Διπλωματική Εργασία :

Μία αρχιτεκτονική για **Collectors** με την χρήση γεννήτριας κώδικα

Πατεράκης Παναγιώτης

Επιβλέπων Καθηγητής: Σαμολαδάς Βασίλης
Μέλος Επιτροπής: Πετράκης Ευριπίδης
Μέλος Επιτροπής: Χριστοδουλάκης Σταύρος
Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών
Πολυτεχνείο Κρήτης
Χανιά 2005

Επιβλέπων Καθηγητής: Σαμολαδάς Βασίλης

Μέλος Επιτροπής: Πετράκης Ευριπίδης

Μέλος Επιτροπής: Χριστοδουλάκης Σταύρος

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Περίληψη

Σκοπός της εργασίας αυτής είναι η κατασκευή ενός Framework για την επεξεργασία εξωτερικών event πάνω σε ένα application server.Πέρα από την μοντελοποίηση και σχεδίαση του συστήματος , έχει υλοποιηθεί και ένα πρωτότυπο σύστημα στον JBoss application server. Η υλοποίηση κάνει χρήση πολύπλοκων τεχνικών γέννησης κώδικα παρέχοντας με αυτόν τον τρόπο την δυνατότητα ανάπτυξης πολύπλοκων εφαρμογών σε μικρό χρονικό διάστημα.

Στην υλοποίηση του συστήματος παρέχουμε Event listeners για κάποια από το βασικά πρωτόκολλα internet. Τα πρωτόκολλα αυτά είναι : Telnet Client,FTP Client,SSH Client,HTTP server ενώ δίνουμε την δυνατότητα στους χρήστες να επιλέγουν και τον τρόπο επικοινωνίας των Event Listener που γεννιούνται δίνοντας τους την δυνατότητα να επιλέγουν μεταξύ JMS Topic και Remote JMX Notifications.

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή

1.1.1 Εισαγωγή

Τα τελευταία χρόνια γίνεται μεγάλη προσπάθεια για να αυτοματοποιηθούν πολύπλοκες διαδικασίες στην ανάπτυξη εφαρμογών. Διαδικάσιες όπως η ανάπτυξη τετριμμένων πρωτοκόλλων θα μπορούσαν να αντικασταθούν απο αυτοματοποιημένες εργασίες με σκοπό την μείωση στον χρόνο ανάπτυξης πολύπλοκων εφαρμογών όπως επίσης και στην ομοιογένεια της ανάπτυξης εφαρμοργών. Τέτοια παραδείγματα συναντάμε συχνά σε εφαρμογές που μας ενδιαφέρει περισσότερο η ανάπτυξη του Bussiness Logic της εφαρμογής απο την ανάπτυξη κορεσμένων κομματίων κώδικα όπως για παράδειγμα ενός Telnet Client. Στην δουλειά μας προσπαθούμε να αυτοματοποιήσουμε τέτοιες διαδικασίες με την χρήση γεννήτριας κώδικα για την δημιουργία τόσο client όσο και server για τα βασικά πρωτόκολλα internet όπως το Telnet ,FTP ,HTTP ,SSH (βλέπε [11, 13, 12, 14]). Παρόμοιες δουλειές συναντάμε συχνα στην ανάπτυξη πολύπλοκων εφαρμογών τών νέων γλωσσών προγραμματισμού όπως τα WorkFlow.

1.1.2 Εισαγωγή στις τεχνολογίες

Για να αναπτύξουμε το Framework για την γέννηση των πρωτοκόλλων που επιλέξαμε να υλοποιήσουμε βρεθήκαμε στο δίλημμα του τί τεχνολογίες θα έπρεπε να χρησιμοποιήσουμε για την υλοποιήση του συστήματος. Παραδοσιακές τεχνολογίες για την γέννηση ενδιάμεσου κώδικα όπως ο Flex και ο Bison δεν θα ήταν και πολύ χρήσιμες γιατί το κόστος ανάπτυξης με

την χρήση τους θα ήταν πολύ μεγάλο. Για τον λόγο αυτό επιλέξαμε να χρησιμοποιήσουμε νέα εργαλεία για software engineering όπως το Velocity [8] και το XDoclet [9]. Επίσης για λόγους ταχύτητας στο runtime όπως επίσης και η ανάγκη για να είναι διαχειρίσημα τα components μας χρησιμοποιήσαμε την τεχνολογία JMX [1] για την υλοποίηση των collectors ή αλλιώς event listeners . Οι collectors είναι σχεδιασμένοι έτσι ώστε να τρέχουν στον JBoss Application Server [2]. Σαν βοηθητικό σύστημα έχουμε κατασκευάσει και έναν JMX HTTP Adaptor με την βοήθεια του Jakarta Struts Framework [7]. Τέλος θα πρέπει να αναφέρουμε ότι για την επεξεργασία τών metadata του κάθε collector χρησιμοποιήσαμε τον Castor XML parser [10].

1.1.3 Εισαγωγή στο **Framework**

Σκοπός αυτού του Framework είναι να δώσει την δυνατότητα στους χρήστες τους να κατασκευάζει εύκολα πρωτόκολλα με την χρήση της γλώσσας Java δίνοντας τους παράλληλα την ιδιότητα του Event Listener ώστε να μπορούν να συλλέγουν δεδομένα. Παρόλο που η κατασκευή του πρωτοκόλλου είναι transparent ο χρήστης θα πρέπει να γνωρίζει καλά τις παραμέτρους που απαιτούνται για την υλοποίηση κάποιου απο τα πρωτόκολλα που του παρέχουμε.

Στην διατριβή αυτή ασχοληθήκαμε με τα παρακάτω θέματα:

- Πιστοποίηση της ορθότητας του πρωτοκόλλου και κατασκευή ενεργής βιβλιοθήκης με την χρήση γεννητικού προγραματισμού.
- Άμεσο Deployment της βιβλιοθήκης ώστε να είναι πρός χρήση την ίδια στιγμή.
- Πρόσβαση στους collectors από ξεχωριστά κομμάτια Facets με σκοπό την απλοποίηση της διαδικασίας σύνδεσης τους με εξωτερικά συστήματα.

1.1.4 Συνέχεια

Στην συνέχεια αυτής της διατριβής θα εξετάσουμε τα εξής θέματα:

- Στο κεφάλαιο Σχετική δουλειά συζητάμε τόσο για τα εργαλεία που χρησιμοποιήσαμε, όσο και για δουλειά που έχει γίνει στο Publish/Subscribe μοντέλο.

- Στο κεφάλαιο Αρχιτεκτονική Συστήματος συζητάμε για την αρχιτεκτονική του συστήματος μας όπως επίσης και για τις σχεδιαστικές επιλογές(class diagrams.
- Στο κεφάλαιο Υλοποίηση Συστήματος συζητάμε για τα πρωτόκολλα που επιλέξαμε να υλοποιήσουμε όπως επίσης και για την πληροφορία που θα πρέπει να συνοδεύει το καθε ένα απο αυτά.
- Στο κεφάλαιο Μελλοντική Δουλειά και Συμπεράσματα συζητάμε για το τί μπορεί να γίνει στο μέλλον για να είναι το σύστημα πιο ολοκληρωμένο όπως επίσης και πιο ικανό για χρήση στον πραγματικό κόσμο.
- Στο Παράρτημα Α' παρατίθενται τα XSD του συστήματος.
- Στο Παράρτημα Β' παρατίθενται τα JAVADOC του API μας.

Κεφάλαιο 2

Σχετική Δουλειά

2.1 Περί **JMX** Γενικά

2.1.1 Τι είναι το **JMX ?**

Το JMX [1] είναι ένα ενοποιημένο framework για την εκτέλεση κομματιών από JAVA κώδικα σε μία σύγχρονη IT αρχιτεκτονική. Πριν από το JMX,δεν υπήρχε κάποια συγκεκριμένη προσέγγιση στην JAVA για να ξεκινάς,να διαχειρίζεσαι,να επιβλέπεις και να σταματάς διαφορετικά κομμάτια από μία εφαρμογή ή και ακόμα διαφορετικές εφαρμογές. Η διαχείριση τον εφαρμογών έχει επιτευχθεί μέσα από μία συλλογή από διαδικασίες και τυχαίου κώδικα διαχείρισης.Το JMX υπόσχεται να αντικαταστήσει με μία γενική αρχιτεκτονική τις τυχαίες αρχιτεκτονικές διαχείρισης για κάθε ξεχωριστή εφαρμογή.

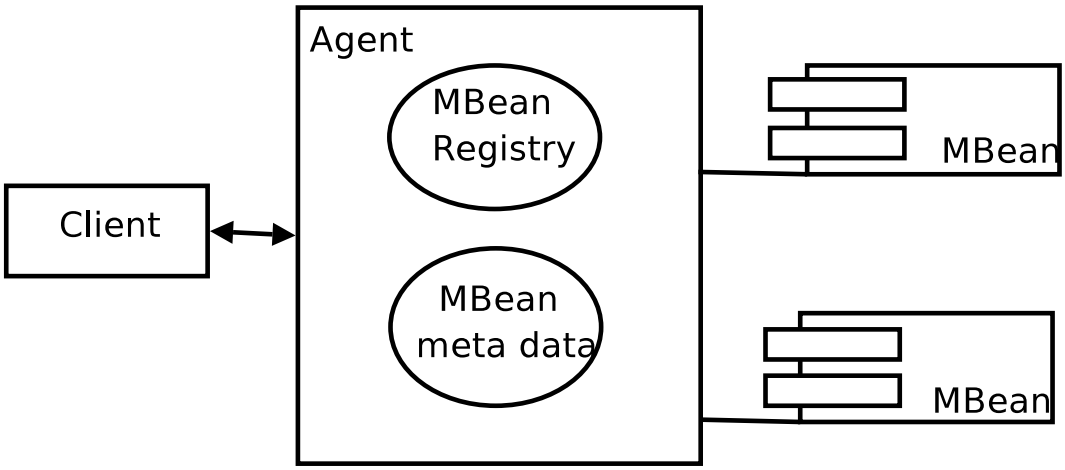
2.1.2 **MBean Component Types**

Τά MBeans είναι κομμάτια που υλοποιούν ένα management interface,είτε στατικά ,είτε δυναμικά. Στήν πρώτη περίπτωση το management interface είναι ένα Java interface.Στην δεύτερη περίπτωση το management interface ορίζεται από ένα σετ με κλάσεις που περιέχουν τα metadata.

