ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μηχανισμός υποστήριξης αποφάσεων με κριτήρια Ποιότητας Υπηρεσίας (Quality of Service) σε Υπηρεσιοστρεφείς Αρχιτεκτονικές (Service Oriented Architectures)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΑΝΑΓΙΩΤΗ Π. ΠΕΤΡΙΔΗ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Αναπληρώτρια Καθηγήτρια Ε.Μ.Π

Αθήνα, Σεμπτέβριος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μηχανισμός υποστήριξης αποφάσεων με κριτήρια Ποιότητας Υπηρεσίας (Quality of Service) σε Υπηρεσιοστρεφείς Αρχιτεκτονικές (Service Oriented Architectures)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΑΝΑΓΙΩΤΗ Π. ΠΕΤΡΙΔΗ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Αναπληρώτρια Καθηγήτρια Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 28 Σεμπτεβριου 2007

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Καγιάφας Ελευθέριος	Λούμος Βασίλειος
Αναπλ. Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπλ. Καθηγητής Ε.Μ.Π.

Αθήνα, Σεμπτέβριος 2007

.....

ΠΑΝΑΓΙΩΤΗΣ Π. ΠΕΤΡΙΔΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Παναγιώτης Πετρίδης
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Σκοπός της παρούσας εργασίας αυτής είναι η παρουσίαση ενός μηχανισμού ο οποίος ενεργοποιεί την χρησιμοποίηση ενός βοηθητικού συστατικού, με την ονομασία Quality of Service (QoS) History, το οποίο περιέχει πληροφορίες σχετικά με την συμπεριφορά του συστήματος του φορέα παροχής υπηρεσιών στο παρελθόν συναρτήσει με τα SLA πρότυπα. Η διαδικασία απόφασης στην πλευρά του εξυπηρετητή ενισχύεται όσο τα περιεχόμενα του αρχείου μπορούν να χρησιμοποιηθούν απόλυτα ως κριτήριο για την βελτίωση της προσφοράς του παροχέα και επομένως και την επιχειρηματική του δύναμη. Μελετώντας τις προηγούμενες συνεργασίες μεταξύ του παροχέα και των πελατών μπορεί να παρέχει τα αναγκαία δεδομένα για την βελτίωση της παρεχόμενης υπηρεσίας (π.χ εάν μια υπηρεσία δεν συμπεριφέρεται σωστά ήδη δύο φορές, μπορεί να βοηθήσει (i) στην εύρεση του λόγου και (ii) να βοηθήσει στην αναδιαμόρφωση των προσφερόμενων δυνατοτήτων της υπηρεσίας) Αυτό είναι εφικτό μέσα από το SLA-based accounting, παρακολούθηση και αξιολόγηση των σχετικών με QoS όρων διαπραγμάτευσης, επομένως, με την χορήγηση ενός λεπτομερούς σκίτσου της επίδοσης συγκεκριμένων προσφορών σε συγκεκριμένες αιτήσεις στο παρελθόν.

Σε αυτή τη λογική, αποσκοπούμε στην ενεργοποίηση ενός μηχανισμού ο οποίος θα κρατά τις δικές του καταχωρήσεις σχετικά με την ιστορία επίδοσης για κάθε μέλος της αγοράς του και καθορίζει την επιτυχία των δικών του απαιτήσεων μέσα από το ιστορικό του. Η πρόταση είναι η δημιουργία του μηχανισμού οποίος μπορεί να εγκατασταθεί στην πλευρά του παροχέα και ο οποίος θα ενεργοποιεί την δυναμική κατανόηση του SLA χρησιμοποιώντας την πείρα του σχετικά με το QoS στο παρελθόν και μ’αυτό, να καθορίσει την ανταγωνιστικότητα στην αγορά.

Η έννοια της ποιότητας υπηρεσίας (Quality of Service - QoS) είναι πολύ σημαντική στον αναπτυσσόμενο χώρο των τεχνολογιών Πολυπλέγματος, αφού η χρήση των συστημάτων αυτών στον επιχειρηματικό κόσμο απαιτεί την εξασφάλιση και επιβεβαίωση της ποιότητας υπηρεσίας μεταξύ των επιχειρήσεων που συμμετέχουν.

Ο μηχανισμός που περιγράφεται στο παρόν κείμενο υλοποιήθηκε με χρήση δικτυακών υπηρεσιών (Web Services) που εγκαθίστανται σε υπολογιστικούς πόρους Grid συστημάτων. Οι δικτυακές υπηρεσίες καλούνται είτε από τους παρόχους των υπηρεσιών (Service Providers) με στόχο τον εσωτερικό έλεγχο και τη βελτίωση των υπηρεσιών, είτε από εξωτερικούς πιστοποιημένους πελάτες με σκοπό την επιβεβαίωση της ποιότητας της υπηρεσίας.

Οι πλατφόρμες και τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση είναι οι πιο διαδεδομένες για αυτή τη χρήση. Συγκεκριμένα έγινε χρήση του Apache Tomcat Server, επίσης ενός εργαλείου διαχείρισης βάσεων δεδομένων και συγκεκριμένα του Oracle Berkeley DB XML 2.3 και προγραμματισμού στη γλώσσα Java για τις υπηρεσίες.

Λέξεις Κλειδιά: <<Ποιότητα Υπηρεσίας, Grid, Java, Oracle Berkeley DB XML, Tomcat server, SLA, ΠάροχοςΥπηρεσίας, Καταναλωτής Υπηρεσίας>>

Abstract

The aim of this thesis is the development of a decision support mechanism in service oriented architectures, and will give advices to the users according to the level of quality of service a provider can give to his customers.

The role of QoS in Grid is fundamental, since it provides the necessary verification and the guarantee between the participants of a business Grid.

This module will be implemented with the use of Web Services which will be deployed in every resource of the Grid system. Those services will be called by the Service Providers, for internal control and improvement, and by authorized users of the Grid, for their confirmation of the service's quality.

The technologies that will be used during the implementation of the module will be at the cutting edge. In detail, tomcat server and Java programming will be used for the web service implementation while Oracle Berkeley DB XML will implement the database in which we will store the useful information about the quality of service a provider can provide to his customers.

Keywords: <<Grid, QoS, SLA, Web Service, Java, Tomcat, Oracle Berkeley DB XML, Service Provider, Service Consumer>>

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την Αναπληρώτρια Καθηγήτρια Θεοδώρα Βαρβαρίγου για την ανάθεση της εργασίας, την παρακολούθηση και παροχή κατευθύνσεων για την υλοποίηση της. Επίσης, δεν θα ήθελα να παραλείψω τη συμβολή των Τσερπέ Κωνσταντίνου, Μενύχτα Αντρέα, και Κυριαζή Δημοσθένη, υποψήφιων διδακτόρων του Εργαστηρίου Τηλεπικοινωνιών, που με αμέριστη υπομονή και συμπαράσταση, συνέβαλαν με τον τρόπο τους στην ολοκλήρωση αυτού του έργου.

Επιπρόσθετα θα ήθελα να ευχαριστήσω την Μαρίνα Μαρκίδου για τις χρήσιμες συμβουλές της καθώς επίσης και για την βοήθεια της.

Τέλος, οφείλω να ευχαριστήσω την οικογένεια μου για την συνεχή συμπαράσταση της προς εμένα αλλά και για τις θυσίες που έχει κάνει για την επιτυχή σταδιοδρομία μου.

Πίνακας περιεχομένων

1 Εισαγωγή στο κεφάλαιο 1

1.1 Αντικείμενο της διπλωματικής 1

1.2 Οργάνωση του εγγράφου 2

2 *Περιγραφή Θέματος* 5

2.1 Σχετικές τεχνολογίες 5

2.1.1 Grid 5

2.1.1.1 Γιατί Grid; 5

2.1.1.2 Τα πλεονεκτήματα του Grid 6

2.1.1.3 Μια άλλη ερμηνεία του Grid 8

2.1.1.4 Ιστορία του Grid 9

2.1.2 OGSA 11

2.1.2.1 Ποιοι είναι οι στόχοι του OGSA ; 11

2.1.2.2 Η αρχιτεκτονική του OGSA 12

2.1.3 Γενικά για τις υπηρεσίες 14

2.1.3.1 XML Web services (Διαδικτυακές Υπηρεσίες) 15

2.1.4 Service Oriented Architecture (Υπηρεστρεφής Αρχιτεκτονική) 16

2.1.5 Web Services Description Language (WSDL) 18

2.1.6 Simple Object Access Protocol (SOAP) 18

2.1.7 Ποιότητα Υπηρεσίας (Quality of Service) 20

2.1.7.1 Η έννοια της Ποιότητας Υπηρεσίας στο Grid 20

2.1.7.2 Η εφαρμογή της Ποιότητας Υπηρεσίας στο Grid 23

2.1.7.3 Η Ιστορία της Ποιότητας της Υπηρεσίας στα Grids 23

2.2 Στόχος 24

2.2.1 Από το ακαδημαϊκό στο επιχειρηματικό Grid (Business Grid) 25

2.2.2 Προσέγγιση λειτουργίας του επιχειρηματικού μοντέλου του Grid 26

2.2.2.1 QoS Provider 28

2.2.2.2 Διαθεσιμότητα (Availability) 29

2.2.2.3 Αποδοτικότητα (Performance) 29

2.2.2.4 Αξιοπιστία (Reliability) 30

3 *Ανάλυση και σχεδίαση* 31

3.1 Βασικό Θεωρητικό Υπόβαθρο 31

3.2 Συνεργαζόμενος μηχανισμός 34

3.2.1 Μηχανισμός Παροχής Πληροφορίας για την Ποιότητα Υπηρεσίας (QoS Information Provisioning Mechanism) 35

3.2.2 Παράμετροι και Δείκτες QoS 35

3.2.3 Περιγραφή QoS Information Provisioning Mechanism 38

3.2.4 Υπηρεσία Παροχής Πληροφορίας QoS (QoS Information Provisioning Service) 41

3.2.5 Λήψη των παραμέτρων ποιότητας 46

3.3 Περιγραφή Αρχιτεκτονικής 50

3.3.1 Μηχανισμός Ιστορικού Πληροφορίας QoS (QoS History) 50

3.3.2 Λειτουργία Μηχανισμού 54

3.3.2.1 Λεπτομερής περιγραφή λειτουργιών μηχανισμού 55

4 Υλοποίηση 59

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία 59

4.1.1 Java 60

4.1.2 Oracle Berkeley DB XML 62

4.1.3 Εξυπηρετητής-Web Server 62

4.1.4 Tomcat 62

4.1.5 AXIS 63

4.2 Λεπτομέρειες Υλοποίησης 63

4.2.1 Server Side 64

4.2.1.1 Αρχείο InsertSLA.java 64

4.2.1.2 Αρχείο RecommendSLA.java 65

4.2.2 Client Side 66

4.2.2.1 Αρχείο Insert.java 66

4.2.2.2 Αρχείο Update.java 67

4.2.2.3 Αρχείο Rate.java 67

4.2.3 Βάση Δεδομένων 69

4.2.3.1 Βάση Δεδομένων Πελάτη 69

4.2.3.2 Βάση Δεδομένων Φορέα Παροχής Υπηρεσιών 71

5 ***Έλεγχος*** 73

5.1 Οδηγός Εγκατάστασης 73

5.1.1 Εγκατάσταση του Tomcat και AXIS 73

5.1.2 Ρύθμιση μεταβλητών συστήματος και βιβλιοθηκών 74

5.1.3 Εγκατάσταση Βάσης Δεδομένων 75

5.1.4 Εγκατάσταση QoS History 75

5.2 Αναλυτική παρουσίαση ελέγχου 76

5.2.1 Τα αποτελέσματα της χρήσης του συστήματος 76

6 ***Επίλογος*** 83

6.1 Σύνοψη και συμπεράσματα 83

6.1.1 Σύνοψη 83

6.1.2 Συμπεράσματα 85

6.2 Μελλοντική Εργασία 86

7 ***Βιβλιογραφία*** 87

Παράρτημα 93

Client Side 93

Insert.java 93

UpdateDocument.java 97

Rate.java 105

DBService.java 113

Services.xml 113

build.xml 114

Server Side 117

InsertSLA.java 117

RecommendSLA.java 121

HistoryServices.java 129

build.xml 130

1 Εισαγωγή στο κεφάλαιο

Στο παρόν κεφάλαιο γίνεται αρχικά προσέγγιση του αντικειμένου της διπλωματικής εργασίας και παρουσιάζεται η πορεία που ακολουθήθηκε για την εκπόνησή της και την ολοκλήρωση του συστήματος λογισμικού που αναπτύχθηκε στα πλαίσια αυτής. Τέλος γίνεται ανάλυση της οργάνωσης του παρόντος τόμου της διπλωματικής εργασίας σε κεφάλαια και περιγραφή του περιεχομένου των κεφαλαίων.

1.1 Αντικείμενο της διπλωματικής

Η αλματώδης εξέλιξη των δικτυακών τεχνολογιών και η απαίτηση υπολογιστικής ισχύος οπουδήποτε στον πλανήτη και οποτεδήποτε, οδήγησε στην ανάπτυξη της Grid τεχνολογίας. Με την διάδοση αυτής της τεχνολογίας και εκτός του ακαδημαϊκού χώρου, ένας συνεχώς αυξανόμενος αριθμός από εμπορικές εφαρμογές άρχισαν να υιοθετούν τεχνολογίες Grids, προκειμένου να βελτιώσουν την απόδοση, την αξιοπιστία και τη διαθεσιμότητά τους με ένα οικονομικό τρόπο. Η ευκολία στην δυναμική διαχείριση των πόρων και άλλες υπηρεσίες που προσφέρονται από τα Grid, έχουν φέρει την τεχνολογία αυτή στο προσκήνιο τα τελευταία χρόνια. Σε αυτό το πλαίσιο, και δεδομένου ότι η αγορά απαιτεί τώρα περισσότερες εγγυήσεις στο Επίπεδο της παρεχόμενης Ποιότητας (QoS), η διπλωματική αυτή εργασία, έχει ως σκοπό την υλοποίηση ενός μηχανισμού, ικανού να παρέχει

πληροφορία σχετικά με την ποιότητα των παρεχόμενων υπηρεσιών από το Grid, σε αρχιτεκτονικές βασισμένες σε υπηρεσίες. Ο μηχανισμός αυτός θα αναπτυχθεί πάνω σε τεχνολογίες Web Services και σκοπός του θα είναι να καταγράφει, να υπολογίζει, να καταχωρεί και να επιστρέφει σε όποιον του ζητήσει δομημένες (μοντελοποιημένες) πληροφορίες σχετικά με την ποιότητα των Grid υπηρεσιών.

Τέλος όλες οι πληροφορίες θα συλλέγονται για μελλοντική επεξεργασία και εξαγωγή συμπερασμάτων με σκοπό την διόρθωση και βελτίωση των παρεχόμενων υπηρεσιών.

Η ανάπτυξη αυτής της υπηρεσίας θα ακολουθήσει την παρακάτω διαδικασία:

- Εξοικείωση με τις σχετικές τεχνολογίες
- Σχεδιασμός του μηχανισμού
- Συγγραφή κώδικα υπηρεσίας
- Συγγραφή κώδικα διασυνδέσεων χρήστη (User Interface)
- Εξαγωγή συμπερασμάτων και συλλογή αποτελεσμάτων

1.2 Οργάνωση του εγγράφου

Το παρόν έγγραφο αποτελείται από 7 κεφάλαια και Παράρτημα. Στις ενότητες των κεφαλαίων αυτών παρουσιάζεται ουσιαστικά και με αναλυτικό τρόπο η διαδικασία ανάπτυξης της εφαρμογής, που περιγράφηκε στην προηγούμενη ενότητα.

Το κεφάλαιο 2 έχει τίτλο «Περιγραφή Θέματος» και τοποθετεί το θέμα ως προς την επιστημονική περιοχή στην οποία ανήκει (Ενότητα «Σχετικές Τεχνολογίες») . Επιπλέον δίνει το στόχο της εργασίας και το πρόβλημα που αυτή καλείται να αντιμετωπίσει (Ενότητα «Στόχος»).

Το κεφάλαιο 3 έχει τίτλο «Ανάλυση και Σχεδίαση» και αφορά το τμήμα της σχεδίασης του

συστήματος λογισμικού. Σε αυτό το κεφάλαιο γίνεται προσπάθεια να περιγραφεί η αρχιτεκτονική και η λειτουργία του συστήματος με την παρουσίαση διαγραμμάτων της δομής του συστήματος και περιπτώσεων χρήσης του.

Το κεφάλαιο 4 με τίτλο «Υλοποίηση», αναφέρεται στα εργαλεία που χρησιμοποιήθηκαν στην εφαρμογή και επεξηγηματικές πληροφορίες σχετικά με αυτή, οι οποίες λειτουργούν επικουρικά στα έγγραφα του κεφαλαίου 3.

Το κεφάλαιο 5 έχει τίτλο «Έλεγχος» και περιλαμβάνει τον Οδηγό Εγκατάστασης του συστήματος και τα αποτελέσματα χρήσης του.

Τέλος, στον «Επίλογος» (κεφάλαιο 6) περιλαμβάνεται η σύνοψη της διπλωματικής εργασίας, καθώς και τα συμπεράσματα που εξήχθησαν με την ολοκλήρωσή της. Το κεφάλαιο 7 περιλαμβάνει τη Βιβλιογραφία και το 8ο (Παράρτημα) αναλυτικά τον κώδικα της εφαρμογής.

2 *Περιγραφή Θέματος*

Στο κεφάλαιο αυτό παρουσιάζονται οι σχετικές με το θέμα και το σύστημα τεχνολογίες, δίνεται η γενική περιγραφή του προς ανάπτυξη συστήματος και προσδιορίζονται οι στόχοι του.

2.1 Σχετικές τεχνολογίες

2.1.1 Grid

Με τον όρο Grid εννοούμε το σύνολο της υποδομής και των υπηρεσιών για τη δημιουργία ενός ενιαίου υπολογιστικά περιβάλλοντος, που αν και γεωγραφικά κατανεμημένο, εμφανίζεται ως μια συμπαγής υπολογιστική πλατφόρμα σε όλους τους χρήστες.

Χαρακτηριστικό της Grid τεχνολογίας είναι η διασύνδεση ετερογενών υπολογιστικών και όχι μόνο συστημάτων, με αποτέλεσμα τη δημιουργία νέων συνόλων υπηρεσιών με αυξημένες υπολογιστικές δυνατότητες και νέους τρόπους αξιοποίησης των διαμοιραζόμενων πόρων.[FK98]

2.1.1.1 Γιατί Grid;

Τι θα σήμαινε για σας αν θα μπορούσατε :

- να αναλύσετε την αξία ενός επενδυτικού χαρτοφυλακίου σε λεπτά αντί για ώρες;
- να ενωθείτε με ερευνητικές ομάδες από όλο τον κόσμο και να επωφεληθείτε από την άμεση ενημέρωση όλων των εξελίξεων;
- να μειώσετε τον χρόνο σχεδίασης των προϊόντων σας στο μισό μειώνοντας εξίσου και τυχόν ατέλειες του;

Πανεπιστημιακά εργαστήρια και άλλοι ερευνητικοί οργανισμοί χρησιμοποιούν τεχνολογία Grid εδώ και αρκετά χρόνια, αντιμετωπίζοντας και λύνοντας πολλά σημαντικά προβλήματα που μέχρι πρότινος φαίνονταν άλυτα. Οι ανταγωνιστικοί ρυθμοί στον ερευνητικό και επαγγελματικό τομέα απαιτούν συνεχείς καινοτομίες, έτσι ώστε να παράγουν αξιόπιστα προϊόντα και υπηρεσίες. Υπάρχει μια έντονη αλλαγή στον τρόπο αξιοποίησης της υπολογιστικής ισχύος, στον τρόπο πρόσβασης αλλά και στον τρόπο αξιολόγησής της. Η χρήση λοιπόν τεχνολογίας Grid θα μπορούσε να βελτιώσει ραγδαία την αποδοτικότητα οργανισμών που αντιμετωπίζουν τις προκλήσεις μιας σύγχρονης απαιτητικής αγοράς.

2.1.1.2 Τα πλεονεκτήματα του Grid

Τα σημερινά λειτουργικά περιβάλλοντα πρέπει να είναι πιο ανθεκτικά, ευέλικτα και ολοκληρωμένα από ποτέ. Πολλοί οργανισμοί σε όλο τον κόσμο βρίσκουν σημαντικά πλεονεκτήματα χρησιμοποιώντας Grid σε ουσιαστικές επαγγελματικές διαδικασίες επιτυγχάνοντας επαγγελματικό και τεχνολογικό κέρδος.

Επαγγελματικό Κέρδος

Επιτάχυνση του χρόνου με αποτελέσματα:

• βελτίωση της παραγωγικότητας και της συνεργασίας

• επίλυση προβλημάτων που θεωρούνταν μέχρι πριν άλυτα

Ενεργοποίηση συνεργασίας και προώθηση της λειτουργικής ευελιξίας:

• ενοποίηση όχι μόνο των τεχνολογιών Internet (IT) αλλά και των ανθρώπινων πόρων

• δημιουργία 'εικονικών οργανισμών' (VOs) από απομακρυσμένα τμήματα και επιχειρήσεις, με σκοπό την κοινή χρήση πόρων και πληροφοριών

Αποτελεσματική προσαρμογή στις διάφορες ανάγκες της αγοράς:

• δημιουργία ευέλικτων και ανθεκτικών λειτουργικών υποδομών

• αντιμετώπιση γρήγορων διακυμάνσεων στις απαιτήσεις/ανάγκες του πελάτη

Αύξηση παραγωγικότητας:

• απεριόριστη πρόσβαση του χρήστη σε υπολογιστικούς, αποθηκευτικούς και πόρους πληροφοριών οποιαδήποτε στιγμή

• ευκολία κινήσεων του εργαζόμενου στη φάση σχεδίασης του προϊόντος, του ερευνητικού έργου κλπ.

Τεχνολογικό Κέρδος

Βελτιστοποίηση υποδομών:

• συγκέντρωση της διαχείρισης φόρτου εργασίας (workload)

• παροχή αποθηκευτικού χώρου για εφαρμογές με υψηλές απαιτήσεις

• μείωση των χρονικών επαναλήψεων

Αύξηση της πρόσβασης σε δεδομένα και της συνεργασίας:

• κοινόχρηστα δεδομένα και παγκόσμια διανομή τους

• δημιουργία συνεργασιών μεταξύ οργανισμών και επιχειρήσεων

2.1.1.3 Μια άλλη ερμηνεία του Grid

Η υπαρκτή πλέον τεχνολογική υποδομή και τα πολλά πλεονεκτήματα της τεχνολογίας Grid, όχι μόνο σε ερευνητικό αλλά και σε επαγγελματικό τομέα, έχουν φέρει στην πραγματικότητα πολλές εφαρμογές Grid. Ακούμε για Compute Grids, Data Grids, Bio Grids, Cluster Grids, Tera Grids, αλλά ποια είναι η διαφορά και ποια η ομοιότητα όλων αυτών; Συνολικά θα μπορούσαμε να πούμε ότι τα Grids θα έπρεπε να αξιολογηθούν σε επίπεδο εφαρμογών, επαγγελματικής αξίας, επιστημονικής αποτελεσματικότητας και όχι αρχιτεκτονικής. Ο ορισμός που δώσαμε στην αρχή, σε αυτό το σημείο δεν φαίνεται αρκετός και θα παρουσιάσουμε λίγο πιο αναλυτικά ποιο σύστημα μπορεί να θεωρηθεί Grid. Μπορούμε να συγκεντρώσουμε τα χαρακτηριστικά ενός τέτοιου συστήματος σε τρεις ιδιότητες που όταν και οι τρεις ικανοποιούνται, τότε μιλάμε για Grid. Αρχικά πρέπει το σύστημα αυτό να συντονίζει τους πόρους, οι οποίοι δεν πρέπει να υποβάλλονται σε κεντρικό έλεγχο. Η κεντρική ιδέα της τεχνολογίας Grid είναι η συνεργασία και η ανάπτυξη κατάλληλων προτύπων ασφαλείας και επικοινωνίας πόρων που ανήκουν σε διαφορετικά σύνολα διαχείρισης , πχ οι υπολογιστές ενός εργαστηρίου, τα εργαστήρια ενός πανεπιστημίου μεταξύ τους, τα πανεπιστήμια μιας πόλης κλπ. Η ύπαρξη κεντρικού ελέγχου σε ολόκληρο το πλέγμα υπολογιστών θα το μετέτρεπε αυτομάτως σε τοπικό σύστημα διαχείρισης. Η δεύτερη βασική ιδιότητα είναι η χρήση γνωστών πρωτοκόλλων για την αντιμετώπιση θεμελιωδών προβλημάτων όπως η πρόσβαση σε πόρους, η πιστοποίηση του χρήστη, η εξουσιοδότηση του χρήστη κλπ. Για να είναι το σύστημα αυτό όσο το δυνατόν

προσβάσιμο από περισσότερους χρήστες, έχει μεγάλη σημασία τα βασικά πρωτόκολλα επικοινωνίας να είναι διαδεδομένα, πρότυπα γενικής χρήσης. Τέλος, το σύστημα αυτό πρέπει να αποδίδει σημαντική ποιότητα υπηρεσιών προς του χρήστες του. Αυτό σημαίνει ότι η ομαλή συνεργασία όλων των πόρων θα πρέπει να έχει αποτέλεσμα παροχή υπηρεσιών με συγκεκριμένο χρόνο απόκρισης, απόδοση, διαθεσιμότητα και ασφάλεια, έτσι ώστε η παραγωγικότητα και η χρησιμότητα των συνδυασμένων πόρων να είναι πολύ μεγαλύτερη από τις δυνατότητές τους ξεχωριστά. [F02]

2.1.1.4 Ιστορία του Grid

Το ιστορικό των τεχνολογιών διασύνδεσης των υπολογιστών αρχίζει στις αρχές της δεκαετίας του 70 στην Αμερική από την ερευνητική ομάδα ARPA (Advanced Research Projects Agency) που είχε δημιουργηθεί από την κυβέρνηση για στρατιωτικούς σκοπούς. Από την ομάδα αυτή αναπτύχθηκαν πολλά σημαντικά πρωτόκολλα και τεχνολογίες όπως το TCP/IP. Υπό την καθοδήγηση του Dr. J.C.R. Licklider και την συμβολή ερευνητών από διάφορα πανεπιστήμια της χώρας η έρευνα κατέληξε στη δημιουργία του πρώτου δικτύου υπολογιστών, προάγγελο του Internet, γνωστό ως ARPANET, στα 50 kbps. Η ερευνητική δραστηριότητα στον τομέα των δικτύων υπολογιστών συνεχίστηκε, με αποτέλεσμα την δημιουργία του NSFNET [1986], δικτύου στα 56Kbps που συνέδεε τα πέντε NSF κέντρα υπέρ-υπολογιστών.

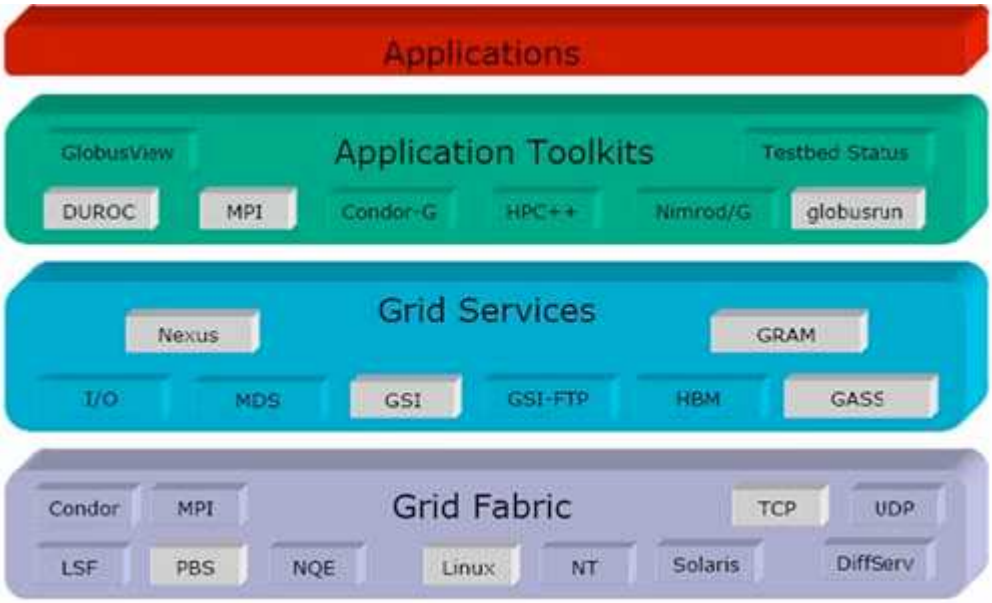
Ως συνέχεια και εξέλιξη αυτών των τεχνολογιών μπορούμε να θεωρήσουμε το πρόγραμμα Condor [1988] του πανεπιστημίου του Wisconsin. Το σύστημα αυτό είναι ένας ‘διαχειριστής

φόρτου εργασίας’ (workload manager), με δυνατότητες παρακολούθησης και διαχείρισης πόρων, δρομολόγησης εργασιών και αποτελεί το πρώτο πρόγραμμα με κατεύθυνση προς

την αξιοποίηση των Grid υπηρεσιών.

Η ανάπτυξη δικτύων υψηλών ταχυτήτων και η ανάγκη για μεγάλη επεξεργαστική ισχύ οδήγησε σε έντονη ερευνητική δραστηριότητα στον τομέα των Grid τεχνολογιών. Η έρευνα αυτή κατέληξε σε ενδιαφέροντα αποτελέσματα με πιο σημαντικά τα προγράμματα LEGION [1993], SRB [1997], GLOBUS [1998]. Το πρώτο βασίζεται στην ιδέα του 'εικονικού υπολογιστή (virtual computer) : όλοι οι πόροι συνδεδεμένοι μεταξύ τους, εμφανίζονται στον χρήστη ως μία εικονική μηχανή, με αρκετά μειονεκτήματα όμως, όπως η πολύπλοκη υλοποίηση και η μικρή αποδοτικότητα. Το SRB (Storage Resource Broker) ήταν μια πλατφόρμα διαχείρισης αποθηκευτικών πόρων που βοήθησε πολύ στην ανάπτυξη των Grid τεχνολογιών, αφού αντιμετώπισε τα προβλήματα μεταφοράς δεδομένων σε Grid περιβάλλον.

Τέλος, το πιο διαδεδομένο σύστημα διαχείρισης Grid υπηρεσιών είναι το GLOBUS, που αναπτύχθηκε στο Argonne National Lab στο πανεπιστήμιο του Berkeley. Το GLOBUS προτυποποίησε πρωτοκόλλα για τη ασφάλεια, μεταφορά δεδομένων, ανακάλυψη πόρων και εκτέλεση εργασιών. Λειτουργεί σε χαμηλό επίπεδο και είναι ανεπτυγμένο σε επίπεδα υπηρεσιών:



Εικόνα 1 : Βασική Grid Αρχιτεκτονική

Η συνέχεια γίνεται με την ανάπτυξη των Web Services [2001] , υπηρεσιών που είναι προσβάσιμες μέσω του διαδικτύου και χρησιμοποιούν συγκεκριμένα πρωτόκολλα περιγραφής (XML, SOAP, WSDL) και τέλος με την εγκαθίδρυση του OGSA (Open Grid Service Architecture) [2002] ως κύριας αρχιτεκτονικής της Grid τεχνολογίας. Το πρότυπο αυτό είναι το πιο σημαντικό για τις τεχνολογίες του Grid, αφού περιγράφει τις δυνατότητες του συστήματος αυτού να αναλύσει την λειτουργία του σε όλα τα επίπεδά του.

2.1.2 OGSA

Οι συντονισμένες μελέτες του GGF (Global Grid Forum) - ενός οργανισμού αποτελούμενου από χρήστες, προγραμματιστές, ερευνητές με σκοπό την ανάπτυξη προτύπων για την υποστήριξη των Grid τεχνολογιών - κατέληξε πρόσφατα (2002) στην συγγραφή του OGSA. Το τελευταίο είναι ένα κείμενο που περιγράφει την δομή ενός Grid συστήματος και συγκεντρώνει όλα τα πρότυπα που χρησιμοποιεί η αρχιτεκτονική αυτή σε κάθε επίπεδο της.

