφές

Ο Διαχειριστής Ειδοποιήσεων παρέχει (δηλαδή έχει τον ρόλο του εξυπηρετητή) τις εξής διεπαφές:

4.4.2.1 Λειτουργικές Διεπαφές

1. Διεπαφή **Παρακολούθησης Πόρου** (**Monitor Resource** Interface): Χρησιμοποιείται για να δηλώσει κάποιος το ενδιαφέρον του για την παρακολούθηση ενός πόρου και τη λήψη ειδοποιήσεων που τον αφορούν. Η κλήση της προκαλεί το ξεκίνημα της διαδικασίας παρακολούθησης εφόσον αυτή ήδη δεν έχει γίνει καθώς και διατήρηση της πληροφορίας του προορισμού των συγκεκριμένων ειδοποιήσεων (Alarm Destinations). Καλείται από τον Συντονιστή και άλλους ενδιαφερόμενους. Δέχεται σαν παράμετρο την διεπαφή ειδοποιήσεων του ενδιαφερόμενου, τον πόρο που θα παρακολουθηθεί και την κλάση επιδιόρθωσης.

2. Διεπαφή **Ανάκλησης Παρακολούθησης Πόρου** (**Stop Monitor Resource** Interface): Χρησιμοποιείται για να δηλώσει κάποιος το ενδιαφέρον του για να σταματήσει την παρακολούθηση ενός πόρου και την λήψη ειδοποιήσεων που τον αφορούν. Η κλήση της προκαλεί το σταμάτημα της διαδικασίας παρακολούθησης εφόσον δεν υπάρχει ενδιαφέρον και από άλλους. Δέχεται σαν παράμετρο τον πόρο και την διεπαφή ειδοποιήσεων του ενδιαφερόμενου για εξακρίβωση της ταυτότητας.

3. Διεπαφή **Παροχής Αναφοράς Ειδοποιήσεων** (**Alarm Summary Report** Interface): Χρησιμοποιείται για να πάρει κάποιος μια αναφορά με όλες τις ειδοποιήσεις που έχουν ληφθεί για κάποιον πόρο μέχρι εκείνη την στιγμή. Δέχεται σαν παράμετρο τον πόρο για τον οποίο ζητείται η αναφορά.

4. Διεπαφή **Παροχής Χάρτη Στατιστικών Πόρων** (**Network Risk Map** Interface): Χρησιμοποιείται για να πάρει κάποιος τον χάρτη των στατιστικών σφαλμάτων που έχουν συμβεί στους πόρους του δικτύου. Καλείται από τον Συντονιστή.

4.4.2.2 Διεπαφές Ειδοποιήσεων

1. Διεπαφή **Λήψης Ειδοποιήσεων** (**Alarm Notification** Interface): Χρησιμοποιείται για την λήψη ειδοποιήσεων σφαλμάτων και επιδιορθώσεων. Η αποστολή των ειδοποιήσεων σφάλματος γίνεται από το EML επίπεδο και προκαλεί άμεσα το ξεκίνημα της διαδικασίας φιλτραρίσματος, ενημέρωσης στατιστικών, αποθήκευσης και προώθησης.

4.5 Εξυπηρετητής Δοκιμών Διαχείρισης Σφαλμάτων

Στο Σχήμα 4-4 φαίνονται οι λειτουργίες του Εξυπηρετητή Δοκιμών καθώς και οι διεπαφές λειτουργιών που παρέχει για να μπορούν άλλες υπολογιστικές οντότητες να τις χρησιμοποιήσουν.

4.5.1 Εσωτερική Λειτουργικότητα

Όπως φαίνεται και από το σχήμα ο Εξυπηρετητής Δοκιμών Διαχείρισης Λαθών είναι υπεύθυνος:

1. Για το ξεκίνημα της δοκιμής ενός πόρου στο δίκτυο (Initiate Testing and Get Result). Το ξεκίνημα γίνεται με την κλήση της κατάλληλης διεπαφής του και είναι μια σύγχρονη διαδικασία. Η διαδικασία αυτή περιλαμβάνει τον εντοπισμό (κεφάλαιο 6 - Θέματα Υλοποίησης), από την πληροφορία του πόρου, της υπολογιστικής οντότητας, στο EML επίπεδο, που είναι υπεύθυνη για την παρακολούθηση του πόρου και την κλήση της κατάλληλης διεπαφής της.

2. Για την αποθήκευση του αποτελέσματος της δοκιμής (Test Result Logging) σε ειδικές οντότητες πληροφορίας (Resources *LOGS*) που υπάρχουν για κάθε πόρο του δικτύου (κεφάλαιο 5).

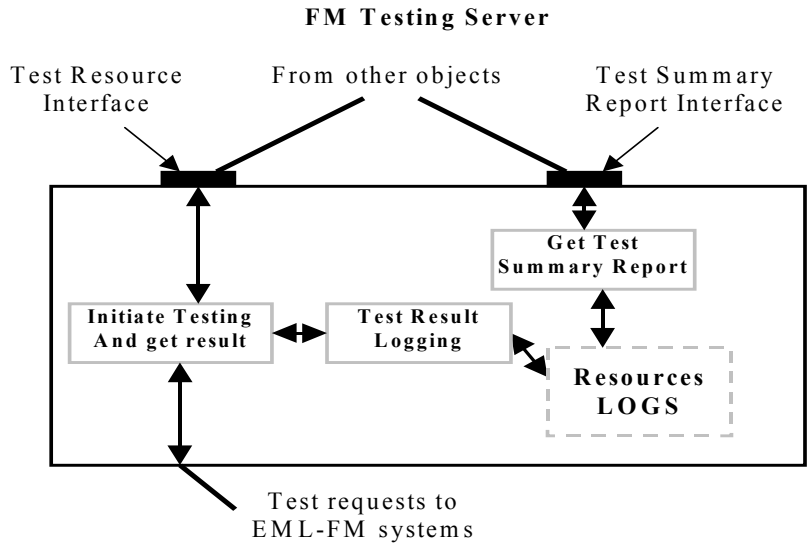
3. Για την παροχή αναφοράς αποτελεσμάτων δοκιμών (Test Summary Report) που έχουν γίνει για κάποιον πόρο όταν αυτό ζητηθεί. Η παροχή αναφοράς γίνεται με την κλήση της κατάλληλης λειτουργικής διεπαφής και προκαλεί την ανάκτηση της πληροφορίας από την οντότητα πληροφορίας που έχει αποθηκευτεί (Resources LOGS) και την μετατροπή της σε κατάλληλη μορφή (κεφάλαιο 5).

4.5.2 Παρεχόμενες Διεπαφές

Ο Εξυπηρετητής Δοκιμών παρέχει (δηλαδή έχει τον ρόλο του εξυπηρετητή) τις εξής διεπαφές:

4.5.2.1 Λειτουργικές Διεπαφές

1. Διεπαφή **Δοκιμής Πόρου** (**Test Resource** Interface): Χρησιμοποιείται για να προκαλέσει κάποιος την δοκιμή ενός συγκεκριμένου πόρου του δικτύου από το σύστημα διαχείρισης σφαλμάτων και να πάρει το αποτέλεσμά της. Καλείται από τον Συντονιστή (ή άλλη εξωτερική οντότητα) κατά την διάρκεια της διαδικασίας εντοπισμού της πρωταρχικής αιτίας του σφάλματος και δέχεται σαν παράμετρο τον πόρο και το είδος της δοκιμής.



Σχήμα 4-4 Ο Εξυπηρετητής Δοκιμών Διαχείρισης Σφαλμάτων (FM Testing Server)

2. Διεπαφή **Παροχής Αναφοράς Αποτελεσμάτων Δοκιμών Πόρου** (**Test Summary Report** Interface): Χρησιμοποιείται για να πάρει κάποιος μια αναφορά με τα αποτελέσματα όλων των δοκιμών που έχουν γίνει σε κάποιον πόρο μέχρι εκείνη την στιγμή. Καλείται από οποιοδήποτε ενδιαφερόμενο και δέχεται σαν παράμετρο τον πόρο και το είδος των δοκιμών που τον θέλει.

4.5.2.2 Διεπαφές Ειδοποιήσεων

Ο Εξυπηρετητής Δοκιμών δεν παρέχει καμιά διεπαφή ειδοποιήσεων.

4.6 Η Γραφική Διεπαφή Χρήστη Διαχείρισης Σφαλμάτων

Στο Σχήμα 4-5 φαίνονται οι λειτουργίες της Γραφικής Διεπαφής Χρήστη καθώς και η διεπαφή ειδοποιήσεων που αυτή παρέχει για να μπορούν άλλες υπολογιστικές οντότητες να της στέλνουν ειδοποιήσεις.

4.6.1 Εσωτερική Λειτουργικότητα

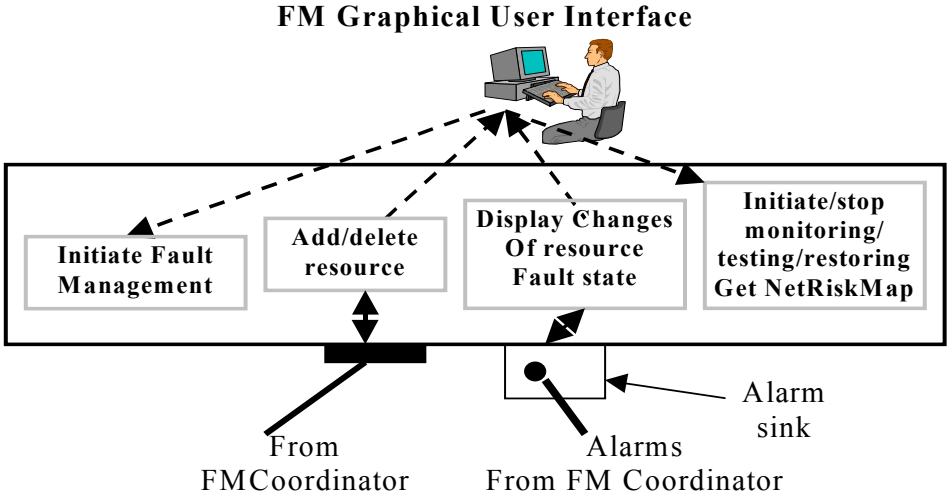
Η εσωτερική λειτουργικότητα της Γραφικής Διεπαφής Χρήστη έχει ως εξής:

1. Παρέχει την δυνατότητα στον χρήστη να (επανα)ξεκινήσει το σύστημα διαχείρισης σφαλμάτων (Initiate Fault Management). Η ενέργεια αυτή του χρήστη προκαλεί την κλήση της διεπαφής επανεκκίνησης του Συντονιστή, την ανάκτηση της πληροφορίας της τοπολογίας και την παρουσίασή της με γραφικό τρόπο.

2. Παρουσιάζει με γραφικό τρόπο τους πόρους του δικτύου και την κατάσταση σφάλματος που αυτοί βρίσκονται με τρόπο που παράλληλα δείχνει την τοπολογική τους συσχέτιση. Η ενημέρωση της πληροφορίας που παρουσιάζεται γραφικά γίνεται με την λήψη (Alarm Sink) ειδοποιήσεων σφάλματος, επιδιόρθωσης και πρωταρχικής αιτίας σφάλματος καθώς και με την κλήση της διεπαφής για ξεκίνημα/σταμάτημα (Add/Delete Resource) διαχείρισης εικονικού μονοπατιού σύνδεσης από τον Συντονιστή.

3. Παρέχει την δυνατότητα στον χρήστη για γραφική παρουσίαση του χάρτη των στατιστικών σφαλμάτων των πόρων του δικτύου (Get Network Risk Map). Η ενέργεια αυτή του χρήστη προκαλεί την κλήση της κατάλληλης διεπαφής του Συντονιστή για την ανάκτηση του χάρτη και την γραφική του παρουσίαση.

4. Τέλος παρέχει την δυνατότητα για ξεκίνημα/σταμάτημα από τον χρήστη της παρακολούθησης, δοκιμής και επιδιόρθωσης ενός πόρου (Initiate/Stop Monitoring/Testing/Restoring).



Σχήμα 4-5 Η Γραφική Διεπαφή Χρήστη (FM Graphical User Interface)

4.6.2 Παρεχόμενες Διεπαφές

Η Γραφική Διεπαφή Χρήστη παρέχει (δηλαδή έχει τον ρόλο του εξυπηρετητή) τις εξής διεπαφές:

4.6.2.1 Λειτουργικές Διεπαφές

1. Διεπαφή **Πρόσθεσης/Αφαίρεσης Πόρου** (**add/delete Resource** Interface): Χρησιμοποιείται για να ενημερώνει ο Συντονιστής την Γραφική Διεπαφή Χρήστη για το ξεκίνημα ή σταμάτημα διαχείρισης ενός εικονικού μονοπατιού σύνδεσης ώστε να ενημερώνεται η πληροφορία τοπολογίας που παρουσιάζεται γραφικά. Την καλεί ο Συντονιστής όταν η Διαχείριση Συνδέσεω καλέσει την διεπαφή **Διαχείρισης Πόρου** (**startFM/stopFM Resource** Interface) και του δώσει την πληροφορία για την δέσμευση/αποδέσμευση ενός εικονικού μονοπατιού σύνδεσης.

4.6.2.2 Διεπαφές Ειδοποιήσεων

1. Διεπαφή **Λήψης Ειδοποιήσεων** (**Alarm Notification** Interface): Χρησιμοποιείται για την λήψη ειδοποιήσεων σφαλμάτων, πρωταρχικής αιτίας σφάλματος και επιδιορθώσεων. Η αποστολή των ειδοποιήσεων γίνεται από τον Συντονιστή.

4.7 Συνολικό Σύστημα

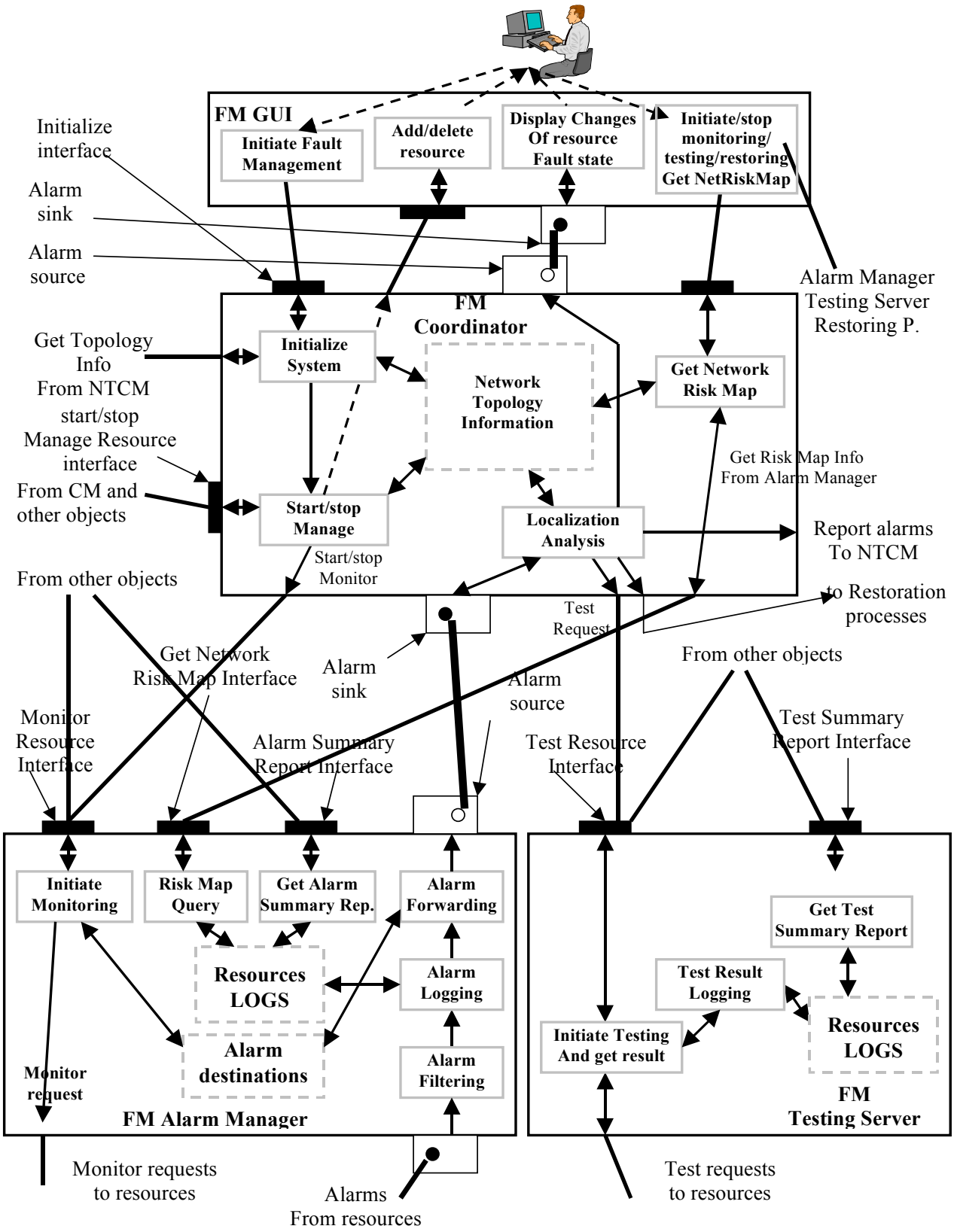
Στο Σχήμα 4-6 φαίνονται οι λειτουργίες και οι αλληλεπιδράσεις των υπολογιστικών οντοτήτων που αποτελούν το σύστημα συνολικά.

4.7.1 Περιγραφή Αλληλεπιδράσεων Υπολογιστικών Οντοτήτων

Στην παράγραφο αυτήν παραθέτονται οι αλληλεπιδράσεις των οντοτήτων, δηλαδή αναφέρεται σε ποιες υπολογιστικές οντότητες και σε ποιες ακριβώς διεπαφές είναι πελάτης κάθε υπολογιστική οντότητα (με βάση το Σχήμα 4.6). Επίσης παραθέτεται ένα παράδειγμα αλληλεπιδράσεων όπου φαίνεται η εκκίνηση του συστήματος και η διαδικασία εντοπισμού της πρωταρχικής αιτίας του σφάλματος καθώς και της αναφοράς του.

4.7.1.1 Συντονιστής Διαχείρισης

Όπως φαίνεται και από το Σχήμα 4-6 ο Συντονιστής Διαχείρισης είναι πελάτης στις εξής διεπαφές λειτουργικότητας:



Σχήμα 4-6 Συνολικό Προτεινόμενο Σύστημα Διαχείρισης Λαθών (Fault Management System)

1. Διεπαφή **Παρακολούθησης Πόρου** (**Monitor Resource** Interface) του Διαχειριστή Ειδοποιήσεων, όπου την χρησιμοποιεί για να δηλώσει ότι ενδιαφέρεται για την παρακολούθηση και λήψη ειδοποιήσεων για κάποιον πόρο.

2. Διεπαφή **Ανάκλησης Παρακολούθησης Πόρου** (**Stop Monitor Resource** Interface) του Διαχειριστή Ειδοποιήσεων, όπου την χρησιμοποιεί για να σταματήσει την παρακολούθηση και λήψη ειδοποιήσεων για κάποιον πόρο.

3. Διεπαφή **Παροχής Αναφοράς Ειδοποιήσεων** (**Alarm Summary Report** Interface) του Διαχειριστή Ειδοποιήσεων, όπου την χρησιμοποιεί όταν χρειαστεί κατά την διαδικασία εντοπισμού της κύριας αιτίας ενός σφάλματος στο δίκτυο την πληροφορία για τις ειδοποιήσεις ενός πόρου.

4. Διεπαφή **Δοκιμής Πόρου** (**Test Resource** Interface) του Εξυπηρετητή Δοκιμών, όπου το χρησιμοποιεί όταν η διαδικασία εντοπισμού της κύριας αιτίας ενός σφάλματος στο δίκτυο απαιτεί την δοκιμή ενός πόρου.

5. Διεπαφή **Παροχής Πληροφορίας Τοπολογίας και Πόρων** του συστήματος NTCM **(**ή κάποιου άλλου συστήματος), όπου το χρησιμοποιεί όταν χρειάζεται την πληροφορία της τοπολογίας φυσικού επιπέδου (κατά την αρχικοποίηση).

6. Διεπαφή **Παροχής Χάρτη Στατιστικών Σφαλμάτων** του Διαχειριστή Ειδοποιήσεων, όπου την χρησιμοποιεί όταν κάποιο άλλο σύστημα του ζητήσει (μέσω της δικής του διεπαφής) τον Χάρτη των Στατιστικών Σφαλμάτων των πόρων του δικτύου.

Ο Συντονιστής Διαχείρισης είναι πελάτης στην εξής διεπαφή ειδοποιήσεων:

1. Διεπαφή **Λήψης Ειδοποιήσεων** (**Alarm Notification** Interface) της Γραφικής Διεπαφής Χρήστη. Μέσω αυτής στέλνει τις ειδοποιήσεις για σφάλματα, για επιδιορθώσεις και για την αναφορά της πρωταρχικής αιτίας του σφάλματος.

2. Διεπαφή **Λήψης Ειδοποιήσεων** (**Alarm Notification** Interface) του συστήματος Διαχείρισης Διαμόρφωσης. Μέσω αυτής στέλνει τις ειδοποιήσεις για σφάλματα, για επιδιορθώσεις και για την αναφορά της πρωταρχικής αιτίας του σφάλματος στο σύστημα Διαχείρισης Διαμόρφωσης.

4.7.1.2 Διαχειριστής Ειδοποιήσεων

Όπως φαίνεται και από το Σχήμα 4-6 ο Διαχειριστής Ειδοποιήσεων είναι πελάτης στην εξής διεπαφή λειτουργικότητας:

1. Διεπαφή **Παρακολούθησης Πόρου** (**Monitor Resource** Interface) της διαδικασίας παρακολούθησης του πόρου στο επίπεδο EML. Την χρησιμοποιεί για να ξεκινήσει/σταματήσει η παρακολούθηση του πόρου.

Ο Διαχειριστής Ειδοποιήσεων είναι πελάτης στην εξής διεπαφή ειδοποιήσεων:

1. Διεπαφή **Λήψης Ειδοποιήσεων** (**Alarm Notification** Interface) του Συντονιστή. Μέσω αυτής του στέλνει τις ειδοποιήσεις που τον αφορούν.

4.7.1.3 Εξυπηρετητής Δοκιμών

Όπως φαίνεται και από το Σχήμα 4-6 ο Εξυπηρετητής Δοκιμών είναι πελάτης στην εξής διεπαφή λειτουργικότητας:

1. Διεπαφή **Δοκιμής Πόρου** (**Test Resource** Interface) της διαδικασίας δοκιμής του πόρου στο επίπεδο EML. Την χρησιμοποιεί για να δοκιμάσει τον πόρο.

Ο Εξυπηρετητής Δοκιμών δεν είναι πελάτης σε καμία διεπαφή ειδοποιήσεων.

4.7.1.4 Γραφική Διεπαφή Χρήστη

Όπως φαίνεται και από το Σχήμα 4-6 η Γραφική Διεπαφή Χρήστη είναι πελάτης στις εξής διεπαφές λειτουργικότητας:

1. Διεπαφή **Επανεκκίνησης Συστήματος,** του Συντονιστή, όπου την χρησιμοποιεί για να επανεκκινήσει το σύστημα, να πάρει και να δείξει με γραφικό τρόπο τους πόρους του δικτύου και την τοπολογία στην οποία βρίσκονται.

2. Διεπαφή **Παροχής Χάρτη Στατιστικών Σφαλμάτων** του Συντονιστή, όπου την χρησιμοποιεί για να πάρει και να δείξει με γραφικό τρόπο τον χάρτη στατιστικών σφαλμάτων πόρων.

Η Γραφική Διεπαφή Χρήστη δεν είναι πελάτης σε καμία διεπαφή ειδοποιήσεων.

4.7.1.5 Παράδειγμα Λειτουργίας Συστήματος

Στο Σχήμα 4-7 φαίνεται το διάγραμμα ενός παραδείγματος των αλληλεπιδράσεων των οντοτήτων (τόσο μεταξύ τους όσο και με εξωτερικές οντότητες). Το παράδειγμα δείχνει την αρχικοποίηση του συστήματος, την ανίχνευση ενός σφάλματος μετά από κάποιο χρόνο, την διαδικασία του εντοπισμού της πρωταρχικής αιτίας και την αναφοράς της.

Τα βήματα που φαίνονται στο σχήμα είναι τα εξής:

1. Η Γραφική Διεπαφή Χρήστη καλεί την διεπαφή Επανεκκίνησης/Αρχικοποίησης του Συντονιστή.

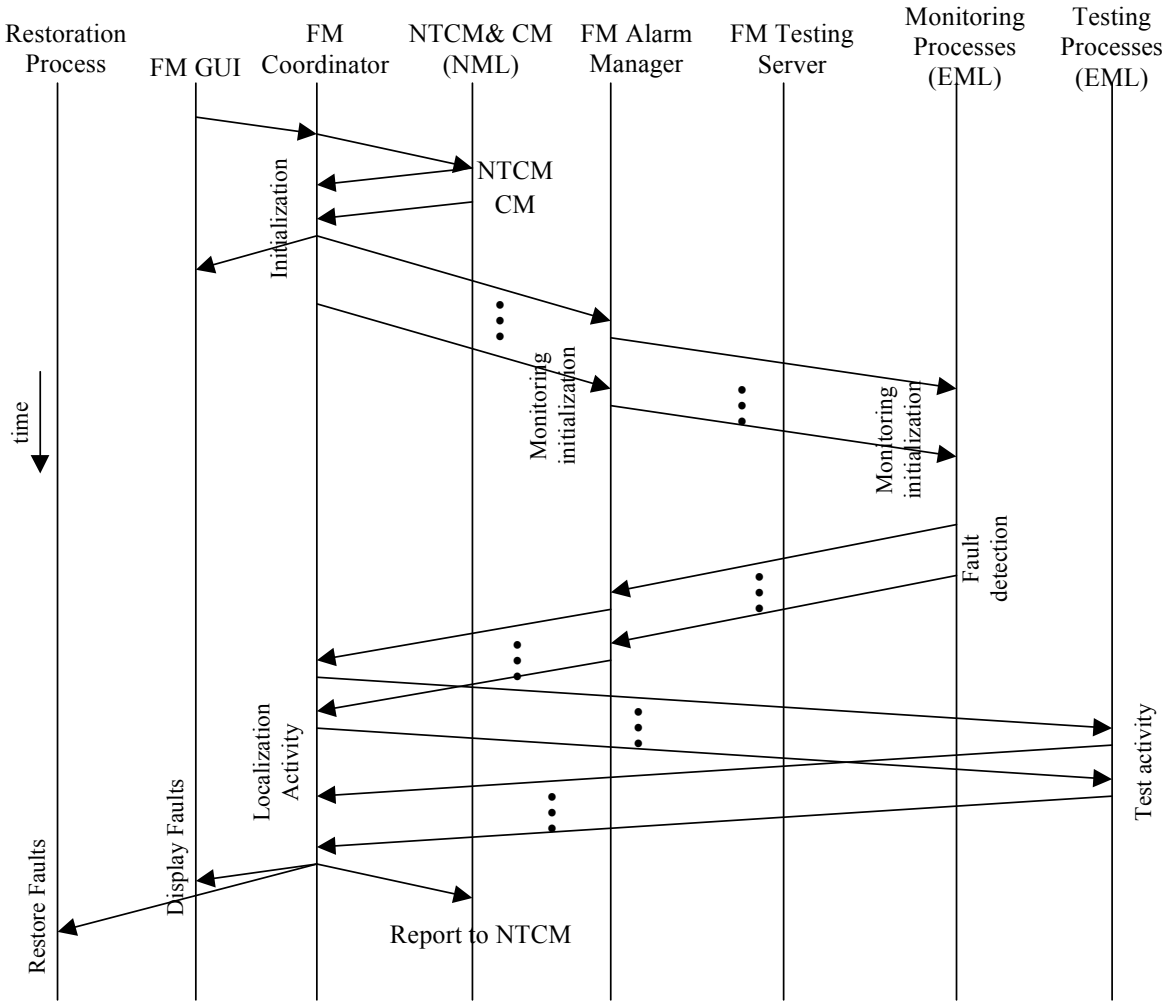
2. Ως αποτέλεσμα ο Συντονιστής καλεί την διεπαφή απόκτησης της πληροφορίας της τοπολογίας των πόρων του δικτύου (ενός συστήματος που ανήκει στο NTCM). Αφού πάρει την πληροφορία, ξεκινάει με βάση αυτήν την κλήση της διεπαφής παρακολούθησης πόρων του Διαχειριστή Ειδοποιήσεων και παράλληλα δίνει την τοπολογία στην Γραφική Διεπαφή Χρήστη.

