ΠΟΛΥΜΟΡΦΙΣΜΟΣ ΓΙΑ ΥΠΗΡΕΣΙΕΣ ΔΙΑΔΙΚΤΥΟΥ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

Ηλία Μάκη

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Φεβρουάριος 2013

ΑΦΙΕΡΩΣΗ

Αφιερώνω αυτή την εργασία στην οικογένειά μου για την υποστήριξη τους όλα αυτά τα χρόνια που διήρκησαν οι σπουδές μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Με το πέρας αυτής της διατριβής, θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα, κ. Απόστολο Ζάρρα, Επίκουρο Καθηγητή του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων, για την εποικοδομητική και αρμονική συνεργασία, καθώς και για το ενδιαφέρον που υπέδειξε, καθ' όλη την διάρκεια της εκπόνησής της. Επίσης θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Διονύση Αθανασόπουλο για την καθοδήγηση του και τις συμβουλές στις δύσκολες στιγμές της εργασίας. Τέλος θα ήθελα να ευχαριστήσω τον κ. Παπαπέτρου Ευάγγελο και τον κ. Βασιλειάδη Παναγιώτη, Επίκουρους Καθηγητές του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων και μέλη της εξεταστικής επιτροπής.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Ηλίας Μάκης του Γεωργίου και της Μαρίας – Φωτεινής. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Φεβρουάριος, 2013, Πολυμορφισμός Για Υπηρεσίες Διαδικτύου. Επιβλέποντας: Απόστολος Ζάρρας.

Στο συμβατικό αντικειμενοστρεφές (ΟΟ) λογισμικό, η ανάπτυξη λογισμικού που μπορεί να είναι εύκολα επεκτάσιμο/προσαρμόσιμο βασίζεται σε δύο έννοιες κλειδιά: την αφαίρεση και τον πολυμορφισμό. Πιο συγκεκριμένα, αν θέλουμε εύκολα να ενσωματώσουμε μια σειρά από εναλλακτικές σχεδιαστικές επιλογές στο λογισμικό ορίζουμε μια αφηρημένη κλάση που αντιπροσωπεύει αυτές τις επιλογές. Στην συνέχεια, αναπτύσσουμε διαφορετικές πολυμορφικές υλοποιήσεις που επαναορίζουν (override) τις μεθόδους της αφηρημένης κλάσης. Το τελευταίο βήμα είναι να αναπτύξουμε το υπόλοιπο του λογισμικού σε σχέση με την αφηρημένη κλάση. Η εξέλιξη από το αντικειμενοστρεφές λογισμικό στο προσανατολισμένο στις υπηρεσίες λογισμικό ανέδειξε τις υπηρεσίες ως μια πολλά υποσχόμενη λύση για τη γρήγορη και χαμηλού κόστους ανάπτυξη εφαρμογών λογισμικού που ενσωματώνουν ανεξάρτητες, και επαναχρησιμοποιήσιμες λειτουργίες, που σχεδιάστηκαν, υλοποιήθηκαν και συντηρούνται από τρίτους. Κατ’ αναλογία με ότι συμβαίνει στο συμβατικό αντικειμενοστρεφές λογισμικό, παρόμοιες υπηρεσίες μπορούν να θεωρηθούν ως εναλλακτικές επιλογές σχεδιασμού που μπορούν να χρησιμοποιηθούν για την ανάπτυξη επεκτάσιμου & προσαρμόσιμου λογισμικού προσανατολισμένο στις υπηρεσίες. Δυστυχώς, στο πεδίο των υπηρεσιών η αξιοποίηση αυτών των εναλλακτικών επιλογών για την κατασκευή επεκτάσιμου και προσαρμόσιμου λογισμικού προσανατολισμένο στις υπηρεσίες δεν είναι τόσο εύκολη όσο και στην περίπτωση του αντικειμενοστραφούς λογισμικού. Τα δύο κύρια προβλήματα είναι: (1) δεν έχουμε κανένα μέσο το οποίο θα βοηθήσει τους προγραμματιστές να αναπτύξουν αφηρημένες υπηρεσίες από ένα σύνολο υπηρεσιών που είναι ήδη διαθέσιμες μέσω του διαδικτύου και (2) δεν έχουμε μηχανισμούς πολυμορφισμού υπηρεσιών οι οποίοι θα επιτρέψουν την σύνδεση ανάμεσα σε αφηρημένες υπηρεσίες

και συγκεκριμένες υπηρεσίες που παίζουν το ρόλο των εναλλακτικών υλοποιήσεων της αφηρημένης υπηρεσίας.

Αυτή η διατριβή επεκτείνει μια υπάρχουσα τεχνική που επιλύει το πρώτο πρόβλημα με ένα μηχανισμό πολυμορφισμού για Υπηρεσίες Διαδικτύου ο οποίος επιλύει το δεύτερο πρόβλημα. Ο προτεινόμενος μηχανισμός δέχεται σαν είσοδο αντιστοιχίσεις ανάμεσα στη διεπαφή μιας αφηρημένης υπηρεσίας και των υπηρεσιών που εκπροσωπούνται από αυτή και με βάση αυτές τις αντιστοιχίσεις μεταφράζει δυναμικά τις κλήσεις που γίνονται με βάση την διεπαφή της αφηρημένης υπηρεσίας σε κλήσεις αντίστοιχων λειτουργιών που προσφέρονται από τις συγκεκριμένες υπηρεσίες που εκπροσωπούνται από την αφηρημένη υπηρεσία.

EXTENDED ABSTRACT IN ENGLISH

Makis, Hlias, MSc, Computer Science Department, University of Ioannina, Greece. February, 2013. Polymorphism For Web Services. Thesis Supervisor: Apostolos Zarras.

In conventional Object – Oriented (OO) software, the development of software that can be easily extended/adapte dis based on two key concepts: abstractions and polymorphism. More specifically, if we want to easily incorporate a number of alternative design options in the software we typically define an abstract class that represents these options. Then, we develop different polymorphic implementations that override the methods of this abstract class. The last step is to develop the rest of the software with respect to the abstract class. The evolution from object – oriented to service – oriented software brought out services as a promising solution to the rapid, low – cost development of software applications that integrate independent, reusable functionalities, designed, implemented and maintained by third parties. In analogy to what happens in conventional object – oriented software, similar services can be seen as alternative design options that can be used to develop extensible & adaptable service – oriented software. Unfortunately, in the service – oriented world exploiting these alternative options towards the construction of extensible & adaptable service oriented software is not as easy as in the case of object – oriented software. The two main problems are: (1) we have no means that would help the developers to develop service abstractions out of a set of existing concrete services that are already available through the Web and (2) we have no polymorphism mechanisms that would allow to bind service abstractions to concrete services. This thesis extends an existing technique that solves the first problem with a polymorphism mechanism for Web Services which solves the second problem. The proposed mechanism takes as input mappings between an abstract service interface and a service represented by it and based on these mappings dynamically translates calls made on the basis of abstract service interface to calls to respective functions offered by these services represented by the abstract service.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Επεκτάσιμο & Προσαρμόσιμο Λογισμικό

Σε γενικές γραμμές, όταν αναπτύσσουμε λογισμικό μπορεί να υπάρχουν πολλές διαφορετικές σχεδιαστικές επιλογές που θα μπορούσαμε να ακολουθήσουμε για ορισμένα τμήματα του λογισμικού ή διαφορετικούς αλγορίθμους ταξινόμησης. Επιπλέον, έχουμε πολλούς διαφορετικούς αλγορίθμους αναζήτησης. Επίσης μπορεί να έχουμε πολλούς διαφορετικούς εναλλακτικούς μηχανισμούς αποθήκευσης που έχουν διαφορετικά χαρακτηριστικά. Έχουμε πολλά μοντέλα συναλλαγών, καθένα με τα πλεονεκτήματα του και τα μειονεκτήματα του. Θα μπορούσαμε να αναφερθούμε σε πολλά τέτοια παραδείγματα πολλαπλών εναλλακτικών επιλογών σχεδιασμού για το ίδιο πρόβλημα· καθένα με τα πλεονεκτήματα και τα μειονεκτήματα του. Σε τέτοιες περιπτώσεις θέλουμε να αναπτύξουμε λογισμικό τέτοιο ώστε να μπορεί εύκολα να προσαρμοστεί/επεκταθεί – ενδεχομένως κατά τον χρόνο της εκτέλεσης – για να χρησιμοποιεί μια επιλογή στην θέση μιας άλλης.