2.1.2.1 Ποιοι είναι οι στόχοι του OGSA ;

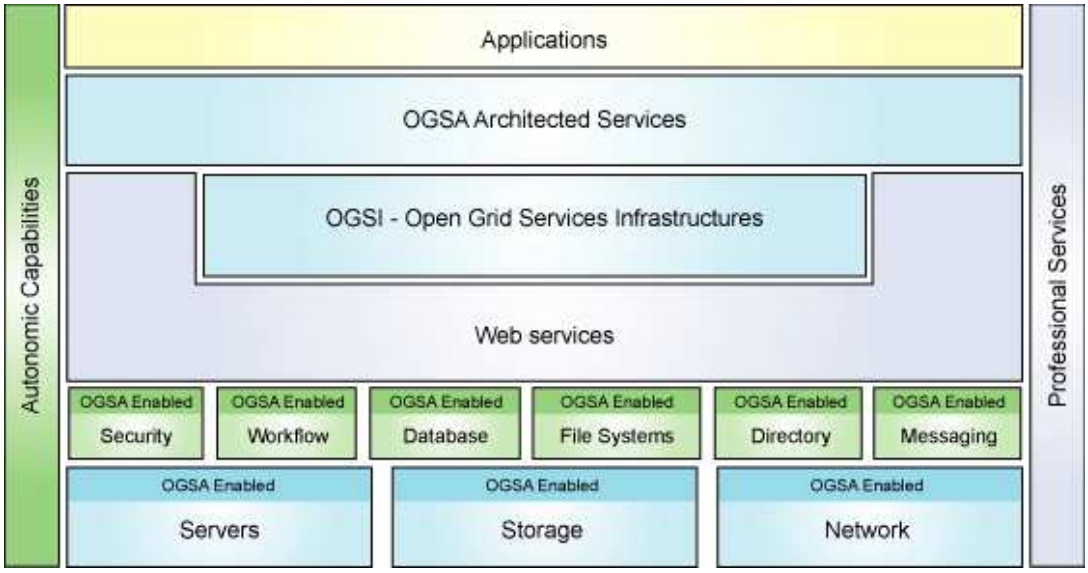
Οι κύριοι στόχοι του OGSA είναι :

- να διαχειρίζεται πόρους απομακρυσμένων ετερογενών συστημάτων
- να αποδίδει σημαντική ποιότητα στις υπηρεσίες που παρέχει (QoS)
- να παρέχει τις βάσεις για αυτόνομες λύσεις διαχείρισης. Η διαφορετικότητα των πόρων που συμμετέχουν σε ένα Grid και ο δυναμικός τρόπος συνεργασίας τους απαιτεί ένα σύστημα διαχείρισης που θα προσαρμόζεται σύμφωνα με τις εκάστοτε ανάγκες.

• Να καθορίζει γνωστά πρότυπα και πρωτόκολλα λειτουργίας. Η δυναμική εισαγωγή και συνεργασία ετερογενών συστημάτων στο δίκτυο του Grid, βασίζεται στην υιοθέτηση γνωστών, ευρείας χρήσης προτύπων.

• Να εκμεταλλεύεται υπάρχουσες τεχνολογίες και να τις προσαρμόζει στο Grid, αν είναι εφικτό. Το OGSA βασίζεται στη τεχνολογία των Web Services (υπηρεσίες διαδικτύου), ένα τομέα πολύ διαδεδομένο αυτή την εποχή.

2.1.2.2 Η αρχιτεκτονική του OGSA



Εικόνα 2 : OGSA Αρχιτεκτονική

Το OGSA αποτελείται από τέσσερα βασικά επίπεδα :

• τους πόρους (φυσικούς και λογικούς)

• τις Web Services συμπεριλαμβανομένου και της OGSI υποδομής που έχει άμεση σχέση με τις υπηρεσίες

• τις υπηρεσίες σχεδιασμένες από το OGSA

• τις Grid εφαρμογές

Επίπεδο φυσικών και λογικών πόρων

Αυτό είναι το πιο χαμηλό επίπεδο από τα τέσσερα και έχει να κάνει με τους πόρους που συμμετέχουν στο Grid. Η έννοια του 'πόρου' στο OGSA είναι πολύ σημαντική και δεν είναι συγκεκριμένη. Ως πόρος (resource) σε ένα Grid μπορεί να θεωρηθεί από τον επεξεργαστή ενός υπολογιστή, μέχρι το τμήμα υπολογιστών μιας μεγάλης εταιρίας. Επίσης εκτός από υπολογιστικές μονάδες, συμμετέχουν και μονάδες αποθήκευσης, βάσεις δεδομένων, δικτυακές υποδομές κλπ. Αυτά που αναφέρθηκαν παραπάνω είναι κυρίως οι φυσικοί πόροι. Εκτός από αυτούς έχουμε και τους λογικούς, κάποια ενδιάμεσα συστήματα (middleware) που χρησιμοποιώντας τους φυσικούς πόρους, προσφέρουν στοιχειώδεις υπηρεσίες, όπως διαχείριση αρχείων ή βάσεων δεδομένων, στο παραπάνω επίπεδο.

Επίπεδο υπηρεσιών διαδικτύου (Web Services)

Μία πολύ βασική θεώρηση του OGSA είναι ότι όλοι οι πόροι του συστήματος αντιστοιχίζονται με υπηρεσίες. Έτσι η OGSI υποδομή (Open Grid Services Infrastructure) χρησιμοποιεί συγκεκριμένες, γνωστές υπηρεσίες δικτύου και πρωτόκολλα όπως XML και WSDL για να δημιουργήσει επαφές και διασυνδέσεις κάθε gird υπηρεσίας με τους πόρους του συστήματος. Η OGSI επεκτείνει τις δυνατότητες των δικτυακών υπηρεσιών έτσι ώστε να παρέχει δυναμικές και αξιόπιστες υπηρεσίες όπως απαιτείται για τη μοντελοποίηση των πόρων.

Επίπεδο Grid υπηρεσιών

Οι υπηρεσίες διαδικτύου και οι δυνατότητες του OGSI παρέχουν τη βασική υποδομή για το επόμενο επίπεδο, των Grid υπηρεσιών. Αυτή την εποχή υπάρχει μεγάλη δραστηριότητα προς την κατεύθυνση προσδιορισμού και προτυποποίησης τέτοιων υπηρεσιών, όπως υπηρεσίες υπολογισμού, πληροφοριών, πυρήνα, κ.α. Καθώς οι υλοποιήσεις αυτών των

υπηρεσιών θα αρχίσουν να εμφανίζονται, η OGSA αρχιτεκτονική θα γίνεται όλο και πιο χρήσιμη βασιζόμενη πάντα στις υπηρεσίες (**S**ervice **O**riented **A**rchitecture).

Επίπεδο εφαρμογών Grid

Με το πέρασμα του χρόνου και όσο οι Grid υπηρεσίες θα αναπτύσσονται, καινούργιες εφαρμογές βασισμένες στην Grid τεχνολογία θα κάνουν την εμφάνισή τους. Αυτές οι εφαρμογές θα χρησιμοποιούν μια ή και περισσότερες Grid υπηρεσίες του προηγούμενου επιπέδου.

2.1.3 Γενικά για τις υπηρεσίες

Τον τελευταίο καιρό έχει γίνει αρκετά μεγάλη συζήτηση γύρω από το θέμα των υπηρεσιών των εφαρμογών. Οι υπηρεσίες τείνουν να γίνουν τμήματα της εφαρμογής που αθροιστικά σχηματίζουν το περιβάλλον αυτής. Φυσικά δεν αποτελούν απλά ένα κομμάτι της εφαρμογής, αλλά έχουν χαρακτηριστικά που τις μετατρέπουν σε μέρος μιας αρχιτεκτονικής προσανατολισμένης στις υπηρεσίες (**S**ervice **O**riented **A**rchitecture).

Ένα από τα χαρακτηριστικά αυτά είναι η αυτονομία από άλλες υπηρεσίες. Αυτό σημαίνει ότι κάθε υπηρεσία είναι υπεύθυνη για το δικό της εύρος λειτουργίας, περιορίζοντας έτσι και εξειδικεύοντάς την σε συγκεκριμένες επαγγελματικές χρήσεις.

Αυτός ο σχεδιασμός έχει ως αποτέλεσμα τη δημιουργία ανεξάρτητων μονάδων, ελαστικά συνδεδεμένων μεταξύ τους με κάποιο πρότυπο πλαίσιο επικοινωνίας. Εξαιτίας αυτής της ανεξαρτησίας που απολαμβάνουν οι υπηρεσίες στο πλαίσιο αυτό, η προγραμματιστική λογική που κάθε υπηρεσία χρησιμοποιεί, δεν χρειάζεται να προσαρμόζεται σε συγκεκριμένη πλατφόρμα ή τεχνολογία. Είναι χαρακτηριστικό των υπηρεσιών ενός

πλαισίου, ο πολυμορφισμός των μερών του με ταυτόχρονη ομαλή συνεργασία.

2.1.3.1 XML Web services (Διαδικτυακές Υπηρεσίες)

Ο πιο ευρέως διαδεδομένος και επιτυχημένος τύπος υπηρεσιών είναι οι XML Services γνωστές ως Web Services.

Αυτός ο τύπος υπηρεσίας έχει δύο βασικές προαπαιτήσεις:

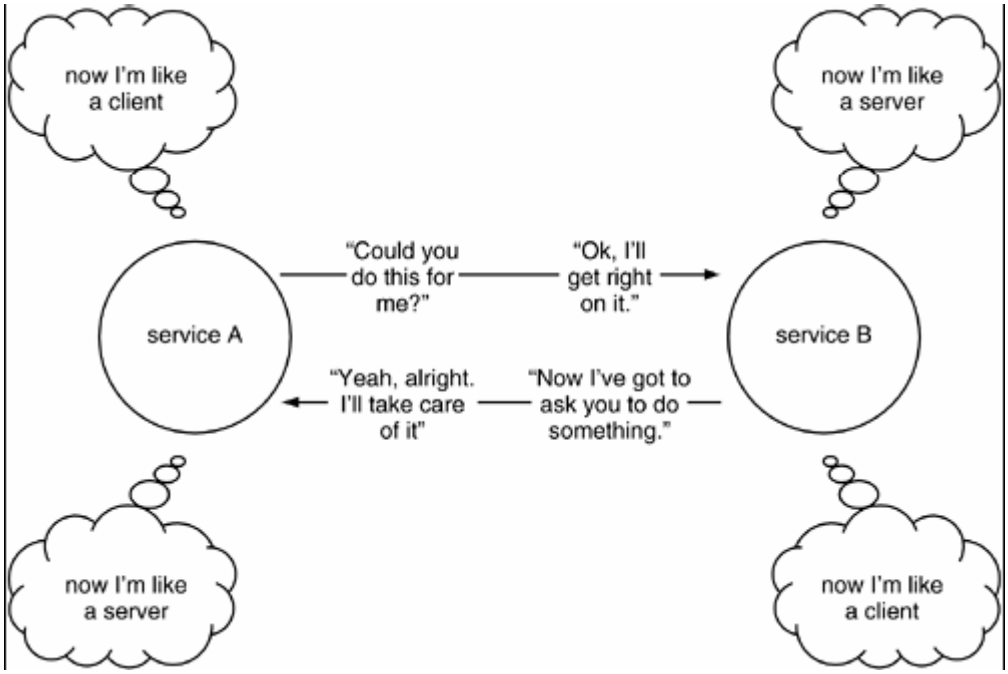
- επικοινωνεί μέσω πρωτοκόλλου Internet (κυρίως HTTP)
- στέλνει και δέχεται δεδομένα μέσα από XML αρχεία

Η ευρεία αποδοχή του μοντέλου των Web Services είχε ως αποτέλεσμα την ανάγκη πρόσθετων τεχνολογιών που βασίζονται σε αυτές και τη δημιουργία καινούργιων προτύπων.

Έτσι η “παραγωγή” τέτοιων υπηρεσιών απαιτεί:

- τη περιγραφή της υπηρεσίας αναλυτικά, τουλάχιστον με ένα WSDL έγγραφο
- τη δυνατότητα μεταφοράς ενός XML εγγράφου χρησιμοποιώντας SOAP μέσω HTTP.

Επιπρόσθετα είναι συνηθισμένο μια υπηρεσία να λειτουργεί και ως πελάτης (client/requestor) και ως πάροχος (provider) υπηρεσίας. Αναλόγως λοιπόν με τη δραστηριότητα της υπηρεσίας κάθε στιγμή, μετατρέπεται από το ένα στο άλλο. Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα αυτής της συμπεριφοράς:

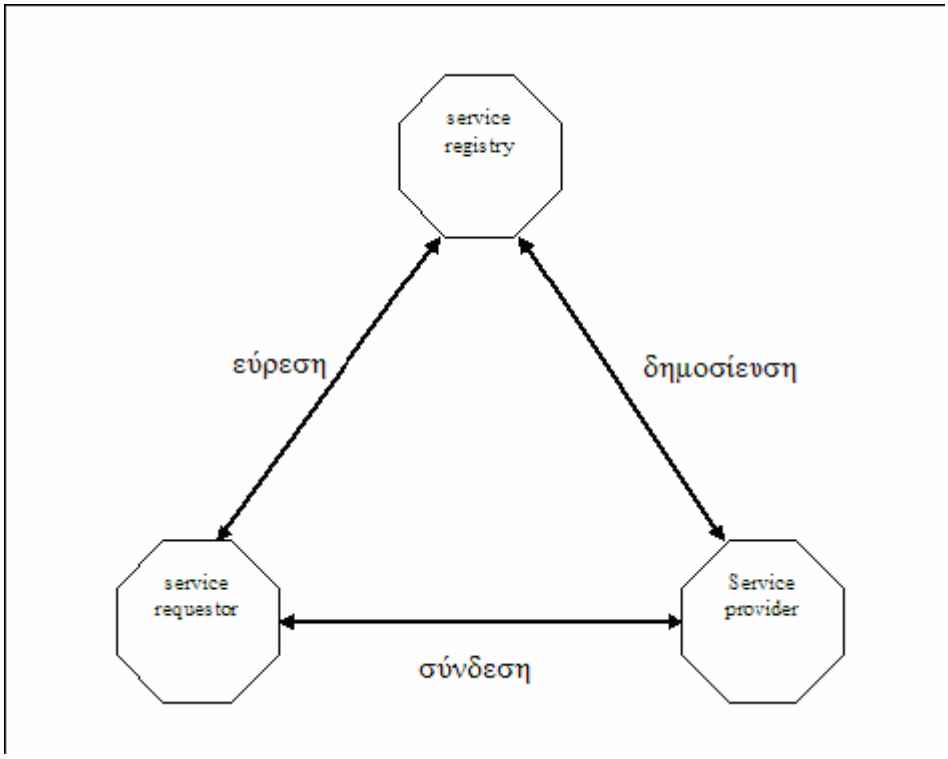


Εικόνα 3 : Ανταλλαγή ρόλων μεταξύ Web services κατά τη διάρκεια μιας επικοινωνίας

2.1.4 Service Oriented Architecture (Υπηρεστρεφής Αρχιτεκτονική)

Το να προσαρμόσεις σε μια εφαρμογή κάποιες υπηρεσίες (Web Services) δεν είναι μια δύσκολη διαδικασία και επιπρόσθετα θα αποδώσει στην εφαρμογή πρόσθετα λειτουργικά χαρακτηριστικά. Τα χαρακτηριστικά αυτά όμως δεν δημιουργούν μια υπηρεσιοστρεφή αρχιτεκτονική (Service Oriented Architecture), αφού η διαφορά τους είναι πιο μεγάλη.

Το SOA πρότυπο είναι ένα σχεδιαστικό μοντέλο με κύριο χαρακτηριστικό την ενσωμάτωση λογικής εφαρμογών μέσα σε υπηρεσίες που θα αλληλεπιδρούν μέσω επικοινωνιακών πρωτοκόλλων. Έτσι η υιοθέτηση σε μια εφαρμογή, SOA δομής σημαίνει αυτόματα την αποδοχή κάποιων σχεδιαστικών αρχών και πρόσθετων τεχνολογιών ως βασικού τμήματος του τεχνικού περιβάλλοντος της.



Εικόνα 4 : Λειτουργία SOA

Σε επίπεδο σχεδιασμού συστημάτων, η χρήση της τεχνολογίας των web services οδηγεί στην υιοθέτηση της λεγόμενης service-oriented αρχιτεκτονικής. Οι βασικοί ρόλοι και λειτουργίες στην αρχιτεκτονική αυτή παρουσιάζονται στην Εικόνα 4[ACK+04]. Η αρχιτεκτονική αυτή υποδεικνύει μια σχέση εξυπηρετητή-πελάτη (server-client) ανάμεσα στον προμηθευτή υπηρεσιών (service provider, που παίζει το ρόλο του server) και τον ζητούντο την υπηρεσία (service-requestor, που παίζει το ρόλο του client). Ο service-provider είναι αυτός που παρέχει την υπηρεσία δεχόμενος μηνύματα κλήσεις από τους requestors. Είναι επίσης υπεύθυνος για τη δημιουργία της περιγραφής της υπηρεσίας (service description) και τη δημοσίευσή της σε κάποιο κατάλογο-οδηγό υπηρεσιών (Universal Description, Discovery and Integration- UDDI). Ο service requestor αναζητά την υπηρεσία και την περιγραφή της σε κάποιο κατάλογο υπηρεσιών (service registry) και στη συνέχεια καλεί κατάλληλα την επιθυμητή υπηρεσία. Ο κατάλογος υπηρεσιών φέρνει ουσιαστικά τις δύο πλευρές, client και server σε επαφή. Η συνέχεια αφορά μόνο τις δύο

άλλες συμμετέχουσες μονάδες (service requestor και service provider).

2.1.5 Web Services Description Language (WSDL)

Οι Web Services χρειάζεται να ορίζονται με συγκεκριμένο τρόπο έτσι ώστε να μπορούν να εντοπιστούν και χρησιμοποιηθούν από άλλες υπηρεσίες και εφαρμογές. Για αυτό το σκοπό δημιουργήθηκε από τον οργανισμό W3C, μια γλώσσα περιγραφής γνωστή ως WSDL (Web Services Description Language).

Η γλώσσα αυτή είναι μια XML δομή εγγράφου, που περιέχει όλες τις πληροφορίες σχετικά με τα δεδομένα εισόδου και εξόδου, τις μεθόδους και ότι άλλο χρειάζεται να γνωρίζει κάποιος για την ακριβή χρήση μιας Web υπηρεσίας.

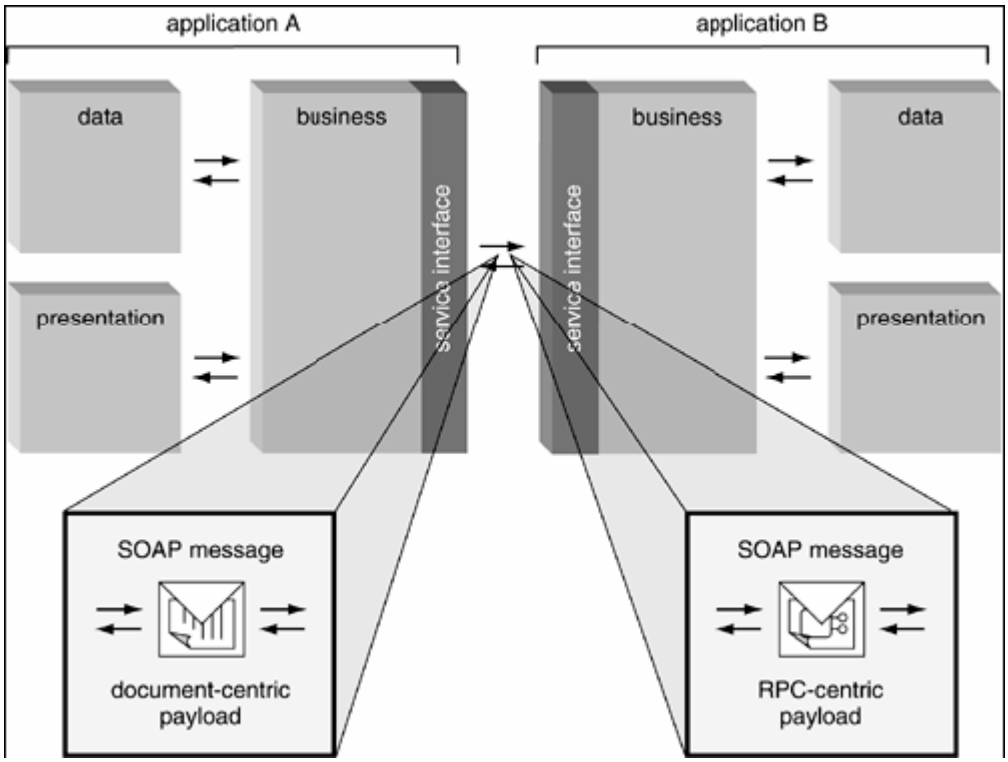
2.1.6 Simple Object Access Protocol (SOAP)

Αν και αρχικά είχε θεωρηθεί ως η τεχνολογία που θα γεφυρώσει το κενό μεταξύ ανόμοιων πλατφορμών βασισμένων σε RPC επικοινωνία, το SOAP έχει εξελιχθεί στο ευρέως χρησιμοποιούμενο πρότυπο επικοινωνίας για τη χρήση XML Υπηρεσιών Διαδικτύου (Web Services). Μετά από αυτή την κατάσταση γίνεται πολλές φορές η παράφραση του ακρωνύμου από Simple Object Access Protocol σε Service-Oriented Architecture (or Application) Protocol.

Το πρωτόκολλο SOAP διαμορφώνει ένα πρότυπο μήνυμα που αποτελείται από ένα XML έγγραφο ικανό να περιγράψει δεδομένα όπως RPC κλήσεις κ.α. Το μήνυμα αυτό μεταφέρεται μεταξύ των υπηρεσιών και των εφαρμογών, χρησιμοποιώντας κυρίως το HTTP πρωτόκολλο δικτύου. Με τον τρόπο αυτό ολοκληρώνεται το πλαίσιο λειτουργίας και

επικοινωνίας στην SOA δομή, αφού με την βοήθεια της περιγραφής WSDL είναι εφικτή η επικοινωνία και συνεργασία οποιωνδήποτε υπηρεσιών στο δίκτυο.

Στην Εικόνα 5 βλέπουμε τη χρήση του πρωτοκόλλου σε εφαρμογές είτε για προτυποποιημένη RPC επικοινωνία είτε για γενική χρήση μεταφοράς μηνύματος.



Εικόνα 5 : RPC και Document Centric επικοινωνία

Το πρωτόκολλο SOAP καθορίζει ένα XML έγγραφο με μια συγκεκριμένη δομή που μέσα του θα εμπεριέχονται οι πληροφορίες. Η βασική δομή ενός τέτοιου εγγράφου είναι η παρακάτω:

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

```
        <soap:Body>
        </soap:Body>
    </soap:Envelope>
```

Εσωτερικά του “σώματος” (Body) του εγγράφου, ενσωματώνεται η πληροφορία που θέλουμε να μεταφέρουμε από τη μία υπηρεσία στην άλλη. Η τεχνολογία που χρησιμοποιήθηκε για απομακρυσμένη κλήση υπηρεσιών είναι η SOAP-RPC, (επέκταση της τεχνολογίας Remote Procedure Call) η οποία εισάγει στο SOAP έγγραφο τις απαιτούμενες πληροφορίες, σύμφωνα και με το WSDL της υπηρεσίας που θα κληθεί και στέλνει το μήνυμα.

Όπως είδαμε και στο σχήμα προηγουμένως, εκτός από την χρήση που περιγράφηκε, το SOAP μπορεί να μεταφέρει οποιαδήποτε άλλη πληροφορία ενσωματωμένη σε XML δομή, μέσα στο “φάκελο” του SOAP μηνύματος, αρκεί βέβαια ο παραλήπτης να γνωρίζει πώς να αποκωδικοποιήσει το έγγραφο αυτό.

2.1.7 Ποιότητα Υπηρεσίας (Quality of Service)

Ο όρος QoS (Quality of Service) χρησιμοποιήθηκε αρχικά ως ορολογία στα δίκτυα υπολογιστών για να αποδώσει την ικανότητα ενός δικτύου να παρέχει καλύτερες υπηρεσίες για συγκεκριμένη κίνηση (network traffic) χρησιμοποιώντας τα διάφορα πρωτόκολλα δικτύου. Η ανάγκη για αξιόπιστα δίκτυα και η βελτίωση των υπηρεσιών που προσέφεραν αυτά δημιούργησε αυτή την παράμετρο, που τώρα θεωρείται ένας πολύ βασικός παράγοντας αξιολόγησης.

2.1.7.1 Η έννοια της Ποιότητας Υπηρεσίας στο Grid

Η ανάπτυξη των Grid στον επιχειρηματικό κόσμο (business Grids) οδήγησε στην υιοθέτηση

της ορολογίας αυτής και στον χώρο των Grid υπηρεσιών. Η σημασία της ποιότητας των παρερχόμενων υπηρεσιών σε ένα Grid σύστημα είναι πολύ μεγάλη, αφού αυτό που κάνει μια υπηρεσία προσιτή στον πελάτη, εκτός από την ίδια την υπηρεσία, είναι η ποιότητά της.

Επίσης η ιδιαιτερότητα των Grid συστημάτων πρώτον ως προς τη συνεργασία ετερογενών συστημάτων και δεύτερον ως προς την αρχιτεκτονική τους (Service Oriented Architecture) έκανε πιο εμφανή την ανάγκη προσδιορισμού της ποιότητας των υπηρεσιών. Όσον αφορά το πρώτο, η συμμετοχή πολλών διαφορετικών συστημάτων στο ίδιο Grid απαιτεί μια κοινή παράμετρο αξιολόγησης αυτών αλλά και των υπηρεσιών που αυτά προσφέρουν. Από την άλλη, η αρχιτεκτονική που χρησιμοποιείται σε τέτοια περιβάλλοντα είναι βασισμένη στις υπηρεσίες, έτσι η εισαγωγή του παράγοντα QoS είναι εφικτή σε όλα τα επίπεδα της δομής του Grid.

Τα χαρακτηριστικά του QoS μπορούν να διαχωριστούν σε δύο κατηγορίες : αυτά που βασίζονται στα ποσοτικά και αυτά που βασίζονται στα ποιοτικά χαρακτηριστικά μιας Grid υποδομής. Τα πρώτα αναφέρονται σε ιδιότητες όπως η απόδοση του επεξεργαστή, η καθυστέρηση του δικτύου, ο διαθέσιμος αποθηκευτικός χώρος κλπ., ενώ τα δεύτερα έχουν σχέση με την αξιοπιστία των υπηρεσιών και την ικανοποίηση του χρήστη.

Ο προσδιορισμός και η συλλογή των πιο συγκεκριμένων παραμέτρων για τον καθορισμό του QoS δεν είναι πάντα μια εύκολη διαδικασία. Κάποιες βασικές πληροφορίες μπορούν να εξαχθούν από προγράμματα παρακολούθησης (monitoring) και άλλα συστήματα μοντέλα προτυποποίησης των πόρων [CIM].

Από την άλλη υπάρχουν αρκετές παράμετροι που εξαιτίας της ιδιομορφίας του Grid συστήματος (δυναμική εισαγωγή ετερογενών υπολογιστικών και μη συστημάτων) δεν είναι εύκολο να προσδιοριστούν. Προσεγγίζοντάς το από τη πλευρά του πελάτη ενός business Grid, και παρατηρώντας τι πληροφορίες το Grid θα μπορούσε να δώσει, καταλήγουμε στις παρακάτω κατηγορίες QoS πληροφορίας:

• Διαθεσιμότητα : η πληροφορία αυτή μπορεί να δοθεί με μορφή ένδειξης, για το αν είναι διαθέσιμο το στιγμιότυπο της υπηρεσίας (service up-and-running), και άλλα απαραίτητα αρχεία και βιβλιοθήκες.

• Απόδοση : ο προσδιορισμός αυτής της παραμέτρου είναι αρκετά δύσκολος εξαιτίας της αδυναμίας μοντελοποίησης ενός γενικού τρόπου εξαγωγής της πληροφορίας.

Παρόλα αυτά μπορούμε να θεωρήσουμε ότι υπολογίζοντας το χρόνο απόκρισης, την ταχύτητα του επεξεργαστή και κάποιες άλλες τιμές ενδεικτικές για την αποδοτικότητα και ικανότητα του συστήματος, έχουμε μια αφηρημένη εικόνα για την παράμετρο αυτή.

• Αξιοπιστία : η παράμετρος αυτή πηγάζει από την επεξεργασία των αποτελεσμάτων της διαθεσιμότητας. Η διαθεσιμότητα μιας υπηρεσίας για μεγάλο διάστημα χρόνου, της αποδίδει αξιοπιστία.

• Κόστος : η παράμετρος αυτή έχει να κάνει αποκλειστικά με την τιμολόγηση των υπηρεσιών σε σχέση με το χρόνο κλπ.

• Στατιστικά στοιχεία για την ποιότητα του αποτελέσματος (Quality of the Result): τα στοιχεία αυτά επιτυγχάνονται από την αξιολόγηση των υπηρεσιών από τους πελάτες, αποδίδοντας ένα βαθμό ικανοποίησης του χρήστη.

• Στατιστικά στοιχεία για την παραβίαση των SLA συμφωνιών : μια ένδειξη για την κατάσταση μιας υπηρεσίας σε σχέση με το συμφωνηθέν SLA θα αποτελούσε χρήσιμη πληροφορία

Οι δύο πρώτες κατηγορίες ανήκουν στα ποσοτικά χαρακτηριστικά, σύμφωνα με τον αρχικό διαχωρισμό, ενώ τα υπόλοιπα αφορούν την ποιότητα της παρεχομένης υπηρεσίας και ανήκουν στα ποιοτικά.. Παρόλο που οι ιδιότητες οι σχετικές με την ποιότητα της υπηρεσίας (ποιοτικά χαρακτηριστικά) είναι πολύ σημαντικές στην αξιολόγηση του QoS, είναι δύσκολο να υπολογιστούν αντικειμενικά, αφού βασίζονται κατά πολύ στην ανάδραση πληροφορίας

από τον χρήστη. Έτσι θα επικεντρώσουμε την προσοχή μας στις πληροφορίες για την διαθεσιμότητα (availability) και αποδοτικότητα (performance).

2.1.7.2 Η εφαρμογή της Ποιότητας Υπηρεσίας στο Grid

Είναι απαραίτητο οι εφαρμογές που θα εκτελεστούν στο Grid να προσδιορίσουν τις απαιτήσεις τους για QoS ως τα χαρακτηριστικά των πόρων που απαιτούνται για την εκτέλεση αυτή (αποθηκευτικό χώρο, εύρος δικτύου, επεξεργαστική ισχύ κλπ.). Επίσης θα πρέπει να εξασφαλίζεται ο εντοπισμός του κατάλληλου πόρου σύμφωνα με τις απαιτήσεις αυτές. Η έρευνα σε αυτό το θέμα έχει καταλήξει σε ένα γενικό μοντέλο διαχείρισης QoS. Σύμφωνα με αυτό η κάθε εφαρμογή υποβάλει τις απαιτήσεις της στον διαχειριστή Grid πόρων (resource manager), ο οποίος υπεύθυνος για την ανάθεση εργασιών στους πόρους. Έτσι ο κάθε πάροχος πόρων (resource provider) πρέπει να υποστηρίζει διαχειριστή ή δρομολογητή για να δέχεται τις αιτήσεις των εφαρμογών. Εκτός όμως από την ανάγκη ύπαρξης QoS πληροφορίας κατά την εκτέλεση μιας εφαρμογής (on demand), είναι απαραίτητη η ύπαρξη αρχείου πληροφοριών με το ιστορικό των παραμέτρων του QoS για λόγους επεξεργασίας και εξαγωγής χρήσιμων αποτελεσμάτων. Συνολικά λοιπόν, μπορούμε να παρατηρήσουμε ότι αφού έχουμε να κάνουμε με QoS σε Grid, δηλ. με πολλές εφαρμογές σε πολλούς πόρους ταυτόχρονα, το SLA απαιτεί να διευκρινίζει το επίπεδο της υπηρεσίας που ο χρήστης επιθυμεί και ο πάροχος πρέπει να προσφέρει.

2.1.7.3 Η Ιστορία της Ποιότητας της Υπηρεσίας στα Grids

Οι παλαιότερες προσπάθειες για την εισαγωγή της έννοιας του QoS στη διαχείριση των πόρων είχαν μεγάλο ενδιαφέρον και σημαντικά αποτελέσματα. Ο Sahai [SGM+03] πρότεινε την υποστήριξη της QoS πληροφορίας σε SLA οντότητες στα ευρείας χρήσης (εμπορικά) Grid. Αν και πολύ ενδιαφέρουσα προσέγγιση, δεν αναπτύχθηκε σημαντικά και είναι ακόμα σε ερευνητικό στάδιο. Ένα γενικό μοντέλο με την ονομασία SNAP (Service Negotiation and Acquisition Protocol) προτάθηκε από τον Czajkowski [CFK+02]. Το μοντέλο αυτό διακρίνει τρία είδη SLA για την ανάθεση των εργασιών στο Grid, Task SLA (TSLA), Resource SLA (RSLA), Bind SLA (BSLA). Το πρώτο από αυτά εμπεριέχει τις απαιτήσεις για την ποιότητα της υπηρεσίας που ζητήθηκε. Ο Keahey πρότεινε την αρχιτεκτονική VAS (Virtual Application Service)[9] για την διαχείριση QoS στα Grid. Η αρχιτεκτονική αυτή εισήγαγε καινούριους τρόπους διαπραγμάτευσης του επιπέδου της ποιότητα των υπηρεσιών και των απαιτήσεων. Τέλος το επικρατέστερο και πληρέστερο μοντέλο για την υποστήριξη QoS στα Grid είναι το General-purpose Architecture for Reservation and Allocation (GARA) [FKL+99]. Το GARA εξασφαλίζει ότι ο χρήστης ή η εφαρμογή θα λάβει συγκεκριμένη ποιότητα υπηρεσίας από τον διαχειριστή των πόρων. Αν και οι δυνατότητες του είναι αξιόλογες, έχει κάποιους περιορισμούς αφού δεν είναι συμβατό με το OGSA και δεν υποστηρίζει SLA.