3. Επίσης δέχεται από την Διαχείριση Συνδέσεων (μέσω της κατάλληλης διεπαφής) την πληροφορία για την δημιουργία των εικονικών μονοπατιών σύνδεσης και αφού τους αναθέσει τα χαρακτηριστικά διαχείρισης, αρχίζει την παρακολούθηση τους (με κλήση της διεπαφής παρακολούθησης πόρων του Διαχειριστή Ειδοποιήσεων).

4. Σε κάθε κλήση της διεπαφής παρακολούθησης πόρων ο Διαχειριστής Ειδοποιήσεων αποθηκεύει την διεπαφή του ενδιαφερόμενου (την διεπαφή Ειδοποιήσεων του Συντονιστή) εντοπίζει (από την πληροφορία του πόρου) την οντότητα παρακολούθησης στο EML επίπεδο και καλεί την κατάλληλη διεπαφή της για να αρχίσει η παρακολούθηση.

5. Όταν κάποιο σφάλμα εντοπιστεί από μια διαδικασία παρακολούθησης στο επίπεδο Διαχείρισης Στοιχείων του δικτύου αυτή αναφέρεται στον Διαχειριστή Ειδοποιήσεων μέσω της διεπαφής

ειδοποιήσεων του. Παράλληλα ανιχνεύονται και ότι άλλα σφάλματα συμβούν και αναφέρονται και αυτά με την σειρά τους στον Διαχειριστή Ειδοποιήσεων. Για κάθε ειδοποίηση που λαμβάνει ο Διαχειριστής Ειδοποιήσεων, την φιλτράρει, την αποθηκεύει, ενημερώνει την πληροφορία του χάρτη στατιστικών σφαλμάτων και την προωθεί στον ενδιαφερόμενο (Συντονιστή).



Σχήμα 4-7 Παράδειγμα αλληλεπιδράσεων κατά την λειτουργία του συστήματος

6. Μόλις ο Συντονιστής λάβει κάποια ειδοποίηση, εξετάζει την πληροφορία της τοπολογίας για να βρει τις αλληλεξαρτήσεις των πόρων. Για κάθε πόρο στον οποίο θα μπορούσε να οφείλεται το σφάλμα, και εφόσον δεν έχει ληφθεί ειδοποίηση γιαυτόν, αρχίζει μία διαδικασία δοκιμής. Μέχρι να τελειώσουν οι διαδικασίες δοκιμής συνεχίζει να δέχεται τις υπόλοιπες ειδοποιήσεις από τον διαχειριστή ειδοποιήσεων. Αν κατά την διάρκεια των δοκιμών μπορεί από τις ειδοποιήσεις που έλαβε να συμπεράνει την πρωταρχική αιτία, την αναφέρει στην Γραφική Διεπαφή Χρήστη, στην υπολογιστική οντότητα η οποία είναι υπεύθυνη για την επιδιόρθωση των πόρωνκαι στην Διαχείριση Διαμόρφωσης (NTCM). Αν δεν μπορεί περιμένει το τέλος όλων των δοκιμών (οι οποίες γίνονται παράλληλα) για να έχει όλα τα δεδομένα και με βάση αυτά εντοπίζει την πρωταρχική αιτία του σφάλματος.

5 Προδιαγραφή Μοντέλου Πληροφορίας Συστήματος

Στο κεφάλαιο αυτό παρουσιάζεται το μοντέλο πληροφορίας που χρησιμοποιεί το σύστημα. Το μοντέλο πληροφορίας προδιαγράφει τις οντότητες (οι οποίες ονομάζονται οντότητες πληροφορίας- information objects) τις οποίες χρησιμοποιούν οι υπολογιστικές οντότητες, που παρουσιάστηκαν στο προηγούμενο κεφάλαιο, για να λειτουργήσουν.

Το προτεινόμενο μοντέλο πληροφορίας είναι βασισμένο στο Μοντέλο Πληροφορίας Πόρων Δικτύου [NRIM] (*Network Resource Information Model - NRIM*) της TINA καθώς και σε πρότυπα (*ITU-T Xseries* [X.722], [X.733], [X.735], [X.738], [X.739] *ITU-T M3100* [M3100] κλπ) της Διεθνής Ένωσης Τηλεπικοινωνιών τα οποία προδιαγράφουν οντότητες πληροφορίας σχετικές με διαχείριση λαθών.

Η πληροφορία που ορίζεται είναι σχετική με τις τρεις από τις πέντε λειτουργίες του τμήματος Διαχείρισης Σφαλμάτων της TINA-*NRA* που καλύπτει το προτεινόμενο σύστημα διαχείρισης (Επιτήρηση Ειδοποιήσεων, Δοκιμή Πόρων και Εντοπισμό Λαθών), την Επιδιόρθωση Σφαλμάτων καθώς και τις άλλες λειτουργίες που περιγράφονται στο κεφάλαιο 3.

Η παρουσίαση του μοντέλου πληροφορίας του προτεινόμενου συστήματος χωρίζεται σε δύο μέρη. Πρώτα γίνεται η παρουσίαση της πληροφορίας που απαιτείται για να μπορεί να επικοινωνίσει ένα άλλο σύστημα με το σύστημά μας και να ανταλλάξουν πληροφορία. Αυτή η πληροφορία γίνεται σε Γλώσσα Ορισμού Διεπαφών [IDL] και περιλαμβάνει τον ορισμό των διεπαφών που προσφέρει η κάθε υπολογιστική οντότητα που περιγράφτηκε στο προηγούμενο κεφάλαιο, τον ορισμό των ειδοποιήσεων που ανταλλάσσονται, και τον ορισμό άλλων οντοτήτων πληροφορίας που ανταλλάσσονται (Χάρτης Τοπολογίας, Χάρτης Στατιστικών, Αποτελέσματα Δοκιμής, Αναφορά Συνόλου Ειδοποιήσεων Πόρου κλπ). Έπειτα ακολουθεί ο ορισμός της πληροφορίας που κρατείται εσωτερικά στο σύστημά μας και που περιλαμβάνει την πληροφορία των πόρων που φυλάγεται, οι οντότητες στις οποίες φυλλάγεται και ο τρόπος οργάνωσης τους.

5.1 Παρουσίαση Εξωτερικής Πληροφορίας και Διεπαφών

Σε αυτήν την υποενότητα παρουσιάζεται το μοντέλο της εξωτερικής πληροφορίας, δηλαδή ορίζεται η πληροφορία την οποία πρέπει να γνωρίζουν τα υπόλοιπα συστήματα για να μπορούν να επικοινωνήσουν με το προτεινόμενο σύστημα διαχείρισης και να χρησιμοποιήσουν την λειτουργικότητά του. Στην αρχή παρουσιάζεται το μοντέλο της πληροφορίας σε Τεχνική Μοντελοποίησης Οντοτήτων (Object Modeling Technique OMT) [OMT], και στην συνέχεια ορίζεται κάθε οντότητα πληροφορίας σε IDL για να μπορούν τα υπόλοιπα συστήματα να την χρησιμοποιήσουν χωρίς αλλαγές και η επικοινωνία να είναι άμεση και χωρίς προβλήματα συμβατότητας.

5.1.1 OMT Διάγραμμα των Οντοτήτων Εξωτερικής Πληροφορίας

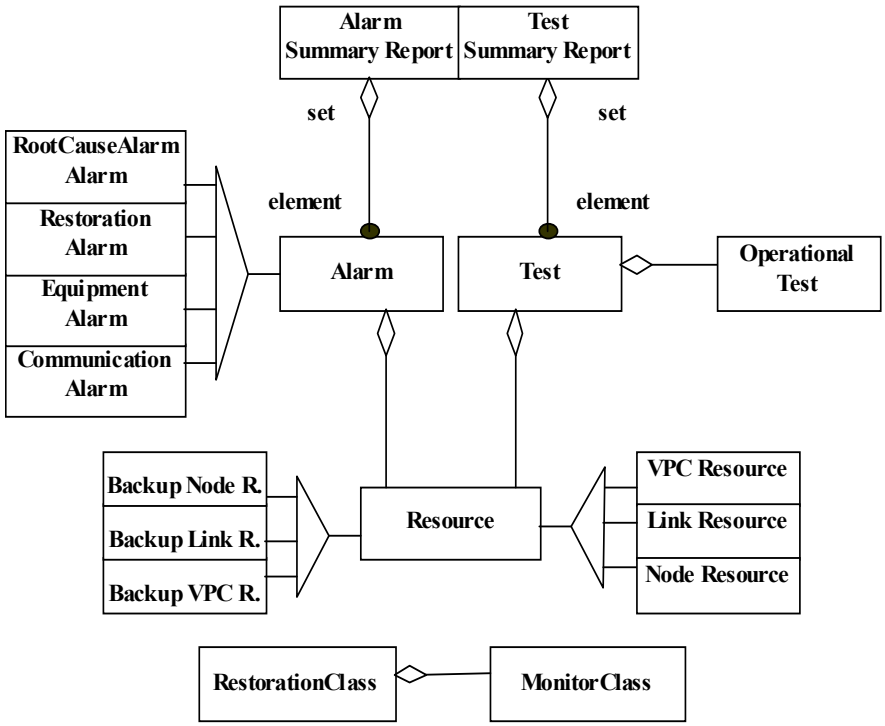
Στο Σχήμα 5-1 βλέπουμε το OMT διάγραμμα των εξής οντοτήτων εξωτερικής πληροφορίας:

1. Πόρος (**Resource**), καθορίζει ένα πόρο του υπό διαχείριση δικτύου και είναι κομμάτι (part-of) πολλών άλλων οντοτήτων (Ειδοποίησης, Δοκιμής κλπ). Περιέχει τον τύπο του πόρου, τον πόρο, και τον πόρο που θα χρησιμοποιηθεί στην θέση του (αν υπάρχει) αν συμβεί σφάλμα σε αυτόν.
2. Πόρος Εικονικό Μονοπάτι Σύνδεσης (**VPC Resource**), καθορίζει έναν πόρο τύπου Εικονικό Μονοπάτι Σύνδεσης και είναι μέρος (part - of) της οντότητας Πόρος ανάλογα με το πεδίο (attribute) **type**.
3. Πόρος Γραμμή (**Link Resource**), καθορίζει έναν πόρο τύπου φυσικής γραμμής του υπό διαχείριση δικτύου και είναι μέρος (part - of) της οντότητας Πόρος ανάλογα με το πεδίο (attribute) **type**.
4. Πόρος Κόμβος (**Node Resource**), καθορίζει έναν πόρο τύπου κόμβου του υπό διαχείριση δικτύου και είναι μέρος (part - of) της οντότητας Πόρος ανάλογα με το πεδίο (attribute) **type**.

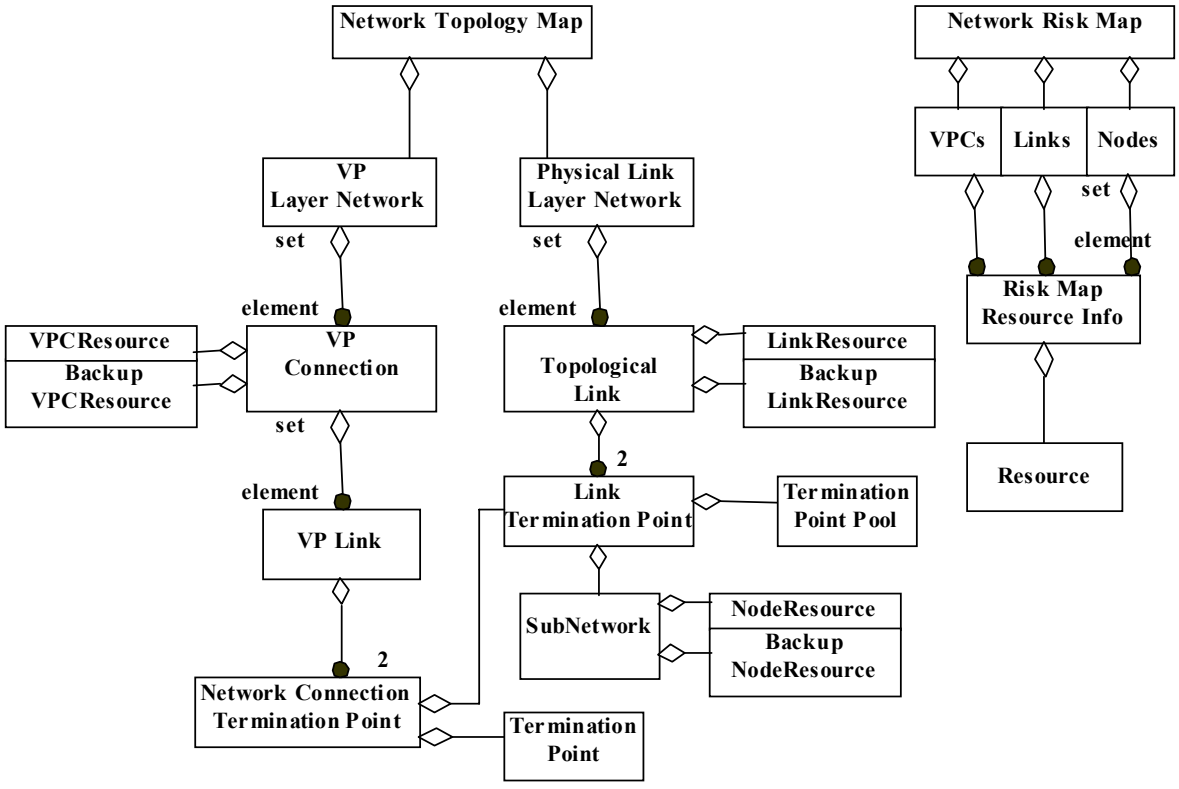
5. Κλάση Παρακολούθησης (**Monitor Class**), καθορίζει τα χαρακτηριστικά (προτεραιότητα) παρακολούθησης.
6. Κλάση Επιδιόρθωσης (**Restoration Class**), καθορίζει τα χαρακτηριστικά (προτεραιότητα) επιδιόρθωσης.
7. Ειδοποίηση (**Alarm**), καθορίζει μια ειδοποίηση που λαμβάνουμε/στέλνουμε και είναι μέρος (part - of) της οντότητας Αναφορά Συνόλου Ειδοποιήσεων (η οντότητα Αναφορά Συνόλου Ειδοποιήσεων είναι ένα σύνολο από οντότητες Ειδοποίηση).
8. Ειδοποίηση Επικοινωνίας (**Communication Alarm**), καθορίζει το περιεχόμενο μιας ειδοποίησης επικοινωνίας και είναι μέρος (part - of) της οντότητας Ειδοποίηση ανάλογα με το πεδίο (attribute) **type**.
9. Ειδοποίηση Εξοπλισμού (**Equipment Alarm**), καθορίζει το περιεχόμενο μιας ειδοποίησης εξοπλισμού και είναι μέρος (part - of) της οντότητας Ειδοποίηση ανάλογα με το πεδίο (attribute) **type**.
10. Δοκιμή (**Test**), καθορίζει μία δοκιμή που γίνεται σε κάποιον πόρο στο υπό διαχείριση δίκτυο και είναι μέρος (part - of) της οντότητας Αναφορά Συνόλου Δοκιμών (η Αναφορά Συνόλου Δοκιμών είναι ένα σύνολο από οντότητες Δοκιμή).
11. Δοκιμή Λειτουργίας (**Operational Test**), καθορίζει το περιεχόμενο (και αποτέλεσμα) μιας δοκιμής λειτουργίας σε έναν πόρο και είναι μέρος (part - of) της οντότητας Δοκιμή ανάλογα με το πεδίο (attribute) **type**.
12. Αναφορά Συνόλου Δοκιμών (**Test Summary Report**), είναι ένα σύνολο από οντότητες Δοκιμή ενός συγκεκριμένου πόρου.
13. Αναφορά Συνόλου Ειδοποιήσεων (**Alarm Summary Report**), είναι ένα σύνολο από οντότητες Ειδοποίηση ενός συγκεκριμένου πόρου.

Στο Σχήμα 5-2 βλέπουμε το OMT διάγραμμα των εξής οντοτήτων εξωτερικής πληροφορίας:

1. Στατιστικά Πόρου (**Risk Map Resource Info**), είναι μέρος (part - of) της οντότητας των οντοτήτων **Links**, **Nodes**, **VPCs** που είναι μέρος του Χάρτη Στατιστικώ (συγκεκριμένα κάθε μια τους είναι ένα σύνολο οντοτήτων Στατιστικά Πόρου).
2. Χάρτης Στατιστικών (**Network Risk Map**), είναι σύνολο οντοτήτων Στατιστικά Πόρου.
3. Σημείο Τερματισμού Γραμμής (**Link Termination Point**), είναι το ένα από τα δύο σημεία που τερματίζει μια γραμμή (κόμβος που τερματίζει και πόρτα του κόμβου που τερματίζει).
4. Γραμμή Τοπολογίας (**Topological Link**) είναι ένα ζεύγος από Σημεία Τερματισμού Γραμμής.
5. Σημείο Τερματισμού Σύνδεσης Δικτύου (**Network Connection Termination Point**) είναι το σημείο τερματισμού μιας σύνδεσης στο δίκτυο (Σημείο Τερματισμού Γραμμής και ο προσδιοριστής του σημείου τερματισμού (εικονικού μονοπατιού - vpi)).

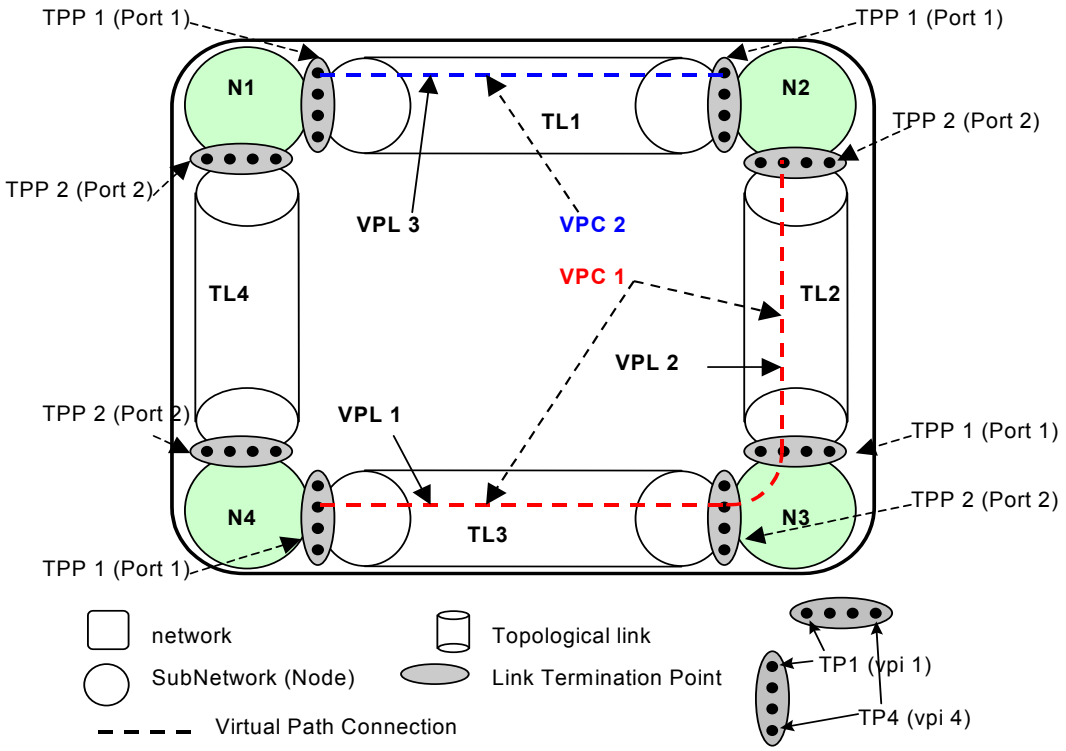


Σχήμα 5-1 OMT Διάγραμμα υποσυνόλου των οντοτήτων εξωτερικής πληροφορίας



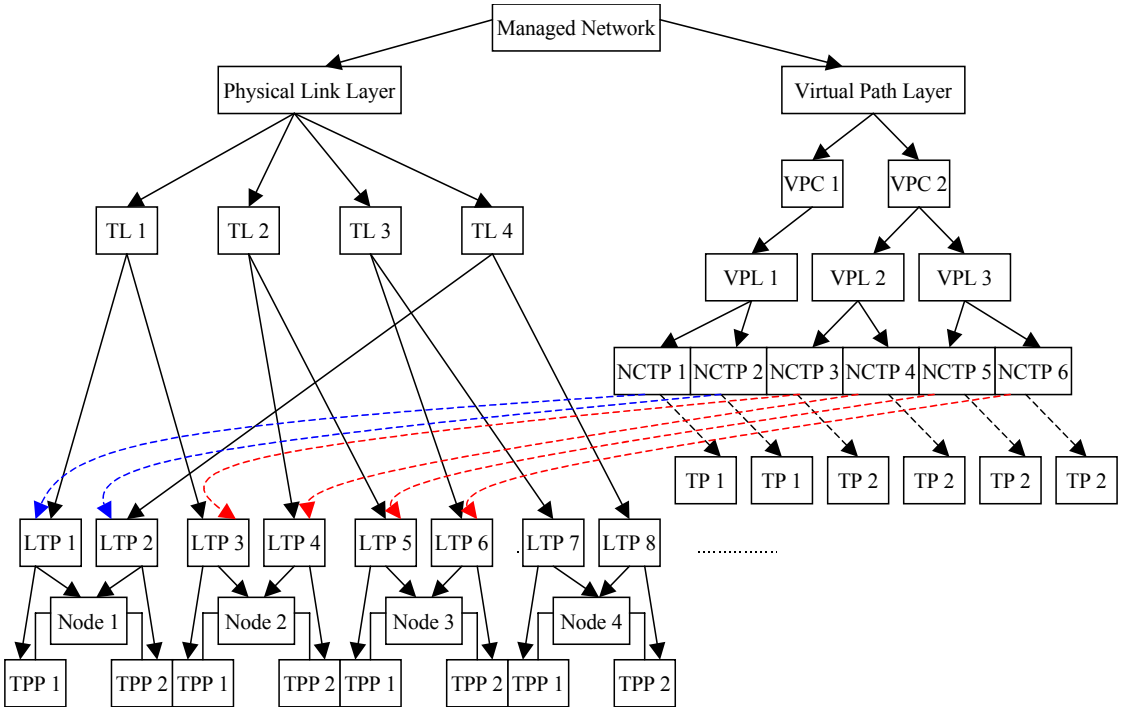
Σχήμα 5-2 OMT Διάγραμμα υποσυνόλου των οντοτήτων εξωτερικής πληροφορίας (Χάρτης Στατιστικών και Χάρτης Τοπολογίας)

6. Γραμμή Εικονικού Μονοπατιού (**Virtual Path Link**) είναι ένα ζεύγος Σημείων Τερματισμού Σύνδεσης Δικτύου (πάνω στην ίδια γραμμή).
7. Σύνδεση Εικονικού Μονοπατιού (**Virtual Path Connection**) είναι ένα σύνολο από Συνδέσεις Γραμμής Εικονικού Μονοπατιού.
8. Επίπεδο Φυσικών Γραμμών Δικτύου (**Physical Link Layer Network**) είναι το σύνολο από Συνδέσεις Φυσικής Γραμμής που υπάρχουν στο υπό διαχείριση δίκτυο.
9. Επίπεδο Συνδέσεων Εικονικού Μονοπατιού Δικτύου (**Virtual Path Layer Network**) είναι το σύνολο Συνδέσεων Εικονικών Μονοπατιών που υπάρχουν στο υπό διαχείριση δίκτυο.
10. Χάρτης Τοπολογίας Δικτύου (**Network Topology Map**) είναι το Επίπεδο Φυσικών Γραμμών Δικτύου και το Επίπεδο Συνδέσεων Εικονικού Μονοπατιού Δικτύου του υπό διαχείριση δικτύου.



Σχήμα 5-3 Παράδειγμα διαχειριζόμενου δικτύου

Για την καλύτερη κατανόηση του Χάρτη Τοπολογίας Δικτύου δίνεται στο Σχήμα 5-3 ένα παράδειγμα υπό διαχείριση δικτύου και στο Σχήμα 5-4 το πως είναι ο χάρτης. Το παράδειγμα του δικτύου μας διαθέτει τέσσερις κόμβους (*SubNetworks*) (*N1, N2, N3, N4*), τέσσερις γραμμές (*Topological Links*) που τους συνδέουν (*L1, L2, L3, L4*) και δύο Εικονικά Μονοπάτια Σύνδεσης, το *VPC1* το οποίο χρησιμοποιεί δύο VPLs (*VPL1* και *VPL2*) και το *VPC2* το οποίο χρησιμοποιεί ένα VPL (*VPL1*). Στο Σχήμα 5-4 φαίνεται η μορφή του Χάρτη της Τοπολογίας.