Το JMX specification [1] ορίζει τέσσερις διαφορετικούς τύπους από MBean τρεις εκ•των οποίων είναι δεδομένοι. Οι τύποι αυτοί είναι:

Standard MBean

Dynamic MBean



Σχήμα 2.1: Overview of the agent and MBean components.

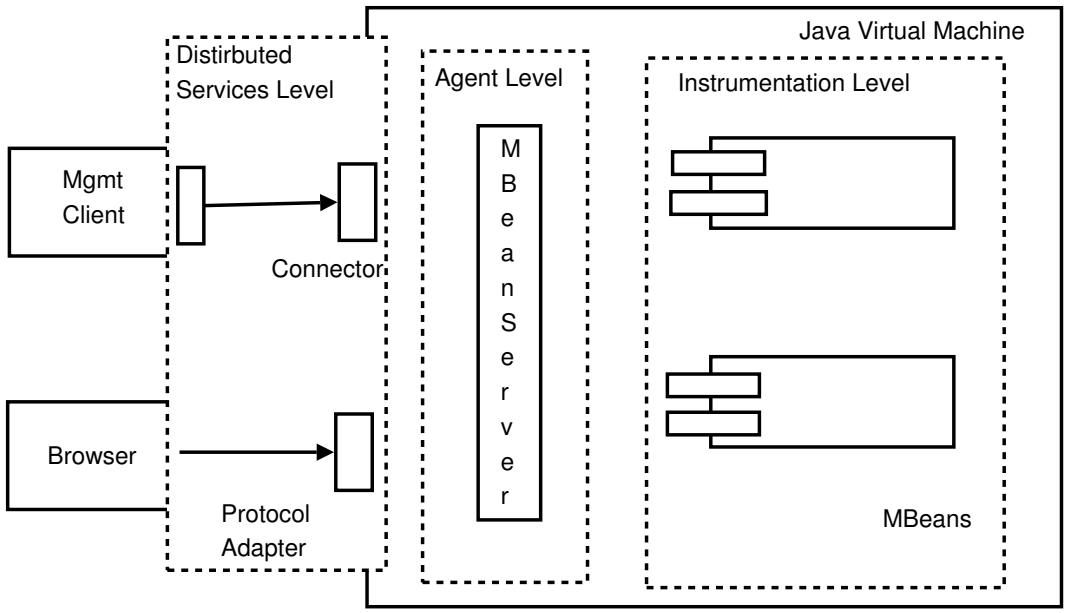
Model MBean

Open MBean

Τα Standard MBean,Dynamic MBean καί Model MBean είναι δεδομένα και καλά ορισμένα στο specification ενώ τα Open MBean είναι στην διάθεση του κατασκευαστή.

Τα Standard MBean δημιουργούνται υλοποιώντας ένα Java interface μαζί με την πληροφορία που θα πρέπει να διαχειρίζεται.Οι άλλοι τρεις τύποι περιγράφουν το management interface μέσα από ένα σετ με κλάσεις που περιέχουν τα metadata τους. Τα Model MBean επεκτείνουν τα Dynamic MBean επιτρέποντας σε επιπλέον περιγραφείς να προστίθενται στο management interface.Τα Open MBeans περιορίζουν τους τύπους αντικειμένων που χρησιμοποιούνται στο management interface σε ένα προκαθορισμένο σετ από κλάσεις που περιγράφουν τους βασικούς τύπους.

Τα MBeans είναι δεσμευμένα σε έναν agent ο οποίος είναι ικανός να τα διαχειρίζεται.Ο agent συμπεριφέρεται σαν την μνήμη για τα MBeans προσφέροντας την δυνατότητα για ερωτήσεις πάνω σε αυτά,όπως επίσης και να τα μεταβάλεις μέσω του management interface.Η σχέση μεταξύ του agent και των MBeans φαίνεται στο σχήμα 2.1.



Σχήμα 2.2: JMX Management Architecture.

Ένα άλλο σημαντικό θέμα το οποίο θα πρέπει να συζητήσουμε στο σχήμα 2.1 είναι οι επικοινωνίες μεταξύ των clients και των MBean.Αυτή γίνεται μόνο μέσω του Agent level και ποτέ απευθείας reference των MBean δεν εκτίθενται εκτός αυτού. Αυτό είναι ένα πολύ σημαντικό χαρακτηριστικό της αρχιτεκτονικής του JMX όπου θα συζητήσουμε και παρακάτω.

2.2 Η αρχιτεκτονική του **JMX**

2.2.1 Γενικά

Το JMX specification χωρίζει την αρχιτεκτονική ενός JMX βασισμένου συστήματος διαχείρισης σε τρία διαφορετικά επίπεδα. Το κάθε επίπεδο έχει ξεχωριστό ρόλο στην αρχιτεκτονική και διευθύνει ξεχωριστά ζητήματα στην αρχιτεκτονική του συστήματος.

Παρακάτω θα εξετάσουμε το κάθε επίπεδο του μοντέλου της αρχιτεκτονικής του JMX ξεχωριστά , αναγνωρίζοντας τον ρόλο του κάθε επιπέδου και το πώς θα πρέπει να χρησιμοποιείται σαν μέρος ενός management συστήματος. Το σχήμα 2.2 δίνει μια περιγραφή της αρχιτεκτονικής.

Το instumentation level ορίζει το πώς θα πρέπει να υλοποιείς τα MBeans έτσι ώστε να κάνεις τις πηγές ικανές για διαχείριση από τα κατάλληλα JMX-συμβατά εργαλεία διαχείρισης.Το instumentation level ορίζει τα MBeans, όπως επίσης και τους τέσσερις διαφορετικούς τύπους τους.

Στο επίπεδο του agent,το JMX specification ορίζει τον ρόλο ενός MBean server.Ο server είναι το κλειδί στην αρχιτεκτονική και είναι υπεύθυνος στο να διαχειρίζεται όλες τις εκτελέσεις μεταξύ των εφαρμογών διαχείρισης και των πηγών διαχείρισης λειτουργώντας ως ένας δίαυλος επικοινωνίας μεταξύ τους. Αυτό απομονώνει τις πηγές διαχείρισης από τον πελάτη , πράγμα που δημιουργεί μεγαλύτερη ευελιξία στο σύστημα.

Τα MBeans μπορούν να ανανεώνονται,να επαναεγκαθίστωνται και να μετακινούνται μεταξύ MBeans server χωρίς να πρέπει υποχρεωτικά να αποκόβουν την επικοινωνία του πελάτη με τον εξυπηρετητή. Οι εφαρμογές διαχείρισης ποτέ δεν απευθύνονται απευθείας στις πηγές διαχείρισης που έχουν αναλάβει. Αντί αυτού το επίπεδο του agent εισαγάγει το θέμα των object names όπου ο πελάτης χρησιμοποιεί μια μη απευθείας σύνδεση με την πηγή.

Τέλος το επίπεδο των distributed services ορίζει την επικοινωνία μεταξύ μίας εφαρμογής διαχείρισης και ενός agent ή ακόμα και μεταξύ δύο agent.

2.3 JBOSS

2.3.1 Τι είναι ο JBoss ?

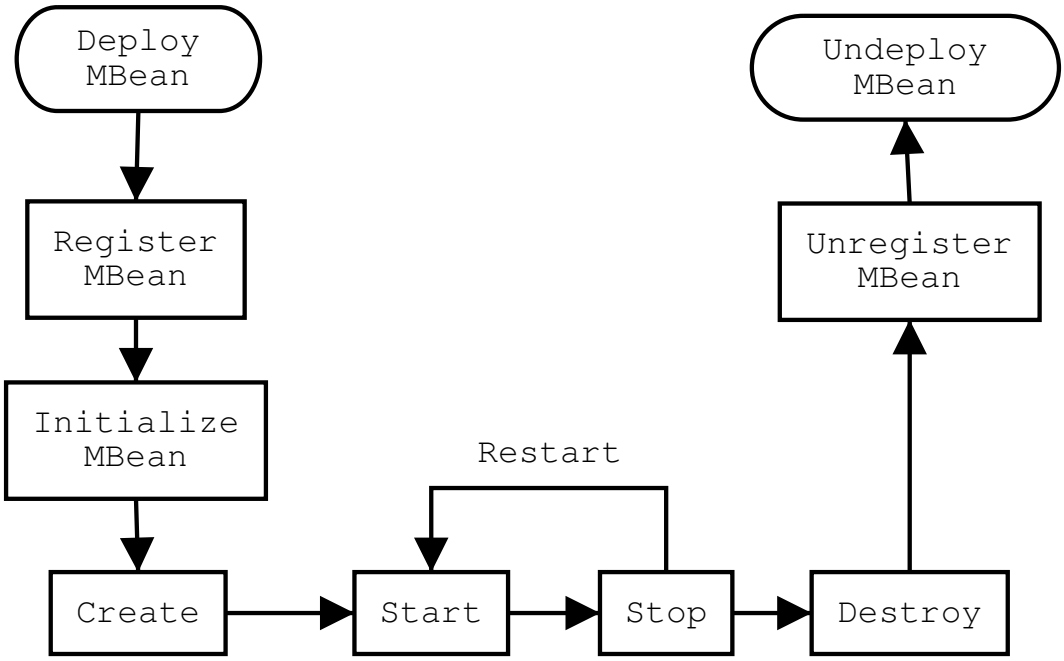
Ο JBoss [2] θεωρείται αυτή την στιγμή ως ο καλύτερος application server. Το JBossMX είναι ένα πολύ καλό παράδειγμα για τις δυνατότητες του JMX πέρα απο την διαχείριση εφαρμογών.Ο JBossMX είναι ο πυρήνας της αρχιτεκτονικής του application server.