2.2 Στόχος

Στην ενότητα αυτή περιγράφεται η ανάπτυξη των τεχνολογιών που αναλύθηκαν προηγουμένως και η χρήση τους πλέον εμπορικά. Αρχικά θα γίνει μια προσέγγιση της τεχνολογίας του Grid στον επιχειρηματικό τομέα και στη συνέχεια θα εξετάσουμε την τεχνολογία αυτή ως κατεξοχήν αντικείμενο προς εμπορική εκμετάλλευση.

2.2.1 Από το ακαδημαϊκό στο επιχειρηματικό Grid (Business Grid)

Όπως μιλήσαμε και νωρίτερα, ο όρος Grid αναφέρεται σε μεγάλης κλίμακας διαχείριση και κοινή χρήση πόρων, με σκοπό την επίλυση δύσκολων προβλημάτων. Αυτή η διαχείριση και ταυτόχρονα και η συμμετοχή στο Grid γίνεται από οργανωμένες ομάδες που ονομάζονται "εικονικοί οργανισμοί" (Virtual Organizations) γνωστοί ως VOs [F01]. Τέτοιοι οργανισμοί μπορούν να θεωρηθούν πάροχοι υπηρεσιών εφαρμογών (application service providers), αποθηκευτικού χώρου (storage service providers), βάσεις δεδομένων, συστήματα προσομοίωσης κ.α.

Η συνεργασία των VOs είναι απαραίτητο να καθορίζεται από συγκεκριμένους κανόνες, έτσι ώστε να είναι ξεκάθαρη η σχέση μεταξύ παροχών υπηρεσιών και καταναλωτών, ποιος επιτρέπεται να χρησιμοποιήσει τους πόρους, ποιους πόρους, κάτω από ποιες συνθήκες κλπ. Οι κανονισμοί αυτοί προτυποποιήθηκαν και έτσι δημιουργήθηκαν τα SLA (Service Level Agreement), συμβόλαια δηλαδή με όλες τις πληροφορίες και απαιτήσεις μεταξύ των παροχών των υπηρεσιών και των αποδεκτών τους.

Η χρήση SLA στις Grid υπηρεσίες έδωσε τις προδιαγραφές για την ανάπτυξη της Grid τεχνολογίας και εκτός του αυστηρά ερευνητικού τομέα. Η αλματώδης εξέλιξη του Internet και γενικά της δικτύωσης των υπολογιστών και ο αμείλικτος ανταγωνισμός των επιχειρήσεων, έδωσαν άλλο νόημα στον όρο e-business. Η ροή των εργασιών στις επιχειρήσεις, από την σχεδιασμό μέχρι την υλοποίηση είναι πιο γρήγορη από ποτέ. Στο χώρο όμως του e-business οι απροσδόκητες αλλαγές στο φόρτο εργασιών, στον όγκο πληροφοριών ή στις απαιτήσεις επεξεργασίας δημιούργησαν την απαίτηση για ευέλικτα, μικρού κόστους και μεγάλης αξιοπιστίας συστήματα. Πάνω σε αυτήν την απαίτηση στηρίχθηκε η Grid τεχνολογία για την δημιουργία του business Grid [LG]. Την υιοθέτηση δηλαδή Grid υποδομών στον επιχειρηματικό κόσμο με σκοπό την εμπορική εκμετάλλευση

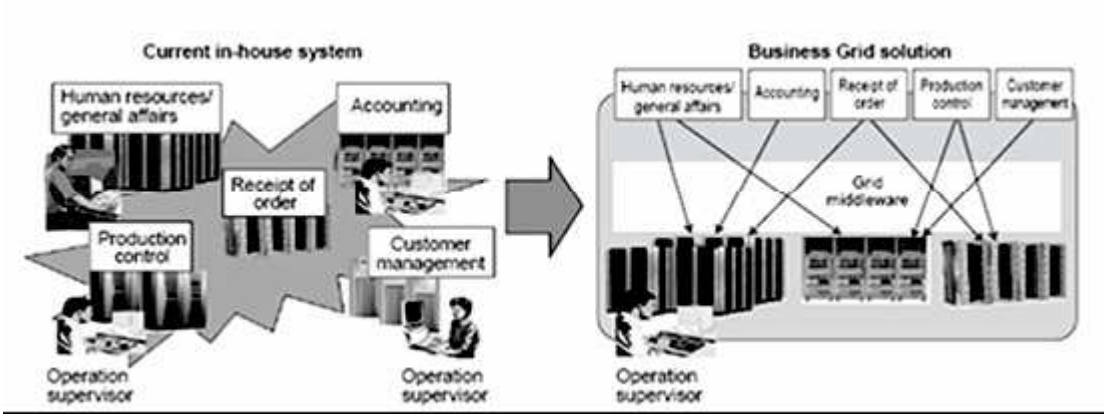
των Grid υπηρεσιών. Πιο συγκεκριμένα οι στόχοι του business Grid είναι :

• την κοινόχρηστη χρήση IT (Information Technology) πόρων, μέσω δικτύωσης υπολογιστών των υπολογιστών απομακρυσμένων κέντρων πληροφοριών (data centers).

• Τη μείωση του κόστους στις επιχειρήσεις με τη χρήση ευέλικτων υποδομών και αυτόνομων συστημάτων ελέγχου

• Την δυναμική εισαγωγή και χρήση πόρων με την ταυτόχρονη διοχέτευση φόρτου εργασίας σε διαφορετικές απομακρυσμένες περιοχές.

• Την απλοποίηση, συνολικά των υποδομών και τη δημιουργία αξιόπιστων συστημάτων που θα υποστηρίζουν διασπορά εργασίας, ανάκτηση κατεστραμμένων πληροφοριών και άλλα λειτουργικά συστήματα και συστήματα ασφαλείας.



Εικόνα 6 : Business Grid

Όπως βλέπουμε και στην Εικόνα 6, η υποδομή του Business Grid βελτιστοποιεί την εκμετάλλευση των πόρων μιας επιχείρησης με αποτέλεσμα την μείωση του κόστους και την αύξηση της παραγωγικότητας.

2.2.2 Προσέγγιση λειτουργίας του επιχειρηματικού μοντέλου του Grid

Αναφερθήκαμε νωρίτερα στην διεύρυνση χρήσης του Grid όχι μόνο στον ερευνητικό τομέα

αλλά και στον επαγγελματικό. Η χρήση αυτή όμως δεν περιορίζεται στην εκμετάλλευση των συστημάτων αυτών εντός μιας επιχείρησης με σκοπό την μείωση κόστους, την αύξηση της παραγωγικότητας και μείωση του κύκλου παραγωγής, αλλά και στην εμπορική εκμετάλλευση αυτών των τεχνολογιών με και νωρίτερα, στα πλαίσια του business Grid, την δυνατότητα εμπορικής εκμετάλλευσης των Grid υπηρεσιών. Η απαίτηση της αγοράς για εξειδίκευση στην παροχή υπηρεσιών και η συγκεκριμένη αρχιτεκτονική του Grid (Service Oriented Architecture) συστήματος, αν δεν μας δίνουν ένα συγκεκριμένο μοντέλο λειτουργίας, μας δίνουν τουλάχιστον τις προδιαγραφές αυτού.

Για να περιγράψουμε τη λειτουργία ενός τέτοιου συστήματος θα αναφέρουμε τους ρόλους και τις οντότητες που συμμετέχουν σε αυτό. Έτσι μπορούμε να ξεχωρίσουμε τον πάροχο της υπηρεσίας (Service Provider). Αυτός ο όρος είναι πολύ γενικός και μπορεί να διασπαστεί και σε άλλες υπό-οντότητες, αφού οι παρεχόμενες υπηρεσίες που σχετίζονται είναι υπηρεσίες δικτύου, πόρων (αποθηκευτικών, υπολογιστικών κλπ.), εφαρμογών (application services) κ.α.

Κάποιες από τις υπηρεσίες αυτές μπορούν να παρέχονται από το ίδιο άτομο ή να είναι διαμοιρασμένες σε περισσότερους. Σε κάθε περίπτωση, για την ομαλή και αποδοτική συνεργασία μεταξύ τους υπάρχουν τα συμβόλαια (Service Level Agreements) που διασφαλίζουν τους όρους συνεργασίας. Όπως αναφέρθηκε και σε προηγούμενα κεφάλαια, τα SLA θα πρέπει να διασφαλίσουν και την ποιότητα των παρεχομένων υπηρεσιών (QoS) και αυτό μπορεί να επιτευχθεί με την εισαγωγή στο σύστημα και ενός άλλου ρόλου, του QoS Provider. Μιας οντότητας δηλαδή, η οποία θα επικοινωνεί με τους άλλους πάροχους υπηρεσιών, θα συλλέγει πληροφορίες και θα προσδιορίζει την ποιότητα των προσφερόμενων υπηρεσιών από και μεταξύ κάθε πλευράς. Ο ρόλος αυτός είναι πολύ σημαντικός αφού διασφαλίζει συνολικά την ποιότητα της Grid υπηρεσίας που παρέχεται και που μας ενδιαφέρει να εκμεταλλευτούμε. Φυσικά η τελευταία οντότητα στο σύστημα

αυτό, που δε πρέπει να ξεχνούμε είναι ο πελάτης που θέλει την χρήση των υπηρεσιών του Grid. Έτσι και αλλιώς γύρω από τις απαιτήσεις του πελάτη γίνονται όλες οι προσπάθειες για τη βελτίωση της ποιότητας των υπηρεσιών.

2.2.2.1 QoS Provider

Αναλύοντας λοιπόν περισσότερο τον μηχανισμό, μιας και σκοπός μας είναι η μοντελοποίηση του και τελικά η υλοποίηση του, θα πρέπει να περιγράψουμε τις προδιαγραφές λειτουργίας του. Η υπηρεσία παροχής QoS πληροφορίας θα πρέπει :

- Να έχει πρόσβαση σε πληροφορίες των πόρων ενός Grid συστήματος που άλλοι χρήστες μπορεί να μην έχουν
- Να έχει υλοποίηση γενικής χρήσης έτσι ώστε να μπορεί να εφαρμοστεί σε διαφόρων τύπων πόρους
- Να είναι συμβατό με τις τεχνολογίες που εφαρμόζονται στα Grid συστήματα
- Η υλοποίησή της να είναι διαθέσιμη σε δημόσια χρήση, έτσι ώστε οποιοσδήποτε επιθυμεί on- demand πληροφορία να μπορεί να την αποκτήσει
- Να γίνεται καταγραφή και επεξεργασία των πληροφοριών αυτών όχι μόνο για τεκμηρίωση της ποιότητα των προσφερόμενων υπηρεσιών αλλά και για βελτίωση της λειτουργίας όλου του Grid συστήματος.

Μιλώντας για την ποιότητα υπηρεσιών σε Grid περιβάλλον, ανατρέχουμε κατευθείαν στους όρους της διαθεσιμότητας, αποδοτικότητας, αξιοπιστίας, κόστους κλπ. Η παραμετροποίηση αυτών των εννοιών και η εξαγωγή της αντίστοιχης πληροφορίας θα μας δώσει τα δεδομένα που χρειαζόμαστε για τον προσδιορισμό του QoS. Έχουμε αναλύσει τις παραμέτρους της ποιότητας των Grid υπηρεσιών σε προηγούμενη ενότητα και σε αυτό το σημείο μας ενδιαφέρουν οι όροι διαθεσιμότητα και απόδοση. Αυτά τα δύο χαρακτηριστικά εμπεριέχουν όλη την πληροφορία μέσω της οποίας μπορεί να γίνει αποτίμηση της αξιοπιστίας, του

κόστους και της τήρησης των συμβολαίων (SLA).

2.2.2.2 Διαθεσιμότητα (Availability)

Με τον όρο διαθεσιμότητα μπορούμε να συμπεριλάβουμε παραμέτρους όπως τη διαθεσιμότητα των εφαρμογών, διαθεσιμότητα των πόρων, διαθεσιμότητα του δικτύου κ.α. Το σημαντικότερο στοιχείο στην ποιότητα μιας υπηρεσίας Grid, αλλά και γενικά μιας οποιαδήποτε υπηρεσίας, είναι το κατά πόσο αυτή και ότι απαιτήσεις έχει, είναι διαθέσιμα. Ποιο αναλυτικά τα χαρακτηριστικά που θεωρούμε έκφραση της διαθεσιμότητας είναι :

- Διαθέσιμος ελεύθερος χώρος
- Εύρος δικτύου
- Απαραίτητες βιβλιοθήκες-αρχεία
- Υποστηριζόμενα πρωτόκολλα υπηρεσιών
- Κατάσταση υπηρεσιών εφαρμογών

Τα παραπάνω όπως παρατηρούμε, είναι οι στοιχειώδεις προϋποθέσεις για την εκτέλεση μιας Grid εφαρμογής. Προσπαθώντας να τηρήσουμε τις αρχικές απαιτήσεις λειτουργίας ενός συστήματος γενικής χρήσης, δε θα αναζητήσουμε παραμέτρους συγκεκριμένους για κάθε είδος συστήματος ή πόρου. Η ισορροπία στην επιλογή αυτή είναι λίγο λεπτή αφού πρέπει να αποκτήσουμε όσον το δυνατόν περισσότερη πληροφορία αλλά παράλληλα να κρατήσουμε την μορφή της QoS υπηρεσίας εφαρμόσιμη σε κάθε σύστημα.

2.2.2.3 Αποδοτικότητα (Performance)

Ένα άλλο χαρακτηριστικό που είναι πολύ σημαντικό και καθοριστικό για την ποιότητα της

υπηρεσίας που παρέχεται είναι η απόδοση των συστημάτων που συμμετέχουν στο Grid. Ο καθορισμός τέτοιων παραμέτρων που θα προσδιορίζουν την απόδοση, είναι αρκετά δύσκολος και διαφέρει από σύστημα σε σύστημα. Επίσης η ταχύτητα με την οποία μπορεί να αλλάξει η κατάσταση ενός συστήματος από αποδεκτής απόδοσης σε μη αποδεκτής δυσκολεύει περισσότερο την μοντελοποίηση τέτοιων μηχανισμών. Λαμβάνοντας υπόψη ότι το μεγαλύτερο μέρος των συστημάτων που συμμετέχουν θα είναι υπολογιστικές μονάδες, κάποιες μελέτες έχουν καταλήξει σε παραμέτρους όπως :

- Συχνότητα ρολογιού επεξεργαστή
- Υπολογισμός MFLOPs (Millions Floating Operations per second)
- Μέσο Εύρος ζώνης μνήμης
- Μέσο εύρος ζώνης σκληρού δίσκου
- Διαθέσιμη μνήμη

Ο προσδιορισμός πιο αξιόπιστων παραμέτρων αποτελεί αντικείμενο επιστημονικού ενδιαφέροντος και πεδίου συζήτησης.

2.2.2.4 Αξιοπιστία (Reliability)

Ένα τρίτο χαρακτηριστικό που είναι εξίσου σημαντικό με τα προαναφερθέντα είναι η αξιοπιστία της παρεχόμενης υπηρεσίας. Η αξιοπιστία της υπηρεσίας έχει άμεση σχέση με την διαθεσιμότητα σε σύγκριση όμως με την χρονική διάρκεια της. Εκτός από αυτό όμως, η αξιοπιστία εξαρτάται και από παραμέτρους όπως η συχνότητα εμφάνισης σφαλμάτων ή την πιθανότητα εμφάνισής τους.

Συνολικά όμως η αξιοπιστία της υπηρεσίας διαπιστώνεται από μακροχρόνια παρακολούθηση των παραμέτρων της διαθεσιμότητας και απόδοσης και συνολικής επεξεργασίας τους.

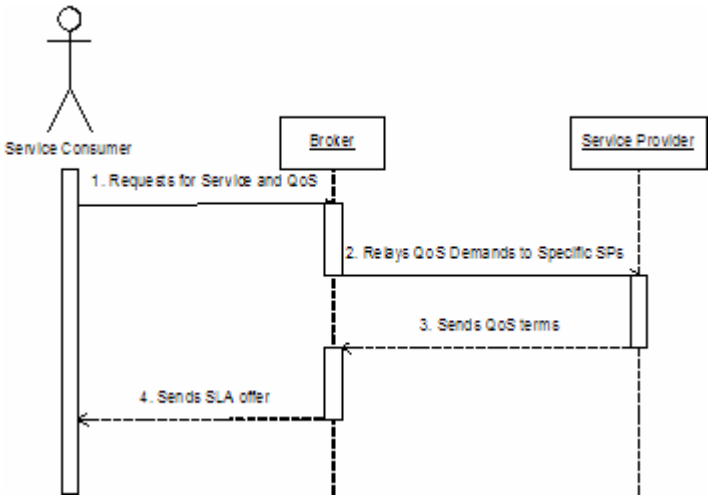
3 *Ανάλυση και σχεδίαση*

Στο κεφάλαιο αυτό θα γίνει μια παρουσίαση και ανάλυση της αρχιτεκτονικής και των λειτουργιών του συστήματος που θα αναπτύξουμε. Μια πρώτη προσέγγιση της αρχιτεκτονικής του μηχανισμού που θα ονομάσουμε “QoS History” φαίνεται παρακάτω.

3.1 Βασικό Θεωρητικό Υπόβαθρο

Γενικώς, ο τρόπος με τον οποίο η ποιότητα υπηρεσίας γίνεται αντιληπτή σε τρέχουσες αρχιτεκτονικές Πλέγματος είναι ως μέρος της διαδικασίας διαπραγμάτευσης SLA όπως περιγράφεται στις εργασίες [BH04],[AAL+04] και [PDD05]. Ο τελικός χρήστης ορίζει τους στόχους/περιορισμούς του και τους αποστέλλει σε πολλούς παρόχους υπηρεσιών μέσω της λειτουργικότητας που προσφέρεται από το μεσάζοντα (συνήθως το SLA Management Service) για την εύρεση των καταλληλότερων φορέων παροχής υπηρεσιών. Εν συνεχεία, οι

φορείς παροχής υπηρεσιών, χρησιμοποιώντας τα εμπιστευτικά επιχειρηματικά μοντέλα τους, καθορίζουν πώς μπορούν να ανταποκριθούν στις προδιαγραφές του τελικού χρήστη και να στείλουν μια προσφορά πίσω στον μεσάζοντα που με τη σειρά του παραδίδει μια προσφορά SLA. Αυτό ολοκληρώνει τον πρώτο κύκλο της διαδικασίας διαπραγμάτευσης SLA ο οποίος κλείνει εάν ο τελικός χρήστης αποφασίσει να αποδεχθεί μια προσφορά ή συνεχίζει εάν ο τελικός χρήστης αποφασίσει να επαναδιαπραγματευθεί



Εικόνα 7: Κλήση υπηρεσιών που πληρούν τα κριτήρια ποιότητας υπηρεσίας στις τρέχουσες αρχιτεκτονικές Πλέγματος

Το ταίριασμα μιας αίτησης για παροχή υπηρεσίας χρησιμοποιώντας το QoS ως κριτήριο επιτυγχάνεται χρησιμοποιώντας μια κοινή "γλώσσα" η οποία είναι κατανοητή και από τις δύο πλευρές. Αυτή η γλώσσα βασικά είναι ένα σύνολο παραμέτρων QoS χαμηλού επιπέδου (τις οποίες παραμέτρους θα αναλύσουμε αργότερα) οι οποίες σχετίζονται με τη λειτουργία των πόρων και τις απαιτήσεις της εφαρμογής.

Οι αδυναμίες μιας τέτοιας μεθόδου μπορεί να είναι οι ακόλουθες:

1. Η αξιοπιστία των παραμέτρων και ακολούθως του φορέα παροχής υπηρεσιών δεν μπορεί να αξιολογηθεί και επομένως δεν υπάρχει κανένα προφανές κίνητρο για τους τελικούς χρήστες για τη συνέχιση μιας εμπιστευόμενης σχέσης.
2. Το σύνολο των παραμέτρων δεν αντανακλά τις απαιτήσεις ποιότητας του τελικού

χρήστη από την υπηρεσία (ή από το σύνολο των υπηρεσιών) αφού η ροή της εργασίας μπορεί να διαφέρει ή να αλλάζει δυναμικά. Για παράδειγμα, ένας τελικός χρήστης επιθυμεί να χρησιμοποιήσει μια σειρά από υπηρεσίες και να δηλώσει συγκεκριμένες απαιτήσεις QoS για καθεμία από αυτές υπό την μορφή παραμέτρων QoS. Ακολούθως οι φορείς παροχής υπηρεσιών απαντάνε ανεξάρτητα με την χορήγηση μιας προσφοράς η οποία αναπαριστά την αξιολόγηση των QoS αναγκών βασισμένη στις δικές τους δυνατότητες παροχή ποιότητας υπηρεσίας. Με αυτό τον τρόπο λαμβάνει χώρα μια αντιστοιχία για καθεμία υπηρεσία. Εντούτοις, το πρόβλημα εάν οι φορείς παροχής υπηρεσιών συναντούν τις απαιτήσεις του τελικού χρήστη πρέπει να εξεταστεί από την σκοπιά της όλης ακολουθίας της υπηρεσίας – αυτό που αποκαλείται end-to-end QoS. Αυτό κάνει το πρόβλημα συνδυαστικό και NP-hard. Διαφορετικές ακολουθίες από υπηρεσίες επικαλούμενες από συγκεκριμένους φορείς παροχής υπηρεσιών μπορούν να οδηγήσουν σε διαφορετική συνολική απόδοση στον τομέα της παροχής QoS. Αυτό μπορεί να οδηγήσει στην καλύτερη αξιοποίηση της υποδομής και στη συνέχεια σε αποδοτικότερες λύσεις κόστους και χρόνου από τις οποίες μπορούν να ωφεληθούν τόσο η πλευρά των τελικών χρηστών όσο και των φορέων παροχής υπηρεσιών. Με άλλα λόγια, οι διαφορετικές ροές εργασίας οδηγούν στη διαφορετική end-to-end παροχή ποιότητας υπηρεσίας [KDM06].

3. Το επίπεδο γνώσης των τελικών χρηστών σχετικά με την εφαρμογή είναι κάτι το οποίο ποικίλει και επομένως οι τεχνικές λεπτομέρειες και άλλες “μη φιλικές” προς τον χρήστη λεπτομέρειες πρέπει να απομονωθούν από αυτόν. Ο τελικός χρήστης πρέπει ουσιαστικά να ανακουφιστεί από το φορτίο να κάνει την επιλογή από μόνος του με το να εξετάζει μόνο έναν κατάλογο προσφορών. Σε πολλές περιπτώσεις ο τελικός χρήστης δεν μπορεί να έχει τα κατάλληλα δικαιώματα στο πλαίσιο του

επιχειρηματικού πλάνου για να αποφασίσει ποια υπηρεσία να επικαλεστεί χρησιμοποιώντας μόνο το κριτήριο του κόστους.

Με βάση τα ανωτέρω, είναι εμφανέστερο, ότι προεκτείνοντας την υποδομή Πλέγματος με τρόπο με τον οποίο να μπορεί να χρησιμοποιηθεί σε εμπορικές εφαρμογές (π.χ. στην παροχή εγγυήσεων κατά τη διαπραγμάτευση των απαιτήσεων QoS) απαιτεί τη γνώση πληροφοριών QoS για καθένα από τα διάφορα επίπεδα μιας διαδικασίας ενδιάμεσου λογισμικού Πλέγματος προκειμένου να προσδιοριστεί το επίπεδο της ποιότητας. Οι διάφορες διαβεβαιώσεις δίνονται από τα περιβάλλοντα του Πλέγματος εκ κατασκευής. Αυτές περιλαμβάνουν τις περιπτώσεις όπου ο τελικός χρήστης ενδιαφέρεται για τις υπηρεσίες Πλέγματος που φιλοξενούνται σε υπολογιστικούς πόρους υψηλών επιδόσεων συμπεριλαμβανομένων των ακριβών εξοπλισμών και των συστημάτων αποθήκευσης δεδομένων τα οποία συνδέονται μέσω αξιόπιστων δικτύων υψηλών ταχυτήτων.

Σε ότι ακολουθεί εισηγούμαστε ένα νέο σχήμα για την παροχή γενικής, end-to-end QoS πληροφορίας. Το σχήμα αυτό αντιμετωπίζει αποδοτικά τα προβλήματα 1) και 2) ενεργώντας σαν τρίτο μέλος, το οποίο εμπιστεύονται και οι δύο πλευρές και συγχρόνως ενσωματώνει το επιχειρηματικό πλάνο του τελικού χρήστη στο ενδιάμεσο λογισμικό Πλέγματος. Επίσης αντιμετωπίζει εν μέρος το πρόβλημα 3) υπό την έννοια όχι μόνο αφήνει το χρήστη να ξεκινήσει τη διαπραγμάτευση SLA χρησιμοποιώντας όρους κατανοητούς γι' αυτόν αλλά επίσης του επιτρέπει να διαπραγματευτεί πάνω σε τέτοιους όρους, όπως θα δούμε παρακάτω.

3.2 Συνεργαζόμενος μηχανισμός

Ο δικός μας μηχανισμός συνεργάζεται με έναν άλλο μηχανισμό ο οποίος καλείται QoS Information Provisioning Mechanism, ο οποίος έχει σαν στόχο την εξαγωγή πληροφοριών

σχετικά με την ποιότητα υπηρεσίας η οποία παρέχεται από κάποιον φορέα παροχής υπηρεσιών. Πιο κάτω θα παρουσιάσουμε αυτόν τον μηχανισμό έτσι ώστε να γίνει πιο κατανοητή η συνολική λειτουργία του μηχανισμού που έχουμε αναπτύξει.

3.2.1 Μηχανισμός Παροχής Πληροφορίας για την Ποιότητα Υπηρεσίας (QoS Information Provisioning Mechanism)

Υπάρχουν δύο εννοιολογικές οντότητες οι οποίες κυριαρχούν σε αυτή την εργασία. Η μία είναι η έννοια των παραμέτρων χαμηλού επιπέδου, που βρίσκονται κοντά στην υποδομή και οι οποίες απεικονίζουν την ικανότητα απόδοσης του φορέα παροχής υπηρεσιών. Τις παραμέτρους αυτές τις καλούμε *παραμέτρους QoS* (QoS parameters). Η άλλη έννοια είναι αυτή των υψηλού επιπέδου παραμέτρων οι οποίες είναι πιο κατανοητές από τον τελικό χρήστη της εφαρμογής. Θα καλούμε τις τελευταίες *δείκτες QoS* (QoS Indices).

3.2.2 Παράμετροι και Δείκτες QoS

Οι παράμετροι QoS χρησιμοποιούνται ευρέως για τις παραμέτρους οι οποίες αναφέρονται στις ιδιότητες τόσο των πόρων όσο και των συγκεκριμένων υλοποιήσεων των υπηρεσιών. Οι ιδιότητες αυτές είναι συνήθως προϊόντα εργαλείων αξιολόγησης και ελέγχου απόδοσης. Οι παράμετροι QoS χρησιμοποιούνται συνήθως ως όροι στις συμβάσεις SLA και ο καταναλωτής υπηρεσιών αναμένεται να εκφράσει τις απαιτήσεις του/της με την ανάθεση των αντίστοιχων τιμών. Είναι γεγονός ότι ένα μεγάλο μέρος αυτών των πληροφοριών μπορεί να εξαχθεί εύκολα στον πραγματικό χρόνο από τα διάφορα τμήματα Πλέγματος όπως υποσυστήματα έλεγχος (monitoring) ή ακόμα και υποσυστήματα που μοντελοποιούν τους πόρους με βάση συγκεκριμένες ιδιότητες (όπως η απόδοση). Αυτό το είδος των

υποσυστημάτων χρησιμοποιεί συνήθως τις χαμηλού επιπέδου προδιαγραφές και πρότυπα, όπως το Common Information Model (CIM) [DMT05], προκειμένου να παράσχει μια γενική μέθοδο μοντελοποίηση που θα είναι σε θέση να εφαρμοστεί σε όλους τους πόρους. Άλλες υλοποιήσεις, κυρίως στις σύγχρονες αρχιτεκτονικές Πλέγματος, εκμεταλλεύονται στην υπηρεσιοστρεφή φύση του περιβάλλοντος, χρησιμοποιώντας πρότυπα που είναι εύκολο υιοθετηθούν και είναι ευθυγραμμισμένα με την OGSA. Παραδείγματα ταξινομημένων παραμέτρων QoS φαίνονται στον Πίνακα 1.

Category	*QoS Parameters*
CPU Capability	
	Flops Count/simple BM (linpack)
	Kernel Version
	Architecture Type
CPU Load	
	System Load
	User Load
	I/O Load
Queue info	
	Policy (how many jobs at a time, max time)
	Length (residency times?)
	Number of running jobs
Inter-processor connections	
	BW Between processors in a cluster
	Latency between processors in a cluster
	MPI messaging
Memory/Cache	
	Capacity
Disk	
	Disk bandwidth (bonnie)
Installation	
	Software and versions
	Environment, variables

Πίνακας 1: Παραδείγματα QoS παραμέτρων

Από την άλλη, οι δείκτες QoS είναι όροι που εισάγονται προκειμένου να εκφράσουν τις απαιτήσεις του καταναλωτή υπηρεσιών και κατά συνέπεια τους επιχειρηματικούς στόχους κατά τρόπο πιο αφηρημένο. Αυτοί οι δείκτες διαμορφώνουν ένα σύνολο μετρικών που δείχνουν το επίπεδο του QoS που ο φορέας παροχής υπηρεσιών μπορεί να προσφέρει και

επομένως θα αναφέρονται ως δείκτες QoS. Οι δείκτες QoS που μελετήσαμε έχουν συνοψιστεί παρακάτω:

- Διαθεσιμότητα της υπηρεσίας
- Αξιοπιστία της υπηρεσίας
- Ποιότητα του αποτελέσματος

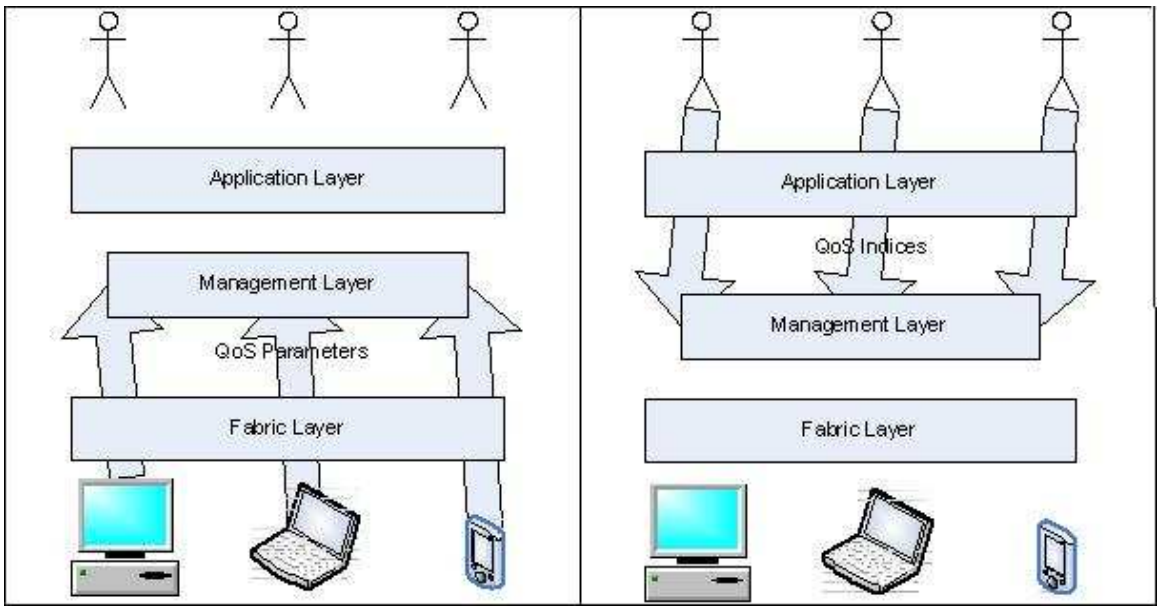
Η έρευνα έχει καταλήξει σε αυτούς τους δείκτες αφού μελετήσαμε τις απαιτήσεις των τελικών χρηστών σε διάφορες εφαρμογές, και συγκεκριμένα:

- Επεξεργασία Ψηφιακών Πολυμέσων (η περίπτωση επεξεργασίας τρισδιάστατων σκηνών για μεταποίηση);
- Μοντελοποίηση Οικονομικών Διαδικασιών (Υπολογισμοί αξίας σε σχέση με ρίσκο);
- Διαχείριση και Βελτιστοποίηση Εφοδιαστικής Αλυσίδας (συνεχής επανασχεδιασμός για λογιστικά μεταφορών με τα δεδομένα να ανανεώνονται σε πραγματικό χρόνο από έξυπνα, μικρά δίκτυα); και,
- Εξόρυξη Δεδομένων σε Περιβάλλοντα Πλέγματος (επεξεργασία μεγάλων βάσεων δεδομένων).

Το συμπέρασμα ήταν ότι οι τελικοί χρήστες αυτών των εφαρμογών είναι πιο πρόθυμοι να πληρώσουν για υπηρεσίες εάν η κλήση τους περιλαμβάνει όρους όπως η διαθεσιμότητα, αξιοπιστία και η ποιότητα του αποτελέσματος. Η απαίτηση για αυτούς τους δείκτες, αποδίδεται στο γεγονός ότι όλες οι περιοχές εφαρμογής έχουν υψηλή ζήτηση σε υπολογιστικούς πόρους (π.χ. υπολογιστική χωρητικότητα και δυνατότητες αποθήκευσης) και επομένως χρειάζονται εγγυήσεις σχετικές με τη διαθεσιμότητα των υπηρεσιών και των πόρων. Υπάρχουν ήδη εφαρμογές που εκτελούν αυτούς τις εργασίες εντούτοις οι τελικοί χρήστες επιδιώκουν καλύτερες επιδόσεις στις τεχνικές υπολογισμού Πλέγματος. Αυτό υπονοεί ότι τα πεδία των εφαρμογών είναι όλα προσανατολισμένα στις επιχειρηματικές ανάγκες και έτσι η ποιότητα του αποτελέσματος έχει μια άμεση επίδραση στο τελικό προϊόν

ενώ η αξιοπιστία παρέχει τις απαραίτητες διαβεβαιώσεις για την ανοχή σε σφάλματα.