Σχήμα 5-4 Χάρτης Τοπολογίας του δικτύου του Σχήματος 5.3

5.1.2 Ορισμός των Οντοτήτων Εξωτερικής Πληροφορίας

Πρώτα ορίζονται, σε Γλώσσα Ορισμού Διεπαφών, κάποιες σταθερές που χρησιμοποιούνται για τον καθορισμό τύπων σε οντότητες πληροφορίας και στην συνέχεια ορίζονται οι οντότητες πληροφορίας:

- Πόρος (**Resource**)
- Πόρος Εικονικού Μονοπατιού Σύνδεσης (**VPC Resource**)
- Πόρος Γραμμή (**Link Resource**)
- Πόρος Κόμβος (**Node Resource**)
- Κλάση Παρακολούθησης (**Monitor Class**)
- Κλάση Επιδιόρθωσης (**Restoration Class**)
- Ειδοποίηση (**Alarm**)
- Ειδοποίηση Επικοινωνίας (**Communication Alarm**)
- Ειδοποίηση Εξοπλισμού (**Equipment Alarm**)
- Δοκιμή (**Test**)
- Δοκιμή Λειτουργίας (**Operational Test**)
- Αναφορά Συνόλου Ειδοποιήσεων (**Alarm Summary Report**)
- Αναφορά Συνόλου Δοκιμών (**Test Summary Report**)
- Στατιστικά Πόρου και Χάρτης Στατιστικών Πόρων (**Network Risk Map**)
- Χάρτης Τοπολογίας Δικτύου (**Network Topology Map**)

Τέλος ορίζονται, και πάλι σε Γλώσσα Ορισμού Διεπαφών, οι διεπαφές των τεσσάρων υπολογιστικών οντοτήτων οι οποίες αποτελούν το προτεινόμενο σύστημα:

- **Διεπαφή Ανταλλαγής Ειδοποιήσεων**, χρησιμοποιείται από κάθε υπολογιστική οντότητα/σύστημα που επιθυμεί να παίρνει ειδοποιήσεις. Αυτή η διεπαφή δίνεται και σαν

όρισμα στον Διαχειριστή Ειδοποιήσεων Διαχείρισης Λαθών στην μέθοδο της παρακολούθησης για να γνωρίζει που θα στέλνει τις ειδοποιήσεις όταν έρχονται.

- **Διεπαφή Συντονιστή Διαχείρισης Λαθών**, παρέχει μεθόδους για να μπορέσει μία άλλη οντότητα να πάρει τον χάρτη των στατιστικών, να μπορέσει η οντότητα Γραφική Διεπαφή Χρήστη να τον αρχικοποιήσει και για να μπορέσει μια οντότητα να εγγραφεί για να παίρνει αποτελέσματα εντοπισμού πρωταρχικής αιτίας σφάλματος.
- **Διεπαφή Διαχειριστή Ειδοποιήσεων Διαχείρισης Λαθών**, παρέχει μεθόδους για το ξεκίνημα/σταμάτημα της παρακολούθησης ενός πόρου για κάποιου είδους ειδοποίηση, για την παροχή Αναφοράς Συνόλου Ειδοποιήσεων για κάποιον πόρο και μέθοδο για την παροχή του Χάρτη Στατιστικών των Πόρων του Δικτύου.
- **Διεπαφή Εξυπηρετητή Δοκιμών Διαχείρισης Λαθών**, παρέχει μέθοδο για να γίνει δοκιμή σε κάποιον πόρο και μέθοδο για την παροχή Αναφοράς συνόλου Δοκιμών για κάποιον πόρο.
- **Διεπαφή Γραφικής Διεπαφής Χρήστη Διαχείρισης Λαθών**, παρέχει μέθοδο για να ενημερώνεται από τον Συντονιστή για νέα εικονικά μονοπάτια σύνδεσης που δημιουργούνται ή για παλιά που σβήνονται.

5.1.2.1 Σταθερές

```
// IDL
// File:    COMMONS.idl
//

#ifndef COMMONS_idl
#define COMMONS_idl

//definitions of alarm types

const string FM_COMMUNICATION_ALARM = "CommunicationAlarm";
const string FM_QOS_ALARM = "QoSAlarm";
const string FM_EQUIPMENT_ALARM = "EquipmentAlarm";
const string FM_RESTORATION_ALARM = "RestorationAlarm";
const string FM_ROOT_CAUSE_ALARM = "RootCauseAlarm";

// definitions of test types

const string FM_OPERATIONAL_TEST = "OperationalTest";
const string FM_QOS_TEST = "QoSTest";


// definitions of test results

const string FM_TEST_RESULT_OK = "OK";
const string FM_RESTORATION_RESULT_NOTOK = "NOTOK";

// definitions of resource types

const string FM_NODE_RESOURCE = "NODE";
const string FM_LINK_RESOURCE = "LINK";
const string FM_VPC_RESOURCE = "VPC";
const string FM_VCC_RESOURCE = "VCC";

// definitions of resource states

const string FM_OPERATIONAL_STATE = "Operational";
const string FM_NOT_OPERATIONAL_STATE = "Not Operational";

// definitions of event types
```

```
const string FM_ALARM_EVENT = "Alarm";
const string FM_TEST_EVENT = "Test";

// definitions of Monitor Classes

const string FM_M_CLASS_A = "A";
const string FM_M_CLASS_B = "B";
const string FM_M_CLASS_C = "C";

// definitions of Restoration Classes

const string FM_R_CLASS_A = "A";
const string FM_R_CLASS_B = "B";
const string FM_R_CLASS_C = "C";


#endif
```

5.1.2.2 Πόρος (Resource)

```
// IDL
// File:    Resource.idl
//

#ifndef Resource_idl
#define Resource_idl

struct Resource {
        string type; // "VPC" or "LINK" or "VCC" or "NODE"
        any rsc;      // VPCResource or LinkResource or VCCResource or NodeResource
        any brsc; // The backup Resource of the Resource (if any
                // VPCResource or LinkResource or VCCResource or NodeResource (as rsc)
};

#endif
```

5.1.2.3 Πόρος Εικονικού Μονοπατιού Σύνδεσης (VPC Resource)

```
// IDL
// File:    VPCResource.idl
//

#ifndef VPCResource_idl
#define VPCResource_idl

struct VPCResource {
        string vpcID; // a string uniquely identifying the resource
        string nodeA; // the first node of the link
        string nodeB; // the second node of the link
        short portA; // the port of the first node of the link
        short portB; // the port of the second node of the link
        long vpiA; // the vpi of the port of the first node of the link
        long vpiB; // the vpi of the port of the second node of the link
};

#endif
```

5.1.2.4 Πόρος Γραμμή (Link Resource)

```
// IDL
// File:    LinkResource.idl
//

#ifndef LinkResource_idl
#define LinkResource_idl

struct LinkResource {
        string linkID; // a string uniquely identifying the resource
        string nodeA; // the first node of the link
        string nodeB; // the second node of the link
        short portA; // the port of the first node of the link
        short portB; // the port of the second node of the link
};

#endif
```

5.1.2.5 Πόρος Κόμβος (Node Resource)

```
// IDL
// File:    NodeResource.idl
//

#ifndef NodeResource_idl
#define NodeResource_idl

struct NodeResource {
        string nodeID; // a string uniquely identifying the resource
        string atmaddr; // the atm address of the resource
        string ipaddr; // the ip address of the resource
        short ports; // the number of ports of the resource
};

#endif
```

5.1.2.6 Κλάση Παρακολούθησης (Monitor Class)

```
// IDL
//  File:    MonitorClass.idl
//

#ifndef MonitorClass_idl
#define MonitorClass_idl


struct MonitorClass {

        string id;  // the id of the class "A" or "B" or "C" ...
        long value; // the maximum time in milliseconds that can be pass
                // before the fault is detected by the monitoring process
};
```

```
#endif
```

5.1.2.7 Κλάση Επιδιόρθωσης (Restoration Class)

```
// IDL
// File:    RestorationClass.idl
//

#ifndef RestorationClass_idl
#define RestorationClass_idl

#include "MonitorClass.idl"

struct RestorationClass {

        string id;  // the id of the class "A" or "B" or "C" ...
        long value; // the maximum time in milliseconds that can be pass
                // after the fault and before the fault is restored
        MonitorClass mc; // the MonitorClass which corresponds to this RestorationClass
};

#endif
```

5.1.2.8 Ειδοποίηση (Alarm)

```
// IDL
// File:    Alarm.idl
//

#ifndef Alarm_idl
#define Alarm_idl

#include "Resource.idl"

struct Alarm {
        Resource  rsc;
        string type; // "Communication" or "QoS" or "Equipment" ...
        any alarm; // CommunicationAlarm or QoSAlarm, or EquipmentAlarm ...
};

#endif
```

5.1.2.9 Ειδοποίηση Επικοινωνίας (Communication Alarm)

```
// IDL
// File:    CommunicationAlarm.idl
//
#ifndef CommunicationAlarm_idl
#define CommunicationAlarm_idl

struct CommunicationAlarm {
        string probablecause;
        string perceivedseverity;
        long emittedtime;
};
#endif
```

5.1.2.10 Ειδοποίηση Εξοπλισμού (Equipment Alarm)

```
// IDL
// File:    EquipmentAlarm.idl
//
#ifndef EquipmentAlarm_idl
#define EquipmentAlarm_idl

struct EquipmentAlarm {
        string probablecause;
        string perceivedseverity;
        long emittedtime;
};
#endif
```

5.1.2.11 Ειδοποίηση Επιδιόρθωσης (Equipment Alarm)

```
// IDL
//  File:    RestorationAlarm.idl
//

#ifndef RestorationAlarm_idl
#define RestorationAlarm_idl

#include "Resource.idl"

struct RestorationAlarm {
        string result;
        Resource brsc; //The backup resource used (if any) for the restoration
        long emittedtime;
};

#endif
```

5.1.2.12 Δοκιμή (Test)

```
// IDL
//  File:    Test.idl
//

#ifndef Test_idl
#define Test_idl

#include "Resource.idl"

struct Test {
        Resource  rsc;
        string type;          // "OperationalTest" ...
        any tst;   // OperationalTest
};

#endif
```

5.1.2.13 Δοκιμή Λειτουργίας (Operational Test)

```
// IDL
//  File:    OperationalTest.idl
//

#ifndef OperationalTest_idl
#define OperationalTest_idl

#include "Resource.idl"

struct OperationalTest {
        long started;       //time the test started
        long ended;         //time the test ended
        string result;      //result of test
};

#endif
```

5.1.2.14 Αναφορά Συνόλου Ειδοποιήσεων (Alarm Summary Report)

```
// IDL
// File:    AlarmSummaryReport.idl
//

#ifndef AlarmSummaryReport_idl
#define AlarmSummaryReport_idl

#include "Alarm.idl"

typedef sequence<Alarm> AlarmSummaryReport;

#endif
```

5.1.2.15 Αναφορά Συνόλου Δοκιμών (Test Summary Report)

```
// IDL
// File:    TestSummaryReport.idl
//

#ifndef TestSummaryReport_idl
#define TestSummaryReport_idl

#include "Test.idl"

typedef sequence<Test> TestSummaryReport;

#endif
```

5.1.2.16 Στατιστικά Πόρου (Risk Map Resource Info) και Χάρτης Στατιστικών Πόρων (Network Risk Map)

```
// IDL
// File:    NetworkRiskMap.idl
```

```
//

#ifndef NetworkRiskMap_idl
#define NetworkRiskMap_idl

#include "Resource.idl"

// The Risk Map info of a resource
struct RiskMapResourceInfo {

        //resource id
        Resource resource;

        //Mean Time Between Failure info for the resource
        long MTBF;

        //Mean Time To Repair info for the resource
        long MTTR;

        //Resource Usage Reliability is the Mean Time of the MTBFs of the
        //Resources that use this Resource
        long RUR;
};

// A sequence of resource risk map info entries
typedef sequence<RiskMapResourceInfo> RiskMapResourceInfoList;

struct NetworkRiskMap {

        //a sequence of risk map info entries of the links of the network
        RiskMapResourceInfoList links;

        //a sequence of risk map info entries of the vpcs of the network
        RiskMapResourceInfoList vpcs;

        //a sequence of risk map info entries of the vccs of the network
        RiskMapResourceInfoList vccs;
};

#endif
```

5.1.2.17 Χάρτης Τοπολογίας Δικτύου (Network Topology Map)

```
// IDL
// File:    NetworkTopologyMap.idl
//

#include "Resource.idl"
#include "MonitorClass.idl"
#include "RestorationClass.idl"

struct SubNetwork {         // A SubNetwork is a managed node in the Managed Network
        Resource node;
        MonitorClass mc;
        RestorationClass rc;
};

typedef short TerminationPointPool;         // A TerminationPointPool is a port of a SubNetwork
typedef long TerminationPoint;              // A TerminationPoint is a vpi of a TerminationPointPool
```

```
// A Link Termination Point is the node and the port in
// which the link terminates
struct LinkTerminationPoint {
        SubNetwork sbnw;
        TerminationPointPool tpp;
};

// A Topological Link is defined by two Link Termination Points
struct TopologicalLink {
        Resource tl;
        MonitorClass mc;
        RestorationClass rc;
        LinkTerminationPoint ltpA;
        LinkTerminationPoint ltpB;
};

// A Network Connection Termination Point is the Link Termination Point
// and the tp (vpi) in which the connection terminates
struct NCTerminationPoint {
        LinkTerminationPoint ltp;
        TerminationPoint tp;
};

// VP Link is defined by two NCTerminationPoints
struct VPLink {
        NCTerminationPoint nctpA;
        NCTerminationPoint nctpB;
};

// VPConnection is a sequence of VPLinks
struct VPConnection {
        Resource vpc;
        MonitorClass mc;
        RestorationClass rc;
        sequence<VPLink> vpls;
};

// NetworkNodes is a sequence of SubNetworks
typedef sequence<SubNetwork> NetworkNodes;

// The PL Layer Network is a sequence of Topological Links
typedef sequence<TopologicalLink> PLLayerNetwork;

// The VP Layer Network is a sequence of VPConnections
typedef sequence<VPConnection> VPLayerNetwork;


// The Network Topology Map is the Physical and VP Layer of the managed Network
struct NetworkTopologyMap {
        VPLayerNetwork vpcs;
        PLLayerNetwork links;
};

#endif
```

5.1.2.18 Διεπαφή Ανταλλαγής Ειδοποιήσεων

// IDL

```
// File:    FMAlarmSupply.idl
//

#include "Resource.idl"
#include "Alarm.idl"

interface FMAlarmSupply {

        // Exception raised when an unknown alarm type is supplied
        exception UnknownAlarmType {
                string reason;
        };

        // Exception raised when the supply failed
        exception AlarmSupplyFailure {
                string reason;
        };

        // Method used to supply an alarm
        void supplyAlarm(in Alarm alarm)
                        raises (UnknownAlarmType, AlarmSupplyFailure);
};
```

5.1.2.19 Διεπαφή Συντονιστή Διαχείρισης Λαθών

```
// IDL
// File:    FMCoordinator.idl
//

#include "Resource.idl"
#include "MonitorClass.idl"
#include "RestorationClass.idl"
#include "NetworkTopologyMap.idl"
#include "NetworkRiskMap.idl"

interface FMCoordinator {

      // Exceptions raised by method calls ...

      // Exception raised when the initializing of the component fails
          exception InitializeFailure {
                string reason;
          };

      // Exception raised when the query of the NetworkRiskMap fails
      exception NetworkRiskMapFailure {
            string reason;
      };

      // Exception raised when the managing of a resource fails
      exception ManagingResourceFailure {
            string reason;
      };

      // Exception raised when the usage info supply fails
      exception UsedResourceFailure {
            string reason;
```

```
        };


        // Methods of the FMCoordinator interface

             // Method used to initialize the component
            NetworkTopologyMap initialize();

            // Method used to query the NetworkRiskMap
            void getNetworkRiskMap(out NetworkRiskMap nrm)
                                raises (NetworkRiskMapFailure);

            // Method used to start the managing of a resource
            void startFMResource(in VPConnection rsc, in string cos)
                                raises (ManagingResourceFailure);
            void stopFMResource(in VPConnection rsc)
                                raises (ManagingResourceFailure);

};
```

5.1.2.20 Διεπαφή Διαχειριστή Ειδοποιήσεων Διαχείρισης Λαθών

```
// IDL
// File:    FMAlarmManager.idl
//

#include "Resource.idl"
#include "NetworkRiskMap.idl"
#include "AlarmSummaryReport.idl"
#include "MonitorClass.idl"

interface FMAlarmManager {

            // Exceptions raised by method calls ...

            // Exception raised when the creation of the NetworkRiskMap fails
            exception NetworkRiskMapFailure {
                        string reason;
            };

            // Exception raised when the creation of an AlarmSummaryReport fails
            exception AlarmSummaryReportFailure {
                        string reason;
            };

            // Exception raised when the monitoring of a resource fails
            exception MonitoringResourceFailure {
                        string reason;
            };

            // Exception raised when the stop of monitoring a resource fails
            exception StopMonitoringResourceFailure {
                        string reason;
            };

            // Methods of the FMAlarmManager interface
```

```
        // Method used to query the NetworkRiskMap
        void getNetworkRiskMap(out NetworkRiskMap nrm)
                        raises (NetworkRiskMapFailure);

        // Method used to query an AlarmSummaryReport for a resource
        void getAlarmSummaryReport(in Resource rsc, out AlarmSummaryReport asr)
                        raises (AlarmSummaryReportFailure);

        // Method used to start the monitoring of a resource
        void monitorResource(in Resource rsc, in string type, in RestorationClass mc, in Object obj)
                        raises (MonitoringResourceFailure);

        // Method used to stop the monitoring of a resource
        void stopMonitorResource(in Resource rsc, in string type, in Object obj)
                        raises (StopMonitoringResourceFailure);

};
```

5.1.2.21 Διεπαφή Εξυπηρετητή Δοκιμών Διαχείρισης Λαθών

```
//IDL
//  File:    FMTestingServer.idl
//

#include "Resource.idl"
#include "TestSummaryReport.idl"

interface FMTestingServer {

        // Exceptions raised by method calls ...

        // Exception raised when the creation of a TestSummaryReport fails
        exception TestSummaryReportFailure {
                string reason;
        };

        // Exception raised when the testing of a resource fails
        exception TestResourceFailure {
                string reason;
        };

        // Methods of the FMTestingServer interface

        // Method used to query a TestSummaryReport for a resource
        void getTestSummaryReport(in Resource rsc, out TestSummaryReport tsr)
                raises (TestSummaryReportFailure);

        // Method used to query a TestSummaryReport for a resource
        void testResource(inout Test tst)
                raises (TestResourceFailure);

};
```

5.1.2.22 Διεπαφή Γραφικής Διεπαφής Χρήστη Διαχείρισης Λαθών

```
// IDL
```

```
// File:    FMGUI.idl
//

#include "NetworkTopologyMap.idl"

interface FMGUI {

        // Exceptions raised by method calls ...

        // Exception raised when the call to addVPCResource fails
        exception AddingVPConnectionFailure {
                string reason;
        };

        // Exception raised when the call to deleteVPCResource fails
        exception DeletingVPConnectionFailure {
                string reason;
        };

        // Methods of the FMGUI interface

        // Method to add a VPConnection in the Graphical Interface
        void addVPCResource(in VPConnection vpc)
                raises (AddingVPConnectionFailure);


        // Method to delete a VPConnection from the Graphical Interface
        void deleteVPCResource(in VPConnection vpc)
                raises (DeletingVPConnectionFailure);

};
```

5.2 Παρουσίαση Εσωτερικής Πληροφορίας

Όπως προαναφέρθηκε η εσωτερική πληροφορία περιλαμβάνει την πληροφορία που φυλάγεται για κάθε πόρο του δικτύου εσωτερικά καθώς και την οργάνωσή της. Στην αρχή παρουσιάζεται το μοντέλο της πληροφορίας σε Τεχνική Μοντελοποίησης Οντοτήτων (*Object Modeling Technique - OMT*), και στην συνέχεια αναλύεται κάθε οντότητα πληροφορίας (τα πεδία και οι μέθοδοι που παρέχει). Στο Παράρτημα Α της εργασίας παραθέτεται ο κώδικας (C++) της υλοποίησης του εσωτερικού μοντέλου πληροφορίας

5.2.1 OMT Διάγραμμα του Μοντέλου

Στο Σχήμα 5-5 φαίνεται το OMT διάγραμμα των οντοτήτων της πληροφορίας που κρατείται εσωτερικά στο σύστημα διαχείρισης. Σε ποια ακριβώς υπολογιστική οντότητα (που περιγράφηκε στο Κεφάλαιο 4) κρατείται η κάθε οντότητα περιγράφεται τόσο στο κεφάλαιο 4 όσο και στα θέματα υλοποίησης (κεφάλαιο 6).

Όπως λοιπόν φαίνεται από το σχήμα υπάρχουν οι εξής οντότητες πληροφορίας:

1. Πεδίο Διαχείρισης Σφαλμάτων (**Fault Management Domain**), είναι μία κλάση που χρησιμοποιείται για σκοπούς κληρονομικότητας κυρίως. Περιλαμβάνει έναν αριθμό από πόρους διαχειριζόμενους ως προς τα σφάλματά τους (Fault Manageable Resources) όπως και μία ή περισσότερες κλάσεις Current Alarm Summary Control, Alarm Severity Assignment Profile.

2. Πόρος Διαχειριζόμενος ως προς τα Σφάλματά του (**Fault Manageable Resource**), είναι η αντιπροσώπευση ενός πόρου που τον διαχειρίζεται το παρόν σύστημα και χρησιμοποιείται κυρίως για λόγους κληρονομικότητας. Περιέχει ένα Ημερολόγιο (Log).

3. Πεδίο Διαχείρισης Σφαλμάτων Εικονικών Μονοπατιών Σύνδεσης (**VPC Fault Management Domain**), κληρονομεί από το Πεδίο Διαχείρισης Σφαλμάτων και όλοι οι πόροι που περιέχει είναι Εικονικά Μονοπάτια Σύνδεσης (VPCs).

4. Πεδίο Διαχείρισης Σφαλμάτων Γραμμών (**Link Fault Management Domain**), κληρονομεί από το Πεδίο Διαχείρισης Σφαλμάτων και όλοι οι πόροι που περιέχει είναι Γραμμές (Links).

5. Πεδίο Διαχείρισης Σφαλμάτων Κόμβων (**Node Fault Management Domain**), κληρονομεί από το Πεδίο Διαχείρισης Σφαλμάτων και όλοι οι πόροι που περιέχει είναι Κόμβοι (Nodes).

6. Πόρος που στέλνει Ειδοποιήσεις (**Alarmable Resource**), είναι ένας πόρος που μπορεί να παρακολουθηθεί για λάθη και αν ένα λάθος συμβεί να αναφερθεί μέσω ειδοποίησης. Κληρονομεί από τον Πόρο Διαχειριζόμενο ως προς τα Σφάλματά του και περιέχει μία οντότητα Ελεγκτής Ειδοποιήσεων (Alarm Control).

7. Πόρος που μπορεί να Δοκιμαστεί (**Testable Resource**), είναι ένας πόρος που μπορεί να δοκιμαστεί για την λειτουργία του. Κληρονομεί από τον Πόρο Διαχειριζόμενο ως προς τα Σφάλματά του.

8. Πόρος Εικονικό Μονοπάτι Σύνδεσης που μπορεί να Δοκιμαστεί (**VPC Testable Resource**), είναι ένα Εικονικό Μονοπάτι Σύνδεσης που μπορεί να Δοκιμαστεί. Κληρονομεί από τον Πόρο που μπορεί να Δοκιμαστεί.

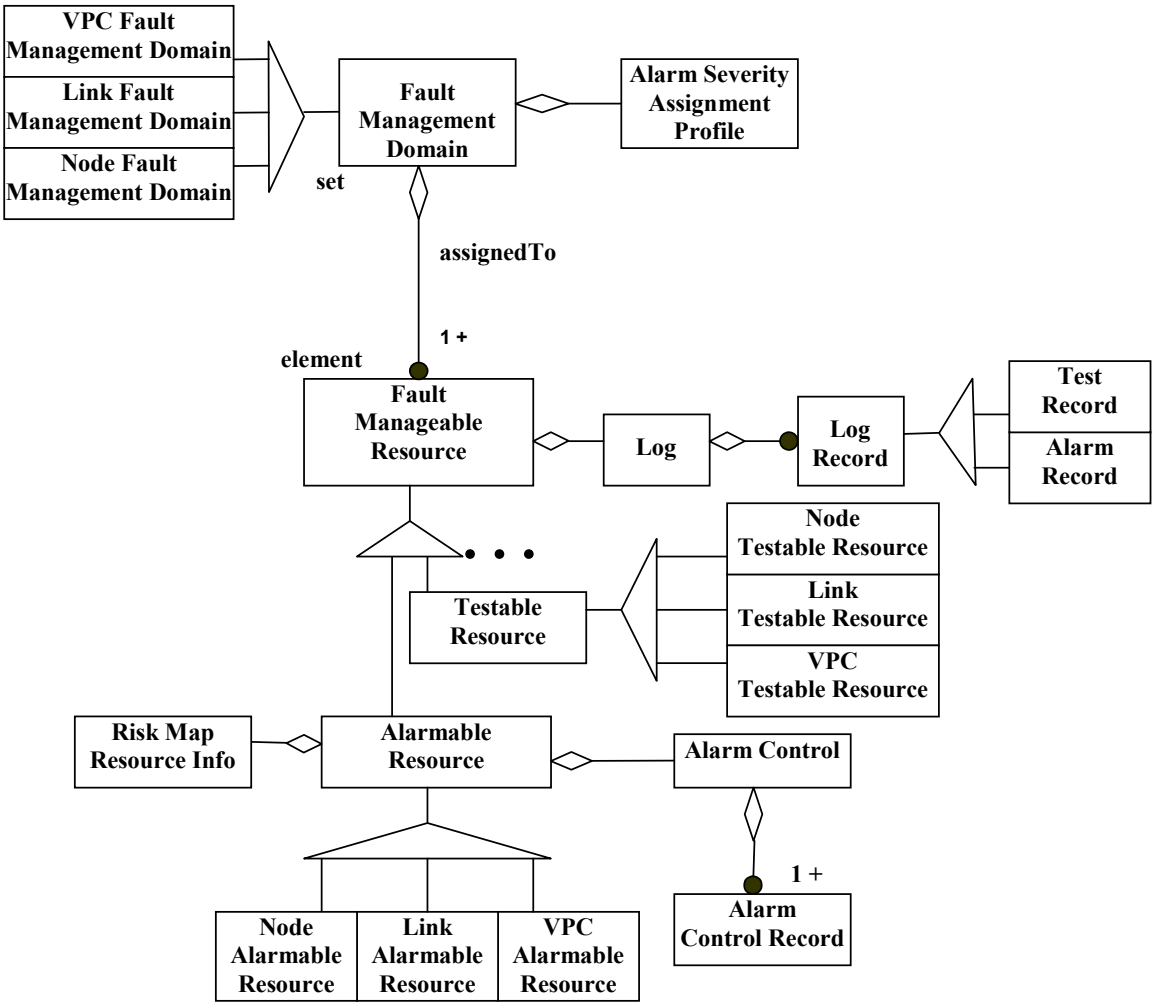
9. Πόρος Γραμμή που μπορεί να Δοκιμαστεί (**Link Testable Resource**), είναι μια Γραμμή Πόρος που μπορεί να Δοκιμαστεί. Κληρονομεί από τον Πόρο που μπορεί να Δοκιμαστεί.

10. Πόρος Κόμβος που μπορεί να Δοκιμαστεί (**Node Testable Resource**), είναι ένας Πόρος Κόμβος που μπορεί να Δοκιμαστεί. Κληρονομεί από τον Πόρο που μπορεί να Δοκιμαστεί.