Το JMX στόν JBoss χρησιμοποιείται όχι μόνο για την διαχείριση δεδομένων αλλά και για να παρέχει ενα καλό επίπεδο διαφοροποίησης σε μια κατανεμημένη πλατφόρμα εφαρμογών. Ο JBoss στηρίζεται πάνω στον JMX microkernel ο οποίος παρέχει διαχείριση πόρων, βασικές

υπηρεσίες όπως class loading, customization, life-cycle management καί service deployment. Όλα τα κομμάτια του JBoss είναι υλοποιημένα σαν MBeans και γιαυτό μπορεί να χρησιμοποιεί έναν MBean Server για την επικοινωνία μεταξύ τους κτλ. Αφού ο JBoss είναι υλοποιημένος απο ξεχωριστά MBean τα οποία μπορούν να διαχειριστούν απο έναν MBean Server μπορεί να επιτύχει έναν αρκετά μεγάλο βαθμό σταθερότητας.

Ο JBoss JMX microkernel δίνει την δυνατότητα για απομακρυνσμένο class loading απο ένα κεντρικό server κάνοντας το installation και το customization τού application server πολύ απλό.Ο JBossMX υποστηρίζει αυτά που θα πρέπει να υποστηρίζει σύμφωνα με το JMX specification συμπεριλαμβάνοντας τα standar , dynamic και model MBeans όπως επίσης και όλες τις βασικές υπηρεσίες για τους agents , όπως το Timer Service, το MLet Service κτλ. Επίσης πέρα απο τις standar προδιαγραφές , ο JBoss παρέχει επιπλέον υπηρεσίες , όπως τα XMBeans, class loader repositories, interceptor-based invocation, και plugable MBean registry.

Ο JBoss σου δίνει τη δυνατότητα να θέτεις MBean attributes απο configuration files. Το life-cycle τών MBeans στον JBoss φαίνεται στο σχήμα 2.3. Το life-cycle περιγράφεται αρκετά καλά απο το σχήμα 2.3.Αυτό που αξίζει συζήτηση είναι η περίπτωση που γίνεται κάποια αλλαγή στα MBean-attributes.Ο JBoss δεν επιτρέπει να λάβει χώρα καμμία αλλαγή μέχρι το MBean να γίνει restart. Αυτό δίνει την δυνατότητα σε πολλά attribute ταυτόχρονα να γίνονται αλλαγές στις τιμές τους και να γίνονται ενεργά ταυτόχρονα χωρίς να υπάρχουν προβλήματα synchronization. Ο JBoss έχει υλοποιημένες τις βασικές υπηρεσίες , όπως για παράδειγμα JNDI, scheduler, mail service, invokers, JBossWeb,JBossMQ σαν MBeans.Τέλος ο JBoss σου δίνει την δυνατότητα να ορίσεις εξαρτήσεις μεταξύ των MBean έτσι ώστε για παράδειγμα ένα MBean να ξεκινάει αφού ξεκινήσει ένα άλλο MBean.

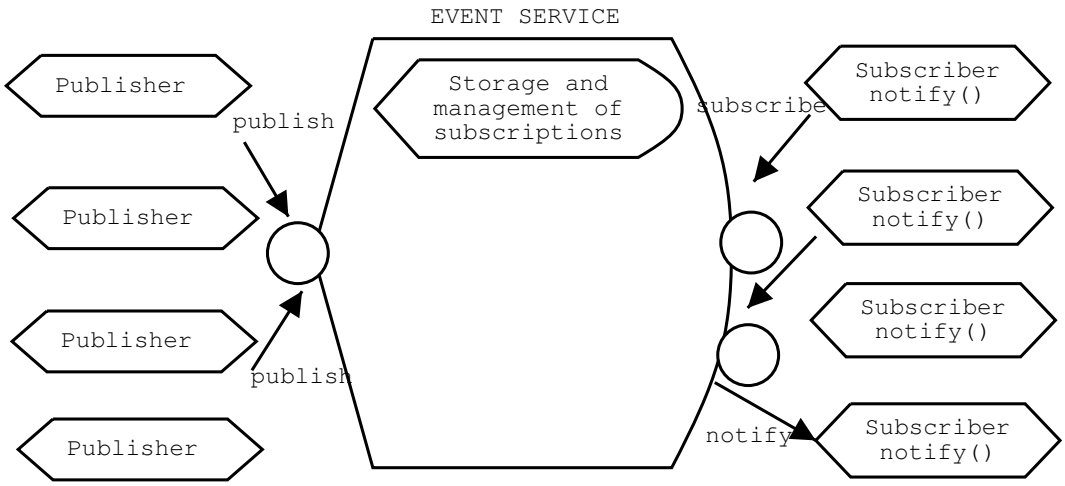


Σχήμα 2.3: The JBoss MBean life-cycle.

2.4 Publish-Subscribe

2.4.1 Περί **Publish-Subscribe** Γενικά

Για να περιγράψουμε το μοντέλο του Publisher/Subscriber [3],[4] θα χρησιμοποιήσουμε ώς παράδειγμα το σχήμα 2.4. Σύμφωνα με το σχήμα ό Publisher δημοσιεύει κάποιο event δηλαδή κάποιο μήνυμα κάθε φορά που αυτό είναι δυνατό. Αντίστοιχα ο Subscriber ζητά να λάβει το event πού τον ενδιαφέρει και το λαμβάνει κάθε φορά που αυτό ειναι διαθέσιμο. Κάθε φορά που ο Publisher δημοσιεύει κάποιο event, αυτό μπαίνει σε κάποιο queue. Η αρχιτεκτονική αυτή ονομάζεται Message-Oriented Middleware (MOM). Το κάθε event καταναλώνεται απο n consumers δηλαδή τα semantics είναι one-of-n.Το Topic-based Publisher/Subscriber model το οποίο χρησιμοποιεί και το JMS είναι μεν στατικό και πολύ απλό αλλά μπορεί να υλοποιηθεί πολύ έξυπνα .



Σχήμα 2.4: A simple Publish/Subscribe System.

2.5 Event-Condition-Action

2.5.1 Event-Condition-Action Rules

Οι Event-Condition-Action Rules [5, 6] είναι κανόνες που βοηθάνε στην επιλογή του action που θα λάβει χώρα με βάση το κάθε event. Δηλαδή για κάθε Event που προκύπτει κάποιο Action θα πρέπει να ενεργήσει με βάση τους κανόνες που ορίσαμε. Θα μπορούσαμε να ορίσουμε τους ECA Rules σάν μία Boolean μέθοδο η οποία πέρνει σαν ορίσματα το Event και το Action και επιστρέφει αντίστοιχα με βάση τον κανόνα της. Ο ψευδοκώδικας για μια τέτοια συνάρτηση φαίνεται παρακάτω.

ECA RULE CODE

Listing 2.1: " ECA RULE CODE"

```
1  boolean rule(Event a, Action b)
2  {
3
4          if(f(a)) // f is the condition
5          then
6                  b(a)
7                  return true;
8          else
9                  return false;
10 }
```

Τα **semantics** ενός λογικού **event** Ένα λογικό event είναι ένα event ένα event με ένα condition το οποίο θα πρέπει να ικανοποιείται για να θεωρήσουμε ότι το event έχει λάβει χώρα όπως φαίνεται στο Listing 2.2.

Semantics Of Logical Events

Listing 2.2: " Semantics Of Logical Events"

```
E: T–> {True, False}

C: T–> {True, False}

E(t) = "True", if an event of type t occurs at
    time t
       "False" otherwise.

C(t) = "True", if the context at time t satisfies
       Constraint C
       "False" otherwise.
```

Συνεπώς ένα λογικό event λαμβάνει χώρα όταν ισχύει η συνθήκη: $Ec(t) = E(t) \wedge C(t)$

Οι λογικές συνθήκες Οι λογικές συνθήκες είναι οι συνθήκες για τα event και όχι απλά κανόνες.Είναι μία Boolean μέθοδος F η οποία παίρνει μέρος κάθε φορά που προκύπτει κάποιο event E.Μία λογική συνθήκη μπορεί να περιγραφεί όπως στο Listing 2.3.

Semantics Of Logical Conditions

Listing 2.3: " Semantics Of Logical Conditions"

```
context(t) –> the context of the occurence of
       an event at time t;

F_condition(t,context(t)) –> Boolean function which
          returns:
          "True" if the condition
          is satisfied
          "False" otherwise
```

Τα **semantics** ενός λογικού **event** σε **ECA rules** Ένας κανόνας καθορίζεται απο ένα event E ,μία συνθήκη C και ένα action A.Συνεπώς ένας κανόνας μπορεί να οριστεί ώς εξής:
$R(E(t), C(t'), A(t'')) = (E(t) \wedge C(t')) \wedge A(t'')$
Όπου το A(t") είναι δεδομένο και πρέπει να εκτελεστεί κάθε φορά ένα event E(t) προκύπτει και ικανοποιείται η συνθήκη C(t').

2.6 Jakarta Struts

2.6.1 Jakarta Struts Γενικά

Το Jakarta Struts πρόγραμμα [7], είναι ένα ανοιχτού κώδικα πρόγραμμα επιχορηγούμενο από τον οργανισμό τής A Softaware.Είναι μία server side υλοποίηση του σχεδιαστικού προτύπου Model-View-Controller(MVC).Το πρόγραμμα Struts,αρχικά ξεκίνησε από τόν Craig McClanahan τόν Μάιο του 2000,αλλά απο τότε το έχει αναλάβει η κοινότητα ανοιχτού κώδικα.

2.6.2 Καταλαβαίνοντας το MVC

Για να κερδίσουμε μία πλήρη κατανόηση τού Struts framework,πρέπει πρώτα να έχουμε μία πλήρη κατανόηση του MVC μοντέλου στο οποίο είναι βασισμένο.Το MVC μοντέλο το οποίο δημιουργήθηκε από την Smalltalk,αφορά τρία κομμάτια.Ένα Model,ένα View και έναν Controller.Ο παρακάτω ορίζει το καθένα απο τα παρακάτω κομμάτια.

Θα συζητήσουμε σε περισσότερη λεπτομέρεια τα παραπάνω στην συνέχεια αυτής της ενότητας. Κάποια από τα πλεονεκτήματα του μοντέλου MVC είναι:

Reliability:Τα κομμάτια των συνδιαλλαγών και της παρουσίασης έχουν ξεκάθαρη διαφοροποίηση , πράγμα που σου επιτρέπει να αλλάζεις την εμφάνιση και την αισθητική χωρίς να πρέπει να επαναμεταγλωττίσεις τον κώδικα του μοντέλου ή του ελεγκτή.