Στον συμβατικό αντικειμενοστρεφές (ΟΟ) λογισμικό, η ανάπτυξη του λογισμικού που μπορεί να είναι εύκολα επεκτάσιμο/προσαρμόσιμο βασίζεται σε δύο έννοιες κλειδιά: την αφαίρεση και τον πολυμορφισμό. Πιο συγκεκριμένα, αν θέλουμε εύκολα να ενσωματώσουμε μια σειρά από εναλλακτικές σχεδιαστικές επιλογές στο λογισμικό ορίζουμε μια αφηρημένη κλάση που αντιπροσωπεύει αυτές τις επιλογές. Στην συνέχεια, αναπτύσσουμε διαφορετικές πολυμορφικές υλοποιήσεις που επαναορίζουν (override) τις μεθόδους της αφηρημένης κλάσης. Το τελευταίο βήμα

είναι να αναπτύξουμε το υπόλοιπο του λογισμικού σε σχέση με την αφηρημένη κλάση. Αυτό μπορεί να γίνει, για παράδειγμα, με παραμετροποίηση των κλάσεων του υπόλοιπου λογισμικού με αναφορές της αφηρημένης κλάσης σε αντικείμενα. Τότε, ενώ εκτελείται το λογισμικό μπορούμε να αρχικοποιήσουμε τα αντικείμενα των κλάσεων περνώντας ως παράμετρο αντικείμενα των συγκεκριμένων υλοποιήσεων που υλοποιούν την αφηρημένη κλάση. Η κλήση μιας μεθόδου βασίζεται στον πολυμορφισμό, δηλ. η σωστή μέθοδος καλείται με βάση την κλάση του αντικειμένου στο οποίο αναφέρεται η αναφορά της αφηρημένης κλάσης. Επιπλέον, κατά τον χρόνο της εκτέλεσης μπορούμε να αλλάξουμε ένα αντικείμενο το οποίο ανήκει σε μια κλάση που υλοποιεί την αφηρημένη κλάση, με ένα αντικείμενο μιας άλλης κλάσης που υλοποιεί την αφηρημένη κλάση.

1.2. Αφηρημένες Υπηρεσίες & Πολυμορφισμός

Η εξέλιξη από το αντικειμενοστρεφές λογισμικό στο προσανατολισμένο στις υπηρεσίες λογισμικό ανέδειξε τις υπηρεσίες ως μια πολλά υποσχόμενη λύση για τη γρήγορη και χαμηλού κόστους ανάπτυξη εφαρμογών λογισμικού που ενσωματώνουν ανεξάρτητες, και επαναχρησιμοποιήσιμες λειτουργίες, που σχεδιάστηκαν, υλοποιήθηκαν και συντηρούνται από τρίτους [30, 13]. Οι υπηρεσίες προσφέρουν τις υπάρχουσες λειτουργίες μέσω προγραμματιζόμενων διεπαφών (web service interface). Στη συνέχεια, η ανάπτυξη λογισμικού προσανατολισμένου στις υπηρεσίες αποτελείται από μια σύνθεση υπηρεσιών, που υλοποιούνται μέσω κλήσεων στις λειτουργίες της υπηρεσίας. Από τεχνικής άποψης, η τεχνολογία υπηρεσιών διαδικτύου επιτρέπει την προσφορά υπηρεσιών οι οποίες είναι προσβάσιμες μέσω της υποδομής του διαδικτύου (web services) [31].

Υπάρχουν πολλά δημόσια μητρώα υπηρεσιών (π.χ., ServiceFinder, WebServiceList, RemoteMethods), τα οποία περιέχουν πληροφορίες σχετικά με μεγάλο αριθμό δημοσιευμένων υπηρεσιών. Ανάμεσα στις υπηρεσίες που δημοσιεύονται είναι δυνατό να βρεθούν σύνολα από υπηρεσίες που προσφέρουν παρόμοια λειτουργικότητα, μέσω διαφορετικών προγραμματιστικών διεπαφών. Οι διεπαφές των παρόμοιων υπηρεσιών μπορεί να διαφέρουν λίγο, ή αρκετά, ακόμα και σε περιπτώσεις υπηρεσιών που προέρχονται από τον ίδιο πάροχο.

Λαμβάνοντας ένα πραγματικό παράδειγμα, η Amazon αποτελεί σημαντικό φορέα παροχής υπηρεσιών που προσφέρει μια ποικιλία από υπηρεσίες μέσω του διαδικτύου. Ανάμεσα στις υπηρεσίες αυτές, η Amazon Simple Queue Service (SQS) επιτρέπει την επικοινωνία μέσω ουρών μηνυμάτων. Από το 2006, έχουν υπάρξει διάφορες παρόμοιες εκδόσεις της υπηρεσίας SQS. Όλες αυτές οι διαφορετικές εκδόσεις παρέχουν 2 διεπαφές. Η πρώτη διεπαφή διαφέρει ελαφρά από την μια έκδοση στην άλλη όσον αναφορά τις εισόδους/εξόδους. Σχετικά με την δεύτερη διεπαφή, οι διαφορές είναι περισσότερες μεταξύ των διαφόρων εκδόσεων· ένα σύνολο από λειτουργίες έχουν προστεθεί, αφαιρεθεί, ή μετονομαστεί.

Παρόμοιες υπηρεσίες μπορούν να χαρακτηρίζονται από διαφορετικές μη-λειτουργικές ιδιότητες, όπως το κόστος, ο χρόνος απόκρισης, η διαθεσιμότητα, και η αξιοπιστία. Στο παράδειγμά μας, οι διαφορετικές SQS εκδόσεις υποθέτουν διαφορετικές πολιτικές τιμολόγησης. Συγκεκριμένα, στις εκδόσεις του 2006 και 2007 οι χρήστες χρεώνονται για κάθε μήνυμα, που αποθηκεύεται στην ουρά του SQS. Αφ' ετέρου, στις εκδόσεις του 2008 και 2009 οι χρήστες χρεώνονται ανά αίτηση στην υπηρεσία SQS.

Κατ' αναλογία με ότι συμβαίνει στο συμβατικό αντικειμενοστρεφές λογισμικό, παρόμοιες υπηρεσίες μπορούν να θεωρηθούν ως εναλλακτικές επιλογές σχεδιασμού που μπορούν να χρησιμοποιηθούν για την ανάπτυξη επεκτάσιμου & προσαρμόσιμου λογισμικού προσανατολισμένο στις υπηρεσίες. Δυστυχώς, στο πεδίο των υπηρεσιών η αξιοποίηση αυτών των εναλλακτικών επιλογών για την κατασκευή επεκτάσιμου και προσαρμόσιμου λογισμικού προσανατολισμένο στις υπηρεσίες δεν είναι τόσο εύκολη όσο και στην περίπτωση του αντικειμενοστραφούς λογισμικού. Τα δύο κύρια προβλήματα είναι:

1. Δεν έχουμε κανένα μέσο το οποίο θα βοηθήσει τους προγραμματιστές να αναπτύξουν αφηρημένες υπηρεσίες από ένα σύνολο υπηρεσιών που είναι ήδη διαθέσιμες μέσω του διαδικτύου.

2. Δεν έχουμε μηχανισμούς πολυμορφισμού υπηρεσιών οι οποίοι θα επιτρέψουν την σύνδεση ανάμεσα σε αφηρημένες υπηρεσίες και συγκεκριμένες υπηρεσίες που παίζουν το πόλο των εναλλακτικών υλοποιήσεων της αφηρημένης υπηρεσίας.

Για την αντιμετώπιση του πρώτου προβλήματος μια πρόσφατη προσπάθεια έχει γίνει από τους Athanasopoulos et. al. στο [5]. Σε αυτήν την εργασία, οι συγγραφείς ορίζουν φορμαλιστικά την αφηρημένη υπηρεσία ως μια έννοια που αντιπροσωπεύει ένα σύνολο υπηρεσιών που προσφέρουν παρόμοιες υπηρεσίες. Επιπλέον, μια αφηρημένη υπηρεσία χαρακτηρίζεται από μια αφηρημένη διεπαφή και αντιστοιχίσεις ανάμεσα σε αυτή την διεπαφή και στις προγραμματιστικές διεπαφές των υπηρεσιών που εκπροσωπούνται. Εν συνεχεία, οι συγγραφείς προτείνουν μια συστηματική προσέγγιση για την εξαγωγή αφηρημένων υπηρεσιών από υπηρεσίες που είναι διαθέσιμες σε όλο το διαδίκτυο. Η προτεινόμενη προσέγγιση βασίζεται σε έναν αλγόριθμο ιεραρχικής ομαδοποίησης.

Αυτή η διατριβή επεκτείνει την προαναφερθείσα εργασία προτείνοντας ένα μηχανισμό ο οποίος δέχεται σαν είσοδο αντιστοιχίσεις ανάμεσα στις διεπαφές της αφηρημένης υπηρεσίας και των υπηρεσιών που εκπροσωπούνται από αυτή και με βάση αυτές τις αντιστοιχίσεις μεταφράζει τις κλήσεις που γίνονται με βάση την αφηρημένη διεπαφή της αφηρημένης υπηρεσίας σε κλήσεις αντίστοιχων λειτουργιών που προσφέρονται από τις συγκεκριμένες υπηρεσίες που εκπροσωπούνται από την αφηρημένη υπηρεσία.