11. Πόρος Εικονικό Μονοπάτι Σύνδεσης που στέλνει Ειδοποιήσεις (**VPC Alarmable Resource**), είναι ένα Εικονικό Μονοπάτι Σύνδεσης που μπορεί να παρακολουθηθεί για σφάλματα και αν ένα σφάλμα συμβεί να αναφερθεί μέσω ειδοποίησης. Κληρονομεί από τον Πόρο που μπορεί να Δοκιμαστεί.

12. Πόρος Γραμμή που στέλνει Ειδοποιήσεις (**Link Alarmable Resource**), είναι μια Γραμμή που μπορεί να παρακολουθηθεί για λάθη και αν ένα λάθος συμβεί να αναφερθεί μέσω ειδοποίησης. Κληρονομεί από τον Πόρο που στέλνει Ειδοποιήσεις.

13. Πόρος Κόμβος που στέλνει Ειδοποιήσεις (**Node Alarmable Resource**), είναι ένας Κόμβος που μπορεί να παρακολουθηθεί για σφάλματα και αν ένα σφάλμα συμβεί να αναφερθεί μέσω ειδοποίησης. Κληρονομεί από τον Πόρο που στέλνει Ειδοποιήσεις.



Σχήμα 5-5 OMT Διάγραμμα των οντοτήτων εσωτερικής πληροφορίας

14. Ελεγκτής Ειδοποιήσεων (**Alarm Control**), είναι μια οντότητα που βρίσκεται σε κάθε Πόρο που στέλνει Ειδοποιήσεις και χρησιμοποιείται για να αποφασιστεί που θα σταλεί η ειδοποίηση του πόρου αυτού. Είναι ένα σύνολο από οντότητες Εγγραφή Ελεγκτή Ειδοποιήσεων (Alarm Control Record).

15. Εγγραφή Ελεγκτή Ειδοποιήσεων (**Alarm Control Record**), βρίσκεται μέσα στην οντότητα Ελεγκτής Ειδοποιήσεων και περιέχει τον προορισμό συγκεκριμένου είδους ειδοποίησης. Όποτε αυτό το είδος ειδοποίησης έρθει ελέγχεται κάθε Εγγραφή Ελεγκτή Ειδοποιήσεων και βρίσκονται οι προορισμοί.

16. Ημερολόγιο (**Log**), βρίσκεται μέσα σε κάθε Πόρο Διαχειριζόμενο ως προς τα Σφάλματά του και χρησιμοποιείται για να φυλαχτούν αποτελέσματα Δοκιμών ή Ειδοποιήσεις που έχουν ληφθεί γι' αυτόν τον Πόρο (ανάλογα αν είναι Πόρος που μπορεί να Δοκιμαστεί ή Πόρος που στέλνει ειδοποιήσεις). Είναι ένα σύνολο από Εγγραφές Ημερολογίου (Log Records).

17. Εγγραφή Ημερολογίου (**Log Record**), χρησιμοποιείται για λόγους κληρονομικότητας.

18. Εγγραφή Δοκιμής (**Test Record**), κληρονομεί από την Εγγραφή Ημερολογίου και χρησιμοποιείται για αποθηκευτεί το περιεχόμενο του αποτελέσματος μιας Δοκιμής.

19. Εγγραφή Ειδοποίησης (**Alarm Record**), κληρονομεί από την Εγγραφή Ημερολογίου και χρησιμοποιείται για αποθηκευτεί το περιεχόμενο μιας Ειδοποιήσεως.

20. Απονομέας Σημαντικότητας Σφάλματος (**Alarm Severity Assignment Profile**), είναι μία λίστα με αντιστοιχίσεις πιθανών αιτιών σφάλματος σε σημαντικότητα σφάλματος και χρησιμοποιείται για να αποδόσει σημαντικότητα σε ένα σφάλμα που έχει γίνει για συγκεκριμένη αιτία.

5.2.2 Ορισμός Οντοτήτων Εσωτερικής Πληροφορίας

Στην υποενότητα αυτή ορίζονται αναλυτικά οι οντότητες πληροφορίας που περιγράφονται στο OMT διάγραμμα που φαίνεται στο Σχήμα 5-5. Για κάθε οντότητα δίνεται η περιγραφή της, από που κληρονομεί, τα πεδία της (attributes) και η περιγραφή τους καθώς και οι μέθοδοι (methods) που προσφέρει και η περιγραφή τους.

5.2.2.1 Πεδίο Διαχείρισης Σφαλμάτων (Fault Management Domain)

5.2.2.1.1 Περιγραφή

Είναι μια οντότητα πληροφορίας που περιέχει έναν αριθμό από πόρους διαχειριζόμενους ως προς τα σφάλματά τους (Fault Manageable Resources) με όμοιο τρόπο.

5.2.2.1.2 Κληρονομικότητα

Δεν κληρονομεί.

5.2.2.1.3 Πεδία

Πεδία	**Περιγραφή**
FaultManagementDomainName	Το όνομα του
FaultManagementDomainType	Ο τύπος του (από εδώ ξεχωρίζει αν περιέχει Γραμμές ή Εικονικά Μονοπάτια Σύνδεσης)
FaultManageableResourceList	Μία λίστα με τους πόρους που περιέχει
AlarmSeverityAssignmentProfile	Μία οντότητα που περιγράφεται παρακάτω και είναι υπεύθυνη για την ανάθεση της κρισιμότητας (Severity) μιας Ειδοποίησης.

5.2.2.1.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetName	Μέθοδος που επιστρέφει το όνομα
GetType	Μέθοδος που επιστρέφει τον τύπο
AddFaultManageableResource	Μέθοδος που προσθέτει έναν πόρο
DeleteFaultManageableResource	Μέθοδος που αφαιρεί έναν πόρο
GetRiskMapResourceInfoList	Μέθοδος που επιστρέφει τα στατιστικά όλων των πόρων του πεδίου διαχείρισης σφαλμάτων

5.2.2.2 Πόρος Διαχειριζόμενος ως προς τα Σφάλματά του (Fault Manageable Resource)

5.2.2.2.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί να διαχειριστεί για τα λάθη του.

5.2.2.2.2 Κληρονομικότητα

Δεν κληρονομεί.

5.2.2.2.3 Πεδία

Πεδία	**Περιγραφή**
ResourceName	Το όνομα του
ResourceType	Ο τύπος του (από εδώ ξεχωρίζει αν μπορεί να Δοκιμαστεί ή στέλνει Ειδοποιήσεις)
ResourceState	Η παρούσα κατάστασή του (από πλευρά λειτουργικότητας)
CreationTime	Ο χρόνος που αρχίσαμε να τον διαχειριζόμαστε
RiskMapResourceInfo	Η πληροφορία στατιστικών για τον κόμβο (κοίτα μοντέλο εξωτερικής πληροφορίας)
RestorationClass	Η κλάση επιδιόρθωσης
Log	Μία οντότητα Ημερολόγιο που χρησιμοποιείται για να φυλάμε τα αποτελέσματα Δοκιμών ή τις Ειδοποιήσεις για αυτόν τον πόρο.

5.2.2.2.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetName	Μέθοδος που επιστρέφει το όνομα
GetType	Μέθοδος που επιστρέφει τον τύπο
ChangeState	Μέθοδος που αλλάζει την κατάσταση του πόρου
AddLogRecord	Μέθοδος που προσθέτει μια εγγραφή ημερολογίου στο Ημερολόγιο
DeleteLogRecord	Μέθοδος που αφαιρεί μια εγγραφή ημερολογίου από το Ημερολόγιο
UpdateRiskMapInfo	Μέθοδος για την ενημέρωση των στατιστικών αυτού του πόρου από την πληροφορία του Ημερολογίου (Log)
GetRiskMapResourceInfo	Μέθοδος για την απόκτηση των στατιστικών (την στιγμή αυτή) για αυτόν τον πόρο

5.2.2.3 Πεδίο Διαχείρισης Σφαλμάτων Εικονικών Μονοπατιών Σύνδεσης (VPC Fault Management Domain)

5.2.2.3.1 Περιγραφή

Είναι μια οντότητα πληροφορίας που περιέχει έναν αριθμό από πόρους διαχειριζόμενους ως προς τα λάθη τους (Fault Manageable Resources) με όμοιο τρόπο και οι οποίοι είναι όλοι τύπου Εικονικά Μονοπάτια Σύνδεσης (VPCs).

5.2.2.3.2 Κληρονομικότητα

Κληρονομεί από το Πεδίο Διαχείρισης Λαθών.

5.2.2.3.3 Πεδία

Δεν υπάρχουν πεδία.

5.2.2.3.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetVPCAlarmableResource	Μέθοδος που επιστρέφει ένα VPC που στέλνει ειδοποιήσεις
GetVPCTestableResource	Μέθοδος που επιστρέφει ένα VPC που δοκιμάζεται
MonitorVPC	Μέθοδος που αρχίζει την παρακολούθηση ενός VPC
StopMonitorVPC	Μέθοδος που σταματάει την παρακολούθηση ενός VPC
TestVPC	Μέθοδος που αρχίζει τη δοκιμή ενός VPC

5.2.2.4 Πεδίο Διαχείρισης Σφαλμάτων Γραμμών (Link Fault Management Domain)

5.2.2.4.1 Περιγραφή

Είναι μια οντότητα πληροφορίας που περιέχει έναν αριθμό από πόρους διαχειριζόμενους ως προς τα λάθη τους (Fault Manageable Resources) με όμοιο τρόπο και οι οποίοι είναι όλοι τύπου Γραμμής (Links).

5.2.2.4.2 Κληρονομικότητα

Κληρονομεί από το Πεδίο Διαχείρισης Λαθών.

5.2.2.4.3 Πεδία

Δεν υπάρχουν πεδία.

5.2.2.4.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetLinkAlarmableResource	Μέθοδος που επιστρέφει ένα Link που στέλνει ειδοποιήσεις
GetLinkTestableResource	Μέθοδος που επιστρέφει ένα Link που δοκιμάζεται
MonitorLink	Μέθοδος που αρχίζει την παρακολούθηση ενός Link
StopMonitorLink	Μέθοδος που σταματάει την παρακολούθηση ενός Link
TestLink	Μέθοδος που αρχίζει τη δοκιμή ενός Link

5.2.2.5 Πεδίο Διαχείρισης Σφαλμάτων Κόμβων (Node Fault Management Domain)

5.2.2.5.1 Περιγραφή

Είναι μια οντότητα πληροφορίας που περιέχει έναν αριθμό από πόρους διαχειριζόμενους ως προς τα λάθη τους (Fault Manageable Resources) με όμοιο τρόπο και οι οποίοι είναι όλοι τύπου Κόμβου (Nodes).

5.2.2.5.2 Κληρονομικότητα

Κληρονομεί από το Πεδίο Διαχείρισης Λαθών.

5.2.2.5.3 Πεδία

Δεν υπάρχουν πεδία.

5.2.2.5.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetNodeAlarmableResource	Μέθοδος που επιστρέφει ένα Node που στέλνει ειδοποιήσεις
GetNodeTestableResource	Μέθοδος που επιστρέφει ένα Node που δοκιμάζεται
MonitorNode	Μέθοδος που αρχίζει την παρακολούθηση ενός Node
StopMonitorNode	Μέθοδος που σταματάει την παρακολούθηση ενός Node
TestNode	Μέθοδος που αρχίζει τη δοκιμή ενός Node

5.2.2.6 Πόρος που στέλνει Ειδοποιήσεις (Alarmable Resource)

5.2.2.6.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί να παρακολουθηθεί για λάθη και αν ένα λάθος συμβεί να αναφερθεί μέσω ειδοποίησης.

5.2.2.6.2 Κληρονομικότητα

Κληρονομεί από τον Πόρο Διαχειριζόμενο ως προς τα Λάθη του.

5.2.2.6.3 Πεδία

Πεδία	**Περιγραφή**
AlarmableResourceType	Ο τύπος του (από εδώ ξεχωρίζει αν είναι VPCAlarmableResource ή LinkAlarmableResource ή NodeAlarmableResource
AlarmControl	Ο Ελεγκτής Ειδοποιήσεων

5.2.2.6.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetType	Μέθοδος που επιστρέφει τον τύπο
AddAlarmControlRecord	Μέθοδος που προσθέτει μια εγγραφή στον Ελεγκτή Ειδοποιήσεων
DeleteAlarmControlRecord	Μέθοδος που αφαιρεί μια εγγραφή από τον Ελεγκτή Ειδοποιήσεων
GetAlarmSummaryReport	Μέθοδος με την οποία παίρνουμε συνολική αναφορά των ειδοποιήσεων γιαυτόν τον πόρο

5.2.2.7 Πόρος που μπορεί να Δοκιμαστεί (Testable Resource)

5.2.2.7.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί δοκιμαστεί για την λειτουργία του.

5.2.2.7.2 Κληρονομικότητα

Κληρονομεί από τον Πόρο Διαχειριζόμενο ως προς τα Λάθη του.

5.2.2.7.3 Πεδία

Πεδία	Περιγραφή
TestableResourceType	Ο τύπος του (από εδώ ξεχωρίζει αν είναι VPCTestableResource ή LinkTestableResource ή NodeTestableResource

5.2.2.7.4 Μέθοδοι

Μέθοδοι	Περιγραφή
GetType	Μέθοδος που επιστρέφει τον τύπο
GetTestSummaryReport	Μέθοδος με την οποία παίρνουμε συνολική αναφορά των δοκιμών γιαυτόν τον πόρο

5.2.2.8 Πόρος Κόμβος που μπορεί να Δοκιμαστεί (Node Testable Resource)

5.2.2.8.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί δοκιμαστεί για την λειτουργία του και είναι τύπου Κόμβος.

5.2.2.8.2 Κληρονομικότητα

Κληρονομεί από τον Πόρο που μπορεί να Δοκιμαστεί.

5.2.2.8.3 Πεδία

Πεδία	Περιγραφή
NodeResource	Η ταυτότητά του (κοίτα ορισμό οντοτήτων εξωτερικής πληροφορίας)

5.2.2.8.4 Μέθοδοι

Δεν υπάρχουν μέθοδοι.

5.2.2.9 Πόρος Γραμμή που μπορεί να Δοκιμαστεί (Link Testable Resource)

5.2.2.9.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί δοκιμαστεί για την λειτουργία του και είναι τύπου Γραμμής.

5.2.2.9.2 Κληρονομικότητα

Κληρονομεί από τον Πόρο που μπορεί να Δοκιμαστεί.

5.2.2.9.3 Πεδία

Πεδία	Περιγραφή
LinkResource	Η ταυτότητά του (κοίτα ορισμό οντοτήτων εξωτερικής

	πληροφορίας)

5.2.2.9.4 Μέθοδοι

Δεν υπάρχουν μέθοδοι.

5.2.2.10 Πόρος Εικονικό Μονοπάτι Σύνδεσης που μπορεί να Δοκιμαστεί (VPC Testable Resource)

5.2.2.10.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί δοκιμαστεί για την λειτουργία του και είναι τύπου Εικονικό Μονοπάτι Σύνδεσης.

5.2.2.10.2 Κληρονομικότητα

Κληρονομεί από τον Πόρο που μπορεί να Δοκιμαστεί.

5.2.2.10.3 Πεδία

Πεδία	**Περιγραφή**
VPCResource	Η ταυτότητά του (κοίτα ορισμό οντοτήτων εξωτερικής πληροφορίας)

5.2.2.10.4 Μέθοδοι

Δεν υπάρχουν μέθοδοι.

5.2.2.11 Πόρος Κόμβος που στέλνει Ειδοποιήσεις (Node Alarmable Resource)

5.2.2.11.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί να παρακολουθηθεί για λάθη και αν ένα λάθος συμβεί να αναφερθεί μέσω ειδοποίησης και είναι τύπου Κόμβος.

5.2.2.11.2 Κληρονομικότητα

Κληρονομεί από τον Πόρο που στέλνει Ειδοποιήσεις.

5.2.2.11.3 Πεδία

Πεδία	**Περιγραφή**
NodeResource	Η ταυτότητά του (κοίτα ορισμό οντοτήτων εξωτερικής πληροφορίας)

5.2.2.11.4 Μέθοδοι

Δεν υπάρχουν μέθοδοι.

5.2.2.12 Πόρος Γραμμή που στέλνει Ειδοποιήσεις (Link Alarmable Resource)

5.2.2.12.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί να παρακολουθηθεί για λάθη και αν ένα λάθος συμβεί να αναφερθεί μέσω ειδοποίησης και είναι τύπου Γραμμής.

5.2.2.12.2 Κληρονομικότητα

Κληρονομεί από τον Πόρο που στέλνει Ειδοποιήσεις.

5.2.2.12.3 Πεδία

Πεδία	**Περιγραφή**
LinkResource	Η ταυτότητά του (κοίτα ορισμό οντοτήτων εξωτερικής πληροφορίας)

5.2.2.12.4 Μέθοδοι

Δεν υπάρχουν μέθοδοι.

5.2.2.13 Πόρος Εικονικό Μονοπάτι Σύνδεσης που στέλνει Ειδοποιήσεις (VPC Alarmable Resource)

5.2.2.13.1 Περιγραφή

Είναι η οντότητα που αντιπροσωπεύει έναν πόρο που μπορεί να παρακολουθηθεί για λάθη και αν ένα λάθος συμβεί να αναφερθεί μέσω ειδοποίησης και είναι τύπου Εικονικό Μονοπάτι Σύνδεσης.

5.2.2.13.2 Κληρονομικότητα

Κληρονομεί από τον Πόρο που στέλνει Ειδοποιήσεις.

5.2.2.13.3 Πεδία

Πεδία	**Περιγραφή**
VPCResource	Η ταυτότητά του (κοίτα ορισμό οντοτήτων εξωτερικής πληροφορίας)

5.2.2.13.4 Μέθοδοι

Δεν υπάρχουν μέθοδοι.

5.2.2.14 Ελεγκτής Ειδοποιήσεων (Alarm Control)

5.2.2.14.1 Περιγραφή

Είναι η οντότητα που βρίσκεται σε κάθε Πόρο που στέλνει Ειδοποιήσεις και χρησιμοποιείται για να αποφασιστεί που θα σταλεί η ειδοποίηση του πόρου αυτού. Είναι ένα σύνολο από οντότητες Εγγραφή Ελεγκτή Ειδοποιήσεων (Alarm Control Record) οι οποίες ψάχνονται όταν φτάσει μια ειδοποίηση για τον πόρο για να βρεθούν οι προορισμοί.

5.2.2.14.2 Κληρονομικότητα

Δεν κληρονομεί.

5.2.2.14.3 Πεδία

Πεδία	**Περιγραφή**
AlarmControlList	Η λίστα με τις Εγγραφές Ελεγκτή Ειδοποιήσεων

5.2.2.14.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
AddAlarmControlRecord	Μέθοδος με την οποία προσθέτουμε μία εγγραφή
DeleteAlarmControlRecord	Μέθοδος με την οποία αφαιρούμε μία εγγραφή
GetDestinationList	Μέθοδος με την οποία παίρνουμε μια λίστα με τους προορισμούς συγκεκριμένου είδους ειδοποίησης

5.2.2.15 Εγγραφή Ελεγκτή Ειδοποιήσεων (Alarm Control Record)

5.2.2.15.1 Περιγραφή

Είναι μια οντότητα που βρίσκεται μέσα στην οντότητα Ελεγκτής Ειδοποιήσεων και περιέχει τον προορισμό συγκεκριμένου είδους ειδοποίησης. Όποτε αυτό το είδος ειδοποίησης έρθει ελέγχεται κάθε Εγγραφή Ελεγκτή Ειδοποιήσεων και βρίσκονται οι προορισμοί.

5.2.2.15.2 Κληρονομικότητα

Δεν κληρονομεί.

5.2.2.15.3 Πεδία

Πεδία	**Περιγραφή**
Destination	Προορισμός της Ειδοποίησης του παρακάτω τύπου
Type	Τύπος Ειδοποίησης

5.2.2.15.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetType	Μέθοδος με την οποία παίρνουμε τον τύπο
GetDestination	Μέθοδος με την οποία παίρνουμε τον προορισμό

5.2.2.16 Ημερολόγιο (Log)

5.2.2.16.1 Περιγραφή

Είναι η οντότητα που βρίσκεται σε κάθε Πόρο Διαχειριζόμενο ως προς τα Λάθη του και χρησιμοποιείται για να φυλαχτούν αποτελέσματα Δοκιμών ή Ειδοποιήσεις που έχουν ληφθεί γι' αυτόν τον Πόρο (ανάλογα αν είναι Πόρος που μπορεί να Δοκιμαστεί ή Πόρος που στέλνει ειδοποιήσεις). Είναι ένα σύνολο από Εγγραφές Ημερολογίου (Log Records).

5.2.2.16.2 Κληρονομικότητα

Δεν κληρονομεί.

5.2.2.16.3 Πεδία

Πεδία	**Περιγραφή**
LogRecordList	Η λίστα με τις Εγγραφές Ημερολογίου

5.2.2.16.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
AddLogRecord	Μέθοδος με την οποία προσθέτουμε μία εγγραφή
DeleteLogRecord	Μέθοδος με την οποία αφαιρούμε μία εγγραφή
GetLogRecordList	Μέθοδος με την οποία παίρνουμε μια λίστα με τις εγγραφές

5.2.2.17 Εγγραφή Ημερολογίου (Log Record)

5.2.2.17.1 Περιγραφή

Είναι η οντότητα που μέσα της φυλάμε είτε το αποτέλεσμα μίας δοκιμής είτε μια ειδοποίηση. Χρησιμοποιείται για λόγους κληρονομικότητας.

5.2.2.17.2 Κληρονομικότητα

Δεν κληρονομεί.

5.2.2.17.3 Πεδία

Πεδία	**Περιγραφή**
EventType	Ο τύπος του συμβάν που φυλάγεται (Ειδοποίηση ή Αποτέλεσμα Δοκιμής)
EventTime	Ο χρόνος που αποθηκεύτηκε
AdditionalInfo	Περαιτέρω πληροφορία

5.2.2.17.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetEventTime	Μέθοδος με την οποία παίρνουμε τον τύπο
GetEventType	Μέθοδος με την οποία παίρνουμε τον χρόνο
GetAdditionalInfo	Μέθοδος με την οποία παίρνουμε την περαιτέρω πληροφορία

5.2.2.18 Εγγραφή Δοκιμής (Test Record)

5.2.2.18.1 Περιγραφή

Είναι η οντότητα που μέσα της φυλάμε το αποτέλεσμα μίας δοκιμής.

5.2.2.18.2 Κληρονομικότητα

Κληρονομεί από την Εγγραφή Ημερολογίου.

5.2.2.18.3 Πεδία

Πεδία	**Περιγραφή**
Test	Η δοκιμή που φυλάγεται (κοίτα ορισμό οντοτήτων εξωτερικής πληροφορίας)

5.2.2.18.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetType	Μέθοδος με την οποία παίρνουμε τον τύπο της δοκιμής
GetTest	Μέθοδος με την οποία παίρνουμε όλη τη δοκιμή

5.2.2.19 Εγγραφή Ειδοποίησης (Alarm Record)

5.2.2.19.1 Περιγραφή

Είναι η οντότητα στην οποία φυλάμε το περιεχόμενο μιας Ειδοποιήσεως.

5.2.2.19.2 Κληρονομικότητα

Κληρονομεί από την Εγγραφή Ημερολογίου.

5.2.2.19.3 Πεδία

Πεδία	**Περιγραφή**
Alarm	Η ειδοποίηση που φυλάγεται (κοίτα ορισμό οντοτήτων εξωτερικής πληροφορίας)

5.2.2.19.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
GetType	Μέθοδος με την οποία παίρνουμε τον τύπο της ειδοποίησης
GetTest	Μέθοδος με την οποία παίρνουμε όλη την ειδοποίηση

5.2.2.20 Απονομέας Σημαντικότητας Ειδοποίησης Σφάλματος (Alarm Severity Assignment Profile)

5.2.2.20.1 Περιγραφή

Είναι η οντότητα που έχει έναν πίνακα με ζευγάρια πιθανής αιτίας σφάλματος και σημαντικότητας της ειδοποίησης σφάλματος. Την χρησιμοποιούμε για να αποδίδουμε σημαντικότητα σε μία ειδοποίηση σφάλματος το οποίο συνέβει για κάποια αιτία (πιθανή αιτία).

5.2.2.20.2 Κληρονομικότητα

Δεν κληρονομεί.

5.2.2.20.3 Πεδία

Πεδία	**Περιγραφή**
SeverityAssignmentList	Ο πίνακας με τις αντιστοιχίσεις

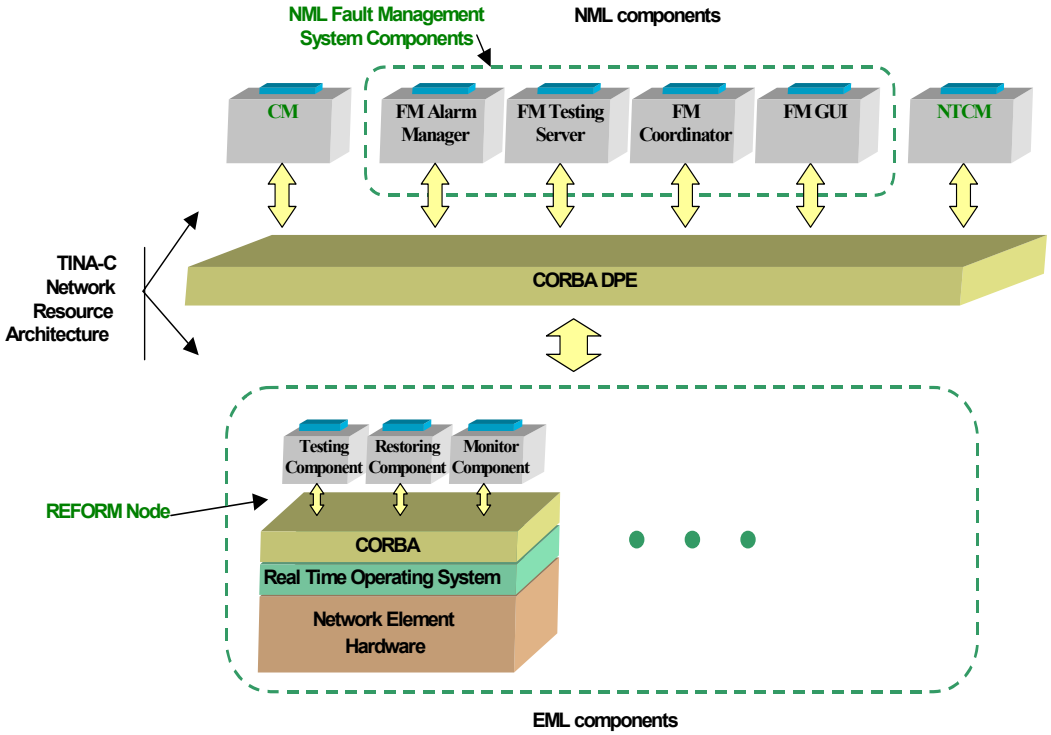
5.2.2.20.4 Μέθοδοι

Μέθοδοι	**Περιγραφή**
SetSeverity	Μέθοδος με την οποία θέτουμε μία καινούργια αντιστοίχιση.
GetSeverity	Μέθοδος με την οποία παίρνουμε μια αντιστοίχιση.