Εντούτοις, είναι κοινά αποδεκτό ότι οι δείκτες QoS δεν μπορούν να καθοριστούν “a priori” αλλά μετά από προσεκτική έρευνα και πειραματισμό μαζί με την πλευρά των τελικών χρηστών των εφαρμογών. Κατά συνέπεια, οι δείκτες QoS είναι προσανατολισμένοι στις εφαρμογές και ορισμένοι από τους χρήστες και είναι εξαιρετικά απίθανο να είναι δυνατό να διατυπωθεί μια μέθοδος που θα λειτουργεί αποτελεσματικά κάτω από όλες τις περιστάσεις και τις περιοχές εφαρμογής. Είναι επίσης σαφές, ότι οι δείκτες QoS είναι αφηρημένες παράμετροι που κλίνουν στην πλευρά του τελικού χρήστη και που φθάνουν έως το επίπεδο διαχείρισης Πλέγματος, ενώ οι παράμετροι QoS είναι πιο σχετικές με το επίπεδο υλικού του περιβάλλοντος Πλέγματος και επεκτείνονται να φθάσουν στο διαχειριστικό στρώμα.



Εικόνα 8: Οι παράμετροι QoS απευθύνονται στα περιβάλλοντα Πλέγματος από τους πόρους, μέσω του στρώματος υλικού, στο διαχειριστικό στρώμα. Οι δείκτες QoS παράγονται από τους τελικούς χρήστες, διαμέσου του στρώματος εφαρμογών διασχίζοντας το διαχειριστικό στρώμα.

3.2.3 Περιγραφή QoS Information Provisioning Mechanism

Προκύπτει από τα ανωτέρω, ότι το ενδιαφέρον τμήμα που σχετίζεται με την έννοια των

δεικτών QoS είναι ο σχεδιασμός του ενδιάμεσου μηχανισμού που απαιτείται προκειμένου να μεταδοθούν οι επιχειρηματικοί στόχοι του καταναλωτή υπηρεσιών άμεσα στο φορέα παροχής υπηρεσιών. Αυτό που θα ήταν ιδανικό, είναι ένας ενδιάμεσος μηχανισμός ικανός να αντιστοιχίσει με ακρίβεια τις επιχειρηματικές ανάγκες των τελικών χρηστών στις δυνατότητες απόδοσης του πάροχου υπηρεσιών. Ένας τρόπος να επιτευχθεί αυτό είναι η επέκταση των υπαρχόντων μηχανισμών διαχείρισης SLA [PDD] με τη λειτουργικότητα της αντιστοίχισης των παραμέτρων QoS σε δείκτες QoS και αντίστροφα. Ονομάζουμε το μηχανισμό αυτό ***Μηχανισμό Παροχής Πληροφορίας QoS*** (QoS Information Provisioning Mechanism-QoS-IPM) και παρουσιάζουμε μία υπηρεσία που υλοποιεί το μηχανισμό αυτό με βάση τις απαιτήσεις της OGSA. Η υπηρεσία αυτή ονομάζεται ***Υπηρεσία Παροχής Πληροφορίας QoS*** (QoS Information Provisioning Service-QoS-IPS).

Πάραυτα, το εγχείρημα της αντιστοίχισης παραμέτρων QoS σε δείκτες QoS είναι ένα πολύπλοκο θέμα. Στο παρελθόν έχουν λάβει χώρα διάφορες προσπάθειες που στοχεύουν στη μέτρηση της απόδοσης ([R05], [GTV03],[TD03]), ωστόσο βασίζονται όλες σε συγκεκριμένες υποθέσεις και καμία δεν είναι γενική αρκετά ώστε να καλύψει τις ανάγκες του QoS-IPM. Ένας τρόπος για να καλυφθεί το εμπόδιο αυτό είναι να βασιστούμε στη γενική απόδοση του πάροχου υπηρεσίας και να μελετήσουμε την παροχή ποιότητας προς τους καταναλωτές κατά τη διάρκεια του χρόνου ζωής του. Προς αυτή την κατεύθυνση, τα SLAs αποδεικνύονται ένα πολύ πολύτιμο εργαλείο, δεδομένου ότι απεικονίζουν τη σχέση προμηθευτή-καταναλωτή υπηρεσιών από την άποψη του πόσο ικανός είναι ο προμηθευτής να καλύψει τις ανάγκες του καταναλωτή. Επομένως, ένα ιστορικό παραβιάσεων του SLA είναι πολύ χρήσιμο σαν πληροφορία όταν συνδυάζεται με τη γνώση της τρέχουσας κατάστασης του προμηθευτή, δεδομένου ότι μπορεί να εξηγήσει εάν ο προμηθευτής είναι σε θέση να ικανοποιήσει μια απαίτηση του καταναλωτή.

Ο όρος ιστορικό παραβιάσεων του SLA υπονοεί την ύπαρξη μίας λίστας από παραμέτρους

QoS μαζί με τις σχετικές τιμές τις οποίες ο πάροχος παραβίασε στο παρελθόν ως μέρος ενός συμβολαίου SLA. Βασικά, αποτελεί την ιστορία της παροχής QoS του πάροχου υπηρεσίας ή το ***Ιστορικό Ποιότητας*** (QoS History). Διαισθητικά, η χρησιμοποίηση των στατιστικών στοιχείων για την αντιστοίχιση παραμέτρων QoS σε δείκτες QoS είναι μια μάλλον ασφαλής μέθοδος, εντούτοις, το αντίθετο δεν μπορεί να απαιτηθεί. Δηλαδή κάποιος δεν μπορεί να αρχίσει από δείκτες QoS και να τους αντιστοιχίσει σε παραμέτρους QoS δεδομένου ότι αυτό μπορεί να οδηγήσει στην απώλεια πολύτιμων πληροφοριών κατά τη διαδικασία.

Επιστρέφοντας στη γενική εικόνα, φαίνεται ότι εάν το σύστημα γνωρίζει δύο είδη πληροφοριών, μπορεί να συναγάγει χρήσιμα συμπεράσματα για την ικανότητα του προμηθευτή να παράσχει QoS. Πιο συγκεκριμένα οι πληροφορίες είναι:

- το ιστορικό QoS των φορέων παροχής υπηρεσιών; και,
- τα τρέχοντα αιτήματα των καταναλωτών υπηρεσιών (μέσω της πρώτης φάσης διαπραγμάτευσης SLA, η οποία είναι όταν στέλνει τις απαιτήσεις).

Αυτή η προαναφερθείσα ικανότητα που μεταφράζεται σε δείκτες QoS μέσω μιας απλής διαδικασίας συσχετισμού μπορεί να δώσει μια προστιθέμενη αξία στα υπολογιστικά περιβάλλοντα Πλέγματος επειδή ο τελικός χρήστης μπορεί άμεσα να έχει πρόσβαση και να διαχειριστεί αυτές τις πληροφορίες (εάν η προσφορά SLA εκφράζεται στους όρους δεικτών QoS) αλλά και επειδή το σύστημα μπορεί να ενεργήσει με βάση αυτές τις πληροφορίες. Το τελευταίο υπονοεί ότι οι δείκτες QoS μπορούν να παραχθούν και να χρησιμοποιηθούν από έναν μηχανισμό που διευκολύνει την υπάρχουσα υποδομή προκειμένου να επιτραπεί στο περιβάλλον Πλέγματος να ενσωματώσει τους επιχειρηματικούς στόχους του τελικού χρήστη. Αυτό είναι δυνατό με τη χρησιμοποίηση των αφηρημένων πληροφοριών που εκφράζονται μέσω των δεικτών QoS σε άλλους μηχανισμούς, εσωτερικά στο Πλέγμα. Με αυτή τη λογική παρουσιάζουμε το σενάριο μηχανισμών αντιστοίχισης ροών εργασίας.

Ο μηχανισμός αντιστοίχισης των ροών εργασίας συνθέτει μια ροή εργασίας από υπηρεσίες

χρησιμοποιώντας ως κριτήριο τις παραμέτρους QoS. Αυτή η ροή εργασίας υπαγορεύει την ακολουθία εκτέλεσης και τον τρόπο με τον οποίο οι υπηρεσίες θα κληθούν για μια συγκεκριμένη εργασία. Η αντικατάσταση των παραμέτρων QoS με δείκτες QoS θα βελτίωνε πολύ τη λειτουργία του μηχανισμού δεδομένου ότι η επιλογή των συγκεκριμένων υπηρεσιών θα επιτυγχανόταν ευκολότερα με τη χρησιμοποίηση λιγότερων κριτηρίων ταξινόμησης. Παράλληλα, θα εισαχθούν νέα επιχειρηματικά πρότυπα δεδομένου ότι η επιλογή βασίζεται πλέον στους επιχειρηματικούς στόχους του τελικού χρήστη, που εκφράζονται μέσω των δεικτών QoS. Τα πλεονεκτήματα της ιδέας αυτής είναι χαρακτηριστικά, αφού οι υπηρεσίες που θα κληθούν και οι πόροι που θα δεσμευτούν για τον τελικό χρήστη θα ικανοποιήσουν πλήρως τις ανάγκες του/της με βάση τις ικανότητες της υποδομής.

Έχοντας εισαγάγει την προστιθέμενη αξία αυτού του μηχανισμού, η εργασία επεκτείνεται προτείνοντας την αρχιτεκτονική του – δηλαδή την αρχιτεκτονική του QoS-IPS.

3.2.4 Υπηρεσία Παροχής Πληροφορίας QoS (QoS Information Provisioning Service)

Η QoS-IPS είναι μία συγκεκριμένη αρχιτεκτονική η οποία στοχεύει στην παροχή μίας υλοποίησης της OGSA του μηχανισμού QoS-IPM. Η αρχιτεκτονική είναι σχεδιασμένη ακολουθώντας τις προδιαγραφές του Web Services Resource Framework (WSRF) [WS1] και επομένως είναι μία σύνθεση από Δικτυακές Υπηρεσίες που διαχειρίζονται καταστάσεις πόρων (Web Services managing stateful resources). Η έννοια της λειτουργίας της είναι βασισμένη στη χρησιμοποίηση των ιστορικών δεδομένων. Η αποθήκευση των τιμών των μετρικών επιτρέπει την πιο σύνθετη ανάλυση για τις αναφορές σε μια πιο πρόσφατη φάση ή

μια σύγκριση των πραγματικών στοιχείων με τα ιστορικά συλλεχθέντα στοιχεία. Και είναι σημαντικό να αποθηκευτούν τα πολύτιμα μετρημένα δεδομένα ως απόδειξη της ποιότητας της υπηρεσίας για μια μεγαλύτερη χρονική περίοδο (απόδειξη της ποιοτικής αναδημιουργίας) σε περίπτωση αξίωσης εξουσιοδότησης από τον πελάτη ή από ρυθμιστική άποψη. Αυτό υπονοεί την ανάγκη για την αρχειοθέτηση ιστορικών, εξαιρετικά σημαντικών δεδομένων.

Η υπηρεσία αποτελείται από δύο θεμελιώδη συστατικά: την ***Υπηρεσία Ιστορικού Ποιότητας*** (QoS History Service) και την ***Υπηρεσία Εκτίμησης Δεικτών Ποιότητας*** (QoS Indices Estimator Service), αλλά αλληπεδρά ή μπορεί να αλληλεπιδράσει με πολλά άλλα. Μια υποχρεωτική, υπάρχουσα αλληλεπίδραση είναι με την υπηρεσία διαχείρισης SLA που είναι αρμόδια για τη διαπραγμάτευση, την παροχή και την υποστήριξη SLA. Η υπηρεσία διαχείρισης SLA, μπορεί να λειτουργήσει ως ενδιάμεσος για τη μεταφορά QoS απαιτήσεων του καταναλωτή υπηρεσιών όταν (the SLA management service) τις λαμβάνει στην πρώτη φάση των διαπραγματεύσεων SLA και χρησιμοποιεί την QoS-IPS προκειμένου να πάρει μία σαφή απάντηση στο κατά πόσον ένας πάροχος μπορεί να εγγυηθεί για την απαιτούμενη ποιότητα.

Οι λειτουργίες που τα δύο συστατικά είναι σε θέση να εκτελέσουν παρουσιάζονται παρακάτω:

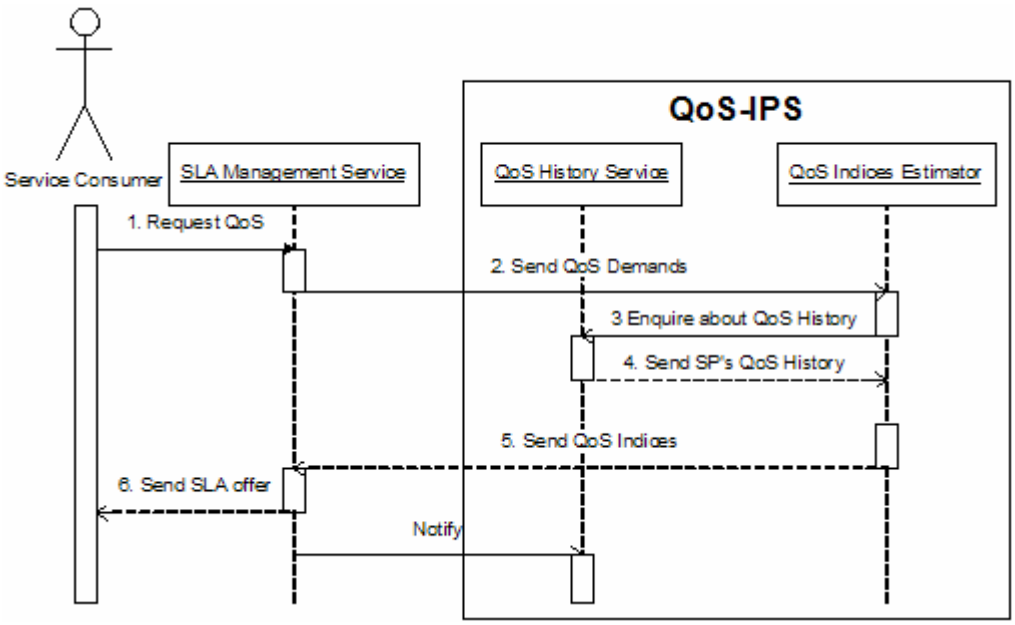
Υπηρεσία Ιστορικού Ποιότητας

- Αποθήκευση του ορίου που έθεσε ο καταναλωτής σε ένα συγκεκριμένο όρο ποιότητας ο οποίος παραβιάστηκε κατά τη διάρκεια ζωής του SLA
- Παρέχει μία διεπαφή έτσι ώστε όταν ερωτηθεί, να επιστρέψει το μέρος του Ιστορικού Ποιότητας που απαιτείται

Υπηρεσία Εκτίμησης Δεικτών Ποιότητας

- Ερωτά το Ιστορικό Ποιότητας σχετικά με το ιστορικό ενός παρόχου υπηρεσιών
- Συγκρίνει τις ανάγκες του καταναλωτή που εκφράζονται με όρους παραμέτρων ποιότητας με το ιστορικό του παρόχου.
- Αποφαίνεται για το αν ένα πάροχος είναι ικανός να παράσχει ποιότητα σε ένα συγκεκριμένο επίπεδο στον καταναλωτή. Με άλλα λόγια για το εάν ο πάροχος μπορεί να ικανοποιήσει τις ανάγκες ενός καταναλωτή.
- Εκτιμά τις τιμές των σχετικών δεικτών ποιότητας και τους επιστρέφει σε καθένα που ενδεχομένως μπορεί να τους ζητήσει.

Με βάση τα παραπάνω, το διάγραμμα ακολουθιών (sequence diagram) για την QoS-IPS στην οποία φαίνονται οι απαραίτητες αλληλεπιδράσεις είναι το ακόλουθο:



Εικόνα 9: Ροή λογικής μέσα στην QoS-IPS μέσω ενός UML διαγράμματος ροής

Άλλες οντότητες που περιλαμβάνονται στο διάγραμμα είναι ο Καταναλωτής Υπηρεσιών και η Υπηρεσία Διαχείρισης SLA. Η ύπαρξη του πάροχου υπηρεσιών υπονοείται και δεν εμφανίζεται προκειμένου να διευκολύνει τον αναγνώστη. Παρακάτω αναλύουμε τη ροή όπως αυτή περιγράφεται στην Εικόνα 9:

1. *Απαίτηση Ποιότητας*: Ο καταναλωτής απαιτεί τη χρήση μίας ή περισσότερων υπηρεσιών και στέλνει τις απαιτήσεις του με όρους QoS προκειμένου να πάρει πίσω μία προσφορά στην οποία μπορεί να συμφωνήσει ή να διαπραγματευτεί. Οι απαιτήσεις εκφράζονται σε μία μορφή ενός συνόλου παραμέτρων ποιότητας μέσω ενός προτύπου (template) που σχετίζεται με την κάθε εφαρμογή. Στην ουσία, ο καταναλωτής συμπληρώνει τα όρια των αντίστοιχων τιμών, ζητώντας εγγυήσεις για την εκτέλεση της εργασίας του. Αυτή είναι η πρώτη φάση της διαπραγμάτευσης SLA, όπως διαχειρίζεται από την Υπηρεσία Διαχείρισης SLA.
2. *Αποστολή Απαιτήσεων Ποιότητας*: Η Υπηρεσία Διαχείρισης SLA διαβάζει το κείμενο ή την υπηρεσία SLA και εξάγει τις παραμέτρους ποιότητας, τις οποίες στη συνέχεια αποστέλλει μέσω ενός μηνύματος SOAP (και ενός προτύπου XML) στον Εκτιμητή Δεικτών Ποιότητας. Επίσης, επικοινωνεί με τους αντίστοιχους πάροχους υπηρεσιών και το μηχανισμό αντιστοίχισης ροών εργασιών αλλά αυτές οι αλληλεπιδράσεις δε φαίνονται στο σχήμα.
3. *Έρευνα για το Ιστορικό Ποιότητας*: Ο Εκτιμητής Δεικτών Ποιότητας πρέπει να μετατρέψει τις παραμέτρους Ποιότητας σε Δείκτες. Προκειμένου να το επιτύχει αυτό, ρωτά το ιστορικό των παραβιάσεων του SLA των αντίστοιχων φορέων παροχής υπηρεσιών. Το αποτέλεσμα μπορεί έπειτα να συγκριθεί με τα τρέχοντα αιτήματα του καταναλωτή υπηρεσιών και να συναγάγει συμπεράσματα από αυτό.
4. *Αποστολή Ιστορικού Ποιότητας ενός Πάροχου Υπηρεσιών*: Η Υπηρεσία Ιστορικού Ποιότητας απαντά με την αποστολή των ορίων για τις παραμέτρους ποιότητας που έχουν παραβιαστεί στο παρελθόν από το φορέα παροχής υπηρεσιών για μια συγκεκριμένη υπηρεσία.
5. *Αποστολή Δεικτών Υπηρεσίας*: Η υπηρεσία Εκτίμησης Δεικτών Ποιότητας χρησιμοποιεί τις απαιτήσεις του καταναλωτή υπηρεσιών και το Ιστορικό Ποιότητας

του πάροχου υπηρεσιών προκειμένου να καταλήξει στο εάν ο πάροχος είναι ικανός να παράσχει την απαιτούμενη ποιότητα και/ή να υπολογίσει τους σχετικούς Δείκτες Ποιότητας.

6. *Αποστολή Προσφοράς SLA*: Η Υπηρεσία Διαχείρισης SLA συλλέγει τις πληροφορίες που απαιτούνται από τον Εκτιμητή Δεικτών Ποιότητας, τον πάροχο και το σύστημα διαχείρισης ροών εργασιών και επιστρέφει μία προσφορά SLA στον καταναλωτή, κλείνοντας τον πρώτο κύκλο του SLA. Ένας νέος κύκλος θα ξεκινήσει εάν ο καταναλωτής αποφασίσει να διαπραγματευτεί τους όρους της προσφοράς, η οποία μπορεί πλέον να εκφραστεί με Δείκτες ποιότητας.
7. *Ενημέρωση*: Αυτή είναι μία ασύγχρονη κλήση και υπονοεί ότι η υπηρεσία διαχείρισης SLA χρησιμοποιεί μηχανισμούς ελέγχου παραβιάσεων SLA και παράγει ένα μήνυμα το οποίο ενημερώνει για την παραβίαση ενός ορίου σε μία παράμετρο ποιότητας. Το μήνυμα αυτό αποστέλλεται στο Ιστορικό Ποιότητας το οποίο εν συνεχεία αποθηκεύει την πληροφορία.

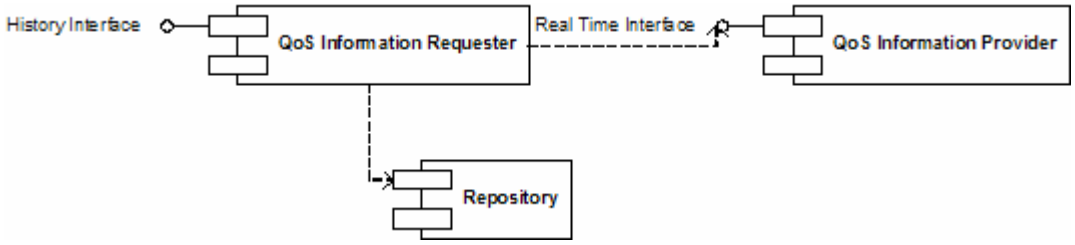
Η λειτουργία της υπηρεσίας QoS-IPS βασίζεται στις προδιαγραφές WSRF, δηλαδή, όλο το περιβάλλον βασίζεται σε Δικτυακές Υπηρεσίες. Επομένως, ξεκινώντας από την πλευρά του πάροχου υπηρεσιών, οι υπηρεσίες αναπτύσσονται ως WS-Resources με τη διάρκεια ζωής τους να ορίζεται από το WS-ResourceLifetime [WS2] ενώ οι παράμετροι ποιότητας εκτίθενται ως ιδιότητες πόρων (resource properties) μέσω του WS-ResourceProperties [WS3]. Το Ιστορικό Ποιότητας είναι μία υπηρεσία καταλόγου η οποία χρησιμοποιεί το WS-Addressing [WS4] για να αποθηκεύει τις λεπτομέρειες που σχετίζονται με την υπηρεσία και το WS-BaseNotification [WS5] για να αποστείλει τις πληροφορίες οι οποίες σχετίζονται με τις παραβιάσεις του SLA. Το πλαίσιο του WS-Security [WS6] χρησιμοποιείται για τη διαχείριση των σκυταλών ασφαλείας (security tokens). Άλλα αρχιτεκτονικά σχεδία που μπορούν να ικανοποιήσουν τις απαιτήσεις ασφαλείας μπορούν να βρεθούν στο [GGL03].

Τέλος, το WSLA [LKD+04] μπορεί να χρησιμοποιηθεί για την αναπαράσταση του SLA, τον έλεγχο και τη δημιουργία, έχοντας ταυτόχρονα στο νου ότι δεν έχει υιοθετηθεί από κάποιο επίσημο Οργανισμό σχετικό με τα Πλέγματα.

3.2.5 Λήψη των παραμέτρων ποιότητας

Η ποιότητα μίας παρεχόμενης υπηρεσίας κατανέμεται σε κάθε υλοποίηση υπηρεσίας ενός περιβάλλοντος Πλέγματος παρά σε ένα επίπεδο υπηρεσίας ή πόρου. Επομένως, οι παράμετροι ποιότητας πρέπει να εξαχθούν από κάθε μία υλοποίηση των υπηρεσιών και συνεπώς στο σχεδιασμό του μηχανισμού πρέπει να ληφθούν υπόψη διάφορες υλοποιήσεις ακόμα και της ίδιας υπηρεσίας, σε διαφορετικές πλατφόρμες και με διαφορετικές ιδιότητες. Όλα αυτά υπονοούν έντονη ετερογένεια και στη βάση αυτή εισαγάγουμε τον όρο ***στιγμιότυπο υπηρεσίας*** (service instance). Προκειμένου να αντιμετωπιστεί αυτό το πρόβλημα η εργασία προχώρησε με το σχεδιασμό δύο συστημάτων. Το ένα από αυτά πρέπει να εγκατασταθεί στην ίδια τοποθεσία με τα στιγμιότυπα των υπηρεσιών –συνήθως στο χώρο του πάροχου υπηρεσιών- και το άλλο μπορεί να αναπτυχθεί οπουδήποτε στην υποδομή Πλέγματος. Ο μηχανισμός αυτός καλείται ***Υπηρεσία Παροχής Παραμέτρων Ποιότητας*** (QoS Parameters Provisioning Service). Από τούδε και εφεξής, ο μηχανισμός αυτός θα αποκαλείται QoS-PPS.

Το διάγραμμα του μοντέλου συστατικών (UML) είναι το ακόλουθο:



Εικόνα 10: Μοντέλο Συστατικών Παροχής Παραμέτρων Ποιότητας

Όπως φαίνεται στην Εικόνα 10, το σύστημα απαρτίζεται από τρία (3) βασικά συστατικά: τον ***Πάροχο Πληροφορίας Ποιότητας*** (QoS information Provider), τον ***Αιτούντα Πληροφορίας Ποιότητας*** (QoS Information Requester) και το σύστημα Αποθήκευσης (Repository).

Ο πάροχος πληροφορίας συλλέγει τις πληροφορίες χαμηλού επιπέδου είτε απευθείας από τα στιγμιότυπα των υπηρεσιών είτε ελέγχοντας την υποδομή του πάροχου υπηρεσιών. Είναι δυνατό να χτιστούν “wrappers” για την εκτέλεση διαφόρων συγκριτικών μετρήσεων επιδόσεων ή “scripts” τα οποία καλού προγράμματα όπως η εντολή “top” σε Linux. Με τον τρόπο αυτό το σύστημα ενημερώνεται για την τρέχουσα κατάσταση του πάροχου υπηρεσιών και ταυτόχρονα, συλλέγει τις απαιτούμενες παραμέτρους ποιότητας. Εν συνεχεία, ο Πάροχος Πληροφορίας Ποιότητας στέλνει τις υπολογισμένες παραμέτρους πίσω στον Αιτούντα.

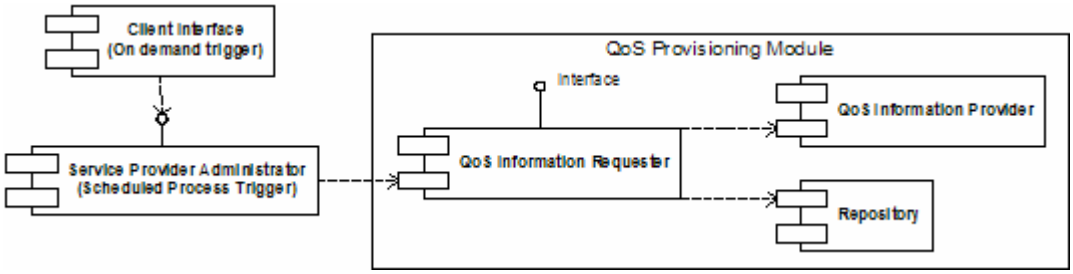
Ο ρόλος του Αιτούντα Πληροφορίας είναι διττός:

1. Ενεργοποιεί τον Πάροχο προκειμένου να ξεκινήσει η διαδικασία συλλογής πληροφοριών σε πραγματικό χρόνο. Η λειτουργία αυτή ενεργοποιείται στέλνοντας ένα αρχείο XML μέσω του πρωτοκόλλου SOAP, υποδεικνύοντας τις παραμέτρους οι οποίες πρέπει να επιστραφούν με την ολοκλήρωση της λειτουργίας του Παρόχου.
2. Διαχειρίζεται τη βάση δεδομένων του συστήματος Αποθήκευσης και δρα ως μία διεπαφή που επεξεργάζεται ερωτήματα για την απόκτηση πληροφορίας από τρίτα μέλη έτσι ώστε να παράσχει πληροφορία πίσω στο σύστημα Αποθήκευσης. Καθώς οι Ανοιχτές Αρχιτεκτονικές Υπηρεσιών πρέπει να παραμένουν ανοιχτές και ευέλικτες, ο μηχανισμός Παροχής Πληροφορίας Ποιότητας πρέπει να επιτρέπει στα άλλα συστατικά να έχουν πρόσβαση στα δεδομένα εφόσον έχουν τα σωστά δικαιώματα. Μία Διεπαφή Ιστορικού επιτρέπει την υποβολή ερωτημάτων τα οποία

χειρίζεται ο Αιτούντος.

Η QoS-PPS, ως ένας σύνθετος μηχανισμός αποτελείται από διάφορα συστατικά που λειτουργούν στο πλαίσιο συγκεκριμένων τομέων. Με αυτή τη λογική, ο Αιτών Πληροφορίες Ποιότητας, το σύστημα Αποθήκευσης και η διεπαφή είναι όλες υπηρεσίες που εγκαταστάθηκαν σε ένα συγκεκριμένο πόρο (δεν υπάρχει συγκεκριμένος λόγος για τον οποίο επιλέγεται αυτός ο πόρος) και το σύστημα Αίτηση Πληροφορίας Ποιότητας σε διάφορα μηχανήματα. Τότε ο Αιτών Πληροφορίας Ποιότητας σποραδικά ερωτά τον Πάροχο Πληροφορίας Ποιότητας και αποθηκεύει τα αποτελέσματα που επιστρέφονται στο σύστημα Αποθήκευσης.

Προκειμένου να επιτευχθούν όλοι οι στόχοι που τίθενται στο σχεδιασμό του μηχανισμού, το συστατικό πρότυπο που παρουσιάζεται στην Εικόνα 10 ενισχύθηκε περαιτέρω με την ενσωμάτωση μερικών νέων οντοτήτων που πρέπει να συμμετέχουν στη διαδικασία.



Εικόνα 11: Εκτεταμένο Μοντέλο Συστατικών της Υπηρεσίας Παροχής Παραμέτρων Ποιότητας

Όπως απεικονίζεται στην Εικόνα 11 οι επεκτάσεις στο πρότυπο σε σύγκριση με την Εικόνα 10 είναι η διαχείριση των φορέων παροχής υπηρεσιών και η διεπαφή των πελατών. Το βασικό μοντέλο συστατικών (Εικόνα 10) είναι η ιδέα της αρχιτεκτονικής του μηχανισμού ενώ η επέκταση δημιουργήθηκε προκειμένου να καλυφτούν οι ανάγκες της εφαρμογής. Αυτοί οι δύο νέοι ρόλοι εφαρμόζουν τις προδιαγραφές του συστήματος για την ανάκτηση των αντίστοιχων πληροφοριών από την υπηρεσία του παρόχου, ώστε να αποτελέσουν

κάποιο σύστημα εσωτερικού ελέγχου/διαχείρισης της υπηρεσίας και ένα μηχανισμό για την εύκολη διανομή των πληροφοριών που η υπηρεσία του παρόχου επιθυμεί να μοιραστεί με άλλες οντότητες μέσα το περιβάλλον. Αυτό είναι μέρος του διαλογικού περιβάλλοντος και απεικονίζεται από τα όρια του συστήματος Παροχής Πληροφορίας Ποιότητας (Εικόνα 11).

Το σύστημα είναι μία σύνθεση διαφόρων μερών και ο τρόπος με τον οποίο αυτά τα μέρη συνδέονται (και κατά συνέπεια ο τρόπος με τον οποίο δένουν με όλη την αρχιτεκτονική Πλέγματος) εξαρτάται από ένα σύνολο αλληλεπιδράσεων. Με βάση αυτά, ο Πάροχος Πληροφορίας αποτελεί τον πυρήνα της Υπηρεσίας Παροχής Πληροφορία Ποιότητας και η ύπαρξή του είναι απαραίτητη σε κάθε πόρο ο οποίος χρησιμοποιείται στο Πλέγμα. Αυτό το συστατικό δεν είναι διαθέσιμο έξω από τα όρια του Πάροχου Υπηρεσίας και η λειτουργία του περιορίζεται στην επιστροφή της απαιτούμενης πληροφορίας στον Αιτούντα. Δεν χρειάζεται να επεξεργαστεί ούτε να αποθηκεύσει δεδομένα καθώς ενεργεί ως ένας απλός ατζέντης. Η νοημοσύνη του συστήματος στρέφεται στο τμήμα Αιτούντος πληροφοριών, το οποίο βρίσκεται επίσης στα όρια του φορέα παροχής υπηρεσιών. Μόνο επικυρωμένοι τρίτοι, οι χρήστες και τα συστατικά μπορούν να έχουν πρόσβαση στον Αιτούντα Πληροφοριών, που λειτουργεί ως μεσολαβητής μεταξύ του προμηθευτή πληροφοριών και των υπολοίπων, μέσα στα όρια του φορέα παροχής υπηρεσιών. Η ευθύνη του είναι να καλέσει την υπηρεσία του Πάροχου Πληροφοριών όποτε αυτό απαιτείται, προκειμένου να υποβληθούν σε επεξεργασία τα δεδομένα που επιστρέφονται και για να τα καταγράψει στο σύστημα Αποθήκευσης. Στη συνέχεια, το σύστημα Αποθήκευσης εγκαθίστανται εσωτερικά στο φορέα παροχής υπηρεσιών και μόνο ο Αιτών Πληροφοριών έχει πρόσβαση σε αυτό.