1.3. Δομή της Διατριβής

Η παρούσα διατριβή αποτελείται από συνολικά, έξι (6) Κεφάλαια, από τα οποία το πρώτο είναι η Εισαγωγή. Το Κεφάλαιο 2 παρουσιάζονται οι σχετικές εργασίες. Συγκεκριμένα παρουσιάζονται οι διάφορες τεχνικές που μπορούν να χρησιμοποιηθούν για την ανάπτυξη επεκτάσιμου και προσαρμόσιμου λογισμικού προσανατολισμένου στις υπηρεσίες καθώς και η συνεισφορά της παρούσας διατριβής σε κάθε μια από τις τεχνικές αυτές. Στο Κεφάλαιο 3 γίνεται η μοντελοποίηση των

αφηρημένων υπηρεσιών και δίνεται και ο αλγόριθμος που χρησιμοποιείται για την εξόρυξη αφηρημένων υπηρεσιών. Στο Κεφάλαιο 4 γίνεται αναφορά στον πολυμορφισμό υπηρεσιών. Πιο συγκεκριμένα παρουσιάζεται ο μηχανισμός παραγωγής πολυμορφικών υπηρεσιών, ο μηχανισμός πολυμορφισμού και τέλος παρουσιάζεται ένας μηχανισμός αναπροσαρμογής. Στο Κεφάλαιο 5 παρουσιάζεται η πειραματική αξιολόγηση του προτεινόμενου μηχανισμού πολυμορφισμού καθώς και η πειραματική αξιολόγηση του μηχανισμού αναπροσαρμογής. Τέλος στο Κεφάλαιο 6 συνοψίζονται τα συμπεράσματα για τον προτεινόμενο μηχανισμό πολυμορφισμού.

ΚΕΦΑΛΑΙΟ 2. ΣΧΕΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

2.1. Προσαρμόσιμο & Επεκτάσιμο Λογισμικό Προσανατολισμένο στις Υπηρεσίες

Η ανάπτυξη επεκτάσιμου και προσαρμόσιμου λογισμικού προσανατολισμένου στις υπηρεσίες, σε γενικές γραμμές, είναι ένα πολύ ενεργό ερευνητικό θέμα τα τελευταία χρόνια. Πρόσφατα, στην εργασία [6] παρέχει μια επισκόπηση των διαφόρων τεχνικών που μπορούν να χρησιμοποιηθούν για αυτό το σκοπό. Οι αναφερόμενες τεχνικές μπορούν να συνδυαστούν για να χειριστούν μεταβολές στις απαιτήσεις και/ή στις ιδιότητες των χρησιμοποιούμενων υπηρεσιών· οι τεχνικές αυτές κυμαίνονται από πολύ απλές μέχρι και πολύ προηγμένες. Οι προηγμένες τεχνικές που αναλύονται στο [6] περιλαμβάνουν τις ακόλουθες κατηγορίες:

- Service substitution: οι τεχνικές αυτές έχουν στόχο την αντικατάσταση μιας υπηρεσίας με μια άλλη.
- Re-composition: οι τεχνικές αυτές στοχεύουν στη συνολική αναδιοργάνωση του συστήματος που είναι προσανατολισμένο στις υπηρεσίες.
- Re-negotiation: οι τεχνικές αυτές στοχεύουν στην εκ νέου διαπραγμάτευση των συμφωνιών που γίνονται με τις υπηρεσίες σχετικά με τη ποιότητα που προσφέρεται από αυτές.

- Re-execution: στην περίπτωση αυτή, το σύστημα επανέρχεται σε μια προηγούμενη σταθερή κατάσταση και ορισμένες διαδικασίες (π.χ., κλήση υπηρεσιών) εκτελούνται ξανά.
- Compensation: οι τεχνικές αυτές στοχεύουν στην αναίρεση των επιπτώσεων ορισμένων διαδικασιών (π.χ., κλήσεις υπηρεσιών) που έλαβαν χώρα πριν από την αντικατάσταση υπηρεσιών.

Στην συνέχεια, θα συζητήσουμε με περισσότερες λεπτομέρειες κάθε μια από τις προαναφερθείσες κατηγορίες και την συμβολή αυτής της διατριβής σε σχέση με αυτές τις τεχνικές.

2.2. Αντικατάσταση Υπηρεσιών

Η αντικατάσταση υπηρεσιών σε μεγάλο βαθμό στηρίζεται στο θεμελιώδη σχεδιαστικό πρότυπο Adaptor [17]. Η βασική ιδέα είναι να οριστεί αντιστοίχιση μεταξύ των λειτουργιών της υπηρεσίας που θα πρέπει να αντικατασταθεί και των λειτουργιών της υπηρεσίας που θα την αντικαταστήσει, η οποία προσφέρει λειτουργίες μέσα από μια διαφορετική διεπαφή. Με βάση μια τέτοια αντιστοίχιση, ένας προσαρμογέας δημιουργείται, ο οποίος επιτρέπει την πρόσβαση στην λειτουργικότητα της υπηρεσίας που έχει αντικαταστήσει την αρχική, μέσω της διεπαφής της αρχικής υπηρεσίας. Με βάση αυτήν την ιδέα, οι τρέχουσες προσεγγίσεις μπορούν να χωριστούν σε δύο κατηγορίες.

Στην πρώτη κατηγορία, έχουμε προσεγγίσεις που αναλύουν τις διεπαφές ανάμεσα στην αρχική υπηρεσία και αυτήν με την οποία θα γίνει η αντικατάσταση, για την εξεύρεση ασυμβατοτήτων και αντιστοιχιών ανάμεσα στις λειτουργιές αυτών των διεπαφών. Στην συνέχεια, με βάση αυτήν την ανάλυση οι απαραίτητοι προσαρμογείς παράγονται αυτόματα ή ημιαυτόματα. Στο [29], για παράδειγμα, οι συγγραφείς προτείνουν μια λύση για την ημι - αυτοματοποιημένη παραγωγή κατά ζεύγη αντιστοιχιών ανάμεσα στην αρχική υπηρεσία και την υπηρεσία που θα την αντικαταστήσει. Η προτεινόμενη τεχνική βασίζεται στην παραδοχή ότι οι διεπαφές της αρχικής υπηρεσίας και αυτής που θα την αντικαταστήσει προέρχονται από την

ίδια δημοφιλή ή τυποποιημένη διεπαφή. Στο [4], προτείνεται μια προσέγγιση η οποία αυτοματοποιεί την παραγωγή των αντιστοιχίσεων ανάμεσα στην αρχική υπηρεσία και την υπηρεσία που θα την αντικαταστήσει, χωρίς να κάνει καμία υπόθεση όσον αναφορά τις διεπαφές των εμπλεκόμενων υπηρεσιών. Επιπλέον, στα [20, 27, 28], το προτεινόμενο πλαίσιο παρέχει μηχανισμούς που επιτρέπουν την ανίχνευση τόσο συντακτικών ασυμβατοτήτων όσο και ασυμβατοτήτων στο πρωτόκολλο χρήσης των υπηρεσιών. Η επίλυση αυτών των ασυμβατοτήτων και η παραγωγή των προσαρμογέων καθοδηγείται από τους χρήστες. Η ανίχνευση ασυμβατότητας στο πρωτόκολλο υποστηρίζεται επίσης από την προσέγγιση που προτείνεται στα [9, 11]· σε αυτήν την περίπτωση οι τεχνικές ανίχνευσης είναι πλήρως αυτοματοποιημένες.