Component	Description
Model	Αντιπροσωπεύει τα αντικείμενα δεδομένων. Το μοντέλο είναι ότι υπολογίζεται και στην συνέχεια παρουσιάζεται στον χρήστη.
View	Υπηρετεί στην παρουσίαση στην οθόνη του μοντέλου. Είναι το αντικείμενο που παρουσιάζει την τωρινή κατάσταση του αντικειμένου τών δεδομένων.
Controller	Ορίζει τον τρόπο με τον οποίο η διεπαφή αντιδρά με την είσοδο του χρήστη.Το μέρος του Controller είναι το αντικείμενο το οποίο επεξεργάζεται το μοντέλο ή το αντικείμενο δεδομένων.

Πίνακας 2.1: The MVC Model

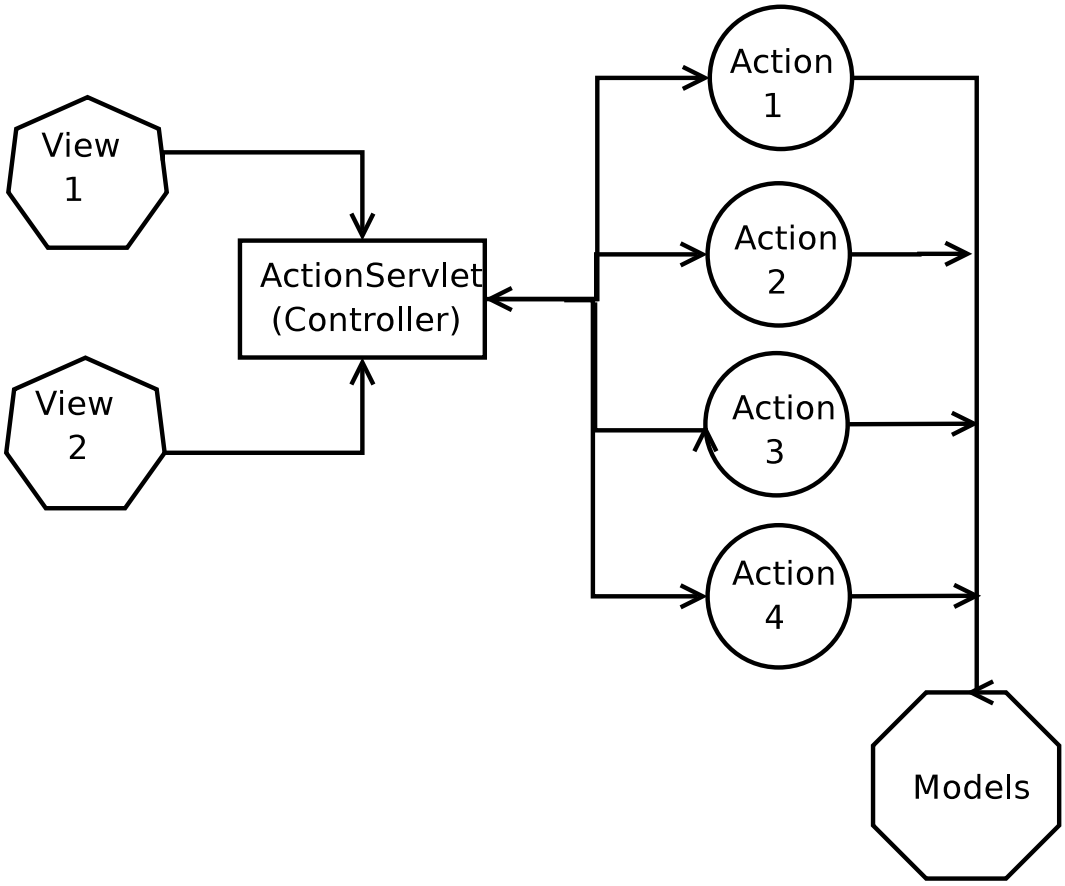
High reuse and adaptability:Το MVC σου επιτρέπει να χρησιμοποιείς πολλαπλούς τύπους όψεων,εκ των οποίων όλοι προσπελάζουν τον ίδιο κώδικα στην πλευρά του server. Αυτό συμπεριλαμβάνει οτιδήποτε από Web browser(HTTP) εως wireless browser(WAP).

Very low development and lifecycle costs:Το MVC κάνει δυνατό να έχεις χαμηλού επιπέδου προγραμματιστές να αναπτύσσουν και να συντηρούν την διεπαφή της εφαρμογής.

Rapid deployment:Ο χρόνος ανάπτυξης μπορεί να μειωθεί σημαντικά, επειδή οι προγραμματιστές του ελεγκτή (Java Developers) εστιάζουν αποκλειστικά στην ανάπτυξη των συνδιαλλαγών ενώ οι προγραμματιστές της όψης (HTML and JSP developers) εστιάζουν αποκλειστικά στην παρουσίαση.

Maintainability:Η δαιφοροποίηση της παρουσίασης απο την λογική της εφαρμογής καθιστά πιο εύκολη την συντήρηση και την αλλαγή εφαρμογών ανεπτυγμένων με τήν χρήση του Struts

The Struts Implementation Of The MVC Το Struts Framework κατέχει μία server side υλοποίηση του MVC μοντέλου με την χρήση ενός συνδυασμού απο JSPs και τυχαίων Jsp tags και ενός JAVA Servlet.Σε αυτήν την ενότητα με συντομία θα περιγράψουμε πως το Struts Framework εντάσει το κάθε μέρος του MVC μοντέλου.Με το τέλος της συζήτησης μας θα έχουμε καταλήξη σε ένα σχήμα όμοιο με το Σχ. 2.5



Σχήμα 2.5: Η υλοποίηση του MVC στο Struts .

Το Σχήμα 2.5 αντιπροσωπεύει την πορεία που ακολουθούν οι περισσότερες εφαρμογές του struts.Η διαδικασία μπορεί να σπάσει σε 5 βασικά βήματα.Τα βήματα αυτά και η ακολουθία τους είναι:

1 Μία αίτηση γίνεται από τήν προηγούμενη διεπαφή που προβαλλόταν.

2 Η αίτηση λαμβάνεται απο το ActionServlet,το οποίο συμπεριφέρεται σαν τον Controller και το ActionServlet αναζητεί το αιτούμενο URI σέ ένα XML αρχείο και αποφασίζει για το όνομα του Action που θα αναλάβει την διαδικασία τής κατάλληλης λογικής.

3 Η κλάση Action διεξάγει την λογική της πάνω στα κομμάτια των μοντέλων που σχετίζονται με την εφαρμογή μας.

4 Όταν η Action ολοκληρώσει την διεργασία της,επιστρέφει τον έλεγχο στό ActionServlet. Σαν μέρος της επιστροφής,η κλάση Action παρέχει ένα κλειδί που δείχνει στα αποτελέσματα της επεξεργασίας της.Το ActionServlet χρησιμοποιεί τα κλειδιά για να αποφασίσει που θα προωθηθούν τα αποτελέσματα για να παρουσιαστούν.

5 Η αίτηση ολοκληρώνεται οταν το ActionServlet απαντά προωθώντας την αιτηση στην διεπαφή που συσχετίζεται με το κλειδί και η διεπαφή παρουσιάζει τα αποτελέσματα που έλαβε απο το Action.

2.7 Jakarta Velocity

2.7.1 Τι είναι το **Velocity?**

Το Jakarta Velocity project [8] είναι μία υλοποίηση σε Java μίας μηχανής template.Ο όρος template σημαίνει κομμάτια απο software που πέρνει σαν είσοδο ένα template ή templates και τα ζευγαρώνει με μεταβλητά δεδομένα για να παράγει μια μορφή εξόδου.Στην περίπτωση του Velocity τα templates είναι απλά αρχεία κειμένου που περιέχουν Velocity Template Language (VTL) μαζί με στατικά περιεχόμενα. Τα κατευθυντήρια της VTL λένε στο Velocity πώς να συνδιάσει τα στατικά περιεχόμενα στο αρχείο template μαζί με τα μεταβλητά δεδομένα απο την εφαρμογή Java για να παράγει την επιθυμητή έξοδο. Τα μεταβλητά δεδομένα μπορούν να

έρθουν απο οπουδήποτε , όπως για παράδειγμα απο μία βάση δεδομένων.Ο μηχανισμός για να μεταφέρεις δεδομένα απο την εφαρμογή στο Velocity είναι πολύ απλός και σου επιτρέπει να δουλεύεις με όλα τα αντικείμενα της Java συμπεριλαμβάνοντας και Collections και πίνακες. Η έξοδος απο το Velocity είναι πάντα κείμενο αλλά η μορφή του κειμένου δεν αποτελεί περιορισμό.Αυτο σημαίνει ότι μπορείς να παράγεις HTML όπως και XML έξοδο αλλά και οποιαδήποτε άλλη μορφή κειμένου όπως Java.

Παρόλο το ότι κάνει το Velocity το κάνει απλά και η χρήση του είναι ιδιαίτερα απλή ,η μηχανή του Velocity είναι πολύ καλά σχεδιασμένη και προσφέρει ένα γρήγορο parser,template caching και plugin-based εξέλιξη.Πολλά εργαλεία έχουν φτιαχτεί με βάση το Velocity.

2.8 XDoclet

2.8.1 **XDoclet** Γενικά

Το XDoclet [9] είναι ένα εργαλείο για να γεννάς κώδικα το οποίο υπόσχεται πολλά πλεονεκτήματα για τις πιό κοινές διαδικασίες της Java.Το XDoclet μπορεί να σε βοηθήσει να παράγεις εφαρμογές πίο γρήγορα και με λιγότερο κόπο.Σε βοηθάει να ξεφύγεις απο την διαδικασία του Deployment Descriptor που πολλές φορές κοστίζει πολύ σε χρόνο αντικαθιστώντας την με λίγα Javadoc tags.Ένα απο τα χαρακτηριστικά του XDoclet είναι το ότι είναι ταυτόχρονα και ένα FrameWork και ένα σέτ απο εφαρμογές γέννησης. Παρόλο που στις λεπτομέρειες κάθε εφαρμογή είναι διαφορετική (για παράδειγμα η γέννηση για EJB είναι διαφορετική απο την γέννηση για Struts) τα κεντρικά θέματα και η χρήση τους έχουν πολλά κοινά.