Τέλος, τα δύο εναπομείναντα συστατικά ασχολούνται με την εκτέλεση της υπηρεσίας του μηχανισμού Παροχής Παραμέτρων Ποιότητας. Το ένα είναι ένας υποδοχέας για το χρονοπρογραμματιστή που ενεργοποιεί την εκτέλεση των συγκεκριμένων ενεργειών του Πάροχου Πληροφοριών Ποιότητας (Διαχειριστής Παρόχου) ενώ το άλλο εξυπηρετεί το

σκοπό της κατόπιν παραγγελίας εκτέλεσης αυτών των ενεργειών, από έναν χρήστη ή ένα συστατικό έξω από τον περιγεγραμμένο μηχανισμό (διεπαφή πελατών).

Σε αυτό το σημείο θα παρουσιάσουμε τις υπηρεσίες που αναπτύχθηκαν για την εφαρμογή του συστήματος Παροχής Παραμέτρων Ποιότητας.

3.3 Περιγραφή Αρχιτεκτονικής

3.3.1 Μηχανισμός Ιστορικού Πληροφορίας QoS (QoS History)

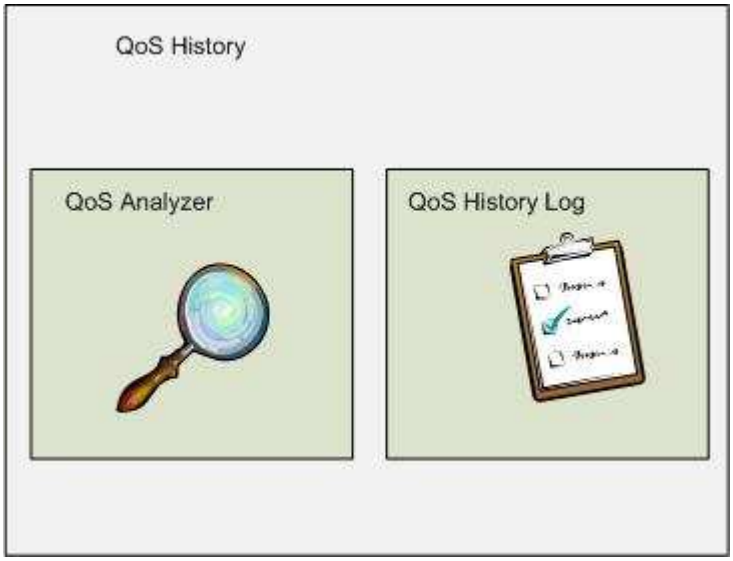
Ο μηχανισμός μας βασίζεται στην αρχή ότι ο πιο αξιόπιστος τρόπος για να προβλέψεις την επίδοση μιας υπηρεσίας σε όρους ποιότητας, είναι η μελέτη της συμπεριφοράς της στο παρελθόν. Επομένως τα αρχεία θα πρέπει να περιέχουν πληροφορίες οι οποίες θα αντικατοπτρίζουν την συμπεριφορά της υπηρεσίας σε προηγούμενες συνεργασίες. Κάθε φορά που καλείται μια υπηρεσία, μια εγγραφή δημιουργείται η οποία παρακολουθεί τις συμφωνηθέντες παραμέτρους οι οποίες ορίζουν την ποιότητα (QoS parameters) και οι σχετικές παραβιάσεις οι οποίες μπορεί να προκύψουν κατά τη διάρκεια του κύκλου ζωής του συμβολαίου. Με αυτό τον τρόπο, ο φορέας καταφέρνει να παρακολουθεί τι έφερε εις πέρας και παρέδωσε στο παρελθόν. Χρησιμοποιώντας αυτή την εγγραφή, την οποία θα ονομάζομε από εδώ και πέρα QoS History, ο φορέας μπορεί να βγάλει χρήσιμα συμπεράσματα σχετικά με τον τρόπο που πρέπει να αντιμετωπίσει μια συγκεκριμένη ανάγκη από ένα πελάτη.

Με μεγαλύτερη λεπτομέρεια, η προαναφερθείσα διαδικασία αντικατοπτρίζεται στην παροχή προτύπων SLA οι οποίες αντιπροσωπεύουν τις ικανότητες της υπηρεσίας από τον πάροχο στον έξω κόσμο. Με αυτά, ο πάροχος αποκρίνεται σε μία απαίτηση υπηρεσίας από ένα πελάτη με την παράδοση τα αντιπροσωπευτικά συγκεκριμένα πρότυπα SLA. Με τον

δυναμικό καθορισμό τους, οι πάροχοι μπορούν να βελτιώσουν την παρεχόμενη ποιότητα και εν συνεχεία την φήμη τους. Αυτή η δυναμικότητα είναι πολύ σημαντική στην επιχειρηματική αγορά , ειδικά για μικρές εταιρείες π.χ. SME αφού συχνά θα πρέπει να συναγωνίζονται με μεγάλους επιχειρηματικούς οργανισμούς.

Τα συστήματα έχουν επεκταθεί κάτω από τις αρχές του OGSA ενεργοποιώντας την ad hoc χειρισμό των διαφόρων μερών και την ανάπτυξη καινούργιων τα οποία θα μπορούν εύκολα να ταιριάξουν χωρίς να λαμβάνεται υπόψη η χρησιμοποιούσα τεχνολογία. Επομένως ο σχεδιασμός του περιγραφόμενου μηχανισμού βασίζεται σ' αυτό, εκτιμούμε ότι προβλέπουμε ότι το συστατικό του QoS History το οποίο αποτελείται από δύο μικρότερα μέρη:

- το QoS History Log και
- το Qos Analyzer



Εικόνα 12: Τα συστατικά μέρη του μηχανισμού QoS History

Το Qos History Log είναι μια αποθήκη στην οποία αποθηκεύονται οι μετρικές οι οποίες εκφράζουν την συμπεριφορά της υπηρεσίας σε σχέση με την συνεργασία της συγκεκριμένης υπηρεσίας και του πελάτη ο οποίος την είχε ζητήσει. Αυτό επιτυγχάνεται με

την παρακολούθηση των παραμέτρων QoS οι οποίοι ορίζονται σε ένα SLA και οι οποίοι αξιολογούνται ανάλογα με τους όρους του SLA. Αυτή η πληροφορία παράγεται απευθείας από πηγές τις οποίες ο παροχέας εμπιστεύεται και αποθηκεύονται επαρκώς στην μορφή δεδομένων του ιστορικού.

Οι πηγές παρουσιάζονται στην αρχιτεκτονική μας ως δύο επιπλέον στοιχεία:

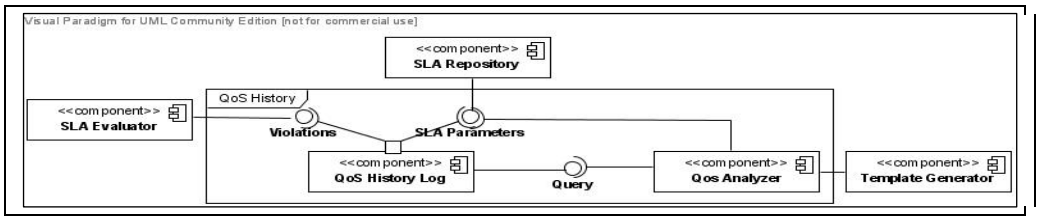
- το SLA Repository
- και το SLA Evaluator

Το SLA Repository είναι το στοιχείο στο οποίο συνοψίζονται τα SLA (τουλάχιστον μέχρι την περίοδο επικύρωσής των SLA). Από την άλλη πλευρά έχουμε τον SLA Evaluator των παρακολουθούμενων δεδομένων. Η κύρια λειτουργικότητα του Evaluator είναι η σύγκριση, ανασκόπηση και κατάταξης της ανάλυσης των παρακολουθούμενων δεδομένων, πάντα σε σχέση με τις παραμέτρους SLA. Είναι επομένως η ιδανική πηγή για να δώσει μια περιγραφή των πραγματοποιημένων ή των επερχόμενων παραβιάσεων οι οποίες έχουν ανιχθευθεί ή , στην καλύτερη περίπτωση, εάν μια υπηρεσία λειτουργεί κατά τη διάρκεια της ζωής της χωρίς κανένα πρόβλημα.

Για να είμαστε ικανοί να αντιμετωπίσουμε πληροφορία η οποία αποθηκεύεται στο QoS History Log, σχεδιάσαμε ένα στοιχείο το οποίο θα έχει τον ρόλο του αναλυτή, το QoS Analyzer. Τα κύρια καθήκοντα αυτού του στοιχείου είναι να χρησιμοποιά την πληροφορία με σκοπό την εξαγωγή συμπερασμάτων σχετικά με τις ανάγκες του πελάτη. Το αποτέλεσμα το οποίο παρέχει αυτό το στοιχείο υπολογισμού μπορεί να χρησιμοποιηθεί για την ενεργοποίηση συγκεκριμένων πολιτικών από πλευράς του ιδιοκτήτη της. Αυτό είναι πολύ βοηθητικό με διαφορετικούς τρόπους. Στη μελέτη του NextGRID επικεντρωνόμαστε και στην επέκταση μηχανισμών που κατανοούν αυτή τη λειτουργία για την δημιουργία εικονικών οργανισμών(Virtual Organizations). Αυτό είναι ένα καλό παράδειγμα, αφού σε ένα NextGRID VO ο διαχειριστής του εικονικού οργανισμού μπορεί είτε να αφαιρέσει ένα

μέλος ή να υποβιβάσει την φήμη του, ενώ οι φορείς μπορούν να αποφασίσουν για την αναδιαμόρφωση των SLA προτύπων τα οποία χρησιμοποιούνται εάν έχουν επίγνωση για το ρίσκο τους.

Οι υψηλού επιπέδου λειτουργίες του παρουσιαζόμενου μηχανισμού παρουσιάζονται στην εικόνα 13:



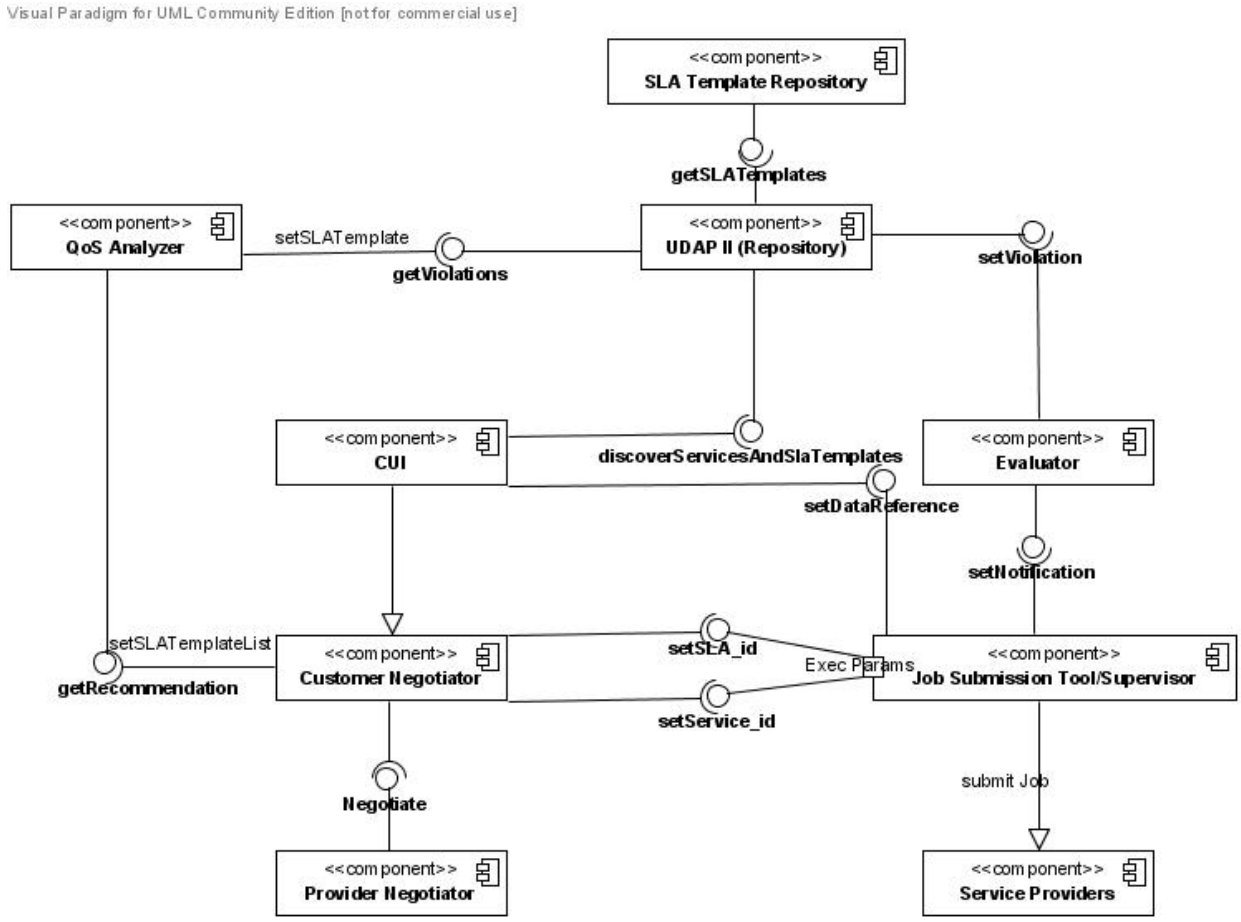
Εικόνα 13: Αλληλεπιδράσεις μεταξύ των διάφορων συστατικών προκειμένου για την αναδιαμόρφωση των SLA προτύπων χρησιμοποιώντας τον μηχανισμό QoS History

Αξίζει να σημειώσουμε ότι ο πιο πάνω μηχανισμός είναι εφαρμοσμένος, με μερικές διαφοροποιήσεις, και στην πλευρά του φορέα παροχής υπηρεσιών και έχει σαν στόχο την ενημέρωση του φορέα σχετικά με τις υπηρεσίες τις οποίες προσφέρει και στην βελτίωση της ποιότητας τους. Είναι στην ουσία ένα συμβουλευτικό εργαλείο. Έχει αναπτυχθεί σαν ένας ανεξάρτητος ο οποίος δεν εγκαθίσταται απευθείας στον φορέα αλλά λειτουργεί σαν ένας μεσολαβητής ο οποίος ελέγχει την υποσχόμενη από τα SLA ποιότητα υπηρεσίας και την πραγματική ποιότητα που φέρνει εις πέρας ο φορέας.

Ο συγκεκριμένος μηχανισμός έχει αναπτυχθεί στην πλευρά του πελάτη ο οποίος χρησιμοποιά αυτό το εργαλείο επίσης σαν συμβουλευτικό για την επιλογή του συγκεκριμένου φορέα ο οποίος ανταποκρίνεται στις απαιτήσεις του σε σχέση με την επιθυμητή ποιότητα υπηρεσίας.

Για σκοπούς πληρότητας παρουσιάζουμε και τον μηχανισμό ο οποίος έχει αναπτυχθεί και στην πλευρά του φορέα παροχής υπηρεσιών. Ένας μηχανισμός ο οποίος παρέχει τις απαραίτητες πληροφορίες στον φορέα για την προσφερόμενη ποιότητα υπηρεσίας από

πλευράς του, έτσι ώστε να μπορεί να αναδιαμορφώσει τις υπηρεσίες του ή τα SLA πρότυπα με στόχο να γίνει πιο ανταγωνιστικός καθώς επίσης και για την εκπλήρωση των επιχειρηματικών πλάνων του.



Εικόνα 14: Αλληλεπιδράσεις που λαμβάνουν χώρα στην πλευρά του φορέα παροχής υπηρεσιών για την παράδοση συμβολαίου SLA και τον έλεγχο για την εκπλήρωση της υποσχόμενης ποιότητα υπηρεσίας από τον φορέα παροχής υπηρεσιών.

3.3.2 Λειτουργία Μηχανισμού

Οι εγγραφές στο regisrty υποθέτουμε ότι έρχονται από του φορείς παροχής υπηρεσιών (μπορεί ο επόπτης του συνολικού μηχανισμού να μπορεί να καταγράψει όλες τις διαθέσιμες υπηρεσίας μετά που οι φορείς παροχής υπηρεσιών του στείλουν μια ειδοποίηση σχετικά με τις παρεχόμενες από αυτούς υπηρεσίες).

Η παράμετρος κλειδί SLA στην οποία στρέφουμε το ενδιαφέρον μας είναι η διαθεσιμότητα.

Επομένως όλα τα SLA πρέπει να περιέχουν αυτή την παράμετρο. Δεν μας δίνετε η δυνατότητα να κάνουμε πολλά με τις άλλες παραμέτρους και επομένως θα πρέπει να στρέψουμε την προσοχή μας σε αυτή την παράμετρο της διαθεσιμότητας.

Ο μηχανισμός QoS Analyzer θα επιχειρήσει ένα πολύ απλουστευμένο υπολογισμό προκειμένου να επιστρέψει μια σύσταση για την σχετική υπηρεσία, και ο οποίος είναι:

$$success_rate = 1 - violation_rate = 1 - \frac{num\ of\ times\ a\ SLA\ was\ violated}{num\ of\ times\ a\ SLA\ was\ used}$$

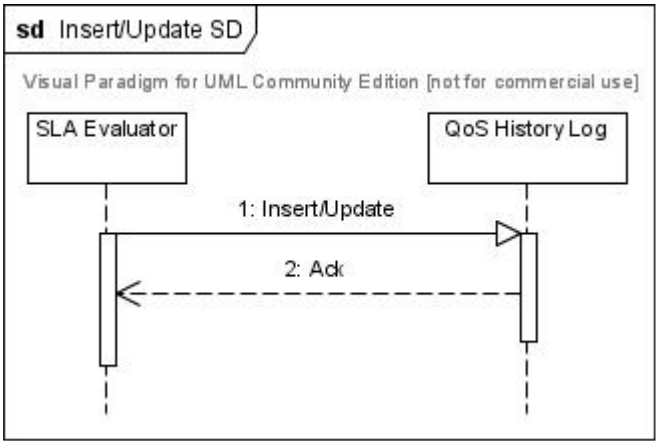
3.3.2.1 Λεπτομερής περιγραφή λειτουργιών μηχανισμού

Από πλευράς υλοποίησης του περιγραφόμενου μηχανισμού οι δύο κύριες συνιστώσες έχουν σχεδιαστεί και ένας ορισμός για την παραπομπή σε αυτές έλαβε χώρα. Όπως έχουμε πει και νωρίτερα ένα εργαλείο διαχείρισης βάσεων δεδομένων το οποίο αποθηκεύει τις εγγραφές υπό μορφή XML αρχείων είχε εγκατασταθεί για την οντότητα QoS History Log. Η μορφή των αρχείων τα οποία αποθηκεύονται στην βάση δεδομένων έχουν σχεδιαστεί κατά μεγάλο μέρος βασιζόμενοι στον συνδυασμό του NextGRID SLA και Event XML σχήματα όπου οι λειτουργικότητες εκτίθενται μέσα από διπροσωπίες υπηρεσιών διαδικτύου. Οι λειτουργίες τις οποίες μπορεί να κάνει το QoS History Log είναι:

- Insert (Εισαγωγή), όποτε ένα καινούργιο πρότυπο SLA χρησιμοποιείται. Μια σειρά από λεπτομέρειες στην μορφή μηνύματος SOAP στέλνονται και αποθηκεύονται στην βάση δεδομένων. Το SLA αντιπροσωπεύεται ως ένα XML έγγραφο το οποίο δημιουργείται χρησιμοποιώντας το σχήμα του SLA, επομένως ταιριάζει τέλεια στο πρότυπο που χρησιμοποιεί η βάση δεδομένων. Συγκεκριμένες παράμετροι ιδιαίτερης σημασίας (εκτός από τις τιμές των κριτηρίων επιτυχίας) που περιέχονται στο πρότυπο του SLA είναι το URI της υπηρεσίας (το οποίο συμπεριφέρεται σαν

μοναδικό αναγνωριστικό) και ο χρόνος ζωής του προτύπου SLA (βλέπε σχήμα 14).

- Update (Ενημέρωση), όποτε χρειάζεται καινούργιες τιμές να προστεθούν ή καινούργιες να διαφοροποιηθούν. Αυτή η λειτουργία ζητείται όποτε συμβεί κάτι σχετιζόμενο με το αποθηκευμένο SLA πρότυπο, όπως μια παραβίαση σε κάποιον όρο (βλέπε εικόνα 15).
- Query (Ερωτήσεις), χρησιμοποιείται όποτε το QoS Analyzer ή μια εμπιστευόμενο τρίτο μέρος χρησιμοποιώ την βάση δεδομένων για την εξαγωγή πληροφοριών.
- Delete (Διαγραφή), χρησιμοποιείται για την διαγραφή εγγραφών, για παράδειγμα, όποτε μια υπηρεσία αφαιρείται από τον εικονικό οργανισμό. Σε αυτό το χρονικό σημείο, αυτή η διαδικασία δεν είναι αυτοματοποιημένη εντούτοις γίνονται προσπάθειες προκειμένου να εξαχθεί πληροφορία από το NextGRID Registry σε σχέση με το πότε μια υπηρεσία είναι διαθέσιμη ή όχι.



Εικόνα 15: Εισαγωγή πληροφοριών στο QoS History Log

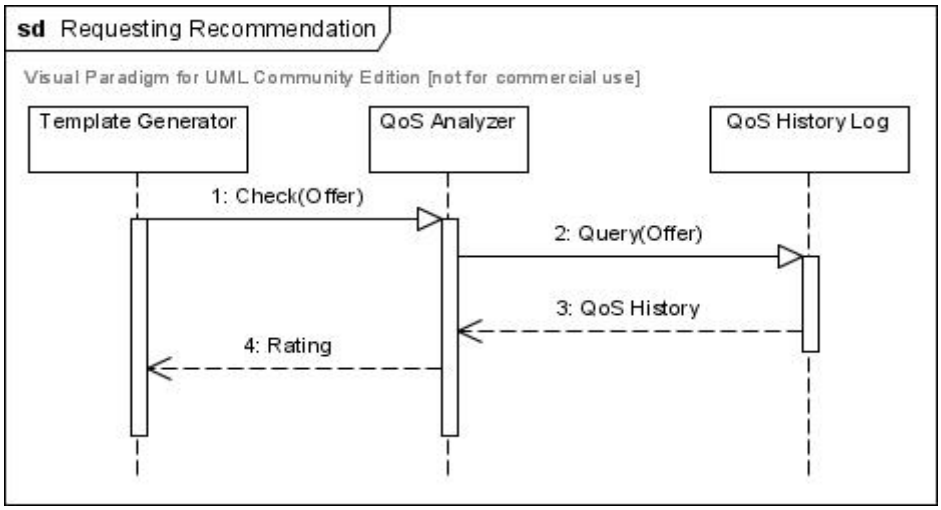
Σύμφωνα με τον σχεδιασμό που παρουσιάστηκε στην Εικόνα 13, το QoS History Log χρειάζεται δύο διαπροσωπείες από μία κοινή θύρα. Η διαπροσωπεία SLA Parameters ισχυρίζεται μια διαδικασία εισαγωγής στην βάση δεδομένων, ενώ η διαπροσωπεία Violations μια ενημέρωση. Αυτό σημαίνει ότι μιας και έχει συμφωνηθεί ένα SLA, το αρχείο αποθηκεύεται στην βάση δεδομένων QoS History XML. Ο συνδυασμός του URI της υπηρεσίας μαζί με τον ακριβή χρόνο στον οποίο έχει υπογραφεί το SLA συμπεριφέρονται

σαν μοναδικά αναγνωριστικά για κάθε εγγραφή. Μόλις ο Evaluator αντιληφθεί μια παραβίαση και ανταποκρίνεται σύμφωνα με ένα σχετικό συμβάν (μέσω ενός μηχανισμού ειδοποίησης), αυτή η πληροφορία επίσης αποθηκεύεται στο Log και συνεργάζεται με το αντίστοιχο SLA και τον συγκεκριμένο όρο SLA (όποτε μπορεί να εφαρμοστεί). Με αυτό τον τρόπο, ένα ιστορικό από παραβιάσεις δημιουργείται για κάθε εμπλεκόμενη υπηρεσία και όλη η πληροφορία για κάποια συγκεκριμένη υπηρεσία μπορεί εύκολα να εξαχθεί.

Αντίστοιχα, το QoS Analyzer χρειάζεται μια διαπροσωπεία για να ερωτά την βάση δεδομένων του Log. Αυτή η διαπροσωπεία προσφέρεται από το QoS History Log (βλέπε εικόνα 13) και ορίζεται σαν μία Xquery στην XML βάση δεδομένων. Αυτή τη χρονική στιγμή, μόνο το URI της υπηρεσίας χρησιμοποιείται ως το αναγνωριστικό της υπηρεσίας, αφού κάθε ερώτηση μπορεί να φέρει σαν αποτέλεσμα διάφορες εγγραφές. Ο Analyzer εν συνεχεία χρησιμοποιά ένα απλό αλγόριθμο για την εκτίμηση της πραγματικής ικανότητας της υπηρεσίας να κρατάει τις υποσχέσεις για ένα συγκεκριμένο όρο SLA, όπως την μέτρηση πόσες φορές η συγκεκριμένη υπηρεσία έχει κληθεί στις οποίες καμία παραβίαση δεν είχε λάβει χώρα στον συγκεκριμένο όρο, πρός του συνολικού αριθμού κλήσης της υπηρεσίας.

Φυσικά πιο περίπλοκοι αλγόριθμοι μπορούν να εφαρμοσθούν εντούτοις αυτό παραμένει σαν μια μελλοντική εργασία. Η κύρια εστίαση αυτής της εργασίας είναι η εφαρμογή ενός πρώτου προτύπου, ενώ η ανάπτυξη είχε σχεδιαστεί με τρόπο ώστε οι αλγόριθμοι να μπορούν να χρησιμοποιηθούν σε σχέση με την πολυπλοκότητα τους για την παράδοση αποτελεσμάτων μεγαλύτερης τιμής. Επιστρέφοντας στην ικανότητα προσέγγισης, το QoS Analyzer το εκφράζει ακολούθως σαν ποσοστό, μια τιμή η οποία δείχνει πόσο συχνά κάποιος μπορεί να πάρει παραβίαση χρησιμοποιώντας ένα συγκεκριμένο όρο από το SLA. Το ποσοστό ακολούθως επιστρέφεται στον αιτητή, ο οποίος είναι ο Template Generator. Το αιτούμενο μήνυμα το οποίο στάλθηκε στο Template Generator κυρίως αποτελείται από δύο

αναγνωριστικά: το αναγνωριστικό της υπηρεσίας και το αναγνωριστικό του προτεινόμενου SLA. Με αυτό τον τρόπο μπορεί να προσαρμοστεί το SLA πρότυπο χρησιμοποιώντας τιμές οι οποίες θα μπορούν να κρατήσουν και να μην παραβιαστούν. Η λογική ροή αναπαρίσταται στην ακόλουθη εικόνα 16.



Εικόνα 16: Έλεγχος για πιθανή προσφορά

4 Υλοποίηση

4.1 Πλατφόρμες και προγραμματιστικά εργαλεία

Η υλοποίηση του μηχανισμού για το ιστορικό για την ποιότητα της υπηρεσίας (QoS History), όπως αναφέραμε θα αποτελείται από διάφορα μέρη χρησιμοποιώντας διαφορετικές τεχνολογίες με σκοπό την προσέγγιση όλων των προδιαγραφών που περιγράφηκαν. Έτσι σε μια γενική ανάλυση του μηχανισμού έχουμε :

- *Insert:* web service υλοποιημένο σε Java κώδικα και εγκατεστημένο στον μηχανισμό ο οποίος είναι ανεξάρτητος και έχει τις αρμοδιότητες αυτές ως μεσολαβητής μεταξύ του παροχέα και του πελάτη. Η υπηρεσία αυτή είναι η πλευρά του Provider, στο μοντέλο που παρουσιάστηκε νωρίτερα, και ο ρόλος της είναι να εισάγει τα SLA πρότυπα τα οποία δίνονται στους πελάτες από τον φορέα.

- *Update:* web service υλοποιημένο σε Java κώδικα και εγκατεστημένο στον μηχανισμό ο οποίος είναι ανεξάρτητος και έχει τις αρμοδιότητες αυτές ως μεσολαβητής μεταξύ του παροχέα και του πελάτη. Η υπηρεσία αυτή είναι η πλευρά του Provider, στο μοντέλο που παρουσιάστηκε νωρίτερα, και ο ρόλος της είναι να ενημερώνει τα ήδη εισαγόμενα SLA πρότυπα τα οποία συμφωνήθηκαν μεταξύ του φορέα και του πελάτη.
- *Rate:* web service υλοποιημένο σε Java κώδικα και εγκατεστημένο στον μηχανισμό ο οποίος είναι ανεξάρτητος και έχει τις αρμοδιότητες αυτές ως μεσολαβητής μεταξύ του παροχέα και του πελάτη. Η υπηρεσία αυτή είναι η πλευρά του Provider, στο μοντέλο που παρουσιάστηκε νωρίτερα, και ο ρόλος της είναι να δίνει ένα ποσοστό σχετικά με την υπηρεσία που έχει επιλέξει ο πελάτης για το κατά πόσο η συγκεκριμένη υπηρεσία θα υποπέσει σε παραβίαση του συγκεκριμένου όρου της διαθεσιμότητας.
- XML Database: Το διαχειριστικό εργαλείο της βάσης δεδομένων που χρησιμοποιήθηκε για την υλοποίηση της βάσης δεδομένων είναι η Oracle Berkeley DB XML 2.3, η οποία δίνει την δυνατότητα της εισαγωγής των εγγραφών σε μορφή XML αρχείων, κάτι το οποίο συμβαδίζει με τα SLA τα οποία έχουνε την ίδια μορφή.

4.1.1 Java

Η επιλογή της JAVA ως κώδικα ανάπτυξης του μηχανισμού έγινε πολύ προσεκτικά βάσει των μεγάλων δυνατοτήτων που αυτή προσφέρει. Σύμφωνα με την κατασκευάστρια της εταιρία την Sun, «Η Java είναι μια απλή, αντικειμενοστρεφής, κατανεμημένη, ερμηνευόμενη, εύρωστη, αρχιτεκτονικά ουδέτερη, μεταφερόμενη, υψηλής απόδοσης, πολυνηματική, και ασφαλής γλώσσα».

Η JAVA, από την αρχή σχεδιάστηκε για εφαρμογές που θα λειτουργούσαν πάνω από το Internet. Έχει έτσι πολλά πλεονεκτήματα σε σχέση με την γρήγορη και ασφαλή διαδικτυακή επικοινωνία κάτι αναγκαίο στις υπηρεσίες του Grid. Εξαιτίας αυτού έχει καταστεί ως μία από τις βασικότερες γλώσσες προγραμματισμού για την ανάπτυξη εφαρμογών και υπηρεσιών διαδικτύου (web services).

Το κυριότερο βέβαια χαρακτηριστικό της JAVA είναι ότι είναι Platform Independent, μπορεί δηλαδή να λειτουργήσει σε κάθε υπολογιστικό σύστημα, είτε αυτό είναι προσωπικός υπολογιστής με Linux ή Windows, είτε ένα οποιοδήποτε ενσωματωμένο σύστημα (embedded system) το οποίο ενσωματώνει αυτή τη τεχνολογία. Αυτή η ιδιαιτερότητα της δίνει και το μεγάλο προβάδισμα αφού στη τεχνολογία Grid είναι χαρακτηριστική η συνεργασία πολλών διαφορετικών συστημάτων. Η έκδοση που θα χρησιμοποιήσουμε για την υλοποίηση μας είναι η Java 2 Platform, Standard Edition 5.0 Update 6 (http://java.sun.com/javaee/downloads/index.jsp) και η πλατφόρμα ανάπτυξης ήταν το Eclipse, ελεύθερο λειτουργικό που διατίθεται στο διαδίκτυο (http://www.eclipse.org/).

Εναλλακτικά της JAVA θα μπορούσε να χρησιμοποιηθεί κάποια άλλη γλώσσα προγραμματισμού όπως C++ ή ακόμα και άλλη πλατφόρμα ανάπτυξης όπως .NET . Η πρώτη εναλλακτική λύση της C++ βρίσκεται πολύ κοντά σε λογική υλοποίησης με την JAVA, αφού είναι και αυτή αντικειμενοστρεφής γλώσσα αλλά δεν είναι τόσο ευέλικτη και προσανατολισμένη σε δικτυακές εφαρμογές. Από την άλλη η επιλογή μιας πλατφόρμας ανάπτυξης όπως το .NET θα μας διευκόλυνε σε κάποια σημεία ανάπτυξης αφού έχει σχεδιαστεί για δικτυακές εφαρμογές, όμως μας περιορίζει αρκετά στα υπολογιστικά συστήματα που υποστηρίζει αφού είναι προσανατολισμένη κυρίως για Windows περιβάλλοντα.