6 Θέματα Υλοποίησης

Στο κεφάλαιο αυτό παρουσιάζονται τα θέματα που αφορούν την υλοποίηση του συστήματος που παρουσιάστηκε στα προηγούμενα κεφάλαια. Η υλοποίηση έγινε σύμφωνα με την Αρχιτεκτονική TINA σε ένα Κατανεμημένο Περιβάλλον Επεξεργασίας [TINA-EM](*TINA-DPE*). Για DPE επιλέχτηκε η υλοποίηση της αρχιτεκτονικής *CORBA*, *Orbix* έκδοση *MT* (Πολλαπλών Νημάτων - *Multi Threaded*) 2.3 της *IONA*.

Στο Σχήμα 6-1 φαίνεται η τοποθέτηση των υπολογιστικών οντοτήτων του συστήματος στο κατανεμημένο περιβάλλον.



Σχήμα 6-1 Τοποθέτηση των οντοτήτων που αποτελούν το προτεινόμενο σύστημα στο TINA DPE

Η Οργάνωση του κεφαλαίου έχει ως εξής: Πρώτα παρουσιάζεται ο τρόπος που οι υπολογιστικές οντότητες του συστήματος εντοπίζουν (με διαφανή τρόπο) η μία την άλλη και τις υπόλοιπες υπολογιστικές οντότητες που χρειάζονται. Ακολουθεί η παρουσίαση των οντοτήτων εκτός συστήματος (υπολογιστικές οντότητες παρακολούθησης, δοκιμής και επιδιόρθωσης πόρων στο EML επίπεδο, υπολογιστική οντότητα Διαχείρισης Διαμόρφωσης, υπολογιστική οντότητα Διαχείρισης Συνδέσεων) οι οποίες απαιτούνται για την λειτουργία του. Στην συνέχεια παρουσιάζεται η υλοποίηση της κάθε οντότητας του συστήματος που περιλαμβάνει την γλώσσα προγραμματισμού που χρησιμοποιήθηκε, την εσωτερική αρχιτεκτονική υλοποίησης, τις διεπαφές καθώς και τις οντότητες πληροφορίας που διατηρεί και ενημερώνει.

6.1 Διαφανής Εντοπισμός Οντοτήτων

Στην αρχιτεκτονική CORBA η μονάδα διευθυνσιοδότησης είναι ο εξυπηρετητής (CORBA server), ο οποίος μπορεί να χειρίζεται μία ή περισσότερες οντότητες, για να εντοπίσεις κάποια οντότητα πρέπει πρώτα να εντοπίσεις τον εξυπηρετητή στον οποίο βρίσκεται. Επειδή όμως ο εντοπισμός του εξυπηρετητή γίνεται με το

όνομα και την πόρτα του μηχανήματος στο οποίο βρίσκεται, το να εντοπίσουμε μία οντότητα δεν είναι καθόλου διαφανές.

Για να μπορέσουν οι διάφορες οντότητες που αποτελούν το σύστημα να εντοπίζουν τις οντότητες με τις οποίες αλληλεπιδρούν (είτε ανήκουν στο σύστημα είτε είναι εξωτερικές) με διαφανή τρόπο, χρησιμοποιήθηκε η υπηρεσία ονομασίας της αρχιτεκτονικής CORBA. Με την υπηρεσία ονομασίας μπορούμε να αντιστοιχήσουμε την κάθε οντότητα σε ένα καθολικό όνομα με το οποίο μπορούμε να την εντοπίζουμε. Ο εξυπηρετητής στον οποίο βρίσκεται η οντότητα εντοπίζεται άμεσα από την υπηρεσία ονομασίας με την χρήση μίας κλήσης, παρέχοντας απλώς το όνομα στο οποίο έχει αντιστοιχιστεί η οντότητα αυτή. Με αυτόν τον τρόπο η κάθε οντότητα μπορεί να εντοπιστεί με διαφανή τρόπο καθώς και να μεταφερθεί με διαφανή τρόπο χωρίς να επηρεάζει την λειτουργικότητα του συστήματος (αρκεί να είναι ενημερωμένη η αντιστοίχηση στην υπηρεσία ονομασίας).

Στην υλοποίηση του συστήματος χρησιμοποιήθηκε η Υπηρεσία Ονομασίας (Naming Service) της αρχιτεκτονικής CORBA και συγκεκριμένα η υλοποίησή της, OrbixNames έκδοση 1.1c της IONA. Το σχήμα της ονομασίας που χρησιμοποιήθηκε είναι ιεραρχικό και βασίζεται στο X.500 [X500] (βλέπε Σχήμα 6-2).

Συγκεκριμένα:

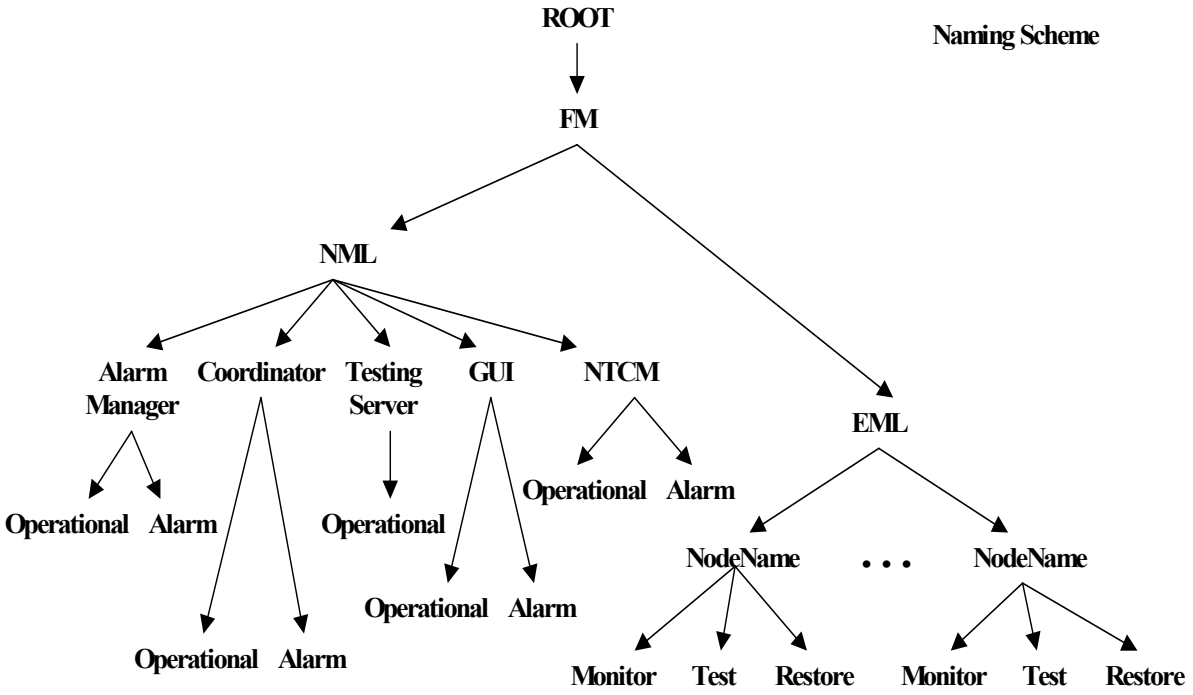
1. Ο Διαχειριστής Σφαλμάτων εγγράφει τη διεπαφή ειδοποιήσεων που προσφέρει στο καθολικό όνομα "*FM*" "*NML*" "*Alarm Manager*" "*Alarm*" και τη διεπαφή λειτουργιών στο όνομα "*FM*" "*NML*" "*Alarm Manager*" "*Operational*". Έτσι οποιαδήποτε οντότητα θέλει να χρησιμοποιήσει την λειτουργική διεπαφή του (ο Συντονιστής) ή την διεπαφή ειδοποιήσεων (οι οντότητες επιδιόρθωσης) μπορεί να τις βρει από την υπηρεσία ονομασίας ανεξαιρέτως, σε ποιο κόμβο του DPE βρίσκεται.

2. Ο Συντονιστής εγγράφει τη διεπαφή ειδοποιήσεων που προσφέρει στο καθολικό όνομα "*FM*" "*NML*" "*Coordinator*" "*Alarm*" και τη διεπαφή λειτουργιών στο όνομα "*FM*" "*NML*" " *Coordinator*" "*Operational*". Έτσι οποιαδήποτε οντότητα θέλει να χρησιμοποιήσει την λειτουργική διεπαφή του (η Γραφική Διεπαφή Χρήστη) ή την διεπαφή ειδοποιήσεων (*NTCM*) μπορεί να τις βρει από την υπηρεσία ονομασίας ανεξαιρέτως, σε ποιο κόμβο του DPE βρίσκεται.

3. Ο Εξυπηρετητής Δοκιμών εγγράφει τη διεπαφή λειτουργιών που προσφέρει στο όνομα "*FM*" "*NML*" " *Testing Server*" "*Operational*". Έτσι οποιαδήποτε οντότητα θέλει να χρησιμοποιήσει την λειτουργική διεπαφή του (Ο Συντονιστής) μπορεί να την βρει από την υπηρεσία ονομασίας ανεξαιρέτως, σε ποιο κόμβο του DPE βρίσκεται.

4. Η γραφική Διεπαφή Χρήστη εγγράφει τη διεπαφή λειτουργιών που προσφέρει στο όνομα "*FM*" "*NML*" "*GUI*" "*Operational*" και την διεπαφή ειδοποιήσεων στο "*FM*" "*NML*" "*GUI*" "Alarm". Έτσι οποιαδήποτε οντότητα θέλει να τις χρησιμοποιήσει (Ο Συντονιστής) μπορεί να τις βρει από την υπηρεσία ονομασίας ανεξαιρέτως, σε ποιο κόμβο του DPE βρίσκεται.

5. Το σύστημα που προσφέρει την τοπολογία του δικτύου (*NTCM*) εγγράφει τη διεπαφή λειτουργιών που προσφέρει στο καθολικό όνομα "*FM*" "*NML*" "*NTCM*" "*Operational*" και την διεπαφή ειδοποιήσεων στο όνομα "*FM*" "*NML*" "*NTCM*" "Alarm". Έτσι οποιαδήποτε οντότητα θέλει να χρησιμοποιήσει τις διεπαφές του (Ο Συντονιστής) μπορεί να τις βρει από την υπηρεσία ονομασίας ανεξαιρέτως, σε ποιο κόμβο του DPE βρίσκεται.



Σχήμα 6-2 Σχήμα Ονομασίας που χρησιμοποιήθηκε για τον διαφανή εντοπισμό των οντοτήτων

6. Οι οντότητες Παρακολούθησης (οι οποίες είναι κοινές για όλους τους πόρους) οι οποίες είναι υπεύθυνες για την παρακολούθηση του κόμβου "*NodeName*" καθώς και των γραμμών και εικονικών μονοπατιών που ξεκινάνε/τερματίζουν σε αυτόν τον κόμβο "*NodeName*", εγγράφουν την λειτουργική διεπαφή που προσφέρουν στο όνομα "*FM*" "*EML*" "*NodeName*" "*Monitor*". Έτσι όταν ο Διαχειριστής Σφαλμάτων θέλει να παρακολουθήσει έναν κόμβο βρίσκει την διεπαφή από το "*FM*" "*EML*" "*NodeName*" "*Monitor*" όπου το "*NodeName*" το βρίσκει από την πληροφορία του πόρου, ενώ όταν θέλει να παρακολουθήσει μία γραμμή ή ένα εικονικό μονοπάτι βρίσκει τις διεπαφές των δύο άκρων από τα ονόματα "*FM*" "*EML*" "*NodeName1*" "*Monitor*" και "*FM*" "*EML*" "*NodeName2*" "*Monitor*" όπου "*NodeName1*" και "*NodeName2*" είναι τα ονόματα των δύο άκρων του πόρου (τα βρίσκει από την πληροφορία του πόρου).

7. Οι οντότητες Δοκιμής και Επιδιόρθωσης εγγράφουν τις λειτουργικές διεπαφές που προσφέρουν ακριβώς όπως και αυτές της Παρακολούθησης αλλά αντί για "*Monitor*" στο όνομα έχουμε αντίστοιχα "*Test*" και "*Restore*". Από αυτά τα ονόματα με παρόμοιο τρόπο όπως και στις οντότητες παρακολούθησης βρίσκουν τις διεπαφές τους.

Οι οντότητες υλοποιήθηκαν με τέτοιο τρόπο ώστε να μπορούν παραμετρικά να χρησιμοποιήσουν άλλο καθολικό όνομα για εγγραφή των διεπαφών τους καθώς και για αναζήτηση των διεπαφών των οντοτήτων με τις οποίες επικοινωνούν.

6.2 Οντότητες Εκτός Συστήματος

Για να μπορέσει να δοκιμαστεί το σύστημα όπως έχει αναφερθεί και κατά την σχεδίασή του, πέρα από τις οντότητες που το αποτελούν απαιτούνται ακόμα και οι:

1. Οντότητες Παρακολούθησης Πόρων
2. Οντότητες Δοκιμής Πόρων
3. Οντότητες Επιδιόρθωσης Πόρων
4. Οντότητα Παροχής Πληροφορίας Τοπολογίας Δικτύου
5. Οντότητα Παροχής Πληροφορίας Τοπολογίας Εικονικών Μονοπατιών Σύνδεσης

Στο παράρτημα Β της εργασίας παραθέτονται οι διεπαφές των οντοτήτων (παρακολούθησης, δοκιμής και επιδιόρθωσης πόρων) καθώς και κάποιες προτάσεις και παρατηρήσεις για την σχεδίαση του συστήματος, στο επίπεδο EML, σύμφωνα με την TINA.

6.2.1 Οντότητες Παρακολούθησης Πόρων

Για οντότητες παρακολούθησης πόρων χρησιμοποιήθηκαν οι οντότητες οι οποίες είχαν σχεδιαστεί και υλοποιηθεί στην πρώτη φάση του συστήματος REFORM και οι οποίες βρίσκονται στο EML επίπεδο. Οι οντότητες αυτές για αλγόριθμο παρακολούθησης τόσο των γραμμών όσο και των εικονικών μονοπατιών χρησιμοποιούν μία προσομοίωση των ροών Λειτουργίας, Διοίκησης και Διατήρησης [OAM] (*OAM flows*).

Επίσης επειδή οι οντότητες αυτές δεν βρίσκονταν στο DPE τους προστέθηκε ο κατάλληλος κώδικας τόσο για να υλοποιούν μια διεπαφή μέσω της οποίας θα προσφέρουν την λειτουργία τους όσο και για να εγγράφουν την διεπαφή αυτή στην υπηρεσία υνομασίας ώστε να μπορούν να τους εντοπίζουν οι ενδιαφερόμενες οντότητες.

6.2.2 Οντότητες Δοκιμής Πόρων

Για οντότητες δοκιμής πόρων χρησιμοποιήθηκαν και πάλι οι οντότητες παρακολούθησης οι οποίες είχαν σχεδιαστεί και υλοποιηθεί στην πρώτη φάση του συστήματος REFORM.

Επειδή οι οντότητες αυτές δεν ήταν για δοκιμή τους προστέθηκε ο κατάλληλος κώδικας για να προσομοιώνουν την δοκιμή ως εξής: Όποτε τους ζητηθεί δοκιμή αρχίζουν ουσιαστικά την διαδικασία παρακολούθησης και αν αυτό καταστεί δυνατόν επιστρέφουν επιτυχημένη δοκιμή (ο πόρος λειτουργεί) και σταματάνε την διαδικασία της παρακολούθησης, ενώ αν δεν καταστεί δυνατόν επιστρέφουν αποτυχημένη δοκιμή (ο πόρος δεν λειτουργεί).

Επίσης και πάλι τους προστέθηκε ο κατάλληλος κώδικας τόσο για να υλοποιούν μια διεπαφή μέσω της οποίας θα προσφέρουν την λειτουργικότητά τους όσο και για να εγγράφουν την διεπαφή αυτή στην υπηρεσία ονομασίας ώστε να μπορούν να τους εντοπίζουν οι ενδιαφερόμενες οντότητες (Εξυπηρετητής Δοκιμών).

6.2.3 Οντότητες Επιδιόρθωσης Πόρων

Για οντότητες επιδιόρθωσης πόρων χρησιμοποιήθηκαν οι οντότητες οι οποίες και πάλι είχαν σχεδιαστεί και υλοποιηθεί στην πρώτη φάση του συστήματος REFORM και οι οποίες βρίσκονται στο EML επίπεδο.

Οι οντότητες αυτές παρέχουν αυτόματη επιδιόρθωση των σφαλμάτων αφού με την ανίχνευση ενός σφάλματος οι οντότητες παρακολούθησης τους στέλνουν μία ειδοποίηση και αυτές κάνουν την επιδιόρθωση με βάση τον πόρο που αναπληρεί (backup resource) τον πόρο στον οποίο παρουσιάζεται το σφάλμα και ο οποίος έχει προκαθοριστεί. Εμείς προσθέσαμε τον απαραίτητο κώδικα ώστε όταν στέλνει την ειδοποίση η οντότητα παρακολούθησης αυτή να περιλαμβάνει την κλαση επιδιόρθωσης και τον αναπληρωματικό πόρο (τα οποία της έχει δώσει ο Διαχειριστής Ειδοποιήσεων) και να χρησιμοποιεί αυτά για να κάνει την επιδιόρθωση. Επίσης τους προστέθηκε και μία διεπαφή στην περίπτωση που πρέπει ο Συντονιστής να προκαλέσει αυτός την επιδιόρθωση, καθώς και τον απαραίτητο κώδικα για να εγγράφουν τη διεπαφή αυτή στην υπηρεσία ονομασίας.

6.2.4 Οντότητα Παροχής Πληροφορίας Τοπολογίας Δικτύου

Για οντότητα που παρέχει την πληροφορία της τοπολογίας του δικτύου χρησιμοποιήθηκε η οντότητα ConfM-NM (Configuration Management - Network Map) που είχε σχεδιαστεί και υλοποιηθεί στην πρώτη φάση του συστήματος REFORM και η οποία βρίσκεται στο NML επίπεδο. Επίσης προστέθηκε ο κώδικας για την δημιουργία μίας διεπαφής ειδοποιήσεων καθώς και για να εγγράφει τις διεπαφές της στην υπηρεσία ονομασίας.

6.2.5 Οντότητα Διαχείρισης Συνδέσεων

Για οντότητα Διαχείρισης συνδέσεων χρησιμοποιήθηκε η οντότητα ConfM-CM (Configuration Management - Connection Management), η οποία είχε σχεδιαστεί και υλοποιηθεί στο σύστημα REFORM. Στην οντότητα αυτή προσθέσαμε τον απαραίτητο κώδικα ώστε μέσω της υπηρεσίας ονομασίας να βρίσκει τη διεπαφή του Συντονιστή για αρχικοποίηση/σταμάτημα διαχείρισησ και να την καλεί όταν δημιουργεί/σβήνει ένα εικονικό μονοπάτι σύνδεσης.

6.3 Υλοποίηση Οντοτήτων

6.3.1 Ο Συντονιστής Διαχείρισης Λαθών (FM Coordinator)

6.3.1.1 Εσωτερική Υλοποίηση

Για την υλοποίηση του Συντονιστή χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++. Ο Συντονιστής υλοποιήθηκε σαν μια διεργασία πολλαπλών νημάτων. Σε κάθε κλήση τόσο της διεπαφής λειτουργιών όσο και της διεπαφής ειδοποιήσεών για την εξυπηρέτηση της κλήσης αναλαμβάνει ένα διαφορετικό νήμα. Επίσης στην διαδικασία εντοπισμού της πρωταρχικής αιτίας του σφάλματος όταν απαιτούνται δοκιμές αυτές γίνονται η κάθε μία από διαφορετικό νήμα για να βελτιωθεί ο χρόνος αφού οι δοκιμές είναι σύγχρονες κλήσεις. Τέλος επειδή διαφορετικά νήματα έχουν πρόσβαση σε κοινές οντότητες πληροφορίας (βλέπε παρακάτω) προστέθηκε ο κατάλληλος κώδικας για να είναι η πληροφορία ασφαλείς (*MT Safe*).

6.3.1.2 Χρήση Οντοτήτων Πληροφορίας

Για την πραγματοποίηση των λειτουργιών του ο Συντονιστής χρησιμοποιεί τις εξής οντότητες πληροφορίας:

1. Την τοπολογία του δικτύου *NetworkTopologyMap* την οποία παίρνει κατά την αρχικοποίηση (τοπολογία φυσικού επιπέδου) από το σύστημα Διαχείρισης Διαμόρφωσης, ενημερώνει από την λήψη πληροφορίας (επιπέδου εικονικών μονοπατιών σύνδεσης VPConnections) από το σύστημα Διαχείρισης Συνδέσεων και χρησιμοποιεί τόσο για να ξεκινήσει και σταματήσει διαδικασίες διαχείρισης σφαλμάτων (παρακολούθηση, δοκιμή, επιδιόρθωση κλπ) όσο και για τον εντοπισμό της πρωταρχικής αιτίας σφάλματος (στην διαδικασία εντοπισμού).
2. Μία λίστα στην οποία φυλάει τις ειδοποιήσεις που έρχονται για να μπορεί να τις εξετάζει κατά την διάρκεια της διαδικασίας εντοπισμού και να βγάζει συμπεράσματα.
3. Οντότητες *NetworkRiskMap* τις οποίες παίρνει από τον Διαχειριστή, συμπληρώνει με την πληροφορία (*RUR*) που υπολογίζει με βάση την τοπολογία και τις επιστρέφει όταν κληθεί η κατάλληλη διεπαφή του.

6.3.1.3 Αλληλεπιδράσεις

Ξεκινώντας ο Συντονιστής εγγράφει τις διεπαφές του στην υπηρεσία ονομάτων, βρίσκει από την υπηρεσία ονομάτων την διεπαφή ειδοποιήσεων της Γραφικής Διεπαφής Χρήστη και την διεπαφή τοπολογίας του NTCM και περιμένει κλήση στην διεπαφή αρχικοποίησής του.

Μόλις κληθεί η διεπαφή αρχικοποίησής του, από την Γραφική Διεπαφή Χρήστη, καλεί την διεπαφή τοπολογίας του NTCM, παίρνει την τοπολογία, την επιστρέφει στη Γραφική Διεπαφή Χρήστη, την ελέγχει, παίρνει την διεπαφή λειτουργιών του Διαχειριστή Σφαλμάτων και ξεκινάει την παρακολούθηση καλώντας την για κάθε πόρο (γραμμή).

Κάθε φορά που δέχεται κλήση στην διεπαφή διαχείρισης πόρου, και του παρέχεται η πληροφορία για ένα καινούργιο εικονικό μονοπάτι σύνδεσης, ενημερώνει την πληροφορία τοπολογίας που διατηρεί, αναθέτει την κατάλληλη κλάση επιδιόρθωσης στον πόρο, ξεκινάει την παρακολούθηση μέσω της κατάλληλης διεπαφής του Διαχειριστή ειδοποιήσεων (στην διεπαφή αυτή δίνει σαν όρισμα και την διεπαφή ειδοποιήσεών του, τον τύπο της ειδοποίησης που τον ενδιαφέρει, τον πόρο και τα χαρακτηριστικά παρακολούθησης) και στέλνει την πληροφορία του καινούργιου πόρου μέσω της κατάλληλης διεπαφής στην Γραφική Διεπαφή Χρήστη ώστε να παρουσιάζει γραφικά την κατάσταση σφάλματος του πόρου.

Μόλις λάβει κάποια ειδοποίηση ξεκινάει ένα νήμα που αναλαμβάνει την διαδικασία εντοπισμού (κοίτα επόμενη παράγραφο), προωθεί την ειδοποίηση στη Γραφική Διεπαφή Χρήστη (καθώς και την πρωταρχική αιτία αν την εντοπίσει) και βρίσκει τις κατάλληλες οντότητες επιδιόρθωσης του/των πόρου/πόρων που έγινε το σφάλμα και τους δίνει την απαραίτητη πληροφορία για την επιδιόρθωση (πόρος αντικατάστασης, κλάση επιδιόρθωσης).

6.3.1.4 Διαδικασία Εντοπισμού Πρωταρχικής Αιτίας Σφάλματος

Η διαδικασία εντοπισμού έχει ως εξής: Με το που λάβει μια ειδοποίηση ξεκινάει ένα νήμα που αναλαμβάνει να κάνει τις δοκιμές στους πόρους που απαιτείται σύμφωνα με την τοπολογία (κάθε δοκιμή γίνεται από διαφορετικό νήμα για παραλληλοποίηση), παράλληλα κοιτάει την λίστα ειδοποιήσεων αν του έχει έρθει κάποια ειδοποίηση σφάλματος με την οποία βγάζει συμπέρασμα που βοηθάει στον εντοπισμό χωρίς να περιμένει το τέλος των δοκιμών. Να δεν μπορεί περιμένει το τέλος των δοκιμών για να βγάλει το τελικό αποτέλεσμα.

Αν έρθει ειδοποίηση για την οποία από την τοπολογία ή προηγούμενο αποτέλεσμα εντοπισμού βγαίνει το συμπέρασμα ότι έχει ήδη εντοπιστεί το λάθος (π.χ. δεν έχει έρθει ακόμα ειδοποίηση επιδιόρθωσης πόρου από τον οποίο εξαρτάται ο πόρος για τον οποίο ήταν η ειδοποίηση ή βρίσκεται υπό εντοπισμό (π.χ. έχει ήδη ξεκινήσει διαδικασία εντοπισμού για πόρο που χρησιμοποιεί αυτόν τον πόρο), τότε δεν ξεκινάει διαδικασία εντοπισμού για αυτήν την ειδοποίηση.

6.3.2 Ο Διαχειριστής Ειδοποιήσεων Διαχείρισης Λαθών (FM Alarm Manager)

6.3.2.1 Εσωτερική Υλοποίηση

Για την υλοποίηση του Διαχειριστή Ειδοποιήσεων χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++. Ο Διαχειριστής υλοποιήθηκε και αυτός σαν μια διεργασία πολλαπλών νημάτων. Σε κάθε κλήση τόσο της διεπαφής λειτουργιών όσο και της διεπαφής ειδοποιήσεων για την εξυπηρέτηση της κλήσης αναλαμβάνει ένα διαφορετικό νήμα. Επειδή και στον Διαχειριστή διαφορετικά νήματα έχουν πρόσβαση σε κοινές οντότητες πληροφορίας (βλέπε παρακάτω) προστέθηκε ο κατάλληλος κώδικας για να είναι η πληροφορία ασφαλείς (*MT Safe*).