2.9 Castor

2.9.1 **Castor** Γενικά

Ο Castor [10] είναι ένα Open Source Framework για XML. Είναι ένας XML Parser όποιος χρησιμοποιεί τον SAX και reflection για να κάνει bind ένα xsd με Java Objects τα οποία γεννάει.Είναι πολύ βολικός στην χρήση αφού σου δίνει την δυνατότητα να επεξεργάζεσαι XML documents χρησιμοποιώντας αντικείνα της Java.Τα αντικείμενα που γεννάει έχουν το

συντακτικό των Java Bean δηλαδή για κάθε attribute έχουν αντίστοιχα τους getters και τους setters. Επίσης υπάρχουν και δύο βασικές μέθοδοι για να κάνεις marshal και unmarshal τα αντικείμενα σε XML documents και το αντίστροφο.

Κεφάλαιο 3

Αρχιτεκτονική Συστήματος

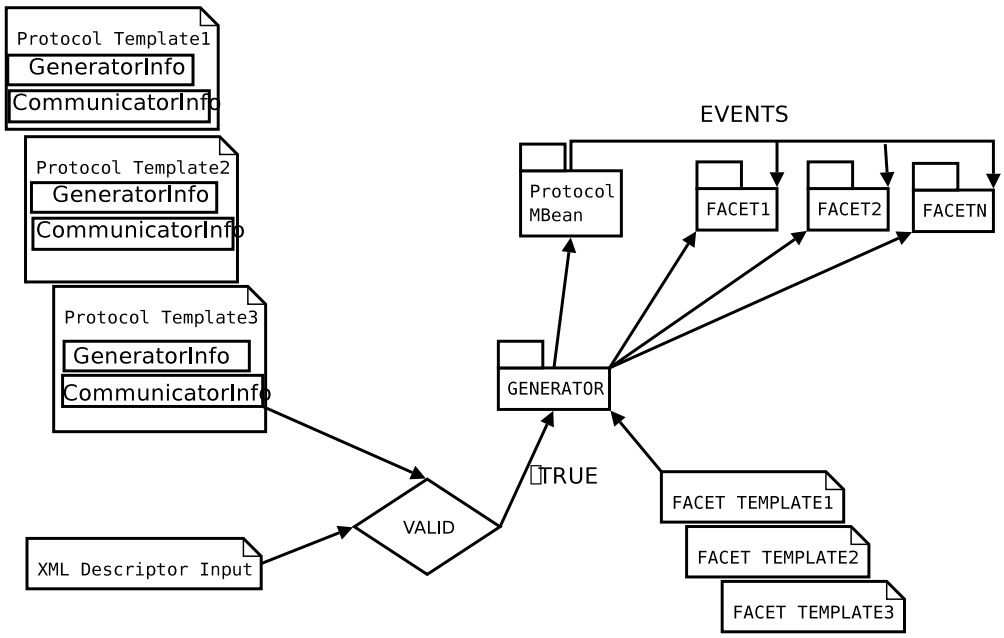
3.1 Αρχιτεκτονική του συστήματος

3.1.1 Τι είναι το σύστημα μας.

Σκοπός αυτής της δουλειάς είναι η δημιουργία collectors με την χρήση γεννήτριας κώδικα. Με τον όρο collectors ουσιαστικά εννοούμε event listeners για τα βασικά πρωτόκολλα internet και όχι μόνο. Για να γίνουμε πιό συγκεκριμένοι θα προχωρίσουμε σε ένα παράδειγμα και θα θεωρήσουμε σαν event μία γραμμή απο τα logs που πετάει το Pam Authentication στο unix. Αυτό θα μπορούσε να είναι ένα event ενώ ένας event listener θα μπορούσε να δημιουργηθεί κάνοντας για παράδειγμα tail -f /var/log/auth.log , το οποίο στην ουσία κάνει Poll στόν file descriptor του αρχείου που γράφονται τα logs του Pam.

Η δουλειά του κάθε collector είναι να μαζεύει αυτά τα events με βάση το πρωτόκολλο που υποστηρίζει και να τα στέλνει σε κάποιο JMS Topic ή να στέλνει κάποιο JMX notification το οποίο έχει επιλέξει ο χρήστης.Η επιλογή του κάθε Topic ή του MBean που κάνει notify γίνεται με βάση κανόνων που ορίζει ο χρήστης.

Ένα τέτοιο σύστημα θα μπορούσε να χρησιμοποιηθεί απο εταιρείες τηλεφωνίας σαν υποστήριξη για ένα σύστημα χρεώσεων Billing Over IP.Επίσης θα μπορούσε να χρησιμοποιηθεί για την συλλογή logs απο διάφορους υπολογιστές σε μία κεντρική βάση, όπως επίσης και για να παρέχει γρήγορα υπηρεσίες σε χρήστες καταγράφοντας τα events για παράδειγμα ,



Σχήμα 3.1: Moulari architecture Steps.

παρέχοντας έναν HTTP Server και μαζεύοντας τα στατιστικά απο αυτόν πολύ εύκολα.

Πέρα απο τούς collectors το σύστημα δίνει την δυνατότητα στον χρήστη να γεννήση και Facet.Η δουλεία του κάθε Facet είναι να κάνει Subscribe σε ένα Topic ή να διαχειρίζεται τα notifications που περνάνε και να τα αξιοποιεί κατάλληλα. Ένα παράδειγμα για Facet θα μπορούσε να είναι ένα Facet που μιλάει με JDBC και να συλλέγει τα Events σε μία βάση δεδομένων. Σκοπός των Facet είναι να μπορεί εύκολα το σύστημα να συνεργάζεται και με άλλες εφαρμογές.

3.1.2 Το **Use Case** Του Συστήματος

Η λειτουργία της διαδικασίας που ακολουθείται για να γεννηθεί ο κώδικας που ζητάμε απο το σύστημα φαίνεται απο το σχήμα 3.1. Τα βήματα που ακολουθούνται περιγράφονται αναλυτικά παρακάτω.

1 Ο χρήστης δίνει σαν είσοδο ένα XML αρχείο το οποίο περιέχει όλη την απαραίτητη πληροφορία για το πρωτόκολλο το οποίο θα πρέπει να γεννηθεί το template το οποίο θα χρησιμοποιήσει , όπως επίσης και για το τί Facets θα γεννηθούν όπως επίσης και για τον τρόπο με τον οποίο θα επικοινωνίση με τα Facet(JMS,JMX NOTIFICATION).

2 Ελέγχεται το κατα πόσο η είσοδος του χρήστη είναι valid τόσο με βάση το xsd της εισόδου όσο και με βάση την πληροφορία του GeneratorInformation του αντίστοιχου template που επέλεξε. Η πληροφορία αυτή είναι βασισμένη στο xsd που βρίσκεται στο Παράρτημα Ά. Για παράδειγμα η πληροφορία που περιέχεται για τον FTPGenerator είναι το Host, Port, Username, Password που είναι η βασική γνώση που απαιτείται για την χρήση του πρωτοκόλλου FTP.Επίσης ελέγχεται και το κατα πόσο είναι σωστή η είσοδος για τον τρόπο με τον οποίο θα επικοινωνίση με τα Facet.Για παράδειγμα εάν ο χρήστης επιλέξει να χρησιμοποιείσει το JMS για επικοινωνία τότε θα πρέπει να έχει δώσει στην είσοδο του το όνομα του Topic και το host.Αντίστοιχει πληροφορία απαιτείται και για τα Facet.

3 Με την χρήση του Velocity [8] γεννιέται ο κώδικας για το MBean του Collector όπως επίσης και ο κώδικας για τα Facets που έχει ζητήσει ο χρήστης να γεννηθούν. Αυτό γίνεται με την ορθή επιλογή του σωστού template που επέλεξε ο χρήστης.

4 Με την χρήση του XDoclet [9] γεννιέται το interface που υλοποιεί το MBean του Collector όπως επίσης και ο Deployment Descriptor του MBean.

5 Ο κώδικας που εχει γεννηθεί γίνεται compile και στην συνέχεια πακετάρεται σε ένα sar (Service Archive).Στην συνέχει ο Generator αναλαμβάνει να κάνει deploy το sar στον JBoss Application Server.Το deploy γίνεται μέσω HTTP/POST στον host που έχει ορίσει ο χρήστης απο τα properties της εφαρμογής.

Απο την στιγμή που ο collector έχει γίνει deploy στόν JBoss είναι διαθέσιμος για εκτέλεση. Ένας απο τους τρόπους που μπορούμε να εκτελέσουμε τις μεθόδους του collector είναι μέσω του HTTP Adaptor. Οι βασικές μέθοδοι είναι startCollector, stopCollector, createCol-

lector , destroyCollector , ενώ δίνεται και η δυνατότητα να γίνουν update τα attributes του πρωτοκόλλου.