4.1.2 Oracle Berkeley DB XML

Η επιλογή της συγκεκριμένης βάσης δεδομένων προέκυψε μετά από μελέτη πολλών εργαλείων βάσεων δεδομένων. Οι λόγοι που μας έχουν οδηγήσει σε αυτή την επιλογή είναι η μεγάλη ευελιξία που μας δίνει το συγκεκριμένο εργαλείο για τον λόγο ότι τα αρχεία που αποθηκεύονται είναι συγκεκριμένης μορφής αλλά με την δυνατότητα προσθήκης επιπλέον πεδίων όπως ορίζονται από την μορφή του XML. Επίσης κάποια επιπλέον πλεονεκτήματα της συγκεκριμένης τεχνολογίας είναι:

- Αποθήκευση συνοπτικών πληροφοριών για τις υπηρεσίες,
    - Μαζί με το ευρύ δίκτυο σχέσεων μεταξύ των στοιχείων των πληροφοριών - αντικείμενα-ιδιότητες-μέθοδοι, αντικείμενο-αντικείμενο κ.λπ.,
    - Χωρίς την απαραίτητη μετάφραση των αντικειμένων ή δομών XML σε μία "επίπεδη" ή συγγενική δομή βάσεων δεδομένων και αντίστροφα.

Περισσότερες πληροφορίες του σχετικού εργαλείου μπορεί να βρει κανείς ανατρέχοντας στην αντίστοιχη ιστοσελίδα (http://www.oracle.com/database/berkeley-db/xml/index.html).

4.1.3 Εξυπηρετητής-Web Server

Ένας web server είναι ένα πρόγραμμα που δέχεται αιτήσεις σε HTTP (HTTP Requests) και απαντά παρέχοντας αρχεία HTML.

4.1.4 Tomcat

Ο Tomcat είναι ένα sevlet container ή servlet engine. Είναι ένα υπερσύνολο ενός web-server, που υποστηρίζει servlets και JSPs καθώς και στατικές σελίδες δεχόμενος requests

στην πόρτα 8080. Είναι ένα εγχείρημα της Apache μέρος του jakarta project και διατίθεται δωρεάν στο διαδίκτυο (http://jacarta.apache.org/tomcat.). Θεωρείται από τους πλέον αξιόπιστους web servers και χρησιμοποιείται ευρέως στην ανάπτυξη δικτυακών υπηρεσιών. Στην παρούσα εφαρμογή χρησιμοποιήθηκε η έκδοση 5.1.

Τα Servlets είναι προγράμματα JAVA τα οποία επεκτείνουν τη λειτουργικότητα ενός web server, παράγοντας δυναμικό περιεχόμενο, αλληλεπιδρώντας με clients με ένα μοντέλο αίτησης-απάντησης. Είναι η τεχνολογία της JAVA για τη δημιουργία δυναμικών σελίδων στο διαδίκτυο. Τα Servlets έχουν τη δυνατότητα να εμπεριέχουν HTML tags στον JAVA κώδικα.

Οι JSPs είναι ουσιαστικά HTML σελίδες οι οποίες περιέχουν και JAVA κώδικα. Είναι δηλαδή το συμμετρικό μοντέλο ως προς τα Servlets και εξυπηρετούν τους ίδιους σκοπούς.

4.1.5 AXIS

Το Apache Axis είναι ένα πλαίσιο εργασίας για την ανάπτυξη δικτυακών υπηρεσιών (web service development framework), βασισμένο σε Java και XML και χρησιμοποιώντας SOAP κωδικοποίηση επικοινωνίας. Το Axis σε συνδυασμό με τον Tomcat Web Server, μας παρέχουν όλες τις απαραίτητες λειτουργίες για την ανάπτυξη και διαχείριση των υπηρεσιών. Το framework αυτό διατίθεται δωρεάν από την ιστοσελίδα του apache.org (http://ws.apache.org/axis/). Η έκδοση που θα χρησιμοποιήσουμε είναι η 2.0.

4.2 Λεπτομέρειες Υλοποίησης

Το σύστημα που σχεδιάζουμε μπορούμε να το χωρίσουμε σε επίπεδο υλοποίησης σε δύο μέρη:

- Server Side : η υλοποίηση της υπηρεσίας του QoS History που θα είναι εγκατεστημένη

στον μεσολαβητή και σε κάθε παροχέα που επιθυμούμε να εντάξουμε στο σύστημα.

• Client Side : η υλοποίηση της εφαρμογής που ο χρήστης θα παίρνει πληροφορίες σχετικά με την παρεχόμενη ποιότητα υπηρεσίας των αντίστοιχων υπηρεσιών που επιθυμεί να καλέσει.

4.2.1 Server Side

Τα απαραίτητα αρχεία/κλάσεις αυτής της υλοποίησης έχουν να κάνουν κυρίως με την υπηρεσία για την εξαγωγή των πληροφοριών από τους πόρους και την υπηρεσία διαχείρισης του μηχανισμού (προσθήκη-αφαίρεση πόρων). Έτσι τα βασικά αρχεία είναι :

• InsertSLA.java : το αρχείο αποτελεί την υλοποίηση της υπηρεσίας για την εισαγωγή των SLA προτύπων στην βάση δεδομένων τα οποία έχουν συμφωνηθεί μεταξύ του πελάτη και του φορέα παροχής υπηρεσιών. Αποτελεί ουσιαστικά τον πυρήνα του QoS History και χρησιμοποιείται ως υπηρεσία (Insert) μέσω του tomcat.

• RecommendSLA.java : είναι η υλοποίηση της υπηρεσίας για την διαχείριση του συστήματος που σχεδιάζουμε. Παρέχει προς το παρόν δυνατότητες προσθήκης και αφαίρεσης πόρων από τον μηχανισμό του QoS Provisioning.

Ολοκληρωμένα ο κώδικας όλων των αρχείων δίνεται στο Παράρτημα, στο τέλος του εγγράφου.

4.2.1.1 Αρχείο InsertSLA.java

Παρακάτω παρουσιάζουμε τις μεθόδους που εμπεριέχονται στην υπηρεσία Insert που δημιουργείται από την εγκατάσταση του κώδικα του αρχείου που εξετάζουμε στον tomcat

server.

OpenEnv: Αυτή η μέθοδος ανοίγει την βάση δεδομένων για να γίνει δυνατή η εισαγωγή των SLA προτύπων που έχουν συμφωνηθεί μεταξύ του φορέα παροχής και του πελάτη. Είναι αναγκαία για την ορθή διατήρηση της συνοχής της βάσης δεδομένων καθώς επίσης και για τον συγχρονισμό σε περίπτωση ταυτόχρονης προσπέλασης της βάσης από διαφορετικούς πελάτες.

CloseEnv: Αυτή η μέθοδος κλείνει την βάση δεδομένων στο τέλος της συναλλαγής με την βάση δεδομένων. Απαραίτητη για τον λόγο ότι πρέπει να αποθηκεύονται σωστά στην βάση δεδομένων και την διατήρηση της συνοχής της βάσης δεδομένων.

4.2.1.2 Αρχείο RecommendSLA.java

Παρακάτω παρουσιάζουμε τις μεθόδους που εμπεριέχονται στην υπηρεσία Recommend που δημιουργείται από την εγκατάσταση του κώδικα του αρχείου που εξετάζουμε στον tomcat server.

OpenEnv, *CloseEnv*: Όπως έχουν παρουσιαστεί στην προηγούμενη παράγραφο και έχουν ως σκοπό την ορθή προσπέλαση της βάσης δεδομένων και την συνοχή στις πληροφορίες και τις εγγραφές τις οποίες περιέχει.

doQuery: Αυτή η μέθοδος έχει σαν σκοπό την εύρεση του συγκεκριμένου SLA προτύπου για το οποίο ενδιαφέρεται ο φορέας σχετικά με την παραδιδόμενη από αυτόν ποιότητα υπηρεσίας.

getNewDocument: Αυτή η μέθοδος έχει σαν στόχο την δημιουργία ενός αρχείου το οποίο θα περιέχει όλες τις πληροφορίες σχετικά με το SLA πρότυπο το οποίο είναι αποθηκευμένο στην βάση δεδομένων. Ο σκοπός είναι για την εύρεση του συγκεκριμένου πεδίου στην εγγραφή στην οποία αναφέρεται η παραβίαση του συμφωνηθέν όρου.

getValue: Αυτή η μέθοδος σε συνεργασία με την προηγούμενη που έχουμε περιγράψει έχει σαν σκοπό την ανάκτηση του συγκεκριμένου πεδίου του καταχωρημένου SLA προτύπου το οποίο σχετίζεται με το εάν ή όχι έχει υποπέσει η συγκεκριμένη υπηρεσία σε παραβίαση του συγκεκριμένου όρου.

4.2.2 Client Side

4.2.2.1 Αρχείο Insert.java

Έχει τις ίδιες λειτουργίες με το InsertSLA.java, δηλαδή καλείται η υπηρεσία Insert η οποία έχει εγκατασταθεί στον tomcat server. Έχει τις ίδιες μεθόδους όπως έχουμε περιγράψει και στην προηγούμενη παράγραφο. Για σκοπούς πληρότητας παραθέτουμε ξανά τις μεθόδους τις οποίες καλεί.

OpenEnv: Αυτή η μέθοδος ανοίγει την βάση δεδομένων για να γίνει δυνατή η εισαγωγή των SLA προτύπων που έχουν συμφωνηθεί μεταξύ του φορέα παροχής και του πελάτη. Είναι αναγκαία για την ορθή διατήρηση της συνοχής της βάσης δεδομένων καθώς επίσης και για τον συγχρονισμό σε περίπτωση ταυτόχρονης προσπέλασης της βάσης από διαφορετικούς πελάτες.

CloseEnv: Αυτή η μέθοδος κλείνει την βάση δεδομένων στο τέλος της συναλλαγής με την βάση δεδομένων. Απαραίτητη για τον λόγο ότι πρέπει να αποθηκεύονται σωστά στην βάση δεδομένων και την διατήρηση της συνοχής της βάσης δεδομένων.

4.2.2.2 Αρχείο Update.java

Παρακάτω παρουσιάζουμε τις μεθόδους που εμπεριέχονται στην υπηρεσία Update που δημιουργείται από την εγκατάσταση του κώδικα του αρχείου που εξετάζουμε στον tomcat server.

OpenEnv, CloseEnv: Όπως έχουν παρουσιαστεί στην προηγούμενη παράγραφο και έχουν ως σκοπό την ορθή προσπέλαση της βάσης δεδομένων και την ορθή συνοχή στις πληροφορίες και τις εγγραφές τις οποίες περιέχει.

doUpdateDocument: Η συγκεκριμένη μέθοδος παίρνει από την βάση δεδομένων το συγκεκριμένο συμφωνηθέν SLA πρότυπο το οποίο σε περίπτωση που παρουσιαστεί μια παραβίαση στο συγκεκριμένο όρο της υπηρεσίας γίνεται ενημέρωση του αρχείου στο οποίο αποθηκεύεται το SLA πρότυπο. Αυτή η διαδικασία γίνεται με την ανάκτηση της εγγραφής και την ενημέρωση του πεδίου στο οποίο γίνεται αναφορά για ενδεχόμενη παραβίαση.

4.2.2.3 Αρχείο Rate.java

Έχει τις ανάλογες λειτουργίες με την κλάση RecommendSLA.java, δηλαδή καλείται η υπηρεσία Recommend που δημιουργείται από την εγκατάσταση του κώδικα του αρχείου που εξετάζουμε στον tomcat server. Έχει τις ίδιες μεθόδους όπως έχουμε περιγράψει και

στην προηγούμενη παράγραφο. Για σκοπούς πληρότητας παραθέτουμε ξανά τις μεθόδους τις οποίες καλεί.

OpenEnv, CloseEnv: Όπως έχουν παρουσιαστεί στην προηγούμενη παράγραφο και έχουν ως σκοπό την ορθή προσπέλαση της βάσης δεδομένων και την συνοχή στις πληροφορίες και τις εγγραφές τις οποίες περιέχει.

doQuery: Αυτή η μέθοδος έχει σαν σκοπό την εύρεση του συγκεκριμένου SLA προτύπου για το οποίο ενδιαφέρεται ο φορέας σχετικά με την παραδιδόμενη από αυτόν ποιότητα υπηρεσίας.

getNewDocument: Αυτή η μέθοδος έχει σαν στόχο την δημιουργία ενός αρχείου το οποίο θα περιέχει όλες τις πληροφορίες σχετικά με το SLA πρότυπο το οποίο είναι αποθηκευμένο στην βάση δεδομένων. Ο σκοπός είναι για την εύρεση του συγκεκριμένου πεδίου στην εγγραφή στην οποία αναφέρεται η παραβίαση του συμφωνηθέν όρου.

getValue: Αυτή η μέθοδος σε συνεργασία με την προηγούμενη που έχουμε περιγράψει έχει σαν σκοπό την ανάκτηση του συγκεκριμένου πεδίου του καταχωρημένου SLA προτύπου το οποίο σχετίζεται με το εάν ή όχι έχει υποπέσει η συγκεκριμένη υπηρεσία σε παραβίαση του συγκεκριμένου όρου.

Πρέπει να σημειώσουμε ότι η ανάπτυξη του μηχανισμού είναι ένα συμμετρικό σύστημα το οποίο λειτουργεί και στην πλευρά του εξυπηρετητή και στην πλευρά του πελάτη. Ο σκοπός είναι για να λειτουργεί σαν συμβουλευτικό εργαλείο στην πλευρά του εξυπηρετητή για την πληροφόρηση του σχετικά με τις παρεχόμενες από αυτόν υπηρεσίες. Η εισαγωγή των SLA

προτύπων από τον εξυπηρετητή γίνεται μετά το τέλος του κύκλου ζωής του SLA και αφού έχει προσδιοριστεί εάν έχει γίνει κάποια παραβίαση στον συγκεκριμένο όρο του προτύπου. Ο λόγος είναι για την ορθή εγκυρότητα των στοιχείων στην βάση δεδομένων και την αξιόπιστη πληροφορία. Επίσης στην πλευρά του πελάτη έχει γίνει η ανάπτυξη του μηχανισμού έτσι ώστε η εισαγωγή των στοιχείων στην βάση δεδομένων να γίνεται μετά την συμφωνία μεταξύ του φορέα και του πελάτη. Η ενημέρωση της βάσης γίνεται κατά την διάρκεια του κύκλου ζωής του SLA σε περίπτωση που κάποια παραβίαση έχει προκύψει. Ουσιαστικά μιλάμε για δύο διαφορετικές βάσεις δεδομένων στις οποίες ο κάθε ένας που συμμετάσχει λαμβάνει τα ανάλογα δεδομένα τα οποία τον αφορούν και τα οποία είναι τόσο αξιόπιστα όσο και πλήρη.

4.2.3 Βάση Δεδομένων

Παρακάτω παρουσιάζεται η μορφή των εγγραφών στην βάση δεδομένων τόσο στην πλευρά του φορέα παροχής υπηρεσιών όσο και από την πλευρά του πελάτη

4.2.3.1 Βάση Δεδομένων Πελάτη

Η μορφή των αρχείων τα οποία αποθηκεύονται στην βάση δεδομένων του πελάτη έχει την κάτωθι μορφή, η οποία μοιάζει κατά πολύ με το πρότυπο SLA το οποίο συμφωνήθηκε μεταξύ του πελάτη και του φορέα.

```
<QoSdb>
        <SLAEpr></SLAEpr>
        <SLATemplateEpr></SLATemplateEpr>
        <StartingDate></StartingDate>
```

```
        <EndDate>EndDate</EndDate>
        <Party>Party</Party>
        <QoEViolation>Yes/No</QoEViolation>
        <BindingTerm>
            <Event>Event</Event>
            <Name>Name</Name>
            <SuccessCriterion>SuccessCriterion</SuccessCriterion>
            <Bound>
                <annotation>
                <documentation>Some text</documentation>
                </annotation>
                <simpleType>
                    <base>
                        <LE>LE</LE>
                    </base>
                </simpleType>
            </Bound>
        </BindingTerm>
    </QoSdb>
```

Είναι δυνατόν για ένα συμφωνηθέν SLA να υπάρχουν πέραν του ενός BindingTerm οι οποίοι ορίζουν και του όρους της επιτυχούς εκτέλεσης και περάτωσης της απαιτούμενης ποιότητας υπηρεσίας. Η εισαγωγή αυτών των επιπλέον όρων είναι εφικτή με την χρήση των αρχείων XML αφού η εισαγωγή και αφαίρεση επιπλέον πεδίων σε μια εγγραφή είναι σχετικά απλή διαδικασία.

4.2.3.2 Βάση Δεδομένων Φορέα Παροχής Υπηρεσιών

Η μορφή των αρχείων τα οποία αποθηκεύονται στην βάση δεδομένων του φορέα παροχής υπηρεσιών έχει την κάτωθι μορφή, η οποία έχει σχέση με τις συμφωνηθέντες υπηρεσίες τις οποίες παρέχει στον πελάτη και κατά πόσο κάποια υπηρεσία έχει υποπέσει σε κάποια παραβίαση ή όχι.

```
<QoE_Information>
    <process>
        <name>paint</name>
        <slaepr>
            <uri>http://paintsla...</uri>
            <slatemplatetype>Gold</slatemplatetype>
        </slaepr>
        <serviceurl>http://paint...</serviceurl>
        <state>COMPLETE</state>
        <time>1605</time>
        <message>outputs</message>
    </process>
</QoE_Information>
```

Στο πεδίο state φαίνεται κατά πόσο μια υπηρεσία έχει υποπέσει σε παραβίαση ή όχι. Αξίζει να σημειώσουμε ότι ο φορέας παροχής υπηρεσιών ενδέχεται να αναλάβει την περάτωση περισσότερων από μία εργασία. Με το πεδίο process εννοούμε την ανατιθέμενη σε αυτόν εργασία. Έτσι το xml αρχείο που έρχεται στον παροχέα ενδέχεται να περιέχει περισσότερες από μια processes, τις οποίες σπάζει σε καθεμία ξεχωριστά και όλες μαζί αποθηκεύονται.

5 *Έλεγχος*

5.1 Οδηγός Εγκατάστασης

5.1.1 Εγκατάσταση του Tomcat και AXIS

Το αρχείο για την εγκατάσταση του Tomcat διατίθεται δωρεάν στο διαδίκτυο στη διεύθυνση (http://jacarta.apache.org/tomcat). Στην παρούσα εφαρμογή χρησιμοποιήθηκε η έκδοση 5.5.1.5.Για την εγκατάσταση σε Linux περιβάλλον, αρκεί να αντιγράψουμε τον κατάλογο που κατεβάσαμε από την ιστοσελίδα, στην τοποθεσία που επιθυμούμε να το εγκαταστήσουμε. Για την εκκίνηση του server εκτελούμε απλά το αρχείο startup.sh στον κατάλογο bin. Σε Windows περιβάλλον, απλά εκτελούμε το αρχείο που κατεβάσαμε από την ιστοσελίδα και ακολουθούμε τις οδηγίες.

Το AXIS προσφέρεται στο διαδίκτυο δωρεάν (http://ws.apache.org/axis/). Για την

εγκατάσταση του αρκεί να αντιγράψουμε τον κατάλογο webapps/axis, της διανομής που κατεβάσαμε από το διαδίκτυο, και να το τοποθετήσουμε μέσα στον κατάλογο webapps του tomcat server. Αφού εκκινήσουμε τον tomcat μπορούμε να επισκεφθούμε την διεύθυνση http://127.0.0.1:8080/axis/, και να επιβεβαιώσουμε την σωστή εγκατάσταση του προγράμματος αυτού. Η έκδοση που χρησιμοποιήσαμε εμείς ήταν η 1.4. Για την ορθή λειτουργία του AXIS απαιτείται η ρύθμιση κάποιων μεταβλητών του συστήματος (system variables). Παρακάτω εξηγούμε ακριβώς την διαδικασία, μαζί με όλες τις υπόλοιπες απαιτήσεις.

5.1.2 Ρύθμιση μεταβλητών συστήματος και βιβλιοθηκών

Εκτός από τα προηγούμενα που αναφέρθηκαν είναι απαραίτητη και η ύπαρξη Java Runtime Environment στο μηχάνημα που αναφερόμαστε. Η προτεινόμενη έκδοση είναι η 1.5.0_06 που προσφέρεται δωρεάν από την ιστοσελίδα της SUN (http://java.sun.com/javase/downloads/index.jsp). Επίσης για την ολοκληρωμένη λειτουργία της Java στην ανάπτυξη Web Services απαιτείται η εγκατάσταση των Xerces API, Javamail API και Jaf API. Αυτό σημαίνει την αντιγραφή των αντίστοιχων αρχείων (xerces.jar, mail.jar, activation.jar) στην βιβλιοθήκη αρχείων του tomcat. Στην εφαρμογή μας χρησιμοποιήσαμε τις εκδόσεις Xerces 1.4.4, Javamail 1.4 και Jaf 1.1.

Για την ρύθμιση των μεταβλητών του συστήματος χρειάζεται να δημιουργήσουμε καινούριες μεταβλητές . Θεωρώντας ότι ο κατάλογος που είναι εγκατεστημένο το AXIS είναι ο usr/axis τοποθετούμε τις εντολές αυτές στο αρχείο ets/bash.bashrc :

```
set AXIS_HOME=/usr/axis
set AXIS_LIB=$AXIS_HOME/lib
```

```
set AXISCLASSPATH=$AXIS_LIB/axis.jar:$AXIS_LIB/commons-discovery.jar:
$AXIS_LIB/commonslogging.
jar:$AXIS_LIB/jaxrpc.jar:$AXIS_LIB/saaj.jar:
$AXIS_LIB/log4j-1.2.8.jar:$AXIS_LIB/xmlapis.
jar:$AXIS_LIB/xercesImpl.jar:$AXIS_LIB /mail.jar:$AXIS_LIB
/activation.jar
export AXIS_HOME; export AXIS_LIB;
export AXISCLASSPATH
```

Εκτός από τις παραπάνω μεταβλητές συστήματος πρέπει να ορίσουμε και τις μεταβλητές για την Java, JAVA_HOME και JRE_HOME που θα περιέχουν το κατάλογο που βρίσκεται εγκατεστημένη η Java και Jre αντίστοιχα.

5.1.3 Εγκατάσταση Βάσης Δεδομένων

Ακολουθώντας τις οδηγίες που υπάρχουν στην ιστοσελίδα της Oracle κατεβάζουμε την διανομή που διατίθεται (http://www.oracle.com/database/berkeley-db/xml/index.html) ανάλογα με το λειτουργικό που χρησιμοποιούμε στο μηχάνημα που θα εγκαταστήσουμε τη βάση. Η προτεινόμενη έκδοση είναι η 2.3 και η εγκατάσταση της εξαρτάται από το λειτουργικό που χρησιμοποιούμε. Για λεπτομέρειες ακολουθήστε οδηγίες που υπάρχουν στην ιστοσελίδα.

5.1.4 Εγκατάσταση QoS History

Για την εγκατάσταση της υπηρεσίας μας χρειάζεται η εξής διαδικασία. Αρχικά πρέπει να κάνουμε deploy τις δικτυακές υπηρεσίες (web services) στον tomcat server. Αρχικά, ενώ ο

tomcat server “τρέχει”, χρειάζεται να εκτελέσουμε την παρακάτω εντολή ενώ βρισκόμαστε στους καταλόγους Insert, Recommend και Update, για να γίνει η δημιουργία των υπηρεσιών, πληκτρολογούμε την εντολή ant και αφού μας πει BUILD SUCCESFUL τότε το μόνο που απομένει είναι η εγκατάσταση των υπηρεσιών στον tomcat server. Αυτό γίνεται με την εντολή "cp /home/ntua/workspace/dbxml/build/HistoryServices.aar /opt/tomcat/webapps/axis2/WEB-INF/services/" , όπου HistoryServices.aar είναι το αρχείο στο οποίο έχουν δημιουργηθεί οι παραπάνω υπηρεσίες.

Η αναλυτική περιγραφή του build αρχείου βρίσκεται στο παράρτημα όπου και μπορείτε να αντέξετε για την ολοκληρωμένη κατανόηση της διαδικασίας ανάπτυξης και εγκατάστασης των υπηρεσιών.

Μετά την ολοκλήρωση των παραπάνω βημάτων είμαστε έτοιμοι για την κλήση των υπηρεσιών που βρίσκονται πλέον εγκατεστημένες στον tomcat server.

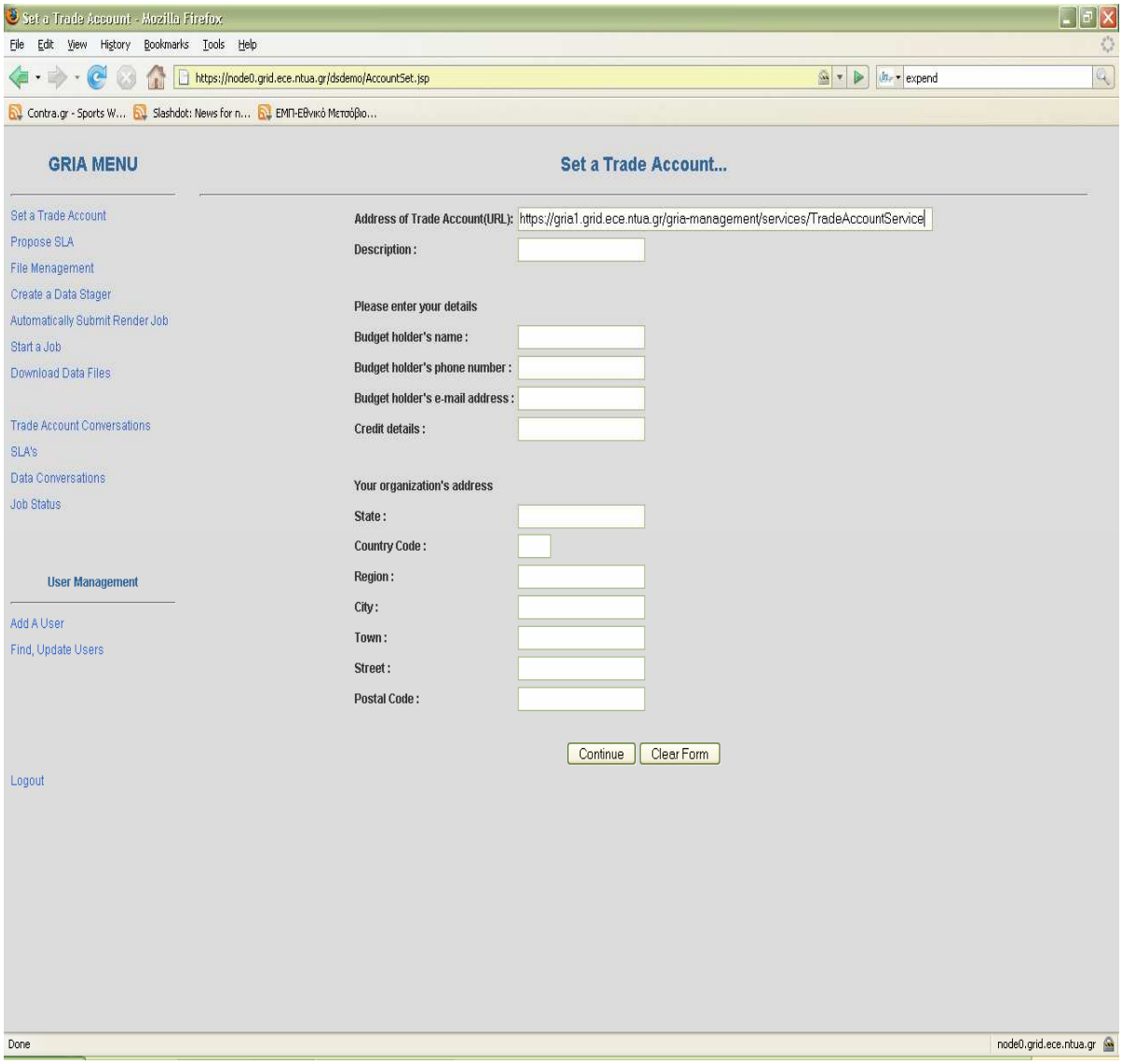
5.2 Αναλυτική παρουσίαση ελέγχου

5.2.1 Τα αποτελέσματα της χρήσης του συστήματος

Στην παράγραφο αυτή θα παρουσιάσουμε την εκτέλεση ενός σεναρίου χρήσης του συστήματος, από ένα πελάτη ο οποίος κάνει κλήση μιας συγκεκριμένης υπηρεσίας και ακολούθως βλέπουμε την περίπτωση όπου κάποια υπηρεσία έχει υποπέσει σε παραβίαση καθώς επίσης και την περίπτωση της ομαλούς διεκπεραίωσης της απαιτούμενης εργασίας. Επίσης θα δείξουμε και το συμβουλευτικό εργαλείο του φορέα παροχής υπηρεσιών ο οποίος έχει γνώσει για την παρεχόμενη ποιότητα των υπηρεσιών του.

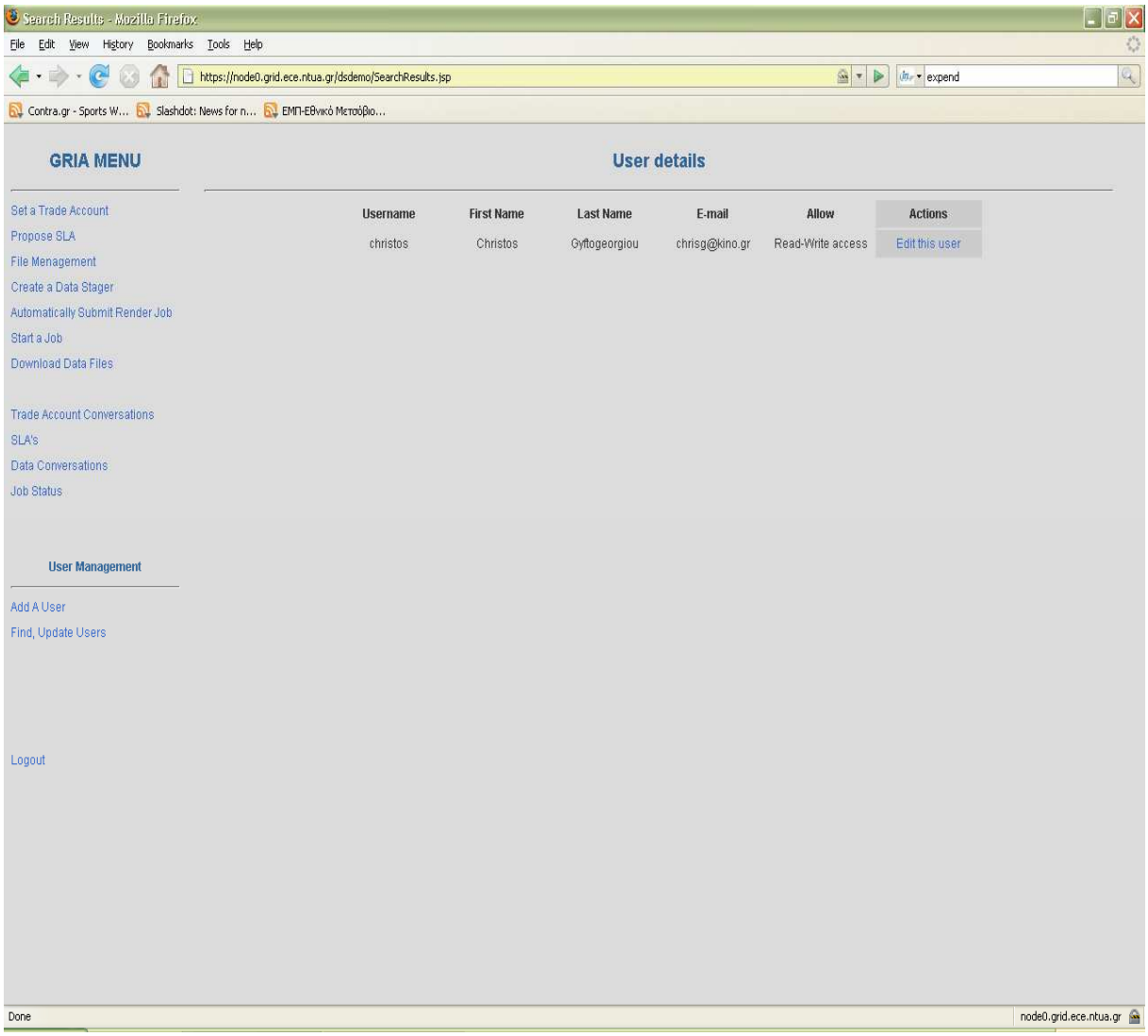
Αρχικά ζητείται από τον χρήστη να εγγραφεί στο σύστημα με το άνοιγμα ενός λογαριασμού σε αυτό και αυτό γίνεται με την συμπλήρωση των αναγκαίων πεδίων που υπάρχουν στην

σελίδα εγγραφής:



Εικόνα 17: Εγγραφή του χρήστη με την συμπλήρωση των στοιχείων του στο σύστημα το οποίο του δίνει την δυνατότητα της χρήσης διαφόρων υπηρεσιών.