Στην δεύτερη κατηγορία, έχουμε προσεγγίσεις οι οποίες υποθέτουν την ύπαρξη των αντιστοιχίσεων μεταξύ της αρχικής και της υποκατάστατης υπηρεσίας. Με βάση αυτήν την υπόθεση, το ζήτημα είναι ότι για μια δεδομένη αρχική υπηρεσία μπορούν να υπάρχουν πολλές υποκατάστατες υπηρεσίες, που χαρακτηρίζονται από συγκεκριμένα ποιοτικά χαρακτηριστικά. Ο στόχος είναι η εύρεση αυτών των εναλλακτικών και η βέλτιστη επιλογή, σε σχέση με τις απαιτήσεις του συστήματος και τα ποιοτικά χαρακτηριστικά τα οποία χαρακτηρίζουν τις εναλλακτικές υπηρεσίες. Για παράδειγμα, οι [25] και [14] είναι παρόμοιες προσεγγίσεις. Το σύστημα που είναι προσανατολισμένο στις υπηρεσίες έχει υλοποιηθεί ως μια BPEL ενορχήστρωση, ενώ οι αντιστοιχήσεις ανάμεσα στις εναλλακτικές υπηρεσίες ορίζονται μέσω XSLT κανόνων μετασχηματισμού. Στο [2] οι συγγραφείς επίσης υποθέτουν BPEL ενορχηστρώσεις, ενώ οι αντιστοιχίσεις καθορίζονται στο πλαίσιο της περιγραφής της υπηρεσίας με SAWSDL, σε σχέση με μια κοινή οντολογία. Οι σημασιολογικές έννοιες που χαρακτηρίζουν τόσο την αρχική υπηρεσία όσο και αυτήν που την αντικαθιστά θα πρέπει να είναι ισοδύναμες. Στο [18] οι συγγραφείς επίσης υποθέτουν μια κοινή οντολογία για τις αντιστοιχίσεις μεταξύ των διεπαφών της αρχικής υπηρεσίας και αυτής που την αντικαθιστά. Τέλος, η [21] επίσης στηρίζεται σε μια κοινή οντολογία που χρησιμοποιείται για να καθορίσει τις αντιστοιχίσεις μεταξύ των διεπαφών των υπηρεσιών. Ωστόσο, η προτεινόμενη προσέγγιση εστιάζεται περισσότερο στην ανίχνευση των αλλαγών που προκαλούν την ανάγκη να προσαρμοστεί το σύστημα, από ότι στην τελική προσαρμογή του συστήματος.

Η προσέγγιση η οποία προτείνεται σε αυτή την διατριβή διαφέρει από τις προαναφερθείσες τεχνικές αντικατάστασης υπηρεσιών καθώς έχουμε να κάνουμε με κάτι ποιο ευρύτερο από την αντικατάσταση μιας υπηρεσίας με μια άλλη. Πιο συγκεκριμένα, ο απώτερος στόχος της προτεινόμενης προσέγγισης είναι να επιτρέψει στο λογισμικό που είναι προσανατολισμένο στις υπηρεσίες να είναι από κατασκευής επεκτάσιμο και προσαρμόσιμο. Για αυτόν το σκοπό, οι προσαρμογείς υπηρεσιών δεν είναι αρκετοί. Αντ΄ αυτού, θα πρέπει να παρέχουμε μηχανισμούς (1) για την εξόρυξη αφηρημένων υπηρεσιών από υπάρχουσες υπηρεσίες και (2) για πολυμορφισμό υπηρεσιών.

2.3. Ανασύνθεση

Στις τυπικές προσεγγίσεις ανασύνθεσης το σύστημα θεωρείται ως μια αφηρημένη ενορχήστρωση, όπου κάθε συγκεκριμένη διαδικασία μπορεί να εκτελεστεί από ένα σύνολο εναλλακτικών υπηρεσιών που μπορούν να αντικαταστήσουν η μια την άλλη. Οι διαφορετικές εναλλακτικές υπηρεσίες χαρακτηρίζονται από διαφορετικές τιμές ορισμένων ποιοτικών χαρακτηριστικών. Η συνολική ποιότητα του συστήματος μετράται από μια συνάρτηση, της οποίας τα αποτελέσματα υπολογίζονται, σε σχέση με τις τιμές των ποιοτικών χαρακτηριστικών των ενορχηστρωμένων υπηρεσιών. Στην συνέχεια ο στόχος των προσεγγίσεων ανασύνθεσης είναι να αντιμετωπιστούν αλλαγές στις απαιτήσεις του συστήματος και/ή αλλαγές στις τιμές των ποιοτικών χαρακτηριστικών των υπηρεσιών. Η αντιμετώπιση τέτοιου είδους αλλαγών ισοδυναμεί με τον υπολογισμό της βέλτιστης ενορχήστρωσης που να ικανοποιεί τις απαιτήσεις του συστήματος, ενώ μεγιστοποιεί την τιμή της συνάρτησης ποιότητας. Η πρώτη προσέγγιση σ’ αυτήν την περιοχή έρευνας προτάθηκε στο [32]. Το συνολικό πρόβλημα της βελτιστοποίησης της ποιότητας υπηρεσιών λύνεται χρησιμοποιώντας την τεχνική βελτιστοποίησης με γραμμικό προγραμματισμό. Η προσέγγιση που προτείνεται στα [1,3], ακολουθεί μια παρόμοια κατεύθυνση· η τεχνική βελτιστοποίησης του γραμμικού προγραμματισμού χρησιμοποιείται για την επίλυση του προβλήματος. Διαφορετικά από τις προαναφερθείσες προσεγγίσεις, στο [12] οι συγγραφείς προτείνουν την χρήση ενός γενετικού αλγορίθμου σε σκοπό να λύσουν το συνολικό πρόβλημα της βελτιστοποίησης της ποιότητας υπηρεσιών που εμπλέκονται στην ανασύνθεση μιας

δεδομένης ενορχήστρωσης υπηρεσιών. Επιπλέον, στο [34] οι συγγραφείς προτείνουν μια μέθοδο η οποία βρίσκει μια λύση που δεν είναι βέλτιστη για το συνολικό πρόβλημα βελτιστοποίησης. Το βασικό κίνητρο για την προτεινόμενη μέθοδο είναι ότι η πολυπλοκότητα των μεθόδων που βρίσκουν τις βέλτιστες λύσεις είναι πολύ υψηλή και κατά συνέπεια, η συνολική απόδοση των μεθόδων αυτών είναι κακή. Πηγαίνοντας ένα βήμα παραπέρα, στα [7, 8] η προτεινόμενη προσέγγιση λύνει το συνολικό πρόβλημα βελτιστοποίησης της ποιότητας υπηρεσιών για σύνολα ανεξάρτητων ενορχηστρώσεων.

Με στόχο την απόδοση, η εργασία [24] εισάγει μια ευρετική μέθοδο που βασίζεται σε τεχνικές ομαδοποίησης για να κατατάξει τους καλύτερες υποψήφιες υπηρεσίες για κάθε διαδικασία, οι οποίες στην συνέχεια χρησιμοποιούνται για να κατευθύνουν την επιλογή σχεδόν βέλτιστων ενορχηστρώσεων υπηρεσιών.

Τέλος, στο [10] οι συγγραφείς προτείνουν μια μέθοδο για την ανασύνθεση των συστημάτων που αποτελούνται από tiles. Το tile αντιστοιχεί σε μια συγκεκριμένη υπηρεσία που μπορεί να απαιτεί την χρήση άλλων υπηρεσιών. Ο στόχος είναι να βρεθεί η βέλτιστη σύνθεση των tiles, με βάση μια συνολική συνάρτηση ποιότητας. Το πρόβλημα λύνεται με την χρήση της τεχνικής του ακέραιου προγραμματισμού για βελτιστοποίηση. Παρόμοια με προσεγγίσεις που επικεντρώνονται στις ενορχηστρώσεις η προτεινόμενη προσέγγιση παρουσιάζει επίσης κακή απόδοση, καθώς το πλήθος των tiles αυξάνει.

Η προσέγγιση που προτείνεται σε αυτήν την διατριβή είναι συμπληρωματική στις προαναφερθείσες προσεγγίσεις ανασύνθεσης. Συγκεκριμένα, όλες αυτές οι προσεγγίσεις υποθέτουν ότι όλες οι διαδικασίες που εκτελούνται από το λογισμικό που είναι προσανατολισμένο στις υπηρεσίες αντιστοιχίζεται σε ομάδες με παρόμοιες υπηρεσίες. Επίσης, υποθέτουν ότι είναι δυνατή η εύκολη εναλλαγή μεταξύ αυτών των υπηρεσιών. Η προτεινόμενη προσέγγιση ταιριάζει απόλυτα με αυτές τις υποθέσεις καθώς οι βασικές έννοιες με τις οποίες ασχολείται είναι (1) οι αφηρημένες υπηρεσίες που αντιπροσωπεύουν ένα σύνολο από παρόμοιες υπηρεσίες και (2) ο πολυμορφισμός υπηρεσιών ο οποίος επιτρέπει την εναλλαγή ανάμεσα στις υπηρεσίες που αντιπροσωπεύονται από μια αφηρημένη υπηρεσία.

2.4. Επαναδιαπραγμάτευση, Εκ Νέου Εκτέλεση & Αναίρεση

Η επαναδιαπραγμάτευση, η εκ νέου εκτέλεση και η αναίρεση δεν είναι τεχνικές που μπορούν να σταθούν μόνες τους με την έννοια ότι συνήθως υποστηρίζουν είτε μια τεχνική αντικατάστασης μιας υπηρεσίας ή μια τεχνική ανασύνθεσης. Συγκεκριμένα, στο [3] και [34] οι συγγραφείς δηλώνουν ότι η επαναδιαπραγμάτευση των συμφωνιών που έχουν γίνει σε επίπεδο υπηρεσιών λαμβάνει χώρα αν δεν είναι δυνατόν να βρεθεί μια βέλτιστη δυνατή διαμόρφωση των υπηρεσιών που πληροί τις ποιοτικές απαιτήσεις του συγκεκριμένου συστήματος.