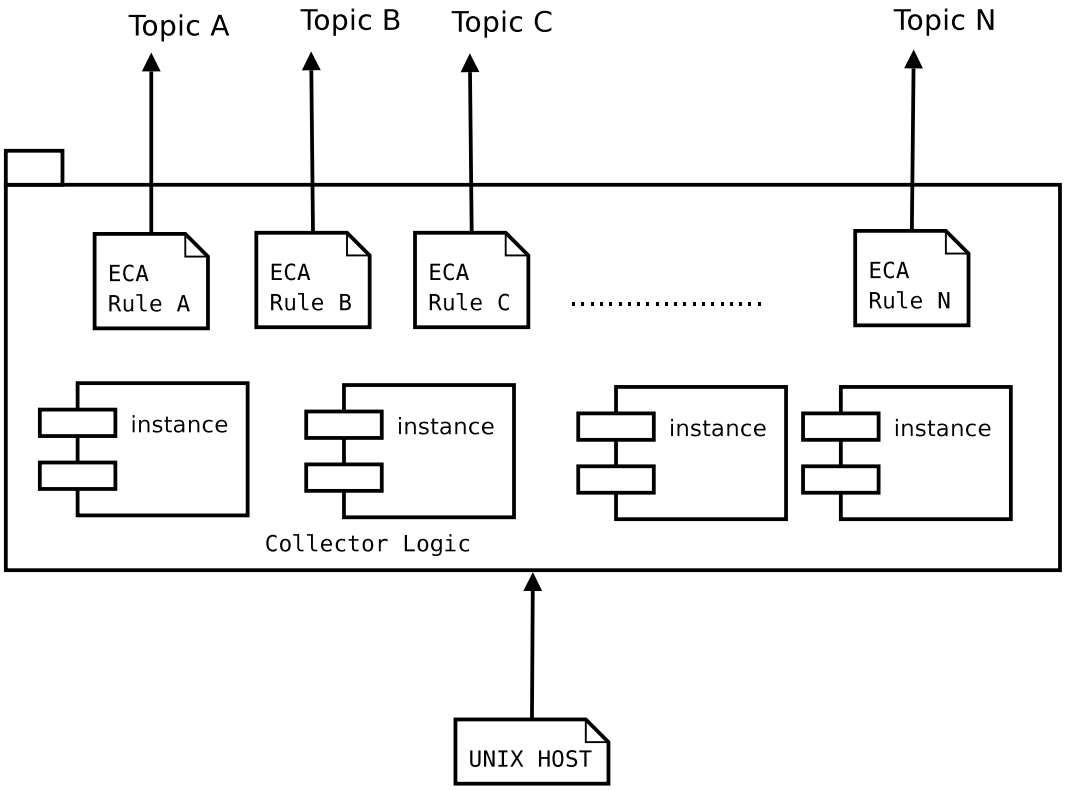
3.1.3 Περιγραφή **Collector**

Στό σχήμα 3.2 φαίνεται αναλυτικά η περιγραφή ενός collector MBean. Ο κάθε collector μπορεί να κάνει publish μηνύματα σε όσα Topics επιλέξει να δημιουργήσει ο χρήστης. Η επιλογή του κάθε Topic γίνεται με την χρήση ECA (Event Condition Action) rules [5] που ορίζει ο χρήστης. Το κάθε MBean μπορεί να περιέχει όσα instances του πρωτοκόλλου επιλέξει ο χρήστης αλλα κάθε φορά χρησιμοποιεί ένα instance για να συλλέξει τα events που τον ενδιαφέρουν. Την λογική του Collector για το πώς θα συλλέγει τα events την καθορίζει ο χρήστης. Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι ECA rules.Ο χρήστης έχει την δυνατότητα να επιλέξει και να δημιουργήσει όσους κανόνες θέλει όπως επίσης του δίνεται η δυνατότητα να επιλέξει και ότι action θέλει αυτός για να αξιοποιήσει τα events .Στην περίπτωσή μας τα action είναι μόνο με JMS publisher όπως επίσης και με Remote JMX Notifications αλλά στον μέλλον θα μπορούσαν εύκολα να υλοποιηθούν και άλλα action όπως για παράδειγμα να δημοσιεύει τα events με CORBA, RMI και ότι άλλο μπορεί κάποιος developer να σκεφτεί. λογική τών κανόνων φαίνεται καλύτερα στο Listing 3.1.

ECA RULE CONDITIONS AND ACTIONS

Listing 3.1: " ECA RULE CONDITIONS AND ACTIONS"

```
1  <base-cond name="a">
2    if foo return true;
3  </base-cond>
4  <base-cond name="b">
5          if bar return true;
6  </base-cond>
7  <base-cond name="c">
8          if foobar return true;
9  </base-cond>
10 <cond>
11 return((a&&b)||c);
12 </cond>
13 <action type="JMS">
```



Σχήμα 3.2: Genarated MBean Instance.

```
14    publish(event);
15  </action>
```

Στο Listing 3.1 φαίνεται ότι ο χρήστης μπορεί να ορίσει όσους βασικούς κανόνες θέλει (Base Conditions) .Με την χρήση τών κανόνων αυτών μπορεί να κάνει πολύπλοκους συνδιασμούς δημιουργώντας με αυτόν τον τρόπο πίο αποτελεσματικούς κανόνες. Επίσης μπορεί να ορίσει και ότι Action θέλει.Στην περίπτωσή μας το action είναι το JMS όπως επίσης και στο παραδειγμά μας.

3.1.4 Collector And Facet Design

Στην ενότητα αυτή θα περιγράψουμε τις σχεδιαστικές επιλογές που επιλέξαμε για την κατασκευή του κάθε Collector όπως επίσης και του κάθε Facet. Ο κάθε collector είναι καταρχήν ένα MBean. Για να αποκτήσει τις ιδιότητες του Standar MBean ακολουθεί τα εξής βήματα:

1 Υλοποιεί τα interface ServiceLifeCycle, MBeanRegistration, ServiceMBean, CollectorMBean ενώ κληρονομεί απο τίς abstact classes NotificationBroadcasterSupport , ServiceMBeanSupport ,Collector.

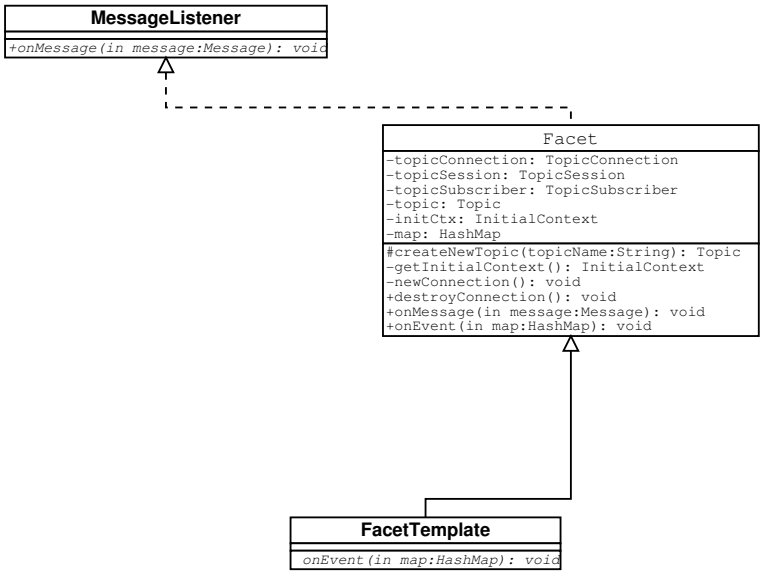
2 Ο κάθε collector που γεννιέται είναι ταυτόχρονα και Publisher σε κάποια Topic.Η υλοποίηση για να είναι Publisher βρίσκεται στην κλάση Collector την οποία κληρονομεί.

Τα παραπάνω βήματα φαίνονται απο τό class diagram του collector που φαίνεται στο σχήμα 3.3.

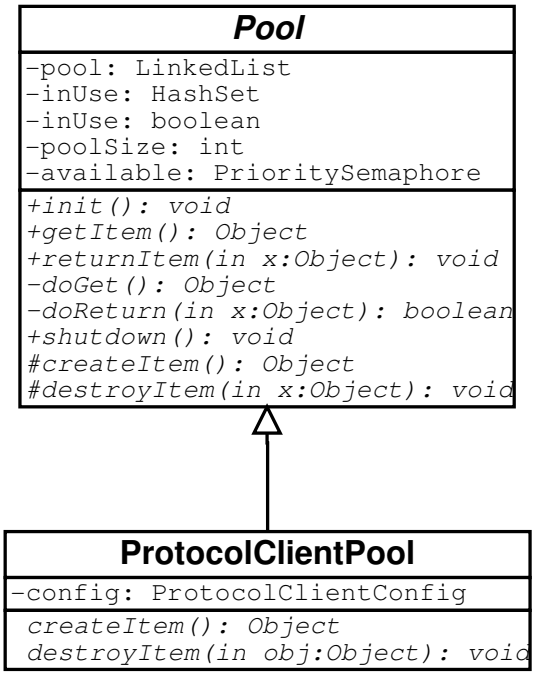
Το κάθε Facet που γεννιέται είναι και Subscriber σε κάποιο Topic. Η υλοποίηση του Subscriber κρύβεται στην abstract class Facet. Το class diagram του κάθε facet φαίνεται στο σχήμα 3.4.

Σε κάποια απο τα πρωτόκολλα που γεννάμε χρησιμοποιούμε το Pattern Pool για τα connection τους.Το class diagram για το Pattern Pool φαίνεται στο σχήμα 3.5.

Σχήμα 3.3: Class Diagram of Protocol Collector.



Σχήμα 3.4: Class Diagram of A Facet Template.



Σχήμα 3.5: Class Diagram of Pool Pattern.

Κεφάλαιο 4

Υλοποίηση Συστήματος

4.1 Υλοποίηση του συστήματος μας

4.1.1 Εισαγωγή

Οι Collectors μπορεί να είναι αρκετά πολύπλοκα components. Στη δουλειά μας προσπαθήσαμε να καλύψουμε αρκετά απο βασικά πρωτόκολλα internet , όπως το Telnet, το FTP ,το SSH και έναν HTTP Server. Τα απαραίτητα attributes που απαιτούνται για την λειτουργία των παραπάνω πρωτοκόλλων , όπως επίσης και η κοινή πληροφορία που απαιτείται για την δημιουργία ενός Collector , φαίνονται παρακάτω.

4.2 Τα Πρωτόκολλα

4.2.1 **Telnet Client**

Το Telnet [11] είναι ένα πρωτόκολλο το οποίο σου δίνει την δυνατότητα να έχεις απομακρυσμένο τερματικό απο κάποιο άλλο υπολογιστή. Στην υλοποίηση μας τα απαραίτητα attributes πού απαιτούνται για την υλοποίηση του φαίνονται στον πίνακα 4.2.1.

Στο Template του πρωτοκολλου Telnet περιέχει και ένα αντικείμενο τύπου TelnetClient με το όνομα telnet. Οι μέθοδοι του αντικειμένου αυτού δίνονται στο Παράστημα Β΄. Επίσης πέρα απο το αντικείμενο client ο χρήστης μπορεί να χρησιμοποιήσει και τα εξής αντικείμενα:

- Syslog. Το αντικείμενο αυτό αντιστοιχεί σε μία γραμμή απο τα Syslog logs.