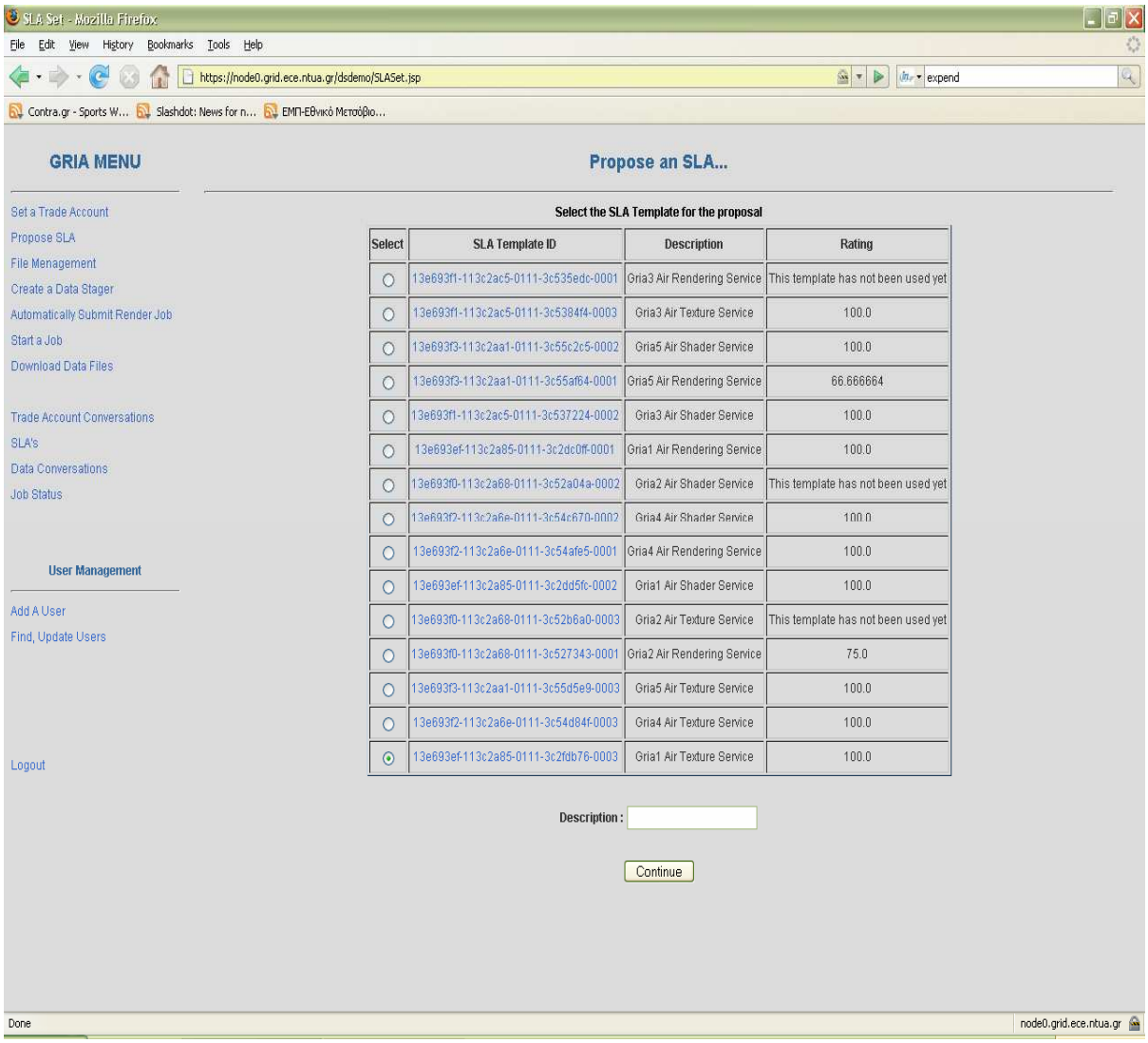
Μετά την επιτυχή εγγραφή του χρήστη στο σύστημα, μπορούμε να δούμε από την κονσόλα διαχείρισης των εγγραφόμενων μελών ότι ο χρήστης έχει εισαχθεί στην λίστα χρηστών του συστήματος καθώς και οι δυνατότητες που έχει ο χρήστης στην χρήση του συστήματος. Αυτό φαίνεται από την παρακάτω εικόνα:



Εικόνα 18: Κονσόλα διαχείρισης του συστήματος η οποία δείχνει τους εγγεγραμμένους χρήστες καθώς επίσης και των προνομίων που έχουν στο σύστημα.

Ακολούθως ο χρήστης επιλέγει το συγκεκριμένο SLA πρότυπο βάση της εφαρμογής συγκεκριμένης υπηρεσίας που έχει σκοπό να καλέσει και βάση της παραδιδόμενης από τον παροχέα ποιότητα υπηρεσίας. Αυτά τα δύο πεδία είναι εμφανή στις στήλες Description και Rating αντίστοιχα. Η πιο πάνω περιγραφόμενη λειτουργία φαίνεται στο σχήμα που ακολουθεί. Μπορούμε να παρατηρήσουμε ότι υπάρχουν υπηρεσίες οι οποίες ανταποκρίνονται 100% στις απαιτήσεις που θέσαμε για διαθεσιμότητα αλλά υπάρχουν και υπηρεσίες οι οποίες ανταποκρίνονται με μικρότερο ποσοστό στον συγκεκριμένο όρο. Επίσης υπάρχουν υπηρεσίες για τις οποίες δεν υπάρχουν σχετικές πληροφορίες αφού οι

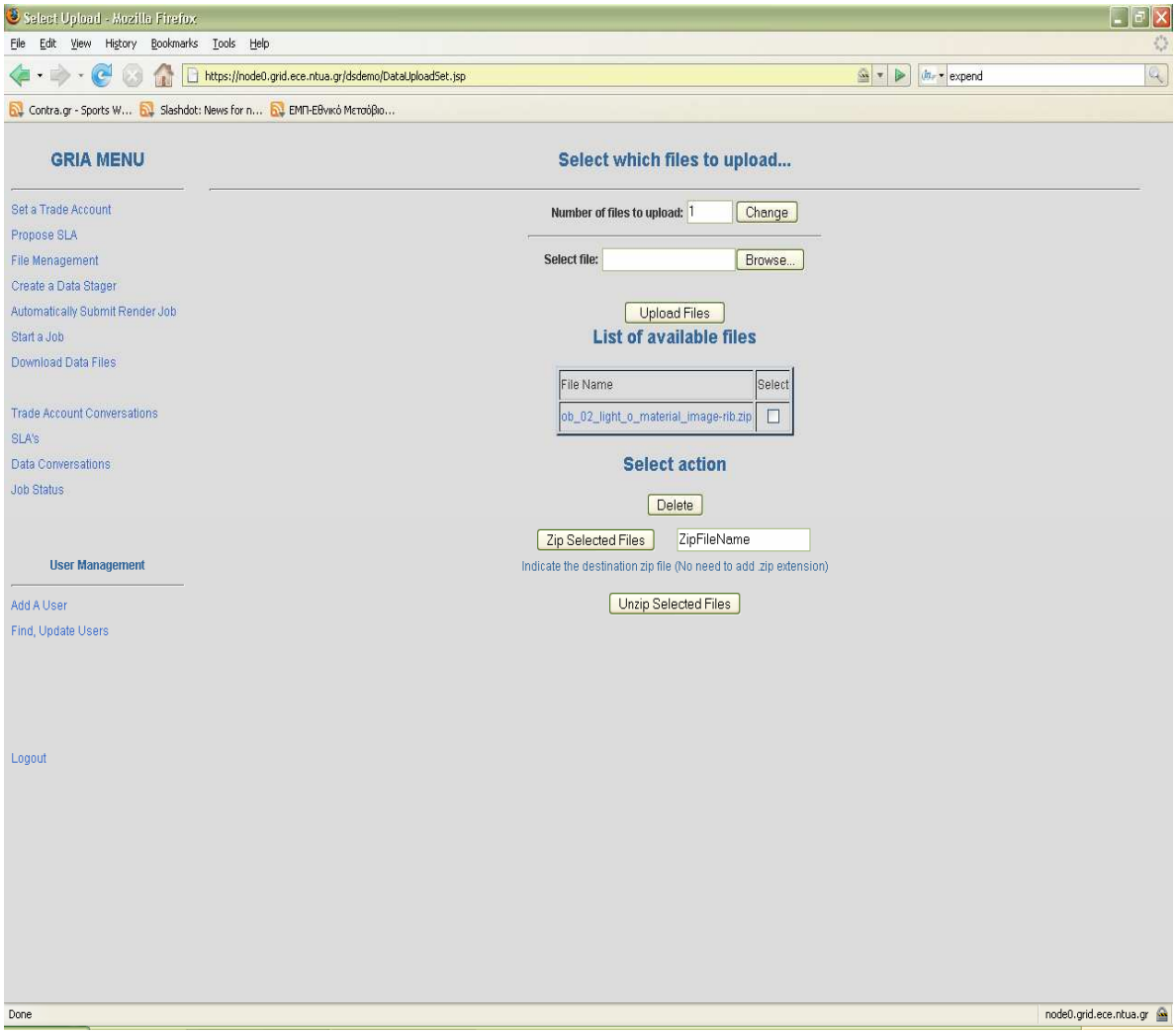
υπηρεσίες αυτές δεν έχουν επιλεγεί από κάποιον χρήστη για να αξιολογηθεί η επίδοση τους από πλευράς διαθεσιμότητας. Αξίζει να σημειώσουμε ότι τα στοιχεία που παίρνει ο μηχανισμός για την εξαγωγή των συγκεκριμένων συμπερασμάτων, παίρνει στοιχεία για την κάθε υπηρεσία από το σύνολο των χρηστών για τους οποίους είναι διαθέσιμες.



Εικόνα 19: Παρουσίαση στον χρήστη όλων των διαθέσιμων υπηρεσιών του συστήματος καθώς επίσης και την περιγραφή της κάθε υπηρεσίας και της κατάταξης της βάση του ιστορικού των παραβιάσεων το οποίο κρατείται για κάθε μία ξεχωριστά.

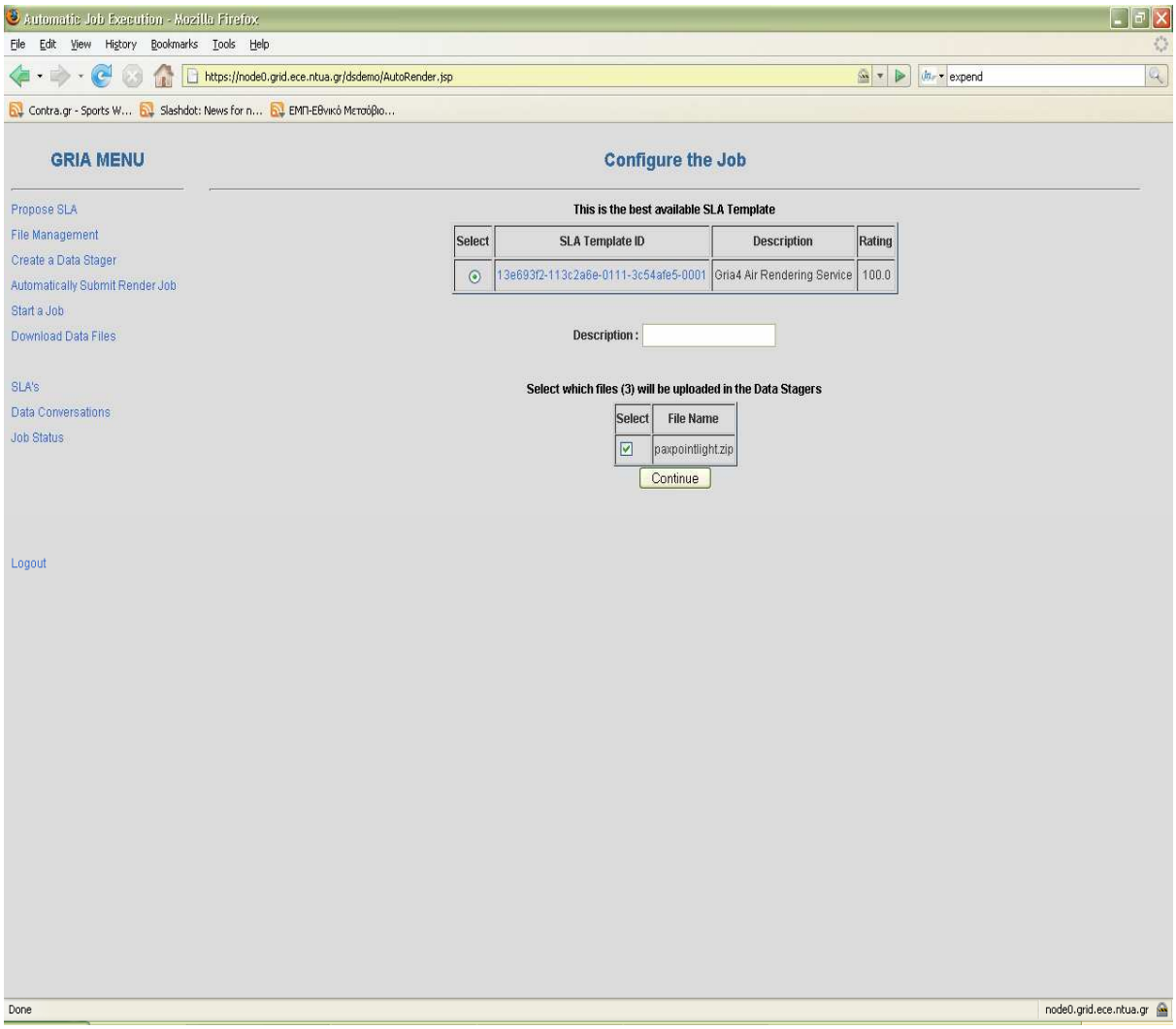
Ακολούθως ο χρήστης επιλέγει μια εκ των προτεινόμενων υπηρεσιών και υποβάλει την αίτηση του για την εκπλήρωση κάποιας συγκεκριμένης εργασίας για την οποία η υπηρεσία

μπορεί να ανταποκριθεί τόσο από πλευράς ικανότητας περάτωσης όσο και από πλευράς ποιότητας υπηρεσίας.



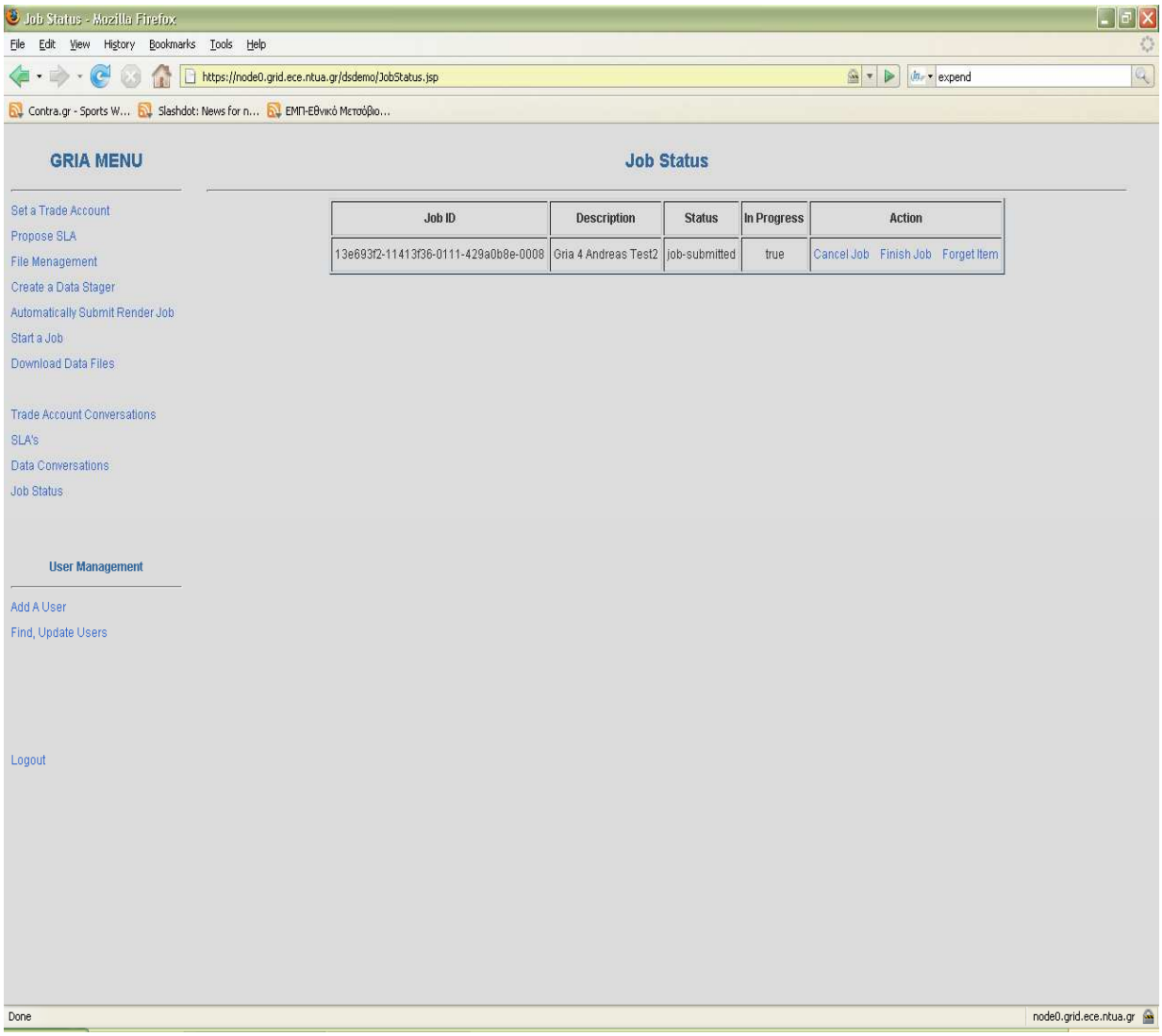
Εικόνα 20: Επιλογή συγκεκριμένης υπηρεσίας από τον χρήστη βάση των κριτηρίων που είχε θέσει ο ίδιος

Ακολούθως δίδεται το δικαίωμα στον χρήστη να περιγράψει και να παραμετροποιήση την εργασία που έχει υποβάλει αναλόγως από την λειτουργική ευχέρεια που του επιτρέπει η υπηρεσία.



Εικόνα 21: Ρύθμιση της υποβαλλόμενης εργασίας του χρήστη στο σύστημα από τον ίδιο.

Μετά την ολοκλήρωση της πιο πάνω διαδικασίας που έχουμε περιγράψει, η υποβληθήσα εργασία του πελάτη περνά στην λίστα των εγγραφόμενων εργασιών που βρίσκονται σε εξέλιξη και τα αποτελέσματα της εργασίας μετά την διεκπεραίωση της θα περάσουν στο ιστορικό της υπηρεσίας. Δηλαδή η υπηρεσία θα κριθεί από το αποτέλεσμα της παραδοθείσας εργασίας και θα καταγραφεί αναλόγως στο ιστορικό της μια επιτυχής ή ανεπιτυχής περάτωση εργασίας, η οποία θα είναι εμφανής στους χρήστης οι οποίοι θα κρίνουν την αξιοπιστία της.



Εικόνα 22: Κονσόλα διαχείρισης εργασιών που έχουν υποβληθεί στο σύστημα. Ο χρήστης μπορεί επίσης να διαχειριστεί τις εργασίες που έχει ο ίδιος υποβάλει στο σύστημα.

6 *Επίλογος*

Στο παρόν κεφάλαιο γίνεται αρχικά μία ανακεφαλαίωση των αποτελεσμάτων της διπλωματικής εργασίας και παρουσιάζονται ορισμένα συμπεράσματα που εξήχθησαν κατά την πορεία εκπόνησή της και με την ολοκλήρωσή της. Στη συνέχεια γίνεται αναφορά σε μελλοντικές επεκτάσεις που η παρούσα εργασία και το σύστημα που αναπτύχθηκε στα πλαίσια αυτής μπορούν να έχουν.

6.1 Σύνοψη και συμπεράσματα

6.1.1 Σύνοψη

Κατά τη διάρκεια αυτής της διπλωματικής υλοποιήσαμε έναν μηχανισμό για την καταγραφή του ιστορικού παραβιάσεων υπηρεσίας (QoS History), Grid εφαρμογών. Χρησιμοποιώντας τεχνολογίες αιχμής όπως SOAP-XML, Web Services υλοποιήσαμε μία υπηρεσία που μας παρέχει με πληροφορίες σχετικές με την παρεχόμενη ποιότητα. Η χρήση βάσης δεδομένων μας βοήθησε στην αποθήκευση των πληροφοριών για μεταγενέστερη ανάκληση και

επεξεργασία. Επίσης αναπτύχθηκε γραφικό περιβάλλον επικοινωνίας με το χρήστη είτε για την ρύθμιση των παραμέτρων των εκτελέσεων αλλά και για την προβολή των αποτελεσμάτων.

Οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση αυτή ήταν :

- Java ως γλώσσα προγραμματισμού για την υλοποίηση της υπηρεσίας
- Eclipse ως η πλατφόρμα ανάπτυξης του κώδικα
- Oracle Berkeley XML βάση δεδομένων για την αποθήκευση των αποτελεσμάτων
- Tomcat Server για την εγκατάσταση της υπηρεσίας
- Linux Fedora για την δοκιμαστική εκτέλεση της υπηρεσίας

Η διαδικασία ανάπτυξης της εφαρμογής περιλάμβανε τα ακόλουθα βήματα:

- Εξοικείωση με το πρόβλημα και τις σχετικές τεχνολογίες
- Βασικός Σχεδιασμός

  Το τμήμα αυτό περιλαμβάνει σχεδιασμό του της υπηρεσίας, τον προσδιορισμό των παραμέτρων εκτέλεσης και των ζητούμενων αποτελεσμάτων.
- Λεπτομερής σχεδιασμός

  Το τμήμα αυτό περιλαμβάνει το αναλυτικό σχεδιασμό των υπηρεσιών που θα χρησιμοποιηθούν, τον σχεδιασμό της βάσης δεδομένων και την επιλογή των απαιτούμενων μονάδων κώδικα για την ανάπτυξη της εφαρμογής.
- Συγγραφή κώδικα και ανάπτυξη της εφαρμογής (deployment)
- Εξαγωγή συμπερασμάτων- Δυνατές Επεκτάσεις

Μετά την ολοκλήρωση καθενός από τα βήματα σχεδίασης και υλοποίησης ακολούθησε έλεγχος και τεκμηρίωση.

6.1.2 Συμπεράσματα

Η ενασχόληση μας με την Grid τεχνολογία, κατά την ανάπτυξη αυτής της διπλωματικής εργασίας μας παρουσίασε τις μεγάλες δυνατότητες και τις προοπτικές αυτής της τεχνολογίας. Βρισκόμαστε ήδη στην ολοκλήρωση του πρώτου κύκλου ανάπτυξης των Grid τεχνολογιών, δηλαδή της λειτουργικής θεμελίωσης του Grid, και βαδίζουμε αυτή τη στιγμή στο επόμενο στάδιο, αυτό της αξιοποίησης και χρήσης του. Η υπηρεσιοστραφής μορφή αυτής της τεχνολογίας (service oriented), μας οδηγεί στην υιοθέτηση του όρου QoS (Ποιότητα Υπηρεσίας) και στη προσπάθεια ορισμού του με συγκεκριμένες παραμέτρους.

Η σημασία παρακολούθησης και επεξεργασίας αυτών των παραμέτρων είναι μεγάλη αφού μπορούν να μας δώσουν μια ένδειξη για την κατάσταση του πάροχου μιας Grid υπηρεσίας, τις δυνατότητές του, τους περιορισμούς του και συνολικά από όλα τα παραπάνω μια πρώτη εικόνα της ποιότητας που προσφέρει. Όταν υπεισέρχονται στην χρήση υπηρεσιών Grid, επιχειρήσεις και συμφωνίες μεταξύ αυτών, η έννοια της ποιότητας της υπηρεσίας παίρνει άλλες διαστάσεις.

Επιπρόσθετα με τα παραπάνω καταλήξαμε ότι υπηρεσίες διαδικτύου (web services) είναι ένα ισχυρό εργαλείο για περιπτώσεις χρήσης σχετικές με Grid, και γενικά απομακρυσμένης εκτέλεσης και παροχής πληροφοριών. Η προσεκτική υλοποίηση αυτών των υπηρεσιών μπορούν να μας προσφέρουν ένα ασφαλές και ευέλικτο μοντέλο επικοινωνίας.

Τέλος η χρήση προγραμματιστικών λύσεων ελεύθερης χρήσης (open source solutions) όπως Java programming, Linux OS, Oracle Berkeley XML DB, Tomcat Server, δεν παρουσίασε κανένα απολύτως πρόβλημα κατά την ανάπτυξη του μηχανισμού, παρέχοντας μας όλες τις απαραίτητες λειτουργίες που επιθυμούσαμε.

6.2 Μελλοντική Εργασία

Το πεδίο έρευνας σχετικά με την ποιότητα υπηρεσιών στην Grid τεχνολογία είναι ένα αρκετά μεγάλο κεφάλαιο στο οποίο υπάρχουν αρκετά ερευνητικά κεφάλαια ανοιχτά και χρίζουν περισσότερης διερεύνησης. Εμείς μελετήσαμε κυρίως την ποιότητα υπηρεσίας από πλευράς κυρίως διαθεσιμότητας, υπάρχουν όμως και άλλοι παράμετροι που σχετίζονται με την ποιότητα υπηρεσίας (τις οποίες έχουμε αναφέρει σε προηγούμενη ενότητα) οι οποίες όμως ακόμα δύσκολα μπορούν να αναλυθούν και να εξαχθούν σίγουρα και αξιόπιστα συμπεράσματα. Το σημαντικό όμως είναι ότι αφού το Grid σαν τεχνολογία έχει περάσει από το ακαδημαϊκό στο επιχειρηματικό μέρος, σίγουρα ένα ζήτημα το οποίο απαιτεί μεγάλη προσοχή και μελέτη είναι η ποιότητα της παρεχόμενης από αυτό υπηρεσίας.

Επίσης ένα άλλο θέμα το οποίο μπορεί να αναλυθεί πιο διεξοδικά είναι και το θέμα του αλγορίθμου με τον οποίο εμείς υπολογίζαμε την διαθεσιμότητα. Σίγουρα η ανάπτυξη ενός πιο ευφυούς αλγορίθμου ο οποίος θα έχει την ικανότητα να προβλέπει τον συγκεκριμένο όρο για μια υπηρεσία, είναι ένα θέμα ανοιχτό για μελέτη.

Γενικά η πτυχή της παρεχόμενης ποιότητας υπηρεσίας χρίζει μεγάλης προσοχής και μεγαλύτερης έρευνας για την παρουσίαση προς τον χρήστη ολοκληρωμένων και αξιόπιστων αποτελεσμάτων τα οποία θα βοηθήσουν τον ίδιο χρήστη αλλά και του φορείς παροχής υπηρεσιών στην βελτίωση των υπηρεσιών τους με σκοπό να γίνονται όλο και πιο ανταγωνιστικοί. Η δική μου άποψη είναι ότι αυτό το θέμα στην τελική είναι κέρδος για τον ίδιο τον χρήστη αφού θα έχει καλύτερες και αξιόπιστες υπηρεσίες.

7 *Βιβλιογραφία*

[FK98] The Grid ,Blueprint for a New Computing Infrastructure Edited by Ian Foster and Carl Kesselman,1998.

[F02] What is the Grid? A three point Checklist by Ian Foster, 2002.

[F01] The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Ian Foster
Mathematics and Computer Science Division, Argonne National Laboratory
Department of Computer Science, The University of Chicago, USA, 2001.

[FKL+99] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservation and coallocation.In Proceedings of the International Workshopon Quality of Service, pages 27–36, 1999.

[CFF+05] From Open Grid Services Infrastructure to WSResource Framework: Refactoring & Evolution,

Karl Czajkowski (Globus Alliance / USC Information Sciences Institute)

Don Ferguson (IBM), Ian Foster (Globus Alliance / Argonne National Laboratory), Jeff Frey (IBM), Steve Graham (IBM), Tom Maguire (IBM), David Snelling (Fujitsu Laboratories of Europe), Steve Tuecke (Globus Alliance / Argonne National Laboratory), May 2005.

[LG] The Business Grid: Providing Transactional Business Processes via Grid Services Frank Leymannand, Kai Güntzel.

[SSK04] Business Grid Computing Project Activities

Andreas Savva, Toshiyuki, Suzuki, Hiro Kishimoto.

(Manuscript received August 27, 2004)

[M03] Web Services Technologies Report for the JISC Technology Watch Service Diane McDonald, University of Strathclyde, May 2003.

[TSD+05] NEXTGrid P4.8.2 Modelling Resources, Kostas Tserpes, Fabrizio Silvestri, Konstantinos Dolkas, Dimos Kyriazis, Andreas Menychtas, September 2005.

[TKL+05] An Open Architecture for Providing QoS Information in Business Grids

Konstantinos Tserpes, Dimosthenis Kyriazis, Antonios Litke, Andreas Menychtas, and Theodora Varvarigou, 2005.

[F02] How does one really characterize Grid computing? By Richard F. Freund, CTO, GridIQ (http://www.Gridtoday.com/02/0819/100249.html

[CIM] Common Infromation Model (CIM), Infrastructure Specification ,DMTF (http://www.dmtf.org/standards/cim/cim_schema_v23/)

A Complete History of the Grid, Dr Rob Baxter Software Development Group Manager ,University of Edinburgh.

[SGM+03] A. Sahai and S. Graupner and V. Machiraju and A. Moorsel. (Specifying and Monitoring) Guarantees in Commercial Grids through SLA.

Proceedings of the 3rd IEEE/ACM CCGrid2003, 2003.

[CFK+02] K. Czajkowski and I. Foster and C. Kesselman and V. Sander and S. Tuecke SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing, 2002.

[KM03] K. Keahey and K. Motawi The Taming of the Grid: Virtual Application Service.Argonne National Laboratory Technical Memorandum No. 262, May 2003.

[RAK+04] Analysis and Provision of QoS for Distributed Grid Applications Rashid J. Al-Ali, Kaizar Amin, Gregor von Laszewski, Omer F. Rana, David W. Walker, Mihael Hategan, and Nestor Zaluzec, April 2004.

[M04] Προγραμματισμός σε Περιβάλλον Πολυπλέγματος, Διπλωματική Εργασία Μενύχτα Αντρέα, 2004.

[ACK+04] G. Alonso, F. Casati, H. Kuno, V. Machiraju. "Web Services: Concepts, Architectures and Applications". Springer-Verlag Berlin Heidelberg 2004

[BH04] G. Bochmann and A. Hafid. "Some Principles for Quality of Service Management", Technical report, Universite de Montreal, 1996.

[AAL+04] R. J. Al-Ali, K. Amin, G. von Laszewski, O. F. Rana, D. W. Walker, M. Hategan, N. J. Zaluzec: "Analysis and Provision of QoS for Distributed Grid Applications", pp. 163-182, Journal of Grid Computing, 2004

[PDD05] Padgett, J., K. Djemame, and P. Dew, "Grid-based SLA Management", Lecture Notes in Computer Science, pp. 1282-1291, 2005

[KDM+06] D. Kyriazis, K. Dolkas, A. Menychtas, T. Varvarigou, A new Workflow Mapping Mechanism for Grids, IST Mobile Summit, Special Session: Grid

and Mobility, June 2006, Myconos

[DMT05] Distributed Management Task Force, Inc., “Common Information Model (CIM) Infrastructure Specification”, DSP0004, Version 2.3, Final, October 4, 2005

[PDD] Padgett, J., K. Djemame, and P. Dew, “Grid-based SLA Management”. Lecture Notes in Computer Science, 2005(3483): p. 1282-1291.

[R05] Raicu. “A Performance Study of the Globus Toolkit® and Grid Services via DiPerF, an automated DIstributed PERformance testing Framework”, University of Chicago, Computer Science Department, MS Thesis, May 2005, Chicago, Illinois.

[GTV03] Gunter, B. Tierney, C. E. Tull, V. Virmani, “On-Demand Grid Application Tuning and Debugging with the NetLogger Activation Service”, 4th International Workshop on Grid Computing, Grid2003, Phoenix, Arizona, November 17th, 2003.

[TD03] G. Tsouloupas, M. Dikaiakos. “GridBench: A Tool for Benchmarking Grids”, 4th International Workshop on Grid Computing, Grid2003, Phoenix, Arizona, November 17th, 2003.