Επιπλέον, οι [33, 15, 16] επικεντρώνονται στην δυναμική αντικατάσταση υπηρεσιών. Για την αντιμετώπιση αυτού του προβλήματος η προτεινόμενη λύση βασίζεται στην αναζήτηση υποψήφιων υπηρεσιών αντικατάστασης μέσα από ένα σύνολο από σημασιολογικά συμβατές υπηρεσίες που μπορούν να χρησιμοποιηθούν στη θέση της υπό αντικατάσταση υπηρεσίας και στην επιλογή μιας μεταξύ αυτών των υποψήφιων· στην καλύτερη περίπτωση η επιλεγμένη υποκατάστατη υπηρεσία είναι τέτοια ώστε η τρέχουσα κατάστασή της να μπορεί να συγχρονιστεί με την κατάσταση της υπηρεσίας που αντικαθιστά. Σε αυτή την περίπτωση η ενορχήστρωση συνεχίζει κανονικά την εκτέλεσή της. Από την άλλη πλευρά, αν η κατάσταση της υποκατάστατης υπηρεσίας δεν μπορεί να συγχρονιστεί με την κατάσταση της υπό αντικατάσταση υπηρεσίας, τότε η προτεινόμενη μέθοδος βρίσκει της διαδικασίες της ενορχήστρωσης που χρησιμοποίησαν δεδομένα που ελήφθησαν από την υπό αντικατάσταση υπηρεσία και αναιρεί την εκτέλεσή τους.

Η προσέγγιση που προτείνεται σε αυτήν την διατριβή μπορεί να υποστηριχθεί από τις υπάρχουσες τεχνικές επαναδιαπραγμάτευσης, εκ νέου εκτέλεσης και αναίρεσης. Πιο συγκεκριμένα, όταν ο πολυμορφισμός υπηρεσιών χρησιμοποιείται για την εναλλαγή μεταξύ των υπηρεσιών που αντιπροσωπεύονται από μια αφηρημένη υπηρεσία, μπορεί να είναι απαραίτητο να χρησιμοποιηθούν τέτοιες τεχνικές για να διατηρηθεί η συνέπεια του λογισμικού που είναι προσανατολισμένο στις υπηρεσίες και χρησιμοποιεί αυτές τις υπηρεσίες.

ΚΕΦΑΛΑΙΟ 3. ΥΠΟΒΑΘΡΟ ΚΑΙ ΑΦΗΡΗΜΕΝΕΣ ΥΠΗΡΕΣΙΕΣ

3.1. Μοντελοποίηση Αφηρημένων Υπηρεσιών

Η έννοια της αφηρημένης υπηρεσίας, μαζί με την τεχνική που επιτρέπει την εξόρυξη αφηρημένων υπηρεσιών από υπάρχουσες υπηρεσίες παρουσιάστηκε στο [5]. Αυτή η εργασία αποτελεί την βάση για αυτή την διατριβή που προσθέτει στην έννοια των αφηρημένων υπηρεσιών ένα μηχανισμό για πολυμορφισμό υπηρεσιών.

Η χρήση των αφηρημένων υπηρεσιών και του πολυμορφισμού υπηρεσιών επιτρέπει την ανάπτυξη επεκτάσιμου και προσαρμόσιμου λογισμικού το οποίο είναι προσανατολισμένο στις υπηρεσίες. Πιο συγκεκριμένα, ο κύριος σκοπός μιας αφηρημένης υπηρεσίας είναι να παρέχει μια αφηρημένη διεπαφή για ένα σύνολο υπηρεσιών που παρέχουν παρόμοιες λειτουργίες. Η αφηρημένη διεπαφή έρχεται μαζί με ένα σύνολο από αντιστοιχίσεις μεταξύ της αφηρημένης διεπαφής και των διεπαφών των υπηρεσιών. Δίνοντας μια τέτοια αντιστοίχιση μεταξύ της αφηρημένης διεπαφής και της διεπαφή μιας υπηρεσίας που εκπροσωπείται, ο μηχανισμός πολυμορφισμού υπηρεσιών μεταφράζει τις αφηρημένες κλήσεις σε λειτουργίες της αφηρημένης διεπαφής σε συγκεκριμένες κλήσεις σε λειτουργίες της υπηρεσίας που εκπροσωπείται.

Στην συνέχεια, μπορούμε να αναπτύξουμε λογισμικό προσανατολισμένο στις υπηρεσίες που χρησιμοποιεί την διεπαφή της αφηρημένης υπηρεσίας, αντί να χρησιμοποιεί τις διεπαφές των υπηρεσιών που εκπροσωπούνται. Αυτό έχει ως πλεονέκτημα ότι το λογισμικό μπορεί εύκολα να προσαρμοστεί, για να χρησιμοποιεί οποιαδήποτε από τις υπηρεσίες που εκπροσωπούνται. Η προσαρμογή μπορεί να γίνει απλά αλλάζοντας την αντιστοίχιση η οποία χρησιμοποιείται από τον μηχανισμό πολυμορφισμού υπηρεσιών για την μετάφραση των κλήσεων αφηρημένων λειτουργιών σε κλήσεις μιας συγκεκριμένης λειτουργίας.

Με αυτήν την έννοια, η αφηρημένη υπηρεσία αντιστοιχεί σε έναν αφηρημένο τύπο, ενώ οι υπηρεσίες που αντιπροσωπεύονται αντιστοιχούν σε υπό τύπους αυτού του αφηρημένου τύπου. Σε αυτό το πλαίσιο, οι Liskov & Wing ορίζουν στο [LW94] μια λίστα από θεμελιωδών κανόνων, οι οποίες εγγυώνται πως ένας τύπος *S* είναι ο σωστός υπό-τύπος ενός τύπου *T*. Μερικά από αυτά τα κριτήρια δεν μπορούν υιοθετηθούν στην περίπτωση μιας αφηρημένης υπηρεσίας. Όσον αναφορά τους κανόνες για τις προ-συνθήκες και τις μετά-συνθήκες [22], υπάρχουν ορισμένες γλώσσες περιγραφής υπηρεσιών που παρέχουν τα μέσα για τον καθορισμό των προ-συνθηκών και των μετά-συνθηκών.

Πίνακας 3.1: Ορισμοί Βασικών Εννοιών

$$ServiceInterface = (n : string, O)$$
$$O = (Op_i : Operation)$$
$$Operation = (n : string, In : Message, Out : Message)$$
$$Message = \{part_i : Part\}$$
$$Part = (n : string, type : BuildinType, lower : int, upper : int)$$
$$ServiceAbstraction = (I : ServiceInterface, R, InterfaceMappings)$$
$$R = \{s_i : ServiceInterface\}$$
$$InterfaceMappings = \{m_{s_i} \mid \forall s_i \in R\}$$
$$m_{s_i} = (OperationMappings_{s_i}, MessageMappings_{io})$$
$$OperationMappings_{s_i} : I.O \rightarrow s_i.O$$
$$MessageMappings_{io} = \{m_{io_k} \mid \forall op_k \in I.O\}$$
$$m_{io_k} = (mi_{ki}, mo_{ki})$$
$$mi_{ki} : op_k.In \rightarrow mop_{s_i}(op_k).In$$
$$mo_{ki} : op_k.Out \rightarrow mop_{s_i}(op_k).Out$$

Ωστόσο, αυτά σπάνια χρησιμοποιούνται στην πράξη. Ως εκ τούτου, οι αφηρημένες υπηρεσίες λαμβάνουν υπόψη τους μόνο τους κανόνες contra-variance και covariance. Εν συντομία ο κανόνας contra-variance ορίζει πως κάθε μέθοδος m_S του τύπου S αντιστοιχίζεται σε μια μέθοδο m_T ενός τύπου *T* τέτοια ώστε ο τύπος κάθε ορίσματος του m_S να είναι υπό - τύπος του τύπου του αντίστοιχου ορίσματος m_T. Ο κανόνας covariance ορίζει ότι ο τύπος του αποτελέσματος της m_T είναι υπό-τύπος του αποτελέσματος της m_S.