Name	Description	Value
host	The Host To Connect	localhost
port	The Port To Connect To	23
endLineChar	The End Of Line Character	/n
exitString	The Exit String	logout
prompt	The Prompt Of The Terminal	$
numberOfConnections	Number of concurrent connections	4
send0	String to Be Sent	pat
wait0	String to Be Waited	.*login:
send1	String to Be Sent	password
wait1	String to Be Waited	.*Password:
send2	String to Be Sent	
wait2	String to Be Waited	
send3	String to Be Sent	
wait3	String to Be Waited	
send4	String to Be Sent	
wait4	String to Be Waited	
send5	String to Be Sent	
wait5	String to Be Waited	
send6	String to Be Sent	
wait6	String to Be Waited	
send7	String to Be Sent	
wait7	String to Be Waited	
send8	String to Be Sent	
wait8	String to Be Waited	
send9	String to Be Sent	
wait9	String to Be Waited	

Πίνακας 4.1: The Telnet Protocol

Name	Description	Value
host	The Host To Connect	localhost
port	The Port To Connect To	21
userName	The Username To connect To	pat
password	The Password To connect To	password
numberOfConnections	Number of concurrent connections	4

Πίνακας 4.2: The FTP Protocol

- Pam. Το αντικείμενο αυτό αντιστοιχεί σε μία γραμμή απο τα Pam logs.
- Utmp. Το αντικείμενο αυτό αντιστοιχεί σε μία εγγραφή απο τα UTMP logs.

4.2.2 FTP Client

Το FTP [12] είναι ένα πρωτόκολλο το οποίο σου δίνει την δυνατότητα να μεταφέρεις αρχεία απο απομακρυσμένο υπολογιστή. Στην υλοποίηση μας τα απαραίτητα attributes που απαιτούνται για την υλοποίηση του φαίνονται στον πίνακα 4.2.2.

Στο Template του πρωτοκόλλου FTP περιέχει και ένα αντικείμενο τύπου FTPClient με το όνομα ftp. Οι μεθόδοι του αντικειμένου αυτού δίνονται στο Παράρτημα Β′.

4.2.3 SSH Client

Το SSH [13] είναι ένα πρωτόκολλο το οποίο σου δίνει την δυνατότητα να τερματικό κάτω απο ασφαλή γραμμή από απομακρυσμένο υπολογιστή. Στην υλοποιησή μας τα απαραίτητα attributes που απαιτούνται για την υλοποίηση του φαίνονται στον πίνακα 4.2.3.

Στο Template του πρωτοκόλλου SSH περιέχει και ένα αντικείμενο τύπου SSHClient με το όνομα ssh. Οι μεθόδοι του αντικειμένου αυτού δίνονται στο Παράρτημα Β′. Επίσης πέρα απο το αντικείμενο client ο χρήστης μπορεί να χρησιμοποιήσει και τα εξής αντικείμενα:

- Syslog. Το αντικείμενο αυτό αντιστοιχεί σε μία γραμμή απο τα Syslog logs.
- Pam. Το αντικείμενο αυτό αντιστοιχεί σε μία γραμμή απο τα Pam logs.
- Utmp. Το αντικείμενο αυτό αντιστοιχεί σε μία εγγραφή απο τα UTMP logs.

Name	Description	Value
host	The Host To Connect	localhost
port	The Port To Connect To	22
userName	The Username To connect To	pat
password	The Password To connect To	password
endLineChar	The End Of Line Character	/n
exitString	The Exit String	logout
prompt	The Prompt Of The Terminal	$
numberOfConnections	Number of concurrent connections	4

Πίνακας 4.3: The SSH Protocol

Name	Description	Value
host	The Host To Connect	localhost
port	The Port To Connect To	8380
minThreads	The minimum number To use in ThreadPool	2
maxThreads	The maximum number To use in ThreadPool	5
currency	The currency Time Limit	100
delay	The delay to load A page	1

Πίνακας 4.4: The HTTP Protocol

4.2.4 **HTTP Server**

Το HTTP [14] είναι ένα πρωτόκολλο το οποίο δίνει την δυνατότητα μεταφοράς HyperText μεταξύ υπολογιστών.Στην υλοποίηση μας τα απαραίτητα attributes πού απαιτούνται για την υλοποίηση του φαίνονται στον πίνακα 4.2.4.

Στο Template του πρωτοκολλου HTTP περιέχει και ένα αντικείμενο τύπου HTTPServer με το όνομα server. Οι μεθόδοι του αντικειμένου αυτού δίνονται στο Παράρτημα Β′.

4.3 **Facets**

4.3.1 Κοινά σημεία μεταξύ τών **Facets**

Το σύστημα δίνει την δυνατότητα στον χρήστη να γεννήσει πέρα απο τον Collector και όσα Facets θέλει τα οποία θα κάνουν Subscribe σε κάποιο Topic και ανάλογα θα διαχειρίζονται το κάθε Event.Στην υλοποίηση μας έχουμε κατασκευάσει τα εξής Facets:

Name	Description	Ocurrence	Value
name	The Name Of The Class	1	FOO
package-name	The Package Name	1	gr.tuc.softnet
facet-generator	The Generators Name	1	SyslogFacet
action	The action type to be used	1	JMS
action-property	The properties of the action	unbounded	name=topic/1
facet-code	The Code That Takes Care Of the Events	1	;
facet-helper-class	Helper Class for Complex Operations	1	

Πίνακας 4.5: The Facet Information

- SMTPFacet. Το Facet αυτο σου δίνει την δυνατότητα να στέλνεις Email τα events.
- SyslogFacet. Το Facet αυτο σου δίνει την δυνατότητα να γράφεις στο Syslog τα events.
- SwingFacet. Το Facet αυτο σου δίνει την δυνατότητα να γράφεις σε ένα Swing GUI τα events.

Επιπλέον πληροφορία για το κάθε Facet ξεχωριστά δίνεται στις παρακάτω ενότητες. Για την δημιουργία του κάθε Facet χρειάζεται η πληροφορία που φαίνεται στον πίνακα 4.3.1.

4.3.2 SMTPFacet

Το Facet αυτο σου δίνει την δυνατότητα να στέλνεις Email τα events. Στο Template του Facet SMTP περιέχεται ένα αντικείμενο τύπου SMTPClientConfig με το όνομα config όπως επίσης και ένα αντικείμενο τύπου SMTPClient. Το κάθε Event αντιστοιχεί σε ένα αντικείμενο τύπου HashMap με όνομα map. Οι μεθόδοι του αντικειμένων αυτών δίνονται στο Παράρτημα Β′.

4.3.3 SyslogFacet

Το Facet αυτο σου δίνει την δυνατότητα να γράφεις στο Syslog τα events. Στο Template του Facet Syslog περιέχεται ένα αντικείμενο τύπου Logger με το όνομα logs . Το κάθε Event αντιστοιχεί σε ένα αντικείμενο τύπου HashMap με όνομα map. Οι μεθόδοι του αντικειμένων αυτών δίνονται στο Παράρτημα Β′.

Name	Description	Ocurrence	Value
name	The Name Of The Class	1	FOO
package-name	The Package Name	1	gr.tuc.softnet
generator-name	The Generators Name	1	SSHGen
collector-event	Protocol Specific Information	1	
collector-queue-out	HashMessage Attributes	unbounded	name=""
Action	The Action to be used	1	
action-type	The Action type(JMS or JMX)	1	JMS
action-connections	The number of open connections	1	2
action-property	The Action properties	unbounded	name="topic/1"
eca-rule	Informations for the rules	unbounded	
name	The name of the rule	1	SU
base-condition	The basic rules information	unbounded	
name	base-codition name	1	condition1
code	Base-condition code	1	return true;
condition	The code of the condition	1	return condition1();
action-type	The Action type(JMS or JMX)	1	JMS
action	The Action code	1	publish(event,"topic/1")
code	Code to collect Events	1	
helper-class	code for a helper class if needed	1	
facet-instance	Facet Specific Information	unbounded	

Πίνακας 4.6: The Collector Information

4.3.4 **SwingFacet**

Το Facet αυτο σου δίνει την δυνατότητα να γράφεις σε ένα Swing GUI τα events.Στο Template του Facet Swing περιέχεται μία private method με το όνομα setMessage(String s) .Το κάθε Event αντιστοιχεί σε ένα αντικείμενο τύπου HashMap με όνομα map.

4.4 Επιπλέον Για τους **Collectors**

4.4.1 Κοινά σημεία μεταξύ τών **Collector**

Όπως αναφέραμε και παραπάνω, για την γέννηση του κάθε collector απαιτείται συγκεκριμένη πληροφορία ανάλογα με το πρωτόκολλο που θέλουμε να χρησιμοποιήσουμε. Επιπλέον όμως απαιτείται και κοινή πληροφορία για το κάθε template που θέλουμε να χρησιμοποιήσουμε. Αυτή φαίνεται στον πίνακα 4.4.1.

Κεφάλαιο 5

Μελλοντική Δουλειά - Συμπεράσματα

5.1 Μελλοντική Δουλειά - Συμπεράσματα

5.1.1 Συμπεράσματα

Στην εργασία αυτή κατασκευάσαμε ένα Framework το οποίο μπορούν εύκολα να χρησιμοποιήσουν προγγραματιστές για να αναπτύξουν εφαρμογές , οι οποίες έχουν την ανάγκη να χρησιμοποιούν event listeners πάνω απο πρωτόκολλα , όπως για παράδειγμα το SSH. Για να το πετύχουμε αυτό χρησιμοποιήσαμε τεχνικές γεννητικού κώδικα , οι οποίες δίνουν την δυνατότητα στους χρήστες να παραμετροποιούν τα components τους και να τα διαμορφώνουν όπως αυτοί επιθυμούν. Τέτοιες εφαρμογές μπορεί να είναι απο εφαρμογές χρεώσεων Billing Mediation που βρίσκουν εφαρμογή σε εταιρείες τηλεφωνίας μέχρι και εφαρμογές Bussiness Logic (Workflow). Επίσης το FrameWork αυτό θα μπορούσε να χρησιμοποιείθεί και απο εταιρείες παροχής υπηρεσιών αφου δίνουμαι την δυνατότητα κατασκεύης πολύπλοκών Component όπως HTTP με πολύ εύκολη παραμετροποίηση όπως επίσης δίνοντας την δυνατότητα να παρακολουθείται η δραστηριότητα των component αυτών.

5.1.2 Μελλοντική Δουλειά

Σκοπός μας είναι για την ολοκλήρωση του Framework να κατασκευαστούν τα παρακάτω.

- Προσθήκη και άλλων collector όπως HTTP Client, SSH Server, FTP Server κτλ.
- Προσθήκη και άλλων Facet όπως JDBC, CJDBC Facet.