[WS1] WSRF, The Web Services Resource Framework (WSRF) v1.2, http://www.oasis-open.org/committees/download.php/17833/wsrf-1.2-os.zip

[WS2] Web Service Resource Lifetime (WS-ResourceLifetime) v1.2 Specification, http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf

[WS3] Web Service Resource Properties (WS-ResourceProperties) v1.2 Specification http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf

[WS4] Web Services Addressing (WS-Addressing),

http://www.w3.org/Submission/ws-addressing/

[WS5] Web Service Base Notification (WS-BaseNotification) v1.2 Specification, http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf

[WS6] Web Service Security (WS-Security) Core Specification v1.1, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[GGL03] L. Gymnopoulos, S. Dritsas, S. Gritzalis, C. Lambrinoudakis, "Grid Security Review", 2nd International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS 2003), pp. 100-111, September 2003, St. Petersburg, Russia, Springer-Verlag

[xvii] H. Ludwig, A. Keller, A. Dan, R. P. King, R. Franck, "Service Level Agreement Language Specification", http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

The Next Generation Grid (NEXTGrid), www.nextGrid.org

IBM About Grid Computing http://www-1.ibm.com/Grid

JAVA Online Help http://www.jguru.com

Apache Tomcat http://jakarta.apache.org/tomcat/

W3C –XML Techonologies http://www.w3.org

Global Grid Forum (GGF) http://www.Gridforum.org/

Παράρτημα

Client Side

Insert.java

```
package xmldb;

import java.io.*;

import com.sleepycat.dbxml.*;
import com.sleepycat.db.*;

public class Insert {
	private String message = null;
	private String myEnvPath = "/opt/dbxml/environment/dsdb";
	private String containerName = "myData.dbxml";
	// DBXML handles
	private Environment myEnv = null;

	private XmlManager mgr = null;
	private XmlContainer container = null;
	private XmlTransaction txn = null;
	// Utility function to clean up objects, exceptions or not,
	// containers and environments must be closed.

	public Insert() {
	}
```

```
public Insert(String myEnvPath) {
        this.myEnvPath = myEnvPath;
}

public String insertDB(String xmlString) {
        File path2myDbEnv = new File(myEnvPath);
        if (!path2myDbEnv.exists()) {
                path2myDbEnv.mkdirs();
        }
        try {
                // Open an environment
                openEnv();
                // Start a transaction
                txn = mgr.createTransaction();

                // Get an update context.
                XmlUpdateContext updateContext = mgr.createUpdateContext();

                // Put the saved document, asking DB XML to generate a name
                XmlDocumentConfig docConfig = new XmlDocumentConfig();
                docConfig.setGenerateName(true);

                container.putDocument(txn, "", xmlString, updateContext,
        docConfig);

         // Commit the writes. This causes the container write operations
         // to be saved to the container.
                txn.commit();
                message = "Insert completed successfully!";
        } catch (XmlException e) {
                try {
                        if (txn != null)
                                txn.abort();
```

```
                } catch (XmlException xe) {
                        xe.printStackTrace();
                }
                e.printStackTrace();
                message = e.getMessage();
        } catch (DatabaseException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } finally {
                closeEnv();
        }
        return message;
} // End insert

private void closeEnv() {
        if (container != null) {
                try {
                        container.close();
                } catch (XmlException xe) {
                        System.err.println("closeEnv: container: " +
                                                xe.toString());
                        xe.printStackTrace();
                }
        }

        if (mgr != null) {
                try {
                        mgr.close();
                } catch (XmlException xe) {
                        System.err.println("closeEnv: mgr: " +xe.toString());
                        xe.printStackTrace();
                }
        }
}
```

```
private void openEnv() throws DatabaseException {
        System.out.println("opening env");
        // Set up the environment.
        EnvironmentConfig myEnvConfig = new EnvironmentConfig();
        myEnvConfig.setAllowCreate(true);
        myEnvConfig.setInitializeCache(true);
        myEnvConfig.setInitializeLocking(true);
        myEnvConfig.setInitializeLogging(true);
        myEnvConfig.setTransactional(true);
        myEnvConfig.setRunRecovery(true);
        myEnvConfig.setThreaded(true);
        myEnvConfig.setErrorStream(System.err);

        try {
                // Open the environment
                myEnv = new Environment(new File(myEnvPath), // Env home
                                myEnvConfig);

        } catch (FileNotFoundException fnfe) {
                System.err.println("openEnv: " + fnfe.toString());
                System.exit(-1);
        }
        try {
                XmlManagerConfig managerConfig = new XmlManagerConfig();
                // Close the environment when the manager closes
                managerConfig.setAdoptEnvironment(true);
                managerConfig.setAllowAutoOpen(true);
                managerConfig.setAllowExternalAccess(true);
                mgr = new XmlManager(myEnv, managerConfig);

                // Open the container
                XmlContainerConfig containerConf = new
                                                        XmlContainerConfig();
```

```
                        containerConf.setTransactional(true);
                        containerConf.setAllowCreate(true);
                        // Declare the container type; that is, whether it is a
                        // node-storage or a whole doc container.If -w is specified
                        // at the command line, the container is set to wholedoc,
                        // otherwise node-storage is used.
                        container = mgr.openContainer(containerName,
                                                                        containerConf);
                } catch (XmlException xe) {
                        System.err.println("TxnGuide: " + xe.toString());
                        xe.printStackTrace();
                }
        }
}
```

UpdateDocument.java

```
package xmldb;

import java.io.*;

import com.sleepycat.db.DatabaseException;
import com.sleepycat.db.Environment;
import com.sleepycat.db.EnvironmentConfig;
import com.sleepycat.db.LockDetectMode;
import com.sleepycat.dbxml.*;

public class UpdateDocument {

        private String message = null;
```

```
private static String collection = "collection('myData.dbxml')";

private String myEnvPath = "/opt/dbxml/environment/dsdb";

private String containerName = "myData.dbxml";

// DBXML handles
private Environment myEnv = null;

private XmlManager mgr = null;

private XmlContainer container = null;

private XmlTransaction txn = null;

// Modifies an XML document that is stored in a DB XML container
private void doUpdateDocument(XmlManager mgr, XmlContainer container,
                String query, XmlQueryContext context, XmlTransaction txn)      {
        String fullQuery = collection + query;

        try {
                XmlUpdateContext updateContext = mgr.createUpdateContext();

                // query for all the documents that we want to update
                XmlResults results = mgr.query(txn, fullQuery, context,
                                                null);

                XmlValue value = results.next();
                while (value != null) {

                        XmlDocument document = value.asDocument();

                // Retrieve the entire document as a single String object
                        String docString = document.getContentAsString();
```

```
                    // This next method just modifies the document string
                    // in a small way.
                    String newDocString = getNewDocument(mgr, document,
                                                        context,docString);

                // Set the document's content to be the new document string
                    document.setContent(newDocString);

                    XmlDocumentConfig docConfig = new
                                                        XmlDocumentConfig();
                    docConfig.setGenerateName(true);
                    container.putDocument(txn, "", newDocString,
                                                updateContext, docConfig);

                    container.deleteDocument(txn, document,
                                                            updateContext);
                    value = results.next();

                }
            } catch (XmlException e) {
            }
        }

        private String getNewDocument(XmlManager dbXML, XmlDocument document,
                XmlQueryContext context, String docString) {

            String retVal = new String();
            try {
                // Get the substring that we want to replace

                String event = getValue(dbXML, document,
                            "string(/QoSdb/QoEViolation)", context);
```

```
                // Find the substring in the original document string
                int pos = docString.indexOf(event);

                if (pos == -1) {
                        // error handler

                } else {

                        // Now we perform the update of the feild Event

                        String newEvent = "Violation";

                        // String newVal = Integer.toString(newInventory);

                        int eventLength = event.length();
                        int endPos = pos + eventLength;

                        StringBuffer strbuff = new StringBuffer(document
                                        .getContentAsString());

                        // Perform the replace
                        strbuff.replace(pos, endPos, newEvent);

                        retVal = strbuff.toString();
                }
        } catch (XmlException x) {
        }

        return retVal;
}

private String getValue(XmlManager mgr, XmlDocument theDocument,
                String query, XmlQueryContext context) {
```

```
        String v = null;
        XmlValue value = null;
        // Perform the query
        // The document provides the context for the query
        try {
                XmlQueryExpression queryExpr = mgr.prepare(query, context);
                XmlResults result = queryExpr.execute(new
                                        XmlValue(theDocument),context, null);

                // We require a result set size of 1.
                if (result.size() != 1) {
                        // Error handler
                }

// Get the value. If we allowed the result set to be larger than
// size 1,
// we would have to loop through the results, processing each as is
// required by our application.

                value = result.next();
                v = value.asString();



        } catch (XmlException xe) {
        }

        return v;

}

public String Update(String slaepr) {
        try {
                // Open an environment
                openEnv();
```

```
        // Start a transaction
        txn = mgr.createTransaction();

        XmlQueryContext context = mgr.createQueryContext();

// we want to update the feild event on a specific xml document
// which sattisfies
// our criteria which are Epr, Starting Date, End Date and party


        String query = "/QoSdb/SLAEpr[text()=\"" + slaepr + "\"]";
        doUpdateDocument(mgr, container, query, context, txn);

// Commit the writes. This causes the container write operations
// to be saved to the container.
        txn.commit();
        message = "Update completed successfully!";

} catch (Exception e) {
// In the event of an error, we abort the operation
// The database is left in the same state as it was in before
// we started this operation.
        try {
                if (txn != null) {
                        txn.abort();
                }
        } catch (XmlException exc) {
                exc.printStackTrace();
                message = exc.getMessage();
        }

} finally {
        closeEnv();
}
```

```
        return message;
    } // End main

    private void closeEnv() {
        if (container != null) {
            try {
                container.close();
            } catch (XmlException xe) {
                System.err.println("closeEnv: container: " +
                                                    xe.toString());
                xe.printStackTrace();
            }
        }

        if (mgr != null) {
            try {
                mgr.close();
            } catch (XmlException xe) {
                System.err.println("closeEnv: mgr: " +
                                                    xe.toString());
                xe.printStackTrace();
            }
        }
    }

    private void openEnv() throws DatabaseException {
        System.out.println("opening env");

        EnvironmentConfig myEnvConfig = new EnvironmentConfig();
        myEnvConfig.setAllowCreate(true);
        myEnvConfig.setInitializeCache(true);
        myEnvConfig.setInitializeLocking(true);
        myEnvConfig.setInitializeLogging(true);
        myEnvConfig.setTransactional(true);
```

```
myEnvConfig.setRunRecovery(true);
myEnvConfig.setThreaded(true);
myEnvConfig.setErrorStream(System.err);
try {
        // Open the environment
        myEnv = new Environment(new File(myEnvPath), // Env home
                        myEnvConfig);

} catch (FileNotFoundException fnfe) {
        System.err.println("openEnv: " + fnfe.toString());
        System.exit(-1);
}
try {
        XmlManagerConfig managerConfig = new XmlManagerConfig();
        // Close the environment when the manager closes
        //managerConfig.setAdoptEnvironment(true);
        managerConfig.setAllowAutoOpen(true);
        managerConfig.setAllowExternalAccess(true);
        mgr = new XmlManager(myEnv, managerConfig);

        // Open the container
        XmlContainerConfig containerConf = new
                                                XmlContainerConfig();
        containerConf.setTransactional(true);
        containerConf.setAllowCreate(true);

// Declare the container type; that is, whether it is a
// node-storage or a whole doc container. If -w is specified
// at the command line, the container is set to wholedoc,
// otherwise node-storage is used.

        container = mgr.openContainer(containerName,
                                                containerConf);
} catch (XmlException xe) {
```

```
                System.err.println("TxnGuide: " + xe.toString());
                xe.printStackTrace();
            }
        }
}
```

Rate.java

```
package xmldb;

import java.io.File;
import java.io.FileNotFoundException;

import com.sleepycat.db.DatabaseException;
import com.sleepycat.db.Environment;
import com.sleepycat.db.EnvironmentConfig;
import com.sleepycat.db.LockDetectMode;
import com.sleepycat.dbxml.XmlContainer;
import com.sleepycat.dbxml.XmlContainerConfig;
import com.sleepycat.dbxml.XmlDocument;
import com.sleepycat.dbxml.XmlException;
import com.sleepycat.dbxml.XmlManager;
import com.sleepycat.dbxml.XmlManagerConfig;
import com.sleepycat.dbxml.XmlQueryContext;
import com.sleepycat.dbxml.XmlQueryExpression;
import com.sleepycat.dbxml.XmlResults;
import com.sleepycat.dbxml.XmlTransaction;
import com.sleepycat.dbxml.XmlValue;

public class Rate {
```

```
private String message = null;

private float vi = 0;

private static String collection = "collection('myData.dbxml')";

private String myEnvPath = "/opt/dbxml/environment/dsdb";

private String containerName = "myData.dbxml";

// DBXML handles
private Environment myEnv = null;

private XmlManager mgr = null;

private XmlContainer container = null;

private XmlTransaction txn = null;

private float doQuery(XmlManager mgr, String query, XmlContainer
        container, XmlQueryContext context, XmlTransaction txn)
{
        // Perform a single query against the referenced container.
        // No context is used for this query.
        String fullQuery = collection + query;

        float rs = 0;
        // Perform the query
        String s = "0";

        try {
                context = mgr.createQueryContext();

                XmlResults results = mgr.query(fullQuery, context, null);
```

```
        // Iterate over the results of the query using an XmlValue object

                XmlValue value;
                while ((value = results.next()) != null) {
                        XmlDocument document = value.asDocument();



                // Retrieve the entire document as a single String object
                        String docString = document.getContentAsString();

                        // Now we compare the vlaue of violation to get
                        // the rank of the service

                        s = getNewDocument(mgr, document, context,docString);
                        if (s.compareTo("Violation") == 0) {
                                vi++;
                        }
                }

                if (results.size() == 0) {
                        rs = -1;
                } else {
                        rs = (100 - (vi / results.size()) * 100);
                }
        } catch (XmlException e) {
                e.printStackTrace();
        }
        return rs;
}

private String getNewDocument(XmlManager dbXML, XmlDocument document,
                XmlQueryContext context, String docString) {

        // Get the substring that we want to check if it is Violation
```

```
        String event = getValue(dbXML, document,

"string(/QoSdb/QoEViolation)",context);

        return event;
}

private static String getValue(XmlManager mgr, XmlDocument theDocument,
                String query, XmlQueryContext context) {

        String v = null;
        XmlValue value = null;
        // Perform the query
        // The document provides the context for the query
        try {
                XmlQueryExpression queryExpr = mgr.prepare(query, context);
                XmlResults result = queryExpr.execute(new
                                        XmlValue(theDocument),context, null);

                // We require a result set size of 1.
                if (result.size() != 1) {
                        // Error handler
                }

// Get the value. If we allowed the result set to be larger than
// size 1,
// we would have to loop through the results, processing each as is
// required by our application.

                value = result.next();
                v = value.asString();
```

```
        } catch (XmlException xe) {
                xe.printStackTrace();
        }

        return v;

}

public String Rating(String slaTempepr) {

        float myresult = 0;
        String finalresult = "0";

        try {
                // Open an environment
                openEnv();
                // Start a transaction
                txn = mgr.createTransaction();

                XmlQueryContext context = mgr.createQueryContext();

        // we want to update the feild event on a specific xml document
        // which sattisfies
        // our criteria which is SLA Template epr

                String query = "/QoSdb/SLATemplateEpr[text()=\"" +
                                        slaTempepr + "\"]";
                myresult = doQuery(mgr, query, container, context, txn);

                finalresult = Float.toString(myresult);
        // Commit the writes. This causes the container write operations
        // to be saved to the container.

                txn.commit();
```

```
                message = finalresult;

        } catch (Exception e) {

        // In the event of an error, we abort the operation
        // The database is left in the same state as it was in before
        // we started this operation.

                try {
                        if (txn != null) {
                                txn.abort();
                        }
                } catch (XmlException exc) {
                        exc.printStackTrace();
                        message = exc.getMessage();
                }

        } finally {
                closeEnv();
        }
        return message;
} // End main

private void closeEnv() {
        if (container != null) {
                try {
                        container.close();
                } catch (XmlException xe) {
                        System.err.println("closeEnv: container: " +
                                                                xe.toString());
                        xe.printStackTrace();
                }
        }
```

```
        if (mgr != null) {
                try {
                        mgr.close();
                } catch (XmlException xe) {
                        System.err.println("closeEnv: mgr: " +xe.toString());
                        xe.printStackTrace();
                }
        }
}

private void openEnv() throws DatabaseException {
        System.out.println("opening env");

        // Set up the environment.
        EnvironmentConfig myEnvConfig = new EnvironmentConfig();
        myEnvConfig.setAllowCreate(true);
        myEnvConfig.setInitializeCache(true);
        myEnvConfig.setInitializeLocking(true);
        myEnvConfig.setInitializeLogging(true);
        myEnvConfig.setTransactional(true);
        myEnvConfig.setRunRecovery(true);
        myEnvConfig.setThreaded(true);
        myEnvConfig.setErrorStream(System.err);
        try {
                // Open the environment
                myEnv = new Environment(new File(myEnvPath),
                                myEnvConfig);

        } catch (FileNotFoundException fnfe) {
                System.err.println("openEnv: " + fnfe.toString());
                System.exit(-1);
        }
        try {
                XmlManagerConfig managerConfig = new XmlManagerConfig();
```

```
                // Close the environment when the manager closes
                // managerConfig.setAdoptEnvironment(true);
                managerConfig.setAllowAutoOpen(true);
                managerConfig.setAllowExternalAccess(true);
                mgr = new XmlManager(myEnv, managerConfig);

                // Open the container
                XmlContainerConfig containerConf = new
                                                XmlContainerConfig();
                containerConf.setTransactional(true);


            // Declare the container type; that is, whether it is a
            // node-storage or a whole doc container. If -w is specified
            // at the command line, the container is set to wholedoc,
            // otherwise node-storage is used.

                container = mgr.openContainer(containerName,
                                                    containerConf);
            } catch (XmlException xe) {
                System.err.println("TxnGuide: " + xe.toString());
                xe.printStackTrace();
            }
        }
}
```

DBService.java

```
package xmldb;
public class DBService {

	public synchronized String Ins(String xmlString) {
		Insert insertmethod = new Insert();
		return insertmethod.insertDB(xmlString);
	}

	public synchronized String Rate(String slaTempepr) {
		Rate ra = new Rate();
		String fe = ra.Rating(slaTempepr);
		return fe;
	}

	public synchronized String Udate(String up) {
		UpdateDocument ud = new UpdateDocument();
		return ud.Update(up);
	}
}
```

Services.xml

```
<service name="DBService" scope="application">
	<description>DB Service</description>
	<messageReceivers>
		<messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
			class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"
```

```
/>
                <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
                        class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
        </messageReceivers>
        <parameter name="ServiceClass" locked="false">
                xmldb.DBService
        </parameter></service>
```

build.xml

```
<project basedir="." default="generate.service">

        <!--<property name="service.name" value="DBService" /> -->

        <property name="dest.dir" value="build" />
        <!--<property name="dest.dir.classes"
                value="${dest.dir}/${service.name}" />-->
        <property name="dest.dir.classes" value="${dest.dir}/classes" />
        <property name="dest.dir.lib" value="${dest.dir}/lib" />
        <property name="axis2.home" value="/opt/axis2" />
        <property name="repository.path"
                value="${axis2.home}/repository/services"/>

        <path id="build.class.path">
                <fileset dir="${axis2.home}/lib">
                        <include name="*.jar" />
                </fileset>
        </path>
        <path id="client.class.path">
                <fileset dir="${axis2.home}/lib">
                        <include name="*.jar" />
```

```
        </fileset>
        <fileset dir="${dest.dir.lib}">
                <include name="*.jar" />
        </fileset>
</path>
<target name="clean">
        <delete dir="${dest.dir}" />
</target>

<target name="prepare">
        <mkdir dir="${dest.dir}" />
        <mkdir dir="${dest.dir.lib}" />
        <mkdir dir="${dest.dir.classes}" />
        <mkdir dir="${dest.dir.classes}/META-INF" />
</target>
<target name="generate.service" depends="clean,prepare">
        <copy file="src/META-INF/services.xml"
tofile="${dest.dir.classes}/META-INF/services.xml" overwrite="true" />

        <javac srcdir="src" destdir="${dest.dir.classes}" includes="xmldb/**">
                <classpath refid="build.class.path" />
        </javac>
        <jar basedir="${dest.dir.classes}" destfile="${dest.dir}/DBService.aar" />
</target>
<target name="rpc.client" depends="clean,prepare">
        <antcall target="rpc.client.compile" />
        <antcall target="rpc.client.jar" />
        <antcall target="rpc.client.run" >
                <param name="uri" value="${uri}" />
        </antcall>
</target>
<target name="rpc.client.compile">
        <javac srcdir="src" destdir="${dest.dir.classes}" includes="xmldb/**">
                <classpath refid="build.class.path" />
```

```
            </javac>
        </target>
        <target name="rpc.client.jar">
            <jar basedir="${dest.dir.classes}" destfile="${dest.dir.lib}/rpc-client.jar"
                includes="xmldb/**" />
        </target>
        <target name="rpc.client.run">
            <java classname="xmldb.InsertClient">
                <classpath refid="client.class.path" />
                <arg value="${uri}" />
            </java>
        </target>
</project>
```

Server Side

InsertSLA.java

```
package qosanalyzer;

import java.io.*;
import com.sleepycat.dbxml.*;
import com.sleepycat.db.*;

public class InsertSLA {

	private String message = null;

	private String myEnvPath = "/opt/dbxml/environment/spdb";

	private String containerName = "spData3.dbxml";
	// DBXML handles
	private Environment myEnv = null;
	private XmlManager mgr = null;
	private XmlContainer container = null;

	private XmlTransaction txn = null;
	public InsertSLA (){

	}

	public InsertSLA(String myEnvPath) {
		this.myEnvPath = myEnvPath;
	}
```

```
public String addSLA(String xmlString) {
        File path2myDbEnv = new File(myEnvPath);

        if (!path2myDbEnv.exists()) {
                path2myDbEnv.mkdirs();
        }

        try {
                // Open an environment
                openEnv();
                // Start a transaction
                txn = mgr.createTransaction();

                // Get an update context.
                XmlUpdateContext updateContext = mgr.createUpdateContext();

                // Put the saved document, asking DB XML to generate a name
                XmlDocumentConfig docConfig = new XmlDocumentConfig();
                docConfig.setGenerateName(true);

                container.putDocument(txn, "", xmlString, updateContext,
                                                                docConfig);

        // Commit the writes. This causes the container write operations
        // to be saved to the container.

                txn.commit();
                message = "Insert completed successfully!";
        } catch (XmlException e) {
                try {
                        if (txn != null)
                                txn.abort();
                } catch (XmlException xe) {
                        xe.printStackTrace();
```

```
                }
                e.printStackTrace();
                message = e.getMessage();
        } catch (DatabaseException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } finally {
                closeEnv();
        }
        return message;
} // End insert

private void closeEnv() {
        if (container != null) {
                try {
                        container.close();

                } catch (XmlException xe) {
                        System.err.println("closeEnv: container: " +
                        xe.toString());
                        xe.printStackTrace();
                }

        }

        if (mgr != null) {
                try {
                        mgr.close();
                } catch (XmlException xe) {
                        System.err.println("closeEnv: mgr: " +
                        xe.toString());
                        xe.printStackTrace();
                }
        }
```

```
}

private void openEnv() throws DatabaseException {
        System.out.println("opening env");

        // Set up the environment.
        EnvironmentConfig myEnvConfig = new EnvironmentConfig();
        myEnvConfig.setAllowCreate(true);
        myEnvConfig.setInitializeCache(true);
        myEnvConfig.setInitializeLocking(true);
        myEnvConfig.setInitializeLogging(true);
        myEnvConfig.setTransactional(true);
        myEnvConfig.setRunRecovery(true);
        myEnvConfig.setThreaded(true);
        myEnvConfig.setErrorStream(System.err);
        try {
                // Open the environment
                myEnv = new Environment(new File(myEnvPath), // Env home
                                myEnvConfig);
        } catch (FileNotFoundException fnfe) {
                System.err.println("openEnv: " + fnfe.toString());
                System.exit(-1);
        }
        try {
                XmlManagerConfig managerConfig = new XmlManagerConfig();
                // Close the environment when the manager closes
                managerConfig.setAdoptEnvironment(true);
                managerConfig.setAllowAutoOpen(true);
                managerConfig.setAllowExternalAccess(true);
                mgr = new XmlManager(myEnv, managerConfig);
                // Open the container
                XmlContainerConfig containerConf = new
                XmlContainerConfig();
                containerConf.setTransactional(true);
```

```
                containerConf.setAllowCreate(true);
            // Declare the container type; that is, whether it is a
            // node-storage or a whole doc container. If -w is specified
            // at the command line, the container is set to wholedoc,
                // otherwise node-storage is used.
                container = mgr.openContainer(containerName,
                        containerConf);
            } catch (XmlException xe) {
                System.err.println("TxnGuide: " + xe.toString());
                xe.printStackTrace();
            }
        }

}
```

RecommendSLA.java

```
package qosanalyzer;

import java.io.File;
import java.io.FileNotFoundException;

import com.sleepycat.db.DatabaseException;
import com.sleepycat.db.Environment;
import com.sleepycat.db.EnvironmentConfig;
import com.sleepycat.dbxml.XmlContainer;
import com.sleepycat.dbxml.XmlContainerConfig;
import com.sleepycat.dbxml.XmlDocument;
import com.sleepycat.dbxml.XmlException;
import com.sleepycat.dbxml.XmlManager;
import com.sleepycat.dbxml.XmlManagerConfig;
```

```
import com.sleepycat.dbxml.XmlQueryContext;
import com.sleepycat.dbxml.XmlQueryExpression;
import com.sleepycat.dbxml.XmlResults;
import com.sleepycat.dbxml.XmlTransaction;
import com.sleepycat.dbxml.XmlValue;

public class ReccomendSLA {

	private String message = null;

	private float vi = 0;

	private static String collection = "collection('spData3.dbxml')";

	private String myEnvPath = "/opt/dbxml/environment/spdb";

	private String containerName = "spData3.dbxml";

	// DBXML handles
	private Environment myEnv = null;

	private XmlManager mgr = null;

	private XmlContainer container = null;

	private XmlTransaction txn = null;

	private float doQuery(XmlManager mgr, String query, XmlContainer
	container,XmlQueryContext context, XmlTransaction txn)
	{
		// Perform a single query against the referenced container.
		// No context is used for this query.

		String fullQuery = collection + query;
```

```
float rs = 0;
// Perform the query
String s = "0";

try {
        context = mgr.createQueryContext();
        XmlResults results = mgr.query(fullQuery, context, null);

// Iterate over the results of the query using an XmlValue object

        XmlValue value;
        while ((value = results.next()) != null) {
                XmlDocument document = value.asDocument();

        // Retrieve the entire document as a single String object

                String docString = document.getContentAsString();



                // Now we compare the vlaue of violation to get
                // the rank of the service

                s = getNewDocument(mgr, document, context,
                                                                docString);
                if (s.compareTo("FAILED") == 0) {
                        vi++;
                }
        }

        if (results.size() == 0) {
                rs = -1;
        } else {
                rs = (100 - (vi / results.size()) * 100);
```

```
                }
        } catch (XmlException e) {
                e.printStackTrace();
        }
        return rs;
}

private String getNewDocument(XmlManager dbXML, XmlDocument document,
                XmlQueryContext context, String docString) {

        String event;
        // Get the substring that we want to check if it is Violation

        event = getValue(dbXML, document,
                                "string(/QoE_Information/process/state)",context);
        return event;
}

private static String getValue(XmlManager mgr, XmlDocument theDocument,
                String query, XmlQueryContext context) {

        String v = null;
        XmlValue value = null;
        // Perform the query
        // The document provides the context for the query
        try {
                XmlQueryExpression queryExpr = mgr.prepare(query, context);
                XmlResults result = queryExpr.execute(new
                                        XmlValue(theDocument),context, null);

                // We require a result set size of 1.
                if (result.size() != 1) {
                        // Error handler
                }
```

```
// Get the value. If we allowed the result set to be larger than
// size 1,
// we would have to loop through the results, processing each as is
// required by our application.

                value = result.next();
                v = value.asString();

        } catch (XmlException xe) {
                xe.printStackTrace();
        }

        return v;

}

public String Recommendation(String slatemplatetype) {

        float myresult = 0;
        String finalresult = "0";

        try {
                // Open an environment
                openEnv();
                // Start a transaction
                txn = mgr.createTransaction();

                XmlQueryContext context = mgr.createQueryContext();

        // we want to update the feild event on a specific xml document
        // which sattisfies
        // our criteria which is SLA Template epr
```

```
                String query =
                        "/QoE_Information/process/slaepr/slatemplatetype[text()=\""
                + slatemplatetype + "\"]";
                myresult = doQuery(mgr, query, container, context, txn);

                finalresult = Float.toString(myresult);
        // Commit the writes. This causes the container write operations
        // to be saved to the container.

                txn.commit();
                message = finalresult;

        } catch (Exception e) {
        // In the event of an error, we abort the operation
        // The database is left in the same state as it was in before
        // we started this operation.
                try {
                        if (txn != null) {
                                txn.abort();
                        }
                } catch (XmlException exc) {
                        exc.printStackTrace();
                        message = exc.getMessage();
                }

        } finally {
                closeEnv();
        }
        return message;
} // End main

private void closeEnv() {
        if (container != null) {
                try {
```

```
                container.close();
            } catch (XmlException xe) {
                System.err.println("closeEnv: container: " +
                xe.toString());
                xe.printStackTrace();
            }
        }

        if (mgr != null) {
            try {
                mgr.close();
            } catch (XmlException xe) {
                System.err.println("closeEnv: mgr: " +
                xe.toString());
                xe.printStackTrace();
            }
        }
    }

    private void openEnv() throws DatabaseException {
        System.out.println("opening env");

        // Set up the environment.
        EnvironmentConfig myEnvConfig = new EnvironmentConfig();
        myEnvConfig.setAllowCreate(true);
        myEnvConfig.setInitializeCache(true);
        myEnvConfig.setInitializeLocking(true);
        myEnvConfig.setInitializeLogging(true);
        myEnvConfig.setTransactional(true);
        myEnvConfig.setRunRecovery(true);
        myEnvConfig.setThreaded(true);
        myEnvConfig.setErrorStream(System.err);
        try {
            // Open the environment
```

```
                myEnv = new Environment(new File(myEnvPath), // Env home
                                myEnvConfig);

            } catch (FileNotFoundException fnfe) {
                System.err.println("openEnv: " + fnfe.toString());
                System.exit(-1);
            }
            try {
                XmlManagerConfig managerConfig = new XmlManagerConfig();
                // Close the environment when the manager closes
                // managerConfig.setAdoptEnvironment(true);
                managerConfig.setAllowAutoOpen(true);
                managerConfig.setAllowExternalAccess(true);
                mgr = new XmlManager(myEnv, managerConfig);

                // Open the container
                XmlContainerConfig containerConf = new
                                                XmlContainerConfig();
                containerConf.setTransactional(true);

            // Declare the container type; that is, whether it is a
            // node-storage or a whole doc container. If -w is specified
            // at the command line, the container is set to wholedoc,
            // otherwise node-storage is used.
                container = mgr.openContainer(containerName,
                containerConf);
            } catch (XmlException xe) {
                System.err.println("TxnGuide: " + xe.toString());
                xe.printStackTrace();
            }
        }
}
```

HistoryServices.java

```
package qosanalyzer;

import qosanalyzer.InsertSLA;
import qosanalyzer.UpdateSLA;
import qosanalyzer.RecomendSLA;

public class HistoryServices {

	public synchronized String Add(String xmlString) {
		InsertSLA addmethod = new InsertSLA();
		return addmethod.addSLA(xmlString);
	}

	public synchronized String Recomend(String slaTempepr) {
		RecomendSLA re = new RecomendSLA();
		String fe = re.Recomendation(slaTempepr);
		return fe;
	}

	public synchronized String Update(String up, String type) {
		UpdateSLA updatemethod = new UpdateSLA();
		return updatemethod.Update(up,type);
	}

	public synchronized String Vtype(String slaepr) {
		ViolationType vtypemethod = new ViolationType();
		return vtypemethod.Vtype(slaepr);
	}

}
```

build.xml

```
<project basedir="." default="generate.service">

	<!--<property name="service.name" value="HistoryServices" /> -->

	<property name="dest.dir" value="build" />
	<!--<property name="dest.dir.classes"	value="${dest.dir}/${service.name}" />-
->
	<property name="dest.dir.classes" value="${dest.dir}/classes" />
	<property name="dest.dir.lib" value="${dest.dir}/lib" />
	<property name="axis2.home" value="/opt/axis2" />
	<property name="repository.path"
	value="${axis2.home}/repository/services"/>

	<path id="build.class.path">
		<fileset dir="${axis2.home}/lib">
			<include name="*.jar" />
		</fileset>
	</path>

	<path id="client.class.path">
		<fileset dir="${axis2.home}/lib">
			<include name="*.jar" />
		</fileset>
		<fileset dir="${dest.dir.lib}">
			<include name="*.jar" />
		</fileset>

	</path>
	<target name="clean">
		<delete dir="${dest.dir}" />
	</target>
```

```
<target name="prepare">
        <mkdir dir="${dest.dir}" />
        <mkdir dir="${dest.dir.lib}" />
        <mkdir dir="${dest.dir.classes}" />
        <mkdir dir="${dest.dir.classes}/META-INF" />
</target>

<target name="generate.service" depends="clean,prepare">

        <copy file="src/META-INF/services.xml"
                tofile="${dest.dir.classes}/META-INF/services.xml" overwrite="true"
        />

        <javac srcdir="src" destdir="${dest.dir.classes}" includes="qosanalyzer/**">
                <classpath refid="build.class.path" />
        </javac>

        <jar basedir="${dest.dir.classes}" destfile="${dest.dir}/HistoryServices.aar"
/>

</target>

<target name="rpc.client" depends="clean,prepare">
        <antcall target="rpc.client.compile" />
        <antcall target="rpc.client.jar" />
        <antcall target="rpc.client.run" >
                <param name="uri" value="${uri}" />
        </antcall>
</target>

<target name="rpc.client.compile">
        <javac srcdir="src" destdir="${dest.dir.classes}" includes="qosanalyzer/**">
                <classpath refid="build.class.path" />
```

```
            </javac>
        </target>

        <target name="rpc.client.jar">
            <jar basedir="${dest.dir.classes}" destfile="${dest.dir.lib}/rpc-client.jar"
                        includes="qosanalyzer/**" />
        </target>

        <target name="rpc.client.run">
            <java classname="qosanalyzer.InsertSLAClient">
                <classpath refid="client.class.path" />
                <arg value="${uri}" />
            </java>
        </target>
</project>
```