Πιο φορμαλιστικά, η έννοια της αφηρημένης υπηρεσίας ορίζεται με βάση ένα γενικό μοντέλο που αντανακλά τα κύρια χαρακτηριστικά των υπηρεσιών. Σύμφωνα με αυτό το μοντέλο, μια υπηρεσία μπορεί να παρέχει ένα σύνολο από διεπαφές. Μια διεπαφή (Service Interface – Πίνακας 3.1), καθορίζεται από ένα όνομα και ένα σύνολο από λειτουργίες, Ο. Κάθε λειτουργία (Operation – Πίνακας 3.1), χαρακτηρίζεται από ένα όνομα, από ένα μήνυμα εισόδου, *In* και ένα μήνυμα εξόδου, *Out*. Γενικά ένα μήνυμα (Message – Πίνακας 3.1) έχει μια ιεραρχική δομή, που αποτελείται από ένα αριθμό τμημάτων, που χαρακτηρίζονται από το όνομά τους, τους XML τύπους δεδομένων τους καθώς και πάνω και κάτω όρια πληθικότητας· ένα μήνυμα μπορεί να είναι και άδειο. Ο τύπος δεδομένων ενός συγκεκριμένου τμήματος μπορεί να είναι είτε ένας βασικός τύπος ή ένας σύνθετος τύπος. (ένας τύπος, που αποτελείται από επιπλέον βασικούς τύπους ή σύνθετους τύπους δεδομένων). Η διαδικασία εξόρυξης αφηρημένων υπηρεσιών θεωρεί ότι μόνο τα φύλλα της ιεραρχικής δομής του μηνύματος. Ο λόγος αυτής της επιλογής είναι ότι η συγκεκριμένη δομή των δεδομένων εισόδου και εξόδου μιας λειτουργιάς προσθέτει περαιτέρω πολυπλοκότητα, ενώ δεν παρέχει πολύ χρήσιμες πληροφορίες για την διαδικασία εξόρυξης. Ιδανικά, μια αφηρημένη υπηρεσία θα πρέπει να αντιπροσωπεύει το σύνολο των διαθέσιμων υπηρεσιών που έχουν κοινό ένα συγκεκριμένο σύνολο από σημασιολογικά συμβατές λειτουργίες. Η εύρεση μέσα σε ένα σύνολο από υπηρεσίες, των υπηρεσιών αυτών που παρέχουν σημασιολογικά συμβατές λειτουργίες είναι πολύ δύσκολη. Ωστόσο, όπως περιγράφεται στο [5] πολύ συχνά συναντάμε υπηρεσίες που είναι σημασιολογικά συμβατές και παρέχουν συντακτικά παρόμοιες διεπαφές. Με βάση την προηγούμενη παρατήρηση, η ομοιότητα μεταξύ δύο διεπαφών υπηρεσιών εκτιμάται με βάση την μετρική απόστασης D_I, η οποία ορίζεται ως εξής (Πίνακας 3.2):

- Δίνονται δυο διεπαφές s_i και s_j και μιας αντιστοίχησης $OperationMapping_{op_{ij}} \subset s_i.O \times s_j.O$ ανάμεσα στις πιο παρόμοιες διαδικασίες των διεπαφών, η απόσταση $D_I(s_i, s_j)$ ορίζεται ως ο μέσος όρος των κανονικοποιημένων αποστάσεων ανάμεσα στα ονόματα των διεπαφών, και τον μέσο όρο των αποστάσεων ανάμεσα στις λειτουργίες που αντιστοιχίζονται.

- Η απόσταση $D_{op}(op_i, op_j)$ ανάμεσα σε δύο λειτουργίες op_i, op_j ορίζεται ως ο μέσος όρος των κανονικοποιημένων αποστάσεων μεταξύ των ονομάτων των διαδικασιών και του μέσου όρου των αποστάσεων των μηνυμάτων εισόδου και εξόδου.

- Δεδομένου μιας αντιστοίχισης $MessageMapping_{m_{ij}} \subset m_i \times m_j$ ανάμεσα στα πιο όμοια τμήματα δυο μηνυμάτων m_i, m_j, η απόσταση μεταξύ των μηνυμάτων είναι ο μέσος όρος των αποστάσεων ανάμεσα στα τμήματα που αντιστοιχιστήκαν.

- Τέλος, η απόσταση μεταξύ δύο τμήματων μηνυμάτων P_i, P_j ορίζεται ως ο μέσος όρος της κανονικοποιημένης απόστασης μεταξύ των ονομάτων τους και της κανονικοποιημένης απόστασης ανάμεσα στους βασικούς τύπους $ND_T(P_i.type_i, Pj.type_j)$· εφόσον αυτοί οι τύποι ανήκουν στον ίδιο κλάδο της τυπικής ιεραρχίας τύπων της XML τότε η απόσταση ορίζεται η απόλυτη διαφορά του βάθους τους, διαιρεμένο με το μέγιστο ύψος της ιεραρχίας τύπων της XML, αλλιώς υποθέτουμε ότι οι τύποι είναι δεν είναι συμβατοί και τότε $ND_T(P_i.type_i, Pj.type_j) = \infty$.

Με βάση τις προηγούμενες έννοιες μια αφηρημένη υπηρεσία ορίζεται ως μια πλειάδα $ServiceAbstraction = (I, R, InterfaceMappings)$ που αποτελείται από:

- Μια αφηρημένη διεπαφή *I*.

- Ένα σύνολο *R* από διεπαφές υπηρεσιών που εκπροσωπούνται (Πίνακας 3.1).

- Κάθε διαδικασία της αφηρημένης διεπαφής *I* αντιστοιχίζεται, μέσω ενός συνόλου αντιστοιχίσεων *InterfaceMappings*, σε ένα σύνολο από λειτουργίες, που παρέχονται από τις εκπροσωπούμενες διεπαφές. Συγκεκριμένα, το *InterfaceMappings* (Πίνακας 3.1) ορίζεται ως ένα ζευγάρι από πλειάδες. Κάθε τέτοια πλειάδα m_{si} αντιστοιχεί σε μια διεπαφή μιας υπηρεσίας $s_i \in R$ και αποτελείται από:

  - Μια 1-1 συνάρτηση $OperationMappings_{si}$ ανάμεσα στις διαδικασίες του *I* και τις λειτουργίες του s_i.

  - Ένα σύνολο από αντιστοιχίσεις $MessageMappings_{io}$ ανάμεσα στις εισόδους/εξόδους των λειτουργιών του *I* και των εισόδων/εξόδων των αντίστοιχων λειτουργιών του s_i τα οποία πληρούν τους κανόνες contra-variance και covariance. Ειδικότερα, $MessageMappings_{io}$ (Πίνακας 3.1) είναι ένα σύνολο από πλειάδες· κάθε τέτοια πλειάδα m_{iok} αντιστοιχεί σε μια διαδικασία του *I* η οποία αποτελείται από:

    - Μια 1-1 συνάρτηση mi_{ki} η οποία αντιστοιχεί μια είσοδο $i_k \in op_k.In$, σε μια είσοδο $i_{si} \in mop_{si}(op_k).In$ τέτοια ώστε: $(i_k.type \subseteq i_{si}.type) \vee (i_k.upper \leq i_{si}.upper) \vee (i_k.lower \geq i_{si}.lower)$.

    - Μια 1-1 συνάρτηση mo_{ki} η οποία αντιστοιχεί μια είσοδο $o_k \in op_k.Out$ σε μια έξοδο $o_{si} \in mop_{si}(op_k).Out$, τέτοιο ώστε: $(o_{si}.type \subseteq o_k.type) \vee (o_{si}.upper \leq o_k.upper) \vee (o_{si}.lower \geq o_k.lower)$.

Πίνακας 3.2:Απόσταση Ανάμεσα Στις Διεπαφές Των Υπηρεσιών

$$D_I(s_i, s_j) = \frac{NED(s_i.n, s_j.n)}{2} + \frac{\sum_{\forall (op_i, op_j) \in M_{op_{ij}}} D_{op}(op_i, op_j)}{|M_{op_{ij}}|}$$

$$D_{op}(op_i, op_j) = \frac{NED(op_i.n, op_j.n)}{2} + \frac{D_{io}(op_i, op_j)}{2}$$

$$D_{io}(op_i, op_j) = \frac{D_m(op_i.In, op_j.In)}{2} + \frac{D_m(op_i.Out, op_j.Out)}{2}$$

$$D_m(m_i, m_j) = \frac{\sum_{\forall (p_i, p_j) \in M_{m_{ij}}} D_p(p_i, p_j)}{|M_{m_{ij}}|}$$

$$D_p(p_i, p_j) = \frac{NED(p_i.n, p_j.n)}{2} + \frac{ND_T(p_i.type, p_j.type)}{2}$$

3.2. Εξόρυξη Αφηρημένων Υπηρεσιών

Ο απώτερος στόχος της διαδικασίας εξόρυξης είναι να βρει ομάδες από παρόμοιες υπηρεσίες, ο εντοπισμός των αφηρημένων υπηρεσιών για τις ομάδες αυτές, μαζί με τις αντιστοιχίσεις μεταξύ των αφηρημένων διεπαφών και των διεπαφών των ομαδοποιημένων υπηρεσιών. Δεδομένου αυτού του στόχου η διαδικασία εξόρυξης εμπνέεται από την ιεραρχική ομαδοποίηση. Ωστόσο, όπως αναφέρθηκε στο [5], οι τυπικοί αλγόριθμοι ιεραρχικής ομαδοποίησης (π.χ., SLA, CLA, WLA, ULA [23]) δεν έχουν άμεση εφαρμογή. Μετά, θα συζητήσουμε τα βήματα της διαδικασίας εξόρυξης που είναι ειδικά προσαρμοσμένες για την περίπτωση των υπηρεσιών, ενώ ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει στο [5] για περαιτέρω λεπτομέρειες.