- Προσθήκη λογικής DO-UNDO στους Collectors.
- Κατασκευή Framework για Actions δηλαδή Adaptors αντί για Collectors και επικοινωνία μεταξύ τούς.
- Μετατροπή των events σε Workflow Activity.
- Παροχή υπηρεσιών Collector σε Grid Services μέσω του Globus.

Παράρτημα Α΄

Τα **XSD** της εφαρμογής

CollectorDesigner XSD

```
1
2  <?xml version="1.0"?>
3
4  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
5  <annotation>
6    <documentation>
7    Collector Pack instance.The elements are:
8      name: where a valid name is collectors
9         generated class name.
10     generator-name: where a valid generator-name
11        is the name of generator directory.
12     description: The description of the collector.
13     package: The package name
14
15   The Collector-Pack has references to :
16     collector-event.
17     collector-queue-out.
18     collector-instance.
19     facet-instance.
20   </documentation>
21 </annotation>
22   <xsd:element name="collector-pack">
23     <xsd:complexType>
24       <xsd:sequence>
25         <xsd:element name="name"
26     type="xsd:string" use="required"/>
27         <xsd:element name="generator-name"
28     type="xsd:string" use="required"/>
```

```
29          <xsd:element name="description"
30        type="xsd:string" use="optional"/>
31          <xsd:element name="package"
32        type="xsd:string" use="required"/>
33          <xsd:element  ref="collector-event"
34         minOccurs="0" maxOccurs="unbounded"/>
35          <xsd:element  ref="collector-queue-out"
36        minOccurs="0" maxOccurs="unbounded"/>
37          <xsd:element  ref="collector-instance"
38        minOccurs="0" maxOccurs="unbounded"/>
39    <xsd:element ref="facet-instance"
40        minOccurs="1" maxOccurs="unbounded"/>
41
42      <xsd:element ref="eca-rule"
43        minOccurs="1" maxOccurs="unbounded"/>
44          </xsd:sequence>
45      </xsd:complexType>
46    </xsd:element>
47  <annotation>
48    <documentation>
49    Collector Event instance.The elements are.
50      name: where a valid name is a name
51        for collector usage like pam\_unix.
52      description: The description of the
53          Collector-Event
54
55    The Collector-Event has references to :
56      event-property.
57    </documentation>
58  </annotation>
59    <xsd:element name="collector-event">
60      <xsd:complexType>
61       <xsd:sequence>
62        <xsd:element name="name"
63      type="xsd:string" use="required"/>
64        <xsd:element name="description"
65       type="xsd:string"  use="optional"/>
66        <xsd:element  ref="event-property"
67        minOccurs="0" maxOccurs="unbounded"/>
68       </xsd:sequence>
69      </xsd:complexType>
70    </xsd:element>
71
```

```
<annotation>
  <documentation>
  Event-Property instance.The attributes are.
    name: The name of the property like host.
    description: The description
      of the property.
    value: The value of the
      property like localhost.
  </documentation>
</annotation>
 <xsd:element name="event-property">
    <xsd:complexType>
      <xsd:attribute name="name"
     type="xsd:string" use="required"/>
      <xsd:attribute name="description"
     type="xsd:string"  use="optional"/>
      <xsd:attribute name="value"
    type="xsd:string" use="optional"/>
    </xsd:complexType>
 </xsd:element>

<annotation>
  <documentation>
  Collector-queue-out instance describes
    the MapMessage to be sent
  </documentation>
</annotation>

<xsd:element name="collector-queue-out">
    <xsd:complexType>
      <xsd:attribute name="name"
    type="xsd:string" use="required"/>
      <xsd:attribute name="description"
    type="xsd:string"  use="optional"/>
    </xsd:complexType>
  </xsd:element>

<xsd:element name="collector-instance">
    <xsd:complexType>
      <xsd:sequence>
          <xsd:element ref="action"
     minOccurs="1" maxOccurs="unbounded"/>
          <xsd:element name="code"
```

```
115       type="xsd:string"  use="required"/>
116             <xsd:element name="helper-class"
117        type="xsd:string" use="optional"/>
118         </xsd:sequence>
119       </xsd:complexType>
120    </xsd:element>
121
122 <xsd:element name="action">
123      <xsd:complexType>
124        <xsd:sequence>
125    <xsd:element name="action-type"
126        type="xsd:string"  use="required"/>
127    <xsd:element name="action-connections"
128        type="xsd:string"  use="required"/>
129    <xsd:element ref="action-property"
130        minOccurs="1" maxOccurs="unbounded"/>
131        </xsd:sequence>
132      </xsd:complexType>
133 </xsd:element>
134
135 <xsd:element name="action-property">
136       <xsd:complexType>
137         <xsd:attribute name="name"
138       type="xsd:string" use="required"/>
139         <xsd:attribute name="description"
140       type="xsd:string"  use="optional"/>
141         <xsd:attribute name="value"
142        type="xsd:string" use="optional"/>
143       </xsd:complexType>
144  </xsd:element>
145
146
147 <xsd:element name="eca-rule">
148    <xsd:complexType>
149             <xsd:sequence>
150       <xsd:element name="name"
151        type="xsd:string" use="required"/>
152       <xsd:element name="action"
153       type="xsd:string" use="required"/>
154       <xsd:element name="action-type"
155       type="xsd:string" use="required"/>
156       <xsd:element name="condition"
157        type="xsd:string" use="required"/>
```

```
158      <xsd:element ref="base-condition"
159      minOccurs="0" maxOccurs="unbounded"/>
160            </xsd:sequence>
161          </xsd:complexType>
162 </xsd:element>
163 <xsd:element name="base-condition">
164    <xsd:complexType>
165   <xsd:sequence>
166       <xsd:element name="name"
167      type="xsd:string" use="required"/>
168   <xsd:element name="code"
169      type="xsd:string"  use="required"/>
170   </xsd:sequence>
171    </xsd:complexType>
172 </xsd:element>
173
174
175   <xsd:element name="facet-instance">
176     <xsd:complexType>
177       <xsd:sequence>
178       <xsd:element name="name"
179     type="xsd:string" use="required"/>
180       <xsd:element name="package-name"
181     type="xsd:string" use="required"/>
182       <xsd:element name="facet-generator"
183      type="xsd:string" use="required"/>
184       <xsd:element name="action"
185      type="xsd:string"  use="required"/>
186             <xsd:element ref="action-property"
187      minOccurs="0" maxOccurs="unbounded"/>
188       <xsd:element name="facet-code"
189      type="xsd:string"  use="required"/>
190       <xsd:element name="facet-helper-class"
191      type="xsd:string" use="optional"/>
192     </xsd:sequence>
193   </xsd:complexType>
194    </xsd:element>
195
196
197 </xsd:schema>
```

CollectorGeneratorInformation XSD

```
1
2  <?xml version="1.0"?>
3
4  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
5
6    <xsd:element name="collector-generator-information">
7      <xsd:complexType>
8        <xsd:sequence>
9          <xsd:element name="name"
10       type="xsd:string" use="required"/>
11         <xsd:element name="description"
12       type="xsd:string" use="required"/>
13         <xsd:element ref="per-collector-event"
14          minOccurs="0" maxOccurs="unbounded"/>
15   <xsd:element name="queue-name"
16        type="xsd:string" use="required"/>
17   <xsd:element name="code"
18       type="xsd:string" use="required"/>
19   <xsd:element name="helper-class"
20        type="xsd:string" use="optional"/>
21       </xsd:sequence>
22     </xsd:complexType>
23   </xsd:element>
24
25   <xsd:element name="per-collector-event">
26     <xsd:complexType>
27       <xsd:sequence>
28               <xsd:element name="name"
29      type="xsd:string" use="required"/>
30               <xsd:element name="description"
31       type="xsd:string"  use="optional"/>
32               <xsd:element ref="per-event-property"
33      minOccurs="0" maxOccurs="unbounded"/>
34       </xsd:sequence>
35     </xsd:complexType>
36   </xsd:element>
37
38   <xsd:element name="per-event-property">
39     <xsd:complexType>
40       <xsd:attribute name="name"
41       type="xsd:string" use="required"/>
42       <xsd:attribute name="description"
43      type="xsd:string"  use="optional"/>
```

```
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

ConnectorInformation XSD

```

<?xml version="1.0"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="connector-information">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name"
     type="xsd:string" use="required"/>
        <xsd:element name="description"
    type="xsd:string" use="required"/>
        <xsd:element ref="per-connector-property"
        minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="per-connector-property">
    <xsd:complexType>
      <xsd:attribute name="name"
    type="xsd:string" use="required"/>
      <xsd:attribute name="description"
     type="xsd:string"  use="optional"/>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Παράρτημα Β′

Τα **JAVADOC** της εφαρμογής

Βιβλιογραφία

[1] Java Management Extensions White Paper: Dynamic Management for the Service Age http://java.sun.com/products/JavaManagement,1999

[2] JBoss Open Source Application Server,http://www.jboss.org

[3] P. Th. Eugster et al., The Many Faces of Publish/Subscribe ,ACM Computing Surveys,Vol. 35, Issue 2, June 2003.

[4] R.S. Silva Filho,D.F. Redmiles, A Survey of Versability for Publish/Subscribe Insfactures, May 2005

[5] Wieland Schwinger, Logical Events In ECA Rules, Master Thesis University of Skovde Sweden, November 1995

[6] M.Berndtsson,B.Lings:"Logical Events and ECA-Rules",Technical Report HS-IDA-TR-95-004,Dept of Computer Science,University of Skovde,Sweden July 1995.

[7] The Jakarta Struts Framework , http://struts.apache.org

[8] The Jakarta Velocity Template Engine , http://jakarta.apache.org/velocity/

[9] XDoclet Open Source Code Generator , http://xdoclet.sourceforge.net/

[10] Castor Open Source XML Project , http://www.castor.org/

[11] The Telnet Protocol RFC, http://www.faqs.org/rfcs/rfc854.html ,May 1983

[12] The FTP Protocol RFC, http://www.faqs.org/rfcs/rfc959.html ,October 1985

[13] The SSH-2 Protocol RFC, http://ietf.org/html.charters/secsh-charter.html , May 2002

[14] The HTTP Protocol RFC , http://www.faqs.org/rfcs/rfc2616.html , June 1999

[15] Jetty Java HTTP Server, http://jetty.mortbay.org/jetty/