6.3.2.2 Χρήση Οντοτήτων Πληροφορίας

Για την πραγματοποίηση των λειτουργιών του ο Διαχειριστής χρησιμοποιεί τις εξής οντότητες πληροφορίας:

1. Μία οντότητα *NodeFaultManagementDomain* στην οποία αποθηκεύει τους *NodeAlarmableResources* που παρακολουθούνται, καθώς και τα στοιχεία που τους αφορά (στοιχεία στατιστικών σφαλμάτων, στοιχεία ενδιαφερόμενων προορισμών ειδοποιήσεων, αποθήκευση ειδοποιήσεων κλπ).
2. Μία οντότητα *LinkFaultManagementDomain* στην οποία αποθηκεύει τους *LinkAlarmableResources* που παρακολουθούνται, καθώς και τα στοιχεία που τους αφορά (στοιχεία στατιστικών σφαλμάτων, στοιχεία ενδιαφερόμενων προορισμών ειδοποιήσεων, αποθήκευση ειδοποιήσεων κλπ).
3. Μία οντότητα *VPCFaultManagementDomain* στην οποία αποθηκεύει τους *VPCAlarmableResources* που παρακολουθούνται, καθώς και τα στοιχεία που τους αφορά (στοιχεία στατιστικών σφαλμάτων, στοιχεία ενδιαφερόμενων προορισμών ειδοποιήσεων, αποθήκευση ειδοποιήσεων κλπ).
4. Οντότητες *NetworkRiskMap* τις οποίες δημιουργεί, γεμίζει με την πληροφορία που έχει και επιστρέφει όταν κληθεί η κατάλληλη διεπαφή του.
5. Οντότητες *AlarmSummaryReport* τις οποίες δημιουργεί, γεμίζει με την πληροφορία που έχει και επιστρέφει όταν κληθεί η κατάλληλη διεπαφή του.

6.3.2.3 Αλληλεπιδράσεις

Ξεκινώντας ο Διαχειριστής Ειδοποιήσεων εγγράφει τις διεπαφές του στην υπηρεσία ονομάτων.

Μόλις κληθεί η διεπαφή παρακολούθησης πόρου που έχει, δημιουργεί την οντότητα που εσωτερικά θα αντιπροσωπεύει τον πόρο (*NodeAlarmableResource*, *LinkAlarmableResource*, *VPCAlarmableResource*), αποθηκεύει σε αυτήν την πληροφορία που παίρνει (τύπος ειδοποίησης, κλάση παρακολούθησης, προορισμός ειδοποίησης κλπ), βρίσκει.από την πληροφορία του πόρου μέσω της Υπηρεσίας Ονομασίας τις οντότητες παρακολούθησης και ξεκινάει την παρακολούθηση.

Μόλις κληθεί η διεπαφή για σταμάτημα της παρακολούθησης, εξετάζει την ταυτότητα της οντότητας που την κάλεσε για λόγους ασφάλεια και αν δεν υπάρχει άλλος ενδιαφερόμενος για τον συγκεκριμένο πόρο βρίσκει.από την πληροφορία του πόρου μέσω της Υπηρεσίας Ονομασίας τις οντότητες παρακολούθησης και σταματάει την παρακολούθηση, αλλιώς (αν υπάρχει και άλλος ενδιαφερόμενος) απλώς σβήνει τον ενδιαφερόμενο από την λίστα των ενδιαφερομένων για τις ειδοποιήσεις του πόρου (*AlarmControl*).
Μόλις λάβει κάποια ειδοποίηση, την φιλτράρει (ελέγχοντας αν ήδη την έχει πάρει), την αποθηκεύει στο ημερολόγιο (*LOG*) του πόρου, ελέγχει για τους ενδιαφερόμενους και τους στέλνει την ειδοποίηση. Αν η κλήση στην διεπαφή κάποιου ενδιαφερόμενου αποτύχει αυτός βγαίνει από την λίστα των ενδιαφερομένων.

Μόλις κληθεί η διεπαφή για αναφορά συνόλου ειδοποιήσεων για έναν πόρο δημιουργεί μια οντότητα *AlarmSummaryReport* την γεμίζει με τα δεδομένα που έχει αποθηκεύσει από τις ειδοποιήσεις για τον πόρο αυτόν και την επιστρέφει.

Μόλις κληθεί η διεπαφή για Χάρτη Στατιστικών δημιουργεί μια οντότητα *NetworkRiskMap* την γεμίζει με τα δεδομένα που έχει υπολογίσει από τις ειδοποιήσεις για τον πόρο αυτόν και την επιστρέφει (δεν βάζει τιμή στο πεδίο *RUR,* το πεδίο αυτό παίρνει τιμή από τον Συντονιστή).

6.3.3 Ο Εξυπηρετητής Δοκιμών Διαχείρισης Λαθών (FM Testing Server)

6.3.3.1 Εσωτερική Υλοποίηση

Για την υλοποίηση του Εξυπηρετητή Δοκιμών χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++. Ο Εξυπηρετητής υλοποιήθηκε και αυτός σαν μια διεργασία πολλαπλών νημάτων. Σε κάθε κλήση της διεπαφής λειτουργιών για την εξυπηρέτηση της κλήσης αναλαμβάνει ένα διαφορετικό νήμα. Επειδή και στον Εξυπηρετητή διαφορετικά νήματα έχουν πρόσβαση σε κοινές οντότητες πληροφορίας (βλέπε παρακάτω) προστέθηκε ο κατάλληλος κώδικας για να είναι η πληροφορία ασφαλείς (MT Safe).

6.3.3.2 Χρήση Οντοτήτων Πληροφορίας

Για την πραγματοποίηση των λειτουργιών του ο Διαχειριστής χρησιμοποιεί τις εξής οντότητες πληροφορίας:

1. Μία οντότητα *NodeFaultManagementDomain* στην οποία αποθηκεύει τους *NodeTestableResources* που δοκιμάζονται, καθώς και τα στοιχεία που τους αφορά (αποθήκευση δοκιμών κλπ).
2. Μία οντότητα *LinkFaultManagementDomain* στην οποία αποθηκεύει τους *LinkTestableResources* που δοκιμάζονται, καθώς και τα στοιχεία που τους αφορά (αποθήκευση δοκιμών κλπ).
3. Μία οντότητα *VPCFaultManagementDomain* στην οποία αποθηκεύει τους *VPCTestableResources* που δοκιμάζονται, καθώς και τα στοιχεία που τους αφορά (αποθήκευση δοκιμών κλπ).
4. Οντότητες *TestSummaryReport* τις οποίες δημιουργεί, γεμίζει με την πληροφορία που έχει και επιστρέφει όταν κληθεί η κατάλληλη διεπαφή του.

6.3.3.3 Αλληλεπιδράσεις

Ξεκινώντας ο Εξυπηρετητής Δοκιμών εγγράφει τις διεπαφές του στην υπηρεσία ονομάτων.

Μόλις κληθεί η διεπαφή δοκιμής πόρου που έχει, δημιουργεί την οντότητα που εσωτερικά θα αντιπροσωπεύει τον πόρο (*NodeTestableResource*, *LinkTestableResource*, *VPCTestableResource*), αποθηκεύει σε αυτήν την πληροφορία που παίρνει (τύπος δοκιμής, στοιχεία πόρου κλπ), βρίσκει.από την πληροφορία του πόρου μέσω της Υπηρεσίας Ονομασίας τις οντότητες δοκιμής και ξεκινάει την παρακολούθηση.

Μόλις κληθεί η διεπαφή για αναφορά συνόλου δοκιμών για έναν πόρο δημιουργεί μια οντότητα *TestSummaryReport* την γεμίζει με τα δεδομένα που έχει αποθηκεύσει από τις δοκιμές για τον πόρο αυτόν και την επιστρέφει.

6.3.4 Η Γραφική Διεπαφή Χρήστη (FM Graphical User Interface)

6.3.4.1 Εσωτερική Υλοποίηση

Για την υλοποίηση της Γραφικής Διεπαφής Χρήστη χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java λόγω της ευκολίας που παρέχει στην απεικόνιση γραφικών. Η Γραφική Διεπαφή Χρήστη υλοποιήθηκε σαν μια διεργασία ενός νήματος.

6.3.4.2 Χρήση Οντοτήτων Πληροφορίας

Για την πραγματοποίηση των λειτουργιών της η Γραφική Διεπαφή Χρήστη χρησιμοποιεί τις εξής οντότητες πληροφορίας:

1. Μία οντότητα *NetworkTopologyMap* την οποία χρησιμοποιεί για να δείξει γραφικά τους πόρους του δικτύου και την κατάσταση σφάλματός τους στην μορφή της τοπολογίας του.
2. Μία οντότητα *NetworkRiskMap* την οποία παίρνει με μια κλήση στην κατάλληλη διεπαφή του Συντονιστή και την δείχνει γραφικά.

6.3.4.3 Αλληλεπιδράσεις

Ξεκινώντας η Γραφική Διεπαφή Χρήστη εγγράφει την διεπαφή ειδοποίησης του στην υπηρεσία ονομάτων.

Παρέχει λειτουργία αρχικοποίησης/επανεκκίνησης μέσω της οποίας καλείται η διεπαφή αρχικοποίησης του Συντονιστή και ξεκινά/επαναξεκινά το σύστημα.

Παρέχει λειτουργία γραφικής αναπαράστασης του Χάρτη των Στατιστικών μέσω της οποίας καλείται η διεπαφή του Χάρτη Στατιστικών του Συντονιστή, αποκτάται ο χάρτης και δείχνεται με γραφικό τρόπο.

Μόλις λάβει κάποια ειδοποίηση ενημερώνει την γραφική διεπαφή που έχει και που δείχνει την τοπολογία του δικτύου και την κατάσταση σφάλματος των πόρων του.

7 Δοκιμές Συστήματος

Στο κεφάλαιο αυτό παρουσιάζονται τέσσερις δοκιμές του συστήματος. Οι δοκιμές γίνονται σε δύο διαφορετικά δίκτυα τόσο όσον αφορά την φυσική τοπολογία όσο και την τοπολογία επιπέδου εικονικών μονοπατιών σύνδεσης (VPCs). Επίσης οι δοκιμές γίνονται με διαφορετικά σενάρια σφαλμάτων, δηλαδή με διαφορετική ακολουθία σφαλμάτων καθώς και με διαφορετικό συσχετισμό.

Οι τρεις από τις δοκιμές αφορούν σενάριο σφάλματος και δίνεται το αποτέλεσμα του αλγορίθμου εντοπισμού πρωταρχική αιτίας σφάλματος καθώς και η απεικόνιση του δικτύου στην Γραφική Διεπαφή Χρήστη (GUI). Η τέταρτη δοκιμή αφορά ένα απλό σενάριο ακολουθίας σφαλμάτων και τα παραγόμενα στατιστικά, τα οποία λόγο του περιορισμένου χρόνου (αφού κανονικά το σύστημα έπρεπε να τρέχει για βδομάδες), έχουν πολλαπλασιαστεί με συντελεστές οι οποίοι προέρχονται (για τις επιδιορθώσεις) από πραγματικές μετρήσεις του συστήματος REFORM.

Λόγω του ότι το σύστημα δεν έχει ενωθεί ακόμα με το EML επίπεδο του συστήματος REFORM, η λειτουργικότητα του επιπέδου αυτού προσομοιώνεται. Δηλαδή για αρχικοποίηση των διαδικασιών παρακολούθησης δεν καλούνται οι οντότητες του συστήματος REFORM αλλά δικές μας οντότητες, που δεν κάνουν παρακολούθηση, και το σενάριο σφαλμάτων γίνεται μέσω μιας γεννήτριας ειδοποιήσεων σφαλμάτων. Επίσης λόγω μη δυνατότητας προς το παρόν για χρησιμοποίηση της Διαχείρισης Σύνδεσης η τοπολογία του επιπέδου εικονικών μονοπατιών διαβάζεται από αρχείο.

Η πρώτη δοκιμή που παρουσιάζεται αφορά ένα δίκτυο ATM με 5 κόμβους, 8 φυσικές γραμμές και 4 εικονικά μονοπάτια σύνδεσης, όπου συμβαίνει ένα σε μία φυσική γραμμή, από την οποία παιρνάνε 2 εικονικά μονοπάτια, και σε 1 εικονικό μονοπάτι.

Η δεύτερη δοκιμή γίνεται στο ίδιο δίκτυο με την πρώτη αλλά τώρα συμβαίνει σφάλμα σε μία γραμμή, από την οποία παιρνάει ένα εικονικό μονοπάτι, και σε δύο εικονικά μονοπάτια.

Η τρίτη δοκιμή γίνεται σε ένα δίκτυο ATM με 4 κόμβους, 5 φυσικές γραμμές και 4 εικονικά μονοπάτια σύνδεσης όπου συμβαίνει σφάλμα κόμβου.

Τέλος η τέταρτη δοκιμή, των στατιστικών, γίνεται στο ίδιο δίκτυο με την τρίτη και παράγονται και παρουσιάζονται με γραφικό τρόπο τα στατιστικά μετά από την ακολουθία μίας σειράς σφαλμάτων.

7.1Δοκιμή σε Δίκτυο 5 Κόμβων και 8 Φυσικών γραμμών (σφάλμα γραμμής και εικονικού μονοπατιού σύνδεσης)

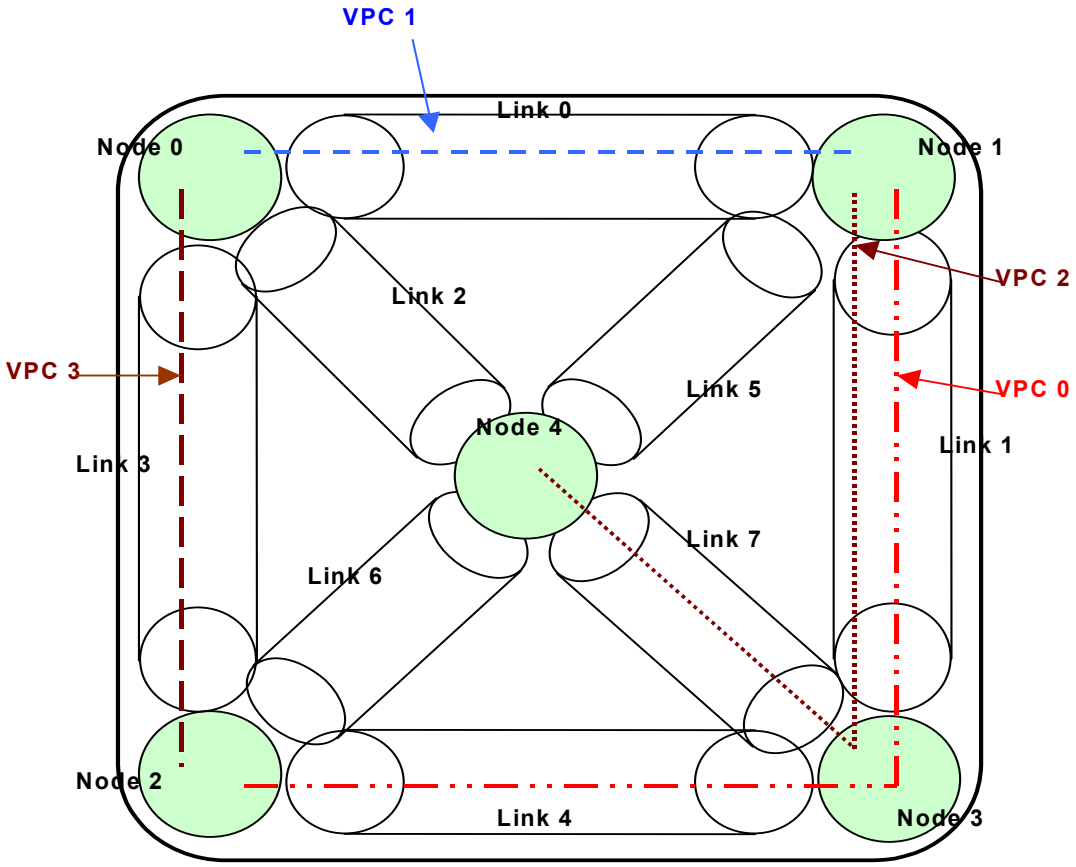
Η δοκιμή αυτή έγινε σε ένα ATM δίκτυο 5 κόμβων (0, 1, 2, 3, 4) και 8 φυσικών γραμμών (0, 1, 2, 3, 4, 5, 6, 7) το οποίο φαίνεται στο Σχήμα 7-1. Όπως φαίνεται στο σχήμα έχουν δημιουργηθεί 4 εικονικά μονοπάτια σύνδεσης (0, 1, 2, 3).

Στο Σχήμα 7-2 βλέπουμε την τοπολογία του δικτύου όπως φαίνεται στην Γραφική Διεπαφή Χρήστη αμέσως μετά την αρχικοποίηση του συστήματος. Τα βήματα που γίνονται είναι τα εξής:

1. Ο χρήστης του συστήματος δίνει μέσω της Γραφικής Διεπαφής εντολή για αρχικοποίηση.
2. Ο συντονιστής επικοινωνεί με το σύστημα Διαχείρισης Διαμόρφωσης από το οποίο λαμβάνει την φυσική τοπολογία.
3. Έπειτα διαβάζεται η τοπολογία του επιπέδου εικονικών μονοπατιών.

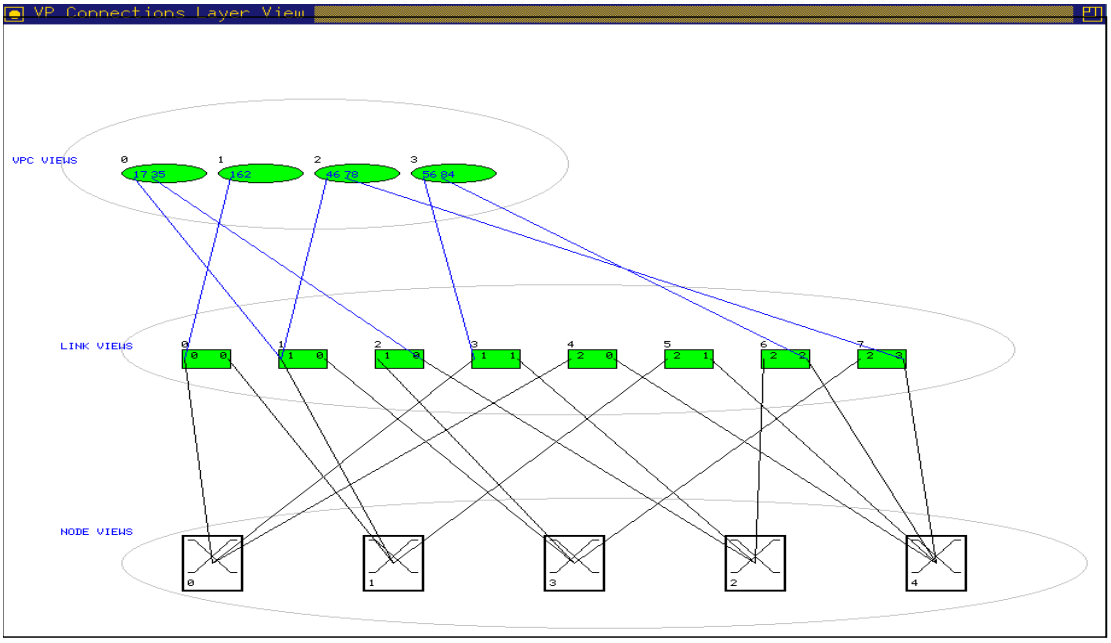
4. Για κάθε φυσική γραμμή και εικονικό μονοπάτι ο Συντονιστής επικοινωνεί με τον Διαχειριστή Ειδοποιήσεων, καλώντας την κατάλληλη διεπαφή (monitorResource()) για να αρχίσει η παρακολούθηση.
5. Από την πληροφορία που περνάει ως παράμετρο στην διεπαφή που καλεί ο Συντονιστής, ο Διαχειριστής Σφαλμάτων, μέσω της υπηρεσίας ονομασίας εντοπίζει της οντότητες παρακολουθήσεως και καλεί τις διεπαφές τους για να αρχίσει η διαδικασία παρακολούθησης.

Το σενάριο σφάλματος έχει ως εξής: κάποια χρονική στιγμή συμβαίνει σφάλμα στην γραμμή 1 και στο VPC 3. Οι ειδοποιήσεις που δημιουργούνται αφορούν την γραμμή 1 και τα VPCs 0, 2 και 3 (αφού τα 0 και 2 χρησιμοποιούν την γραμμή 1).

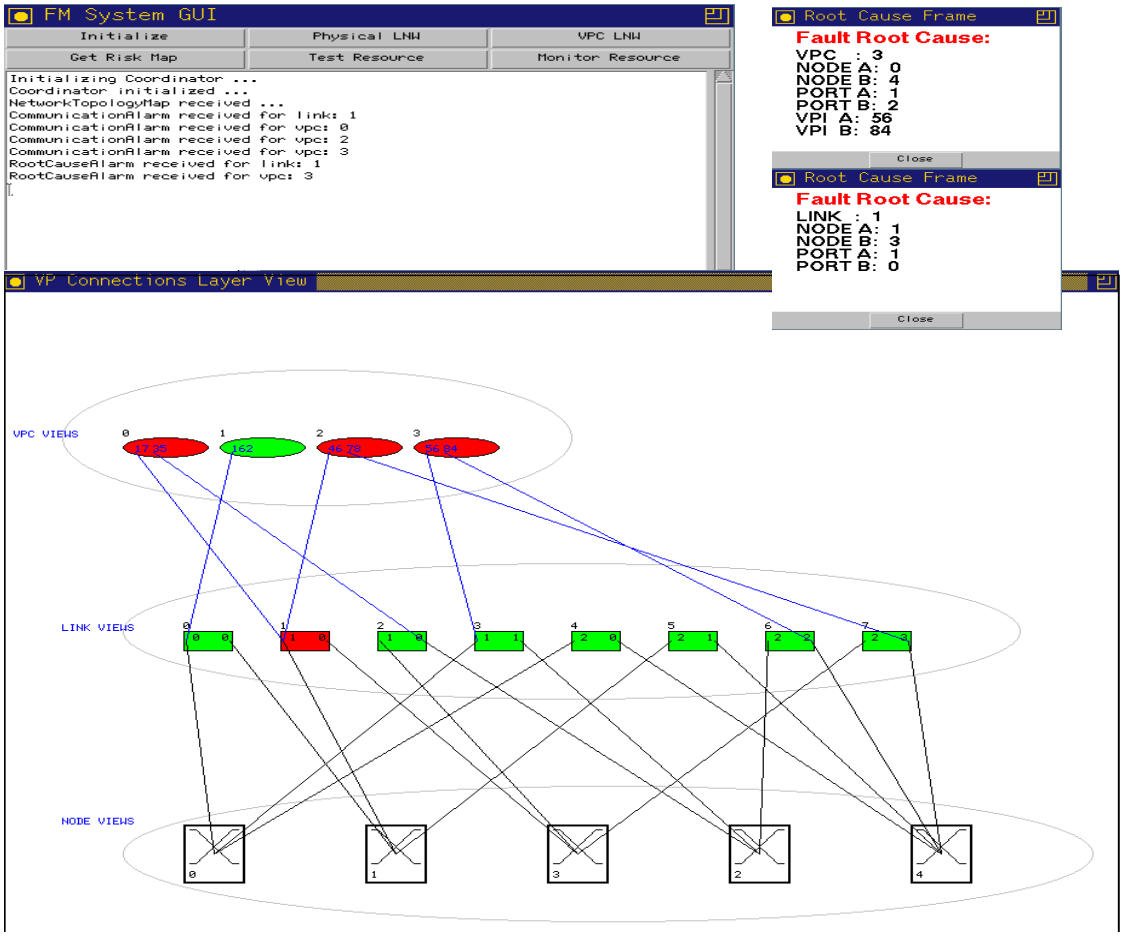


Σχήμα 7-1 Δίκτυο πρώτης και δεύτερης δοκιμής (5 κόμβοι, 7 φυσικές γραμμές)

Οι ειδοποιήσεις λαμβάνονται, αρχίζει ο αλγόριθμος εντοπισμού της πρωταρχικής αιτίας του σφάλματος και εντοπίζονται σωστά οι δύο πρωταρχικές αιτίες σφάλματος. Η μόνη δοκιμή που χρειάστηκε να γίνει ήταν στους κόμβους 1 και 3 (αφού η γραμμή εξαρτάται από αυτούς) όπου και ήταν πετυχημένες (προσομοιώθηκαν). Στο Σχήμα 7-3 φαίνονται τα αποτελέσματα στην γραφική διεπαφή.



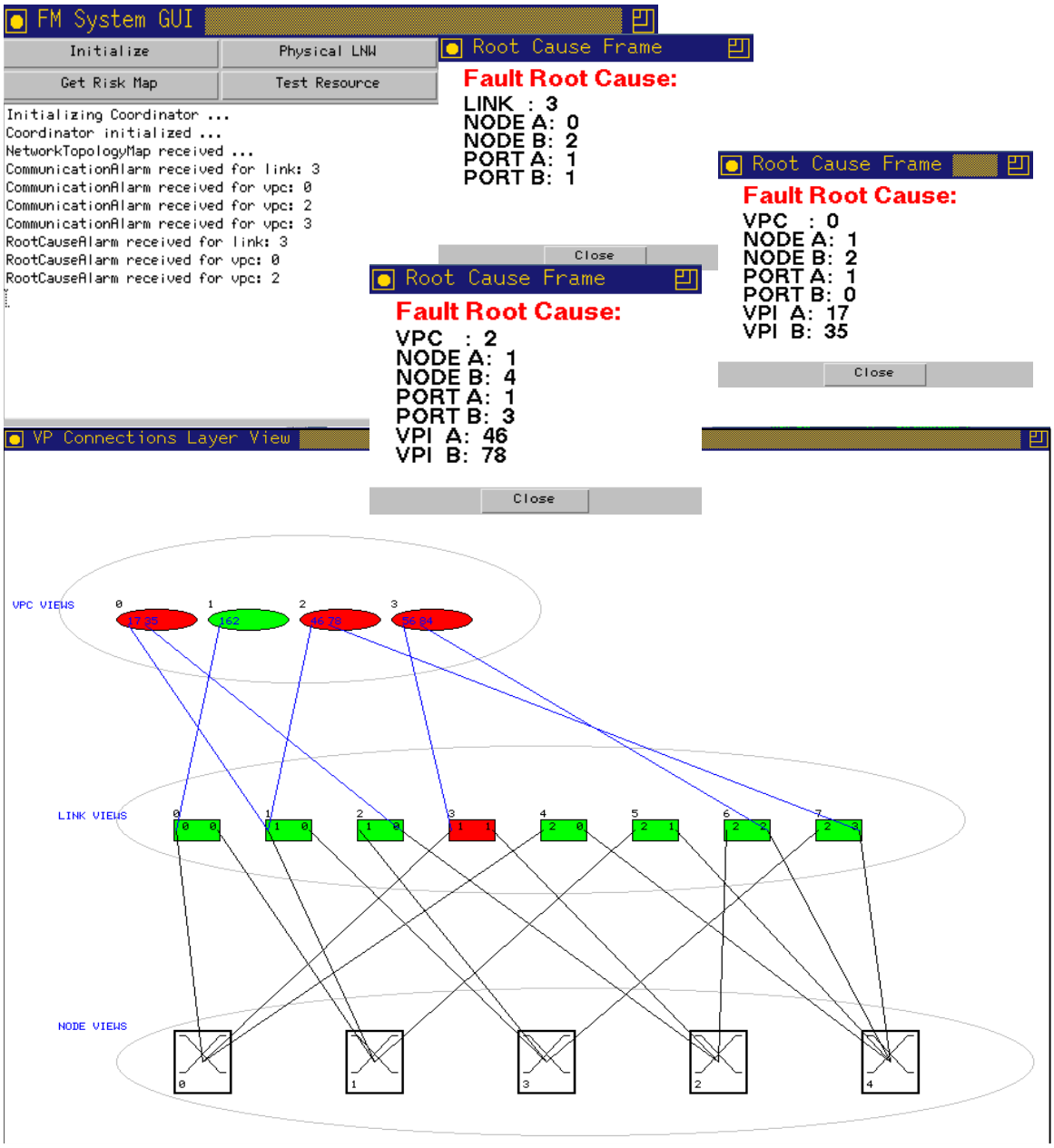
Σχήμα 7-2 Γραφική Διεπαφή αμέσως μετά το διάβασμα της τοπολογίας της πρώτης και δεύτερης δοκιμής



Σχήμα 7-3 Παρουσίαση αποτελέσματος τόσο της κατάσταση των πόρων του δικτύου όσο και της πρωταρχικής αιτίας των σφαλμάτων για την πρώτη δοκιμή

7.2Δοκιμή σε Δίκτυο 5 Κόμβων και 8 Φυσικών γραμμών (σφάλμα γραμμής και δύο εικονικών μονοπατιών σύνδεσης)

Η δοκιμή αυτή έγινε στο ίδιο ακριβώς δίκτυο ATM της πρώτης δοκιμής Σχήμα 7-1. Όπως και για την πρώτη δοκιμή έτσι και για αυτήν η τοπολογία του δικτύου όπως φαίνεται στην Γραφική Διεπαφή Χρήστη αμέσως μετά την αρχικοποίηση του συστήματος φαίνεται στο Σχήμα 7-2. Τα βήματα που γίνονται είναι επίσης τα ίδια:



Σχήμα 7-4 Παρουσίαση αποτελέσματος τόσο της κατάσταση των πόρων του δικτύου όσο και της πρωταρχικής αιτίας των σφαλμάτων για την δέυτερη δοκιμή

1. Ο χρήστης του συστήματος δίνει μέσω της Γραφικής Διεπαφής εντολή για αρχικοποίηση.
2. Ο συντονιστής επικοινωνεί με το σύστημα Διαχείρισης Διαμόρφωσης από το οποίο λαμβάνει την φυσική τοπολογία.