Η διαδικασία της εξόρυξης δέχεται σαν είσοδο ένα σύνολο από διεπαφές $S = \{ s_i: ServiceInterface\}$. Η έξοδος του αλγορίθμου είναι ένα σύνολο από ιεραρχικά δομημένες αφηρημένες υπηρεσίες $A = \{a_l: ServiceAbstraction\}$. Για το σκοπό αυτό, η διαδικασία εξόρυξης εκτελεί αναδρομικά τα εξής βήματα:

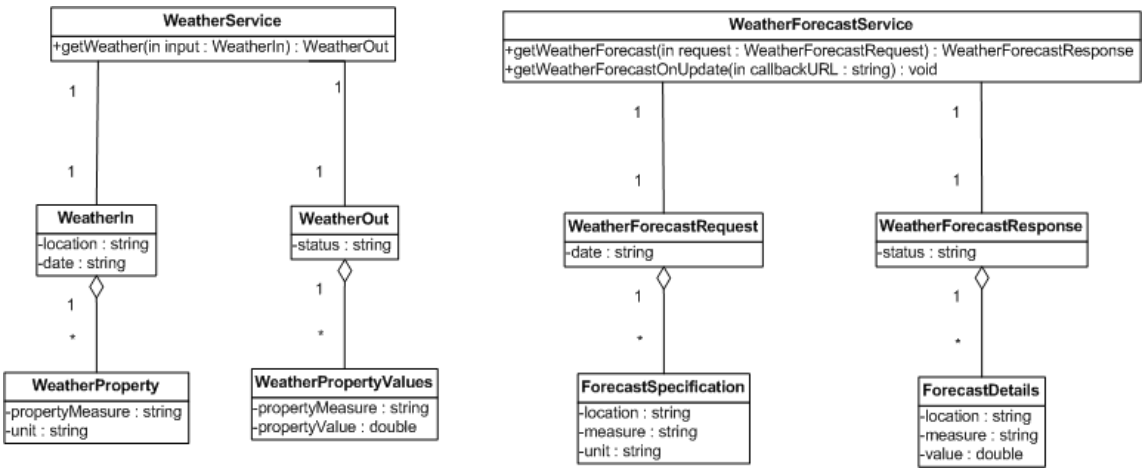
1. Για κάθε ζευγάρι από διεπαφές $s_i \in S$, $s_j \in S$ η διαδικασία υπολογίζει την απόσταση $D_I(s_i, s_j)$. Για τον σκοπό αυτό, οι ποιο όμοιες λειτουργίες $(op_i, op_j) \in s_i.O \times s_j.O$ (δηλ. οι αντιστοιχίσεις $OperationMappings_{op_{ij}}$ - Ενότητα 3.1) βρίσκονται λύνοντας το αντίστοιχο μέγιστο σταθμισμένο πρόβλημα σε διμερές γράφημα [26]. Οι κόμβοι του γραφήματος αντιστοιχούν στις λειτουργίες op_i και op_j, ενώ οι ακμές αντιστοιχούν στις αποστάσεις μεταξύ των λειτουργιών. Η εύρεση των αποστάσεων δύο λειτουργιών $op_i \in s_i.O$ και $op_j \in s_j.O$ περιλαμβάνει την εύρεση των πιο όμοιων ζευγαριών μεταξύ των τμημάτων για τα μηνύματα εισόδου (αντίστοιχα για τα μηνύματα εξόδου) των λειτουργιών (δηλ. οι αντιστοιχίσεις $MessageMapping_{m_{ij}}$ – Ενότητα 3.1). Αυτό το πρόβλημα επιλύεται επίσης λύνοντας το πρόβλημα μέγιστης σταθμισμένης αντιστοίχισης σε ένα διμερές γράφημα που αντιπροσωπεύει τα μηνύματα εισόδου (αντίστοιχα τα μηνύματα εξόδου). Σημειώστε ότι σε αυτό το βήμα είναι πιθανόν να υπολογιστεί μια απόσταση μεταξύ δυο διεπαφών που ισούται με ∞. Αυτή η περίπτωση μπορεί να προκύψει εάν η καλύτερη δυνατή αντιστοίχιση ανάμεσα στα αποτελέσματα των μηνυμάτων περιέχουν τουλάχιστον ένα ζευγάρι από ασύμβατους τύπους. Σε μια τέτοια περίπτωση, θεωρούμε ότι δεν είναι δυνατόν να δημιουργηθεί μια αφηρημένη υπηρεσία από τις δύο διεπαφές.

2. Με βάση τις υπολογισμένες αποστάσεις το πιο παρόμοιο ζεύγος από διεπαφές (s_i, s_j) επιλέγεται και μια αφηρημένη διεπαφή a κατασκευάζεται ως εξής: $\alpha.I$ προέρχεται από το s_i ή s_j. Ας υποθέσουμε ότι $\alpha.I$ προέρχεται από το s_i, το όνομα του $\alpha.I$ προέρχεται από το όνομα του s_i συμπληρωμένο με τον όρο “abstract”·για κάθε ζευγάρι λειτουργιών που ταιριάζουν (op_i, op_j) που βρέθηκε στο προηγούμενο βήμα, η $\alpha.I$ περιλαμβάνει την αντίστοιχη λειτουργία op_a που βασίζεται στην ίδια σύμβαση ονομασίας. Η δομή του μηνύματος της εισόδου (αντίστοιχα της εξόδου) του op_a ακολουθεί την δομή του μηνύματος της εισόδου (αντίστοιχα της εξόδου) του op_i. Τα μέρη που αποτελούν το μήνυμα της εισόδου (αντίστοιχα της εξόδου) του op_a αντιστοιχούν στα ζευγάρια των στοιχείων των μηνυμάτων των op_i, op_j που αντιστοιχίστηκαν στο προηγούμενο βήμα.

3. Η αφηρημένη υπηρεσία a περιλαμβάνεται στα αποτελέσματα π.χ., $A = A \cup \{a\}$. Επιπλέον, οι υπηρεσίες που εκπροσωπούνται από το a αφαιρούνται από το σύνολο εισόδου, π.χ., $S = S – a.D$. Τελικά το $a.I$ περιέχεται στο S, π.χ., $S = S \cup \{a.I\}$, έτσι ώστε να εξυπηρετούν την κατασκευή υψηλότερου επιπέδου αφηρημένες υπηρεσίες.

4. Η διαδικασία επαναλαμβάνει τα βήματα (1) έως (3), έως ότου το σύνολο εισόδου περιλαμβάνει μόνο ένα στοιχείο, δηλαδή η ρίζα της ιεραρχίας A που προκύπτει, η οποία γενικεύει όλες τις διαθέσιμες διεπαφές υπηρεσιών, ή μέχρι να μην μπορούν να κατασκευαστούν άλλες αφηρημένες υπηρεσίες.

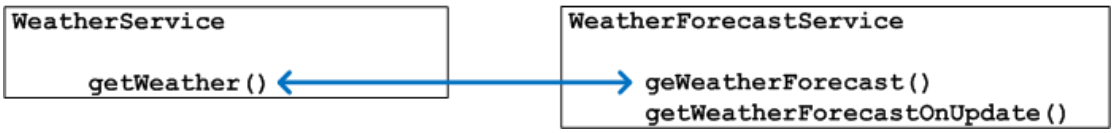
Ας δούμε ένα παράδειγμα, Σχήμα 3.1 δείχνει δύο παρόμοιες υπηρεσίες που χρησιμοποιούνται για να παίρνουμε αναφορές για τον καιρό. Συγκεκριμένα, WeatherService παρέχει την λειτουργία getWeather(), η οποία αναφέρει, για μια συγκεκριμένη ημερομηνία και περιοχή, τις τιμές για ένα δεδομένο σύνολο ιδιοτήτων του καιρού.