3. Έπειτα διαβάζεται η τοπολογία του επιπέδου εικονικών μονοπατιών.
4. Για κάθε φυσική γραμμή και εικονικό μονοπάτι ο Συντονιστής επικοινωνεί με τον Διαχειριστή Ειδοποιήσεων, καλώντας την κατάλληλη διεπαφή (monitorResource()) για να αρχίσει η παρακολούθηση.
5. Από την πληροφορία που περνάει ως παράμετρο στην διεπαφή που καλεί ο Συντονιστής, ο Διαχειριστής Σφαλμάτων, μέσω της υπηρεσίας ονομασίας εντοπίζει της οντότητες παρακολουθήσεως και καλεί τις διεπαφές τους για να αρχίσει η διαδικασία παρακολούθησης.

Εδώ όμως το σενάριο σφάλματος έχει ως εξής: κάποια χρονική στιγμή συμβαίνει σφάλμα στην γραμμή 3 και στα VPC 0 και 2. Οι ειδοποιήσεις που δημιουργούνται αφορούν την γραμμή 3 και τα VPCs 0, 2 και 3 (αφού το 3 χρησιμοποιεί την γραμμή 3).

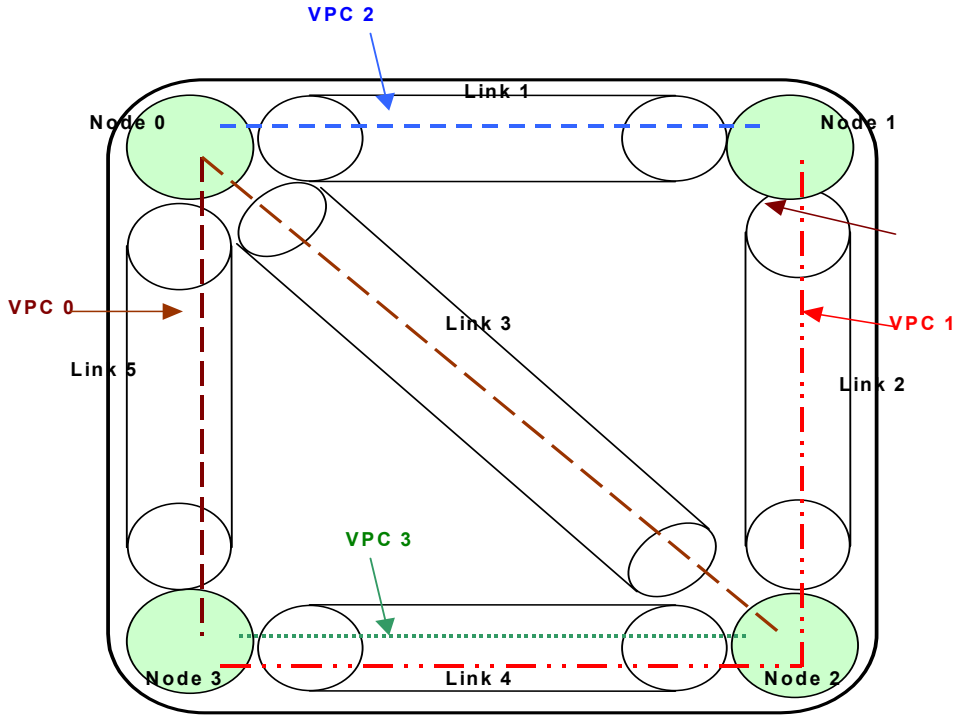
Οι ειδοποιήσεις λαμβάνονται, αρχίζει ο αλγόριθμος εντοπισμού της πρωταρχικής αιτίας του σφάλματος και εντοπίζονται σωστά οι τρεις πρωταρχικές αιτίες σφάλματος. Η μόνες δοκιμές που χρειάστηκε να γίνει ήταν στους κόμβους 0 και 2 (αφού η γραμμή εξαρτάται από αυτούς) 0 όπου και ήταν πετυχημένες (προσομοιώθηκαν). Στο Σχήμα 7-4 φαίνονται τα αποτελέσματα στην γραφική διεπαφή.

7.3Δοκιμή σε Δίκτυο 4 Κόμβων και 5 Φυσικών γραμμών (σφάλμα κόμβου)

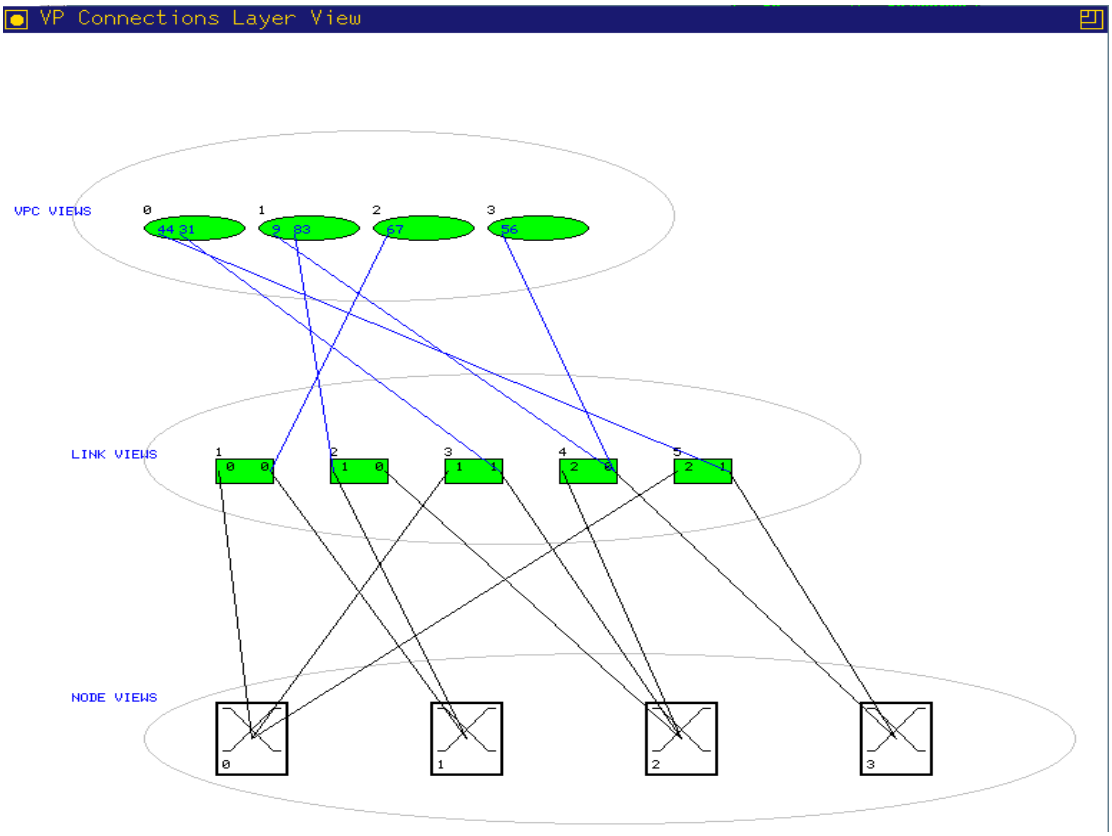
Η δοκιμή αυτή έγινε σε ένα ATM δίκτυο 4 κόμβων (0, 1, 2, 3) και 5 φυσικών γραμμών (1, 2, 3, 4, 5) το οποίο φαίνεται στο Σχήμα 7-5. Όπως φαίνεται στο σχήμα έχουν δημιουργηθεί 4 εικονικά μονοπάτια σύνδεσης (0, 1, 2, 3).

Στο Σχήμα 7-6 βλέπουμε την τοπολογία του δικτύου όπως φαίνεται στην Γραφική Διεπαφή Χρήστη αμέσως μετά την αρχικοποίηση του συστήματος. Τα βήματα που γίνονται είναι τα εξής:

1. Ο χρήστης του συστήματος δίνει μέσω της Γραφικής Διεπαφής εντολή για αρχικοποίηση.
2. Ο συντονιστής επικοινωνεί με το σύστημα Διαχείρισης Διαμόρφωσης από το οποίο λαμβάνει την φυσική τοπολογία.
3. Έπειτα διαβάζεται η τοπολογία του επιπέδου εικονικών μονοπατιών.
4. Για κάθε φυσική γραμμή και εικονικό μονοπάτι ο Συντονιστής επικοινωνεί με τον Διαχειριστή Ειδοποιήσεων, καλώντας την κατάλληλη διεπαφή (monitorResource()) για να αρχίσει η παρακολούθηση.
5. Από την πληροφορία που περνάει ως παράμετρο στην διεπαφή που καλεί ο Συντονιστής, ο Διαχειριστής Σφαλμάτων, μέσω της υπηρεσίας ονομασίας εντοπίζει τις οντότητες παρακολούθησης και καλεί τις διεπαφές τους για να αρχίσει η διαδικασία παρακολούθησης.



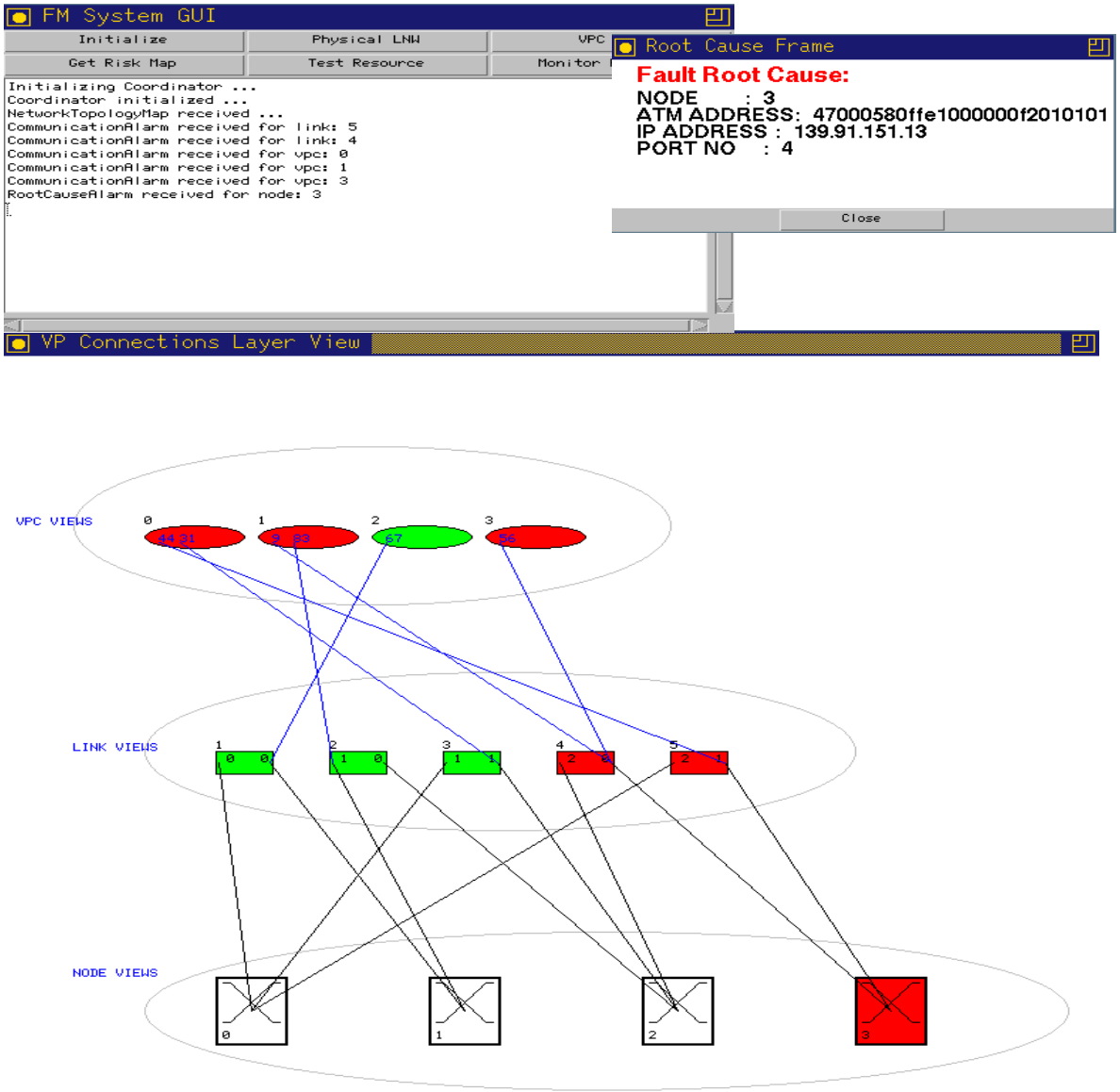
Σχήμα 7-5 Δίκτυο τρίτης και τέταρης δοκιμής (4 κόμβοι, 5 γραμμές 4 VPCs)



Σχήμα 7-6 Γραφική Διεπαφή αμέσως μετά το διάβασμα της τοπολογίας της τρίτης και τέταρτης δοκιμής

Το σενάριο σφάλματος έχει ως εξής: κάποια χρονική στιγμή συμβαίνει σφάλμα στον κόμβο 3. Οι ειδοποιήσεις που δημιουργούνται αφορούν τις γραμμές 4 και 5 καθώς και τα VPCs 0, 1 και 3 (αφού και τα τρία περνάνε από τις γραμμές 4 και 5 οι οποίες έχουν ως ένα άκρο τον κόμβο 3).

Οι ειδοποιήσεις λαμβάνονται, αρχίζει ο αλγόριθμος εντοπισμού της πρωταρχικής αιτίας του σφάλματος και εντοπίζεται σωστά η πρωταρχική αιτία σφάλματος. Η μόνες δοκιμές που χρειάστηκε να γίνουν ήταν στους κόμβους 0, 2, 3 (αφού οι γραμμές εξαρτώνται από αυτούς) όπου και ήταν επιτυχημένες για τους 0 και 2 (προσομοιώθηκαν) ενώ αποτυχημένη για τον 3 (προσομοιώθηκε). Στο Σχήμα 7-7 φαίνονται τα αποτελέσματα στην γραφική διεπαφή.

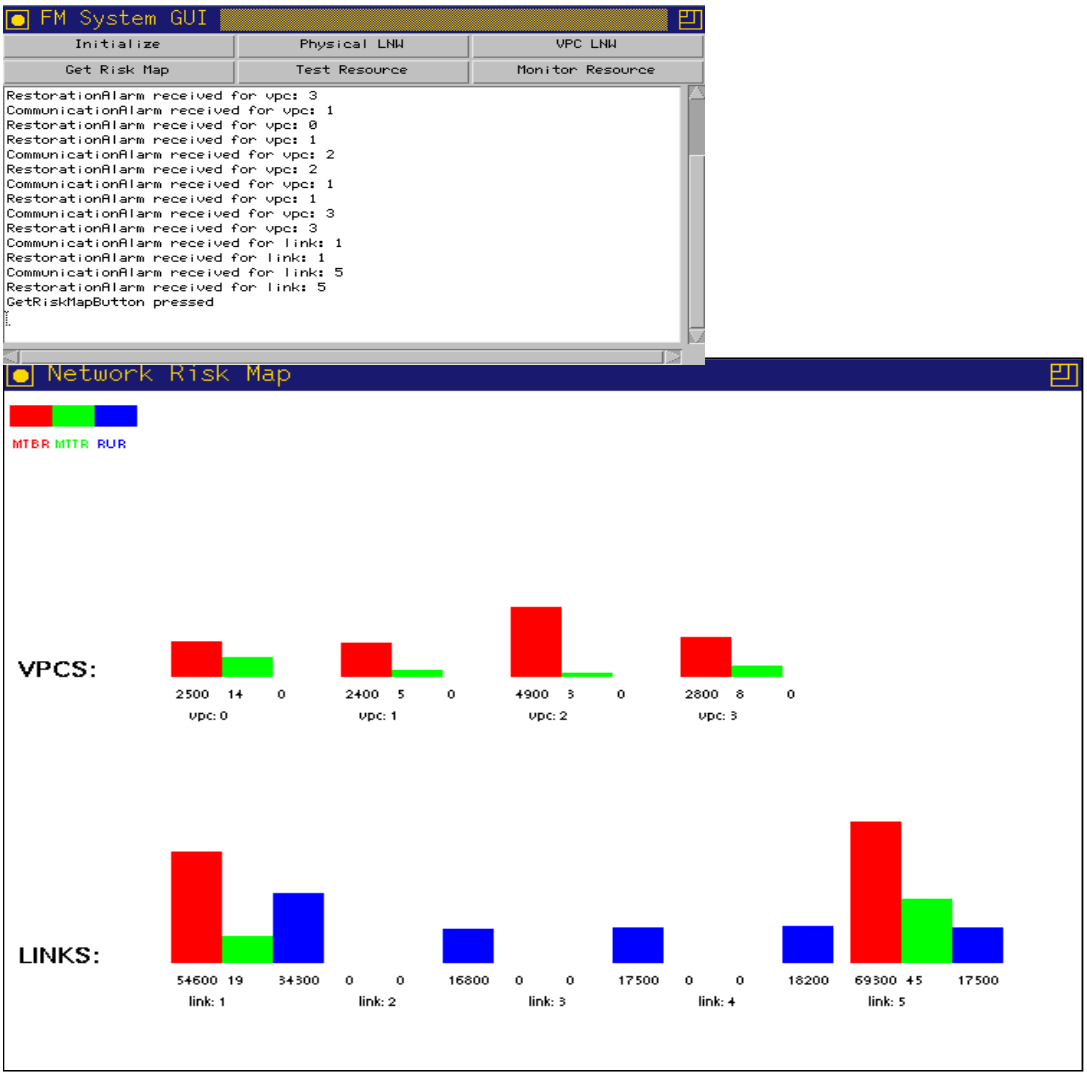


Σχήμα 7-7 Παρουσίαση αποτελέσματος τόσο της κατάσταση των πόρων του δικτύου όσο και της πρωταρχικής αιτίας των σφαλμάτων, για την τρίτη δοκιμή

7.4 Δοκιμή Στατιστικών σε Δίκτυο 4 Κόμβων και 5 Φυσικών γραμμών

Η δοκιμή αυτή έγινε στο ATM δίκτυο της τρίτης δοκιμής (Σχήμα 7-5). Στο Σχήμα 7-6 βλέπουμε την τοπολογία του δικτύου όπως φαίνεται στην Γραφική Διεπαφή Χρήστη αμέσως μετά την αρχικοποίηση του συστήματος. Τα βήματα που γίνονται είναι τα ίδια με την τρίτη δοκιμή:

1. Ο χρήστης του συστήματος δίνει μέσω της Γραφικής Διεπαφής εντολή για αρχικοποίηση.
2. Ο συντονιστής επικοινωνεί με το σύστημα Διαχείρισης Διαμόρφωσης από το οποίο λαμβάνει την φυσική τοπολογία.
3. Έπειτα διαβάζεται η τοπολογία του επιπέδου εικονικών μονοπατιών.
4. Για κάθε φυσική γραμμή και εικονικό μονοπάτι ο Συντονιστής επικοινωνεί με τον Διαχειριστή Ειδοποιήσεων, καλώντας την κατάλληλη διεπαφή (monitorResource()) για να αρχίσει η παρακολούθηση.
5. Από την πληροφορία που περνάει ως παράμετρο στην διεπαφή που καλεί ο Συντονιστής, ο Διαχειριστής Σφαλμάτων, μέσω της υπηρεσίας ονομασίας εντοπίζει τις οντότητες παρακολούθησης και καλεί τις διεπαφές τους για να αρχίσει η διαδικασία παρακολούθησης.



Σχήμα 7-8 Παραγόμενα Στατιστικά της τέταρης δοκιμής

Εδώ όμως το σενάριο έχει ως εξής: Γίνεται μια ακολουθία σφαλμάτων και επιδιορθώσεων στις γραμμές 5 (1 φορά) και 1 (1 φορά), και στα εικονικά μονοπάτια σύνδεσης 0 (1 φορά), 1 (2 φορές), 2 (1 φορά) και 3 (1 φορά). Μετά την ακολουθία αυτή ζητείται από τον χρήστη διαχείρισης τα στατιστικά που έχουν παραχθεί

μέσω της κατάλληλης λειτουργίας της Διεπαφής Χρήστη (κουμπί "Get Net Risk Map"). Αυτή επικοινωνεί με τον Συντονιστή ο οποίος παίρνει τον χάρτη από τον Διαχειριστή Ειδοποιήσεων, ενημερώνει τα πεδία RUR των γραμμών και τον επιστρέφει στην Γραφική Διεπαφή Χρήστη. Το αποτέλεσμα φαίνεται στο Σχήμα 7-8.

8 *Συμπεράσματα και Μελλοντικές Επεκτάσεις*

Στο κεφάλαιο αυτό παρουσιάζονται τα συμπεράσματα της παρούσας εργασίας καθώς και προτεινόμενες μελλοντικές επεκτάσεις της.

8.1 Συμπεράσματα

Σχεδιάστηκε και υλοποιήθηκε ένα σύστημα διαχείρισης σφαλμάτων, στο επίπεδο NML, το οποίο παρέχει τις εξής βασικές λειτουργίες:

1. Την ανάθεση χαρακτηριστικών διαχείρισης σφαλμάτων (παρακολούθησης και επιδιόρθωσης) στους υπό διαχείριση πόρους και την αρχικοποίηση των διαδικασιών διαχείρισης (παρακολούθηση) εντοπίζοντας με διαφανή τρόπο τις υπεύθυνες οντότητες.
2. Τη δυνατότητα παρακολούθησης των πόρων.
3. Τη δυνατότητα δοκιμής των πόρων.
4. Τη διατήρηση ημερολογίων (Logs) των ειδοποιήσεων.
5. Τη διατήρηση της τοπολογία του δικτύου και την δυναμική ενημέρωσή της με την παράλληλη ενημέρωση και στις διαδικασίες διαχείρισης σφαλμάτων των πόρων.
6. Τον εντοπισμό της πρωταρχικής αιτίας σφάλματος στο υπό διαχείριση δίκτυο καθώς και την εξαγωγή στατιστικών στοιχείων για τα σφάλματα που συμβαίνουν στους πόρους του.
7. Την παροχή, μέσω καλά ορισμένης πληροφορίας και διεπαφών, των λειτουργιών αυτών προς άλλα συστήματα.

Παρέχοντας αυτές τις λειτουργίες σχεδιάσαμε και υλοποιήσαμε ένα σύστημα που είναι υπεύθυνο για την διαχείριση σφαλμάτων σε ένα δίκτυο ATM, παρέχοντας όλους εκείνους τους μηχανισμούς που εξασφαλίζουν την βιωσιμότητα του δικτύου μας σε περιπτώσεις σφαλμάτων καθώς και την αποδοτικότερη επανασχεδίασή του.

Με την παροχή της δυνατότητας από το σύστημά μας να συσχετίσουμε:

- τα συμβάντα (events) την συγκεκριμένη χρονική στιγμή (μέσω των ειδοποιήσεων που λαμβάνουμε),
- τα συμβάντα σε παρελθοντικό χρόνο (χρησιμοποιώντας τα ημερολόγια) ,
- την παρούσα κατάσταση του πόρου (μέσα από την δοκιμή του) και
- την τοπολογική σχέση των πόρων,

δείξαμε την δυνατότητα των συστημάτων διαχείρισης σφαλμάτων για παροχή πολύπλοκων μηχανισμών εντοπισμού των αιτιών των σφαλμάτων καθώς και εξαγωγής σημαντικών για άλλες περιοχές της διαχείρισης στατιστικών στοιχείων. Μάλιστα για την καλύτερη απόδειξη αυτής της δυνατότητας υλοποιήσαμε:

- έναν αλγόριθμο εντοπισμού της πρωταρχικής αιτίας του σφάλματος ο οποίος ανταποκρίνεται αποτελεσματικά ακόμα και σε περιπτώσεις σφάλματος που επηρεάζουν πολλούς πόρους.
- έναν αλγόριθμο εξαγωγής στατιστικών στοιχείων για τους πόρους του δικτύου, τόσο για την παρουσία σφαλμάτων στους ίδιους όσο και για την παρουσία σφαλμάτων σε άλλους που συσχετίζονται μαζί τους.

Διαχωρίζοντας το σύστημα σε υπολογιστικές οντότητες με διακριτές λειτουργίες κατανείμαμε τον φόρτο εργασίας του συστήματος και δώσαμε την δυνατότητα για χρήση των επί μέρους οντοτήτων από άλλα συστήματα.