Σχήμα 3.1: Το Παράδειγμα Weather Service

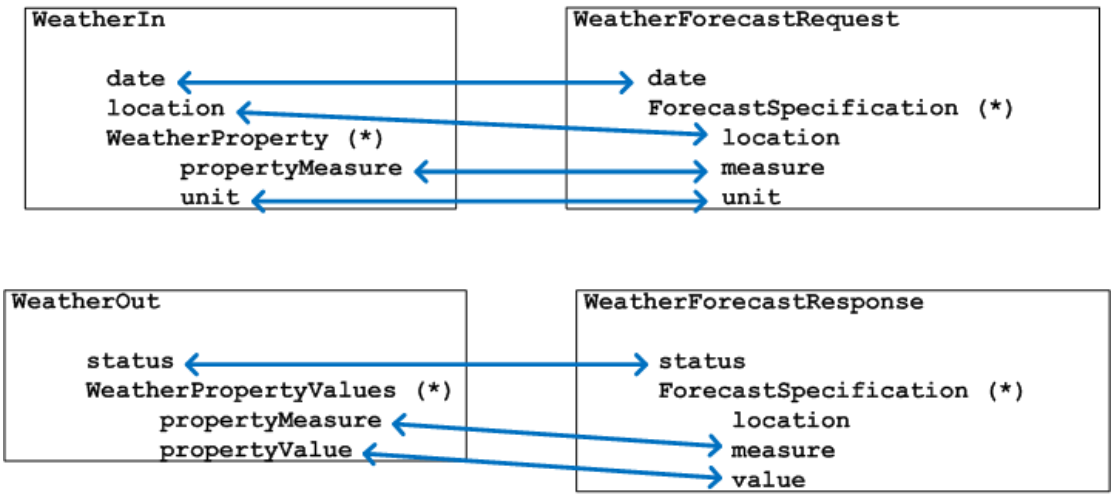
Συγκεκριμένα, το μήνυμα εισόδου της λειτουργίας αποτελείται από το στοιχείο WeatherIn, το οποίο χαρακτηρίζεται από μια περιοχή (π.χ., Αθήνα, Παρίσι) και μια ημερομηνία. Το στοιχείο WeatherIn επιπλέον περιλαμβάνει μια λίστα από στοιχεία WeatherProperty. Κάθε στοιχείο WeatherProperty χαρακτηρίζεται από μια propertyMeasure (π.χ., θερμοκρασία, υγρασία) και μια unit γι’ αυτήν την

propertyMeasure (π.χ., Φαρενάιτ, Κελσίου). Το μήνυμα εξόδου της λειτουργίας αποτελείται από το στοιχείο WeatherOut, το οποίο χαρακτηρίζεται από μια συνολική κατάσταση του καιρού (π.χ., ηλιόλουστος, συννεφιασμένος) και μια λίστα από στοιχεία WeatherPropertyValue. Κάθε στοιχείο χαρακτηρίζεται από μια propertyMeasure (π.χ., θερμοκρασία) και την τιμή της μετρικής αυτής (π.χ., 19 C).



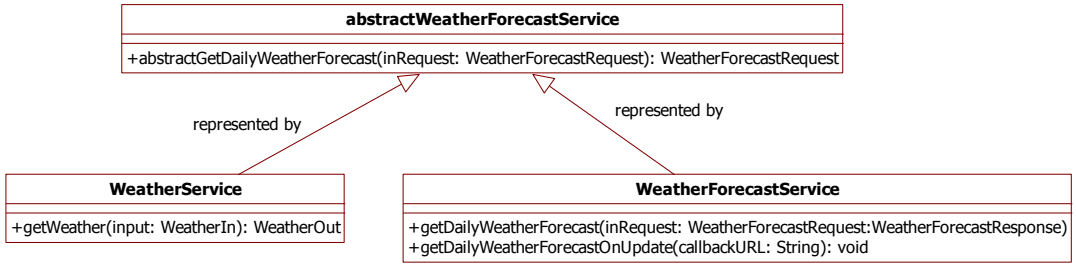
Σχήμα 3.2: Αντιστοιχίσεις των Διαδικασιών του Weather Services

Από την άλλη πλευρά, η υπηρεσία WeatherForecast παρέχει δυο λειτουργίες. Η διαδικασία getWeatherForecast() αναφέρει για μια συγκεκριμένη ημερομηνία τις τιμές ενός συγκεκριμένου συνόλου ιδιοτήτων του καιρού, σε ένα σύνολο από περιοχές. Συγκεκριμένα, το μήνυμα εισόδου της λειτουργίας αποτελείται από ένα στοιχείο WeatherForecastRequest, το οποίο χαρακτηρίζεται από μια ημερομηνία. Το στοιχείο WeatherForecastRequest επιπλέον αποτελείται από μια λίστα από στοιχεία ForecastSpecification. Κάθε στοιχείο ForecastSpecification χαρακτηρίζεται από μια περιοχή (π.χ., Αθήνα, Παρίσι), το measure (π.χ., θερμοκρασία, υγρασία) και μια unit για την μέτρηση αυτή (π.χ., Φαρενάιτ, Κελσίου).



Σχήμα 3.3: Αντιστοιχίσεις Μεταξύ των Μηνυμάτων των ΥπηρεσιώνWeather

Το μήνυμα εξόδου της λειτουργίας αποτελείται από το στοιχείο WeatherForecastResponse, το οποίο χαρακτηρίζεται από μια συνολική κατάσταση του καιρού (π.χ., ηλιόλουστος, συννεφιασμένος) και μια λίστα από στοιχεία ForecastDetails. Κάθε στοιχείο χαρακτηρίζεται από μια περιοχή, μια μέτρηση (π.χ., θερμοκρασία) και την τιμή αυτής της μέτρησης (π.χ., 19C). Η διαδικασία getWeatherForecastOnUpdate(), αναφέρει τις αλλαγές καιρού στέλνοντας πληροφορίες σε ένα συγκεκριμένο URL. Ως εκ τούτου, το μήνυμα εισόδου περιλαμβάνει το δεδομένο URL, ενώ το μήνυμα εξόδου είναι άδειο.



Σχήμα 3.4 : Η Αφηρημένη Υπηρεσία Καιρού

Υποθέτοντας ότι οι παραπάνω δύο υπηρεσίες δίνονται ως είσοδος στην διαδικασία εξόρυξης, παίρνουμε τις αντιστοιχίσεις των διαδικασιών του φαίνονται στο Σχήμα 3.2. Για τις αντιστοιχίσεις αυτών των διαδικασιών, παίρνουμε τις αντιστοιχίσεις μηνυμάτων που φαίνονται στο Σχήμα 3.3. Τέλος, η αφηρημένη υπηρεσία που προκύπτει και εκπροσωπεί τις δύο υπηρεσίες δίνεται στο Σχήμα 3.4.

ΚΕΦΑΛΑΙΟ 4. ΠΟΛΥΜΟΡΦΙΣΜΟΣ ΥΠΗΡΕΣΙΩΝ

4.1. Επεκτάσιμο & Προσαρμόσιμο Λογισμικό Προσανατολισμένο στις Υπηρεσίες

Σε γενικές γραμμές, μπορούμε να υποθέσουμε ότι ένα λογισμικό προσανατολισμένο στις υπηρεσίες αποτελείται από ένα σύνολο από συνεργαζόμενες υπηρεσίες. Ο όρος προσαρμόσιμο αναφέρεται στην ικανότητα να προσαρμόζει τις υπηρεσίες που χρησιμοποιούνται στη λειτουργία του λογισμικού με *ένα συνεπή και κλιμακούμενο τρόπο. Επίσης πρέπει να η προσαρμογή να διαταράσσει όσο το δυνατόν λιγότερο την εκτέλεση του λογισμικού*. Η προσαρμογή μιας υπηρεσίας, αντιστοιχεί στην αλλαγή μιας υπηρεσίας που χρησιμοποιείται με μία άλλη. Για το σκοπό αυτό, συνεπής προσαρμογή σημαίνει ότι η αλλαγή της υπηρεσίας πρέπει να αφήσει το λογισμικό σε μια σωστή κατάσταση, δηλαδή μια κατάσταση από την οποία οι υπηρεσίες που εμπλέκονται μπορούν να συνεχίσουν να λειτουργούν κανονικά, αντί να προχωρούν προς μια κατάσταση σφάλματος [19]. Το να διαταράσσεται όσο το δυνατό λιγότερο η εκτέλεση του λογισμικού σημαίνει ότι η αλλαγή της υπηρεσίας δεν θα πρέπει να αναστείλει την εκτέλεση του συνόλου του λογισμικού, μπλοκάροντας το σύνολο των υπηρεσιών που εμπλέκονται σε αυτό το λογισμικό [KM90]. Κλιμακούμενη προσαρμογή σημαίνει ότι ο μηχανισμός ο οποίος εκτελεί την αλλαγή της υπηρεσίας:

- Δεν θα πρέπει να είναι κεντρικοποιημένη.

- Δεν θα πρέπει να απαιτεί την γνώση όλης της δομής του λογισμικού.