Το σύστημα σχεδιάστηκε και υλοποιήθηκε χρησιμοποιώντας κατανεμημένες οντότητες για τις βασικές λειτουργίες παρακολούθησης και επιδιόρθωσης, μηχανισμούς διαφανούς εντοπισμού των οντοτήτων αυτών και επεκτάσιμο ορισμό της διαχειρίσιμης πληροφορίας. Με αυτόν τον τρόπο εξασφαλίσαμε:

- αποδοτική λειτουργία του συστήματος καθώς και την εύκολη επέκτασή του σε μεγαλύτερα δίκτυα (με περισσότερους διαχειρίσιμους πόρους)
- εύκολη σύνδεση με οντότητες που παρέχουν διαφορετικούς μηχανισμούς παρακολούθησης και επιδιόρθωσης
- εύκολη διεύρυνση της διαχειριζόμενης πληροφορίας ακόμα και για την διατήρηση και χρησιμοποίηση πληροφορίας διαφορετικής από αυτήν της διαχείρισης σφαλμάτων (πληροφορία σχετική με την ποιότητα παροχής υπηρεσιών των πόρων).

Η σχεδίαση και του συστήματος στηρίχθηκε στην αρχιτεκτονική TINA καθώς και σε μια σειρά άλλων αρχιτεκτονικών και προδιαγραφών που αφορούν τις λειτουργίες, τον ορισμό της πληροφορίας και τις διεπαφές επικοινωνίας. Με αυτόν τον τρόπο αποδείχθηκε η χρησιμότητα των υπαρχόντων αρχιτεκτονικών και προδιαγραφών στην σχεδίαση συστημάτων διαχείρισης και εξασφαλίστηκε η δια-λειτουργία του συστήματος με άλλα.

Για την υλοποίηση έγινε χρήση της σύγχρονης τεχνολογίας υλοποίησης κατανεμημένων συστημάτων χρησιμοποιώντας έξυπνους μηχανισμούς ασύγχρονης και σύγχρονης επικοινωνίας. Με την επιλογή αυτή δείξαμε την χρησιμότητα αυτών των τεχνολογιών στην υλοποίηση συστημάτων διαχείρισης, κυρίως για την γρήγορη απόκτηση της, αναγκαίας για τις λειτουργίες τους, πληροφορίας αλλά και για την παροχή των δικών τους λειτουργιών προς άλλα συστήματα.

8.2 Μελλοντικές επεκτάσεις

Οι επεκτάσεις που προτείνουμε αφορούν κυρίως την εκμετάλλευση της παρεχόμενης δυνατότητας συσχέτισης αλλά και την διεύρυνση της πληροφορίας σε άλλους πόρους καθώς και σε άλλα πεδία διαχείρισης. Συγκεκριμένα ως επεκτάσεις προτείνονται:

1. Χρησιμοποίηση της παρεχόμενης λειτουργικότητας για την εφαρμογή έξυπνων αλγορίθμων συσχέτισης και εντοπισμού. Η περιοχή της συσχέτισης της πληροφορίας διαχείρισης παρέχει τέτοιους αλγορίθμους οι οποίοι θα μπορούσαν να τροποποιηθούν για την εκμετάλλευση της υπάρχουσας λειτουργικότητας.

2. Χρησιμοποίηση της παρεχόμενης λειτουργικότητας για την εξαγωγή χρήσιμων στατιστικών στοιχείων για τα σφάλματα, τα οποία θα μπορούσαν να χρησιμοποιήσουν άλλες περιοχές διαχείρισης.

3. Μελέτη επέκτασης της εργασίας σε λήψη ειδοποιήσεων για αλλαγές της ποιότητας υπηρεσιών των εικονικών μονοπατιών σύνδεσης και χρήση της πληροφορίας αυτής για την επαναδιαμόρφωση του επιπέδου τους.

4. Συνεργασία με άλλα συστήματα συσχετισμού των σφαλμάτων σε επίπεδο χαμηλότερο ή υψηλότερο του ATM. Ένα τέτοιο παράδειγμα συνεργασίας θα ήταν με το επίπεδο SDH για τη απόκτηση της πληροφορίας του ποιοι πόροι έχουν ήδη επιδιορθωθεί από τους αυτόματους μηχανισμούς που αυτό παρέχει.

Παράρτημα Α

Υλοποίηση Εσωτερικού Μοντέλου Πληροφορίας

Στο παράρτημα αυτό δίνεται η υλοποίηση του εσωτερικού μοντέλου πληροφορίας του προτεινόμενου συστήματος Διαχείρισης Σφαλμάτων σε γλώσσα C++ (το αρχείο επικεφαλίδας - header file).

Κώδικας Υλοποίησης

```
// C++
// File: InfoModel.h
//
#include "VPCResource.hh"
#include "LinkResource.hh"
#include "NodeResource.hh"
#include "Resource.hh"
#include "AlarmSummaryReport.hh"
#include "TestSummaryReport.hh"
#include "MonitorClass.hh"
#include "NetworkRiskMap.hh"
#include "FMDefines.h"
#include "defines.h"


class Log;
class FaultManageableResource;
class AlarmSeverityAssignmentProfile;
class LogRecord;
class AlarmControl;
class AlarmControlRecord;

class AlarmControlRecordList {
public:
        AlarmControlRecord *acr;
        AlarmControlRecordList *next;
};

class FaultManageableResourceList {
public:
        FaultManageableResource *fmr;
        FaultManageableResourceList *next;
};

class LogRecordList {
public:
        LogRecord *lr;
        LogRecordList *next;
};

class DestinationList {
public:
        CORBA::Object *destination;
        DestinationList *next;
};
```

```
class SeverityAssignmentEntry {

        char *Severity;
        char *ProbableCause;

public:

        SeverityAssignmentEntry(char *severity, char *probablecause);
        virtual ~SeverityAssignmentEntry();

        void setEntry(char *severity, char *probablecause);
        char *getSeverity(char *probablecause);
};

typedef SeverityAssignmentEntry **SeverityAssignmentList;

class FaultManagementDomain {

public:
        char *FaultManagementDomainName;
        char *FaultManagementDomainType;
        FaultManageableResourceList *FMRList;
        AlarmSeverityAssignmentProfile *ASAP;
        int FMRListSize;

        FaultManagementDomain(char *name, char *type);
        virtual ~FaultManagementDomain();

        char *getName();
        char *getType();
        void addFMR(FaultManageableResource *fmr);
        int deleteFMR(FaultManageableResource *fmr);
        RiskMapResourceInfoList *getRiskMapResourceInfoList();
};

class FaultManageableResource {

public:
        char *ResourceName;
        char *ResourceType;
        char *ResourceState;
        long CreationTime;
        Log *log;
        RiskMapResourceInfo *RMRI;

        FaultManageableResource(char *resourcename, char *resourcetype, char *resourcestate);
        virtual ~FaultManageableResource();

        void changeState(char *resourcestate);
        void addLogRecord(LogRecord *lr);
        int deleteLogRecord(LogRecord *lr);
        RiskMapResourceInfo *getRiskMapResourceInfo();
        void updateRiskMapInfo();
};

class TestableResource:public FaultManageableResource {

public:
        char *TestableResourceType;
```

```
        TestableResource(char *testableresourcetype, char *resourcename, char *resourcetype, char
*resourcestate);
        virtual ~TestableResource();

        TestSummaryReport *getTestSummaryReport();
};

class AlarmableResource:public FaultManageableResource {

public:
        char *AlarmableResourceType;
        AlarmControl *AC;

        AlarmableResource(char *alarmableresourcetype, char *resourcename, char *resourcetype, char
*resourcestate);
        virtual ~AlarmableResource();

        AlarmSummaryReport *getAlarmSummaryReport();
        void addAlarmControlRecord(AlarmControlRecord *acr);
};


class Log {

public:
        char *LogName;
        LogRecordList *LRL;
        int CurrentLogSize;

        Log(char *logname);
        virtual ~Log();

        void addLogRecord(LogRecord *lr);
        int deleteLogRecord(LogRecord *lr);

        LogRecordList *getLogRecordList(char *eventtype);
};


class LogRecord {

public:
        char *EventType;
        long EventTime;
        char *AdditionalInfo;

        LogRecord(char *eventtype, char *additionalinfo);
        virtual ~ LogRecord();

        char *getEventType() { return EventType; }
        long getEventTime() { return EventTime; }
        char *getAdditionalInfo() { return AdditionalInfo; }

};

class AlarmRecord:public LogRecord {
        Alarm *alarm;

public:
```

```
        AlarmRecord(Alarm *alrm);
        AlarmRecord(Alarm *alrm, char *addinfo);
        virtual ~AlarmRecord();

        char *getType();
        Alarm *getAlarm();
        Resource *getResource();

};

class TestRecord:public LogRecord {
        Test *test;

public:

        TestRecord(Test *tst);
        TestRecord(Test *tst, char *addinfo);
        virtual ~TestRecord();

        char *getType();
        Test *getTest();
        Resource *getResource();
};

class AlarmControl {

public:
        AlarmControlRecordList *ACRL;

        AlarmControl();
        virtual ~AlarmControl();

        void addAlarmControlRecord(AlarmControlRecord *acr);
        int deleteAlarmControlRecord(AlarmControlRecord *acr);
        DestinationList *getDestinationList(char *alarmtype);
};

class AlarmControlRecord {

public:
        CORBA::Object *Destination;
        char *AlarmType;

        AlarmControlRecord(CORBA::Object *destination, const char *alarmtype);
        virtual ~AlarmControlRecord();

        CORBA::Object *getDestination();
};

class AlarmSeverityAssignmentProfile {

protected:
        SeverityAssignmentList SAL;
        long SeverityAssignmentListSize;

public:
        AlarmSeverityAssignmentProfile();
        virtual ~AlarmSeverityAssignmentProfile();
```

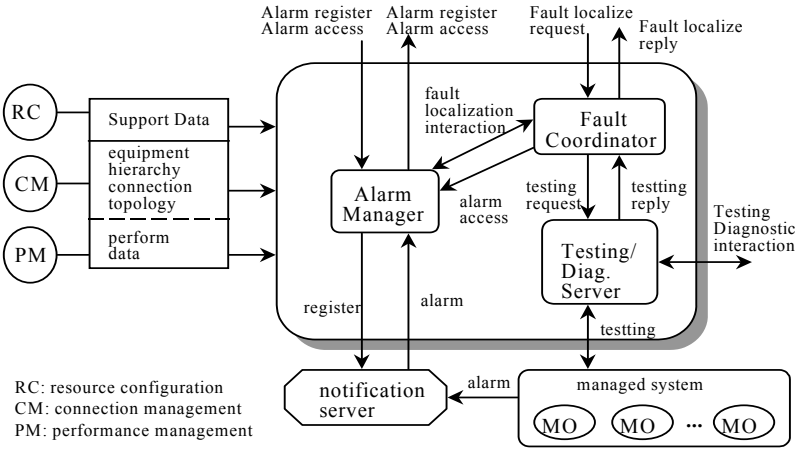
```
        void initialize(char **probablecauses, char **severities);
        void setSeverity(char *probablecause, char *severity);
        char *getSeverity(char *probablecause);
};

class VPCAlarmableResource;
class LinkAlarmableResource;
class NodeAlarmableResource;
class VPCTestableResource;
class LinkTestableResource;
class NodeTestableResource;


class VPCFaultManagementDomain:public FaultManagementDomain {

public:
        VPCFaultManagementDomain(char *name, char *type);
        virtual ~ VPCFaultManagementDomain();

        VPCAlarmableResource *getVPCAlarmableResource(VPCResource *vpconnection);
        VPCTestableResource *getVPCTestableResource(VPCResource *vpconnection);
        int monitorVPC(VPCAlarmableResource *VPCAR);
        int stopMonitorVPC(VPCAlarmableResource *VPCAR);
        int testVPC(VPCTestableResource *VPCTR);
};

class LinkFaultManagementDomain:public FaultManagementDomain {

public:
        LinkFaultManagementDomain(char *name, char *type);
        virtual ~ LinkFaultManagementDomain();

        LinkAlarmableResource *getLinkAlarmableResource(LinkResource *linkconnection);
        LinkTestableResource *getLinkTestableResource(LinkResource *linkconnection);
        int monitorLink(LinkAlarmableResource *LinkAR);
        int stopMonitorLink(LinkAlarmableResource *LinkAR);
        int testLink(LinkTestableResource *LinkTR);

};

class NodeFaultManagementDomain:public FaultManagementDomain {

public:
        NodeFaultManagementDomain(char *name, char *type);
        virtual ~ NodeFaultManagementDomain();

        NodeAlarmableResource *getNodeAlarmableResource(NodeResource *node);
        NodeTestableResource *getNodeTestableResource(NodeResource *node);
        int monitorNode(NodeAlarmableResource *nodeAR);
        int stopMonitorNode(NodeAlarmableResource *nodeAR);
        int testNode(NodeTestableResource *nodeTR);
};

class VPCAlarmableResource:public AlarmableResource {

public:
        VPCResource vpc;

        VPCAlarmableResource(char *state, VPCResource vpcr);
```

```
        virtual ~VPCAlarmableResource();
};

class LinkAlarmableResource:public AlarmableResource {

public:
        LinkResource link;

        LinkAlarmableResource(char *state, LinkResource linkr);
        virtual ~LinkAlarmableResource();

};

class NodeAlarmableResource:public AlarmableResource {

public:
        NodeResource node;

        NodeAlarmableResource(char *state, NodeResource nr);
        virtual ~NodeAlarmableResource();

};

class VPCTestableResource:public TestableResource {

public:
        VPCResource vpc;

        VPCTestableResource(char *state, VPCResource vpcr);
        virtual ~VPCTestableResource();

};

class LinkTestableResource:public TestableResource {

public:
        LinkResource link;

        LinkTestableResource(char *state, LinkResource linkr);
        virtual ~LinkTestableResource();

};

class NodeTestableResource:public TestableResource {

public:
        NodeResource node;

        NodeTestableResource(char *state, NodeResource nr);
        virtual ~NodeTestableResource();

};
```

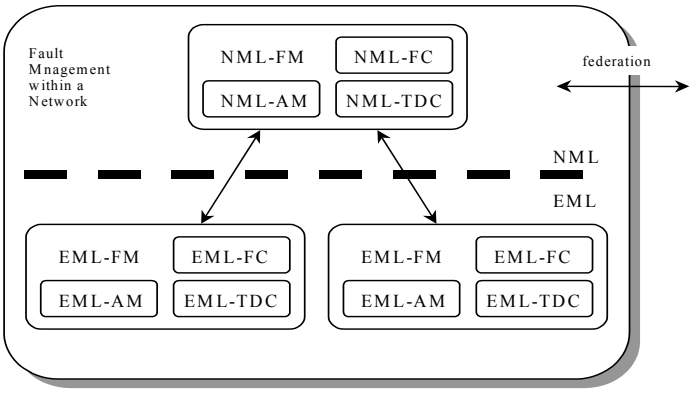
Παράρτημα Β

Περιγραφή επιπέδου Στοιχείων Δικτύου

Στο παράρτημα αυτό περιγράφονται οι λειτουργίες που θα πρέπει να υπάρχουν από ένα σύστημα Διαχείρισης Σφαλμάτων στο επίπεδο EML σύμφωνα με την προτεινόμενη από την TINA αρχιτεκτονική. Επίσης περιγράφονται και οι διεπαφές οι οποίες υλοποιήθηκαν για τις υπολογιστικές οντότητες του επιπέδου αυτού για να μπορέσει να δοκιμαστεί το προτεινόμενο από την εργασία σύστημα.



Σχήμα 1: Αλληλεπιδράσεις Υπολογιστικών Οντοτήτων



Σχήμα 2: Βασική Υπολογιστική Αρχιτεκτονική Διαχείρισης Σφαλμάτων

Υπολογιστικές Οντότητες EML επιπέδου

Σύμφωνα με την αρχιτεκτονική Διαχείρισης Σφαλμάτων που προτείνει η Αρχιτεκτονική Πόρων Δικτύου της TINA, ένα σύστημα διαχείρισης σφαλμάτων στο επίπεδο Στοιχείων του Δικτύου (EML), όπως φαίνεται στα σχήματα 1 και 2, αποτελείται ουσιαστικά από τρεις υπολογιστικές οντότητες:

1. Τον Συντονιστή του επιπέδου Στοιχείων
2. Τον Διαχειριστή Ειδοποιήσεων του επιπέδου Στοιχείων
3. Τον Εξυπηρετητή Δοκιμών του επιπέδου Στοιχείων

Οι λειτουργίες που η κάθε μια από αυτές τις υπολογιστικές οντότητες κάνει σύμφωνα με την TINA περιγράφονται στο Κεφάλαιο 2.

Η συνολική λειτουργικότητα του συστήματος στο επίπεδο των Στοιχείων, περιγραφικά, έχει ως εξής:

Διαχειριστής Σφαλμάτων

Ενημερώνεται από το σύστημα του επιπέδου Δικτύου για τις διαδικασίες παρακολούθησης που πρέπει να αρχίσει/σταματήσει.

Όποτε ανιχνεύσει ένα σφάλμα σε κάποιο πόρο αρχίζει τις εσωτερικές λειτουργίες σε αυτό το σφάλμα (φιλτράρισμα κλπ) και το προωθεί τόσο στον Συντονιστή του ίδιου επιπέδου όσο και στο σύστημα του επιπέδου Δικτύου με μια ειδοποίηση σφάλματος.

Αν έχει διαγνώσει σφάλμα σε κάποιον πόρο, συνεχίζει την διαδικασία παρακολούθησης και αν κάποια στιγμή διαγνώσει επαναφορά λειτουργιών σε αυτόν τον πόρο, το αναφέρει στο σύστημα του επιπέδου του Δικτύου δημιουργώντας μια ειδοποίηση επιδιόρθωσης.

Εξυπηρετητής Δοκιμών

Δέχεται κλήσεις για δοκιμές σε πόρους του στοιχείου στο οποίο βρίσκεται τόσο από το σύστημα του επιπέδου Δικτύου όσο και από τον Συντονιστή του επιπέδου Στοιχείων στο στοιχείο το οποίο βρίσκεται.

Συντονιστής

Δέχεται ειδοποιήσεις από τον Διαχειριστή Ειδοποιήσεων του επιπέδου Στοιχείων, στο στοιχείο που βρίσκεται και προχωράει στην επιδιόρθωση του πόρου εφόσον προχωρήσει ανάλογα με την περίπτωση σε δοκιμές και άλλες διαδικασίες (εντοπισμός κύριας αιτίας κλπ). Επίσης προχωρεί σε επιδιορθώσεις και μετά από κλήσεις από το σύστημα του επιπέδου Δικτύου.

Μόλις τελειώσει η διαδικασία μιας επιδιόρθωσης δημιουργεί την κατάλληλη ειδοποίηση επιδιόρθωσης την οποία στέλνει στο σύστημα του επιπέδου Δικτύου.

Περιγραφή Διεπαφών

Σύμφωνα λοιπόν με την προτεινόμενη από την TINA αρχιτεκτονική υπάρχει ένα σύστημα Διαχείρισης Σφαλμάτων στο NML επίπεδο και πολλά συστήματα στο EML επίπεδο, ένα για κάθε στοιχείο (Σχήμα 2).

Επειδή το σχεδιαζόμενο και υλοποιούμενο σύστημα είναι επιπέδου Δικτύου για να μπορεί να δοκιμαστεί υλοποιήθηκαν όπως περιγράφηκε στο κεφάλαιο 6 και οι οντότητες που αποτελούν το σύστημα στο επίπεδο των Στοιχείων. Σε αυτήν την παράγραφο παραθέτονται οι διεπαφές των τριών υπολογιστικών οντοτήτων του EML επιπέδου σε IDL.

Διεπαφή Συντονιστή EML επιπέδου

Η διεπαφή που υλοποιεί ο Συντονιστής επιπέδου Στοιχείων καλείται από το σύστημα επιπέδου Δικτύου για να γίνει κάποια επιδιόρθωση και έχει ως εξής:

```
//
// File: EML_FMRestore.idl
//

#include "Resource.idl"


interface EML_FMRestore {

        // Exception raised when the restoration of the resource fails
        exception RestorationFailure {
                string reason;
        };

        // Method called in the EML Coordinator in order to restor a resource
        void restore(in Resource rsc)
                raises (RestorationFailure);
};
```

Διεπαφή Διαχειριστή Ειδοποιήσεων EML επιπέδου

Η διεπαφή που υλοποιεί ο Διαχειριστής Ειδοποιήσεων επιπέδου Στοιχείων καλείται από το σύστημα επιπέδου Δικτύου για να αρχίσει/σταματήσει η διαδικασία παρακολούθησης και έχει ως εξής:

```
//
// File:    EML_FMMonitor.idl
//

#include "Resource.idl"


interface EML_FMMonitor {

        // Exception raised when the start/stop monitoring a resource fails
        exception MonitorFailure {
                string reason;
        };

        // Method called in the EML Alarm Manager in order to start monitoring a resource
        void startMonitor(in Resource rsc, in Object obj)
                raises (MonitorFailure);

        // Method called in the EML Alarm Manager in order to stop monitoring a resource
        void stopMonitor(in Resource rsc, in Object obj)
                raises (MonitorFailure);
};
```

Διεπαφή Εξυπηρετητή Δοκιμών EML επιπέδου

Η διεπαφή που υλοποιεί ο Εξυπηρετητής Δοκιμών επιπέδου Στοιχείων καλείται από το σύστημα επιπέδου Δικτύου για να γίνει δοκιμή σε κάποιον πόρο του στοιχείου και έχει ως εξής:

```
//
// File:    EML_FMTest.idl
```

```
//

#include "Test.idl"


interface EML_FMTest {

        // Exception raised when the test on the resource fails
        exception TestFailure {
                string reason;
        };

        // Method called in the EML Testing Server in order to test a resource
        void test(inout Test tst)
                raises (TestFailure);
};
```

Συντομογραφίες

ACTS	Advanced Communications Technologies and Services
AM	Alarm Manager
ATM	Asynchronous Transfer Mode
CM	Connection Management
CO	Computational Object
ConfM-CM	Configuration Management - Connection Management
ConfM-NM	Configuration Management - Network Map
CORBA	Common Object Request Architecture
DIB	Directory Information Base
DPE	Distributed Processing Environment
EML	Element Management Layer
FC	Fault Coordinator
FM	Fault Management
IIOP	Internet Inter-Operation Protocol
IPC	Inter Process Communication
ITU	International Telecommunication Union
MC	Monitor Class
MIB	Managed Information Base
MT	Multi Threaded
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
NEL	Network Element Layer
NML	Network Management Layer
NRA	Network Resource Architecture
NRIM	Network Resource Information Model
NTCM	Network Topology Configuration Management
OAM	Operation Administration Maintenance
ODP	Open Distributed Processing
OMG	Object Management Group
OMT	Object Modeling Technique
ORB	Object Request Broker
RC	Restoration Class
RDN	Relative Distinguished Name
REFORM	REsource Failure and restORation Management
RML	Resource Management Layer
RUR	Resource Usage Reliability
SH	Shelf Healing
SML	Service Management Layer
TDS	Testing Diagnostic Server
TINA	Telecommunication Informational Network Architecture
TMN	Telecommunication Management Network
VCC	Virtual Channel Connection
VPC	Virtual Path Connection

Αναφορές

[D4] REFORM Deliverable D4, “Final REFORM System Specifications and Architecture”, A208/NTUA/WP3/DS/P/008/b1, July 1998

[D7] REFORM Deliverable D7, *“Description of Self-Healing, Load Balancing and Dynamic Routing Capabilities of the REFORM System”,* September 1997

[D11] REFORM Deliverable D11, “*Feedback on the Survivability and Intelligent Resource Management Algorithms in the Phase-1 REFORM System*”, December 1997

[Fuente1] *“The TINA-C Approach to TMN”*, Fuente L., Pavon J. & Moreno C., 1995.

[G803] ITU-T Recommendation G.803, *“Architectures of Transport Networks Based on the SDH”*, 93

[I610] ITU-T Recommendation I.610*, “B-ISDN operation and maintenance principles and functions”,* 1995

[ICM] *“Integrated Communications Management of Broadband Networks”,* D. Griffin (ed.), Crete University Press, March 1997.

[IDL] *“Specification of the Interface Definition Language (IDL)”,* Object Management Group

[M3010] ITU-T Recommendation M.3010, *“Principles for a Telecommunications Management Network (TMN)”,* SG IV, 1996

[M3020] ITU-T Recommendation M.3020, *“TMN interface specification methodology”,* 1995

[M3100] ITU-T Recommendation M.3100, *“Generic Network Information Model”,* 1995

[M3400] ITU-T Recommendation M.3400, *“TMN Management Functions”,* 1992

[NRAv3] *“Network Resource Architecture,”* The TINA Consortium, Version 3.0, Feb. 1997

[NRIM] *“Network Resource Information Model,”* The TINA Consortium, Nov. 1997

[ODMA] ITU-T Draft Recommendation, *“Open Distributed Management Architecture”,* 1995

[ODP] ITU-T Recommendation X.903/ISO 10746-3, *“Basic reference model of Open Distributed Processing”,* 1994

[SMF] ITU-T Recommendation X.730-750, "*Information Technology - Open Systems Interconnection - Systems Management Function",* 92

[TINA] *“An Overview of the Telecommunications Information Networking Architecture (TINA)”,* TINA’95 Conference, Melbourne, Australia, 1995

[TINA-EM] *“Engineering Modelling Concepts (DPE Architecture)”,* The TINA Consortium, Version 2.0, Dec. 1994

[TINA-MA] *“The TINA Management Architecture”,* The TINA Consortium, Version 2.0, Dec 1994

[TINA-RP] *“TINA Reference Points”,* The TINA Consortium, Version 3.1, June 1996

[HSREC] "High Speed & Robust Event Correlation", S. Yemini, E. Mozes, Y. Yemini, D. Ohsie 1995

[KaGe] "A generic Model for Fault Isolation in Integrated Management Systems", Stefan Kaetker, Kurt Geihs, JNSM Vol. 5, No. 2, June 1997

[NNAC] "Using Neural Networks for Alarm Correlation in Cellular Phone Networks", Hermann Wietgrefem Klaus-Dieter Tuchs, Klaus Jobmann, Guido Carls, Peter Frohlich, Wolfgang Nejdl, Sebastian Steinfeld, 1997

[MAC] "Model-Based Alarm Correlation in Cellular Phone Networks", Peter Frohlich, Wolfgang Nejdl, Klaus Jobmann, Hermann Wietgrefem, 1997

[X500] ITU-T Recommendation X.500, " *Information Technology - Open Systems Interconnection - The directory: Overview of concepts, models and services",* 1993

[X501] ITU-T Recommendation X.501, " *Information Technology - Open Systems Interconnection - The directory: Models",* 1993

[X701] ITU-T Recommendation X.701, *"Information Technology - Open Systems Interconnection - Systems Management Overview"*, 1992.

[X710] ITU-T Recommendation X.710, *"Information Technology - Open Systems Interconnection - Common Management Information service"*, 1997

[X721] ITU-T Recommendation X.721, *"Information Technology - Open Systems Interconnection - Structure of Management Information: Deinition of Management Information",* 1992.

[X722] ITU-T Recommendation X.722, *"Information Technology - Open Systems Interconnection - Structure of Management Information: Guidelines for the Definition of Managed Objects (GDMO)",* 1992.

[X733] ITU-T Recommendation X.733, "*Information Technology - Open Systems Interconnection - Systems Management: Alarm Reporting Function*", 1992.

[X735] ITU-T Recommendation X.735, "*Information Technology - Open Systems Interconnection - Systems Management: Log Control Function*", 1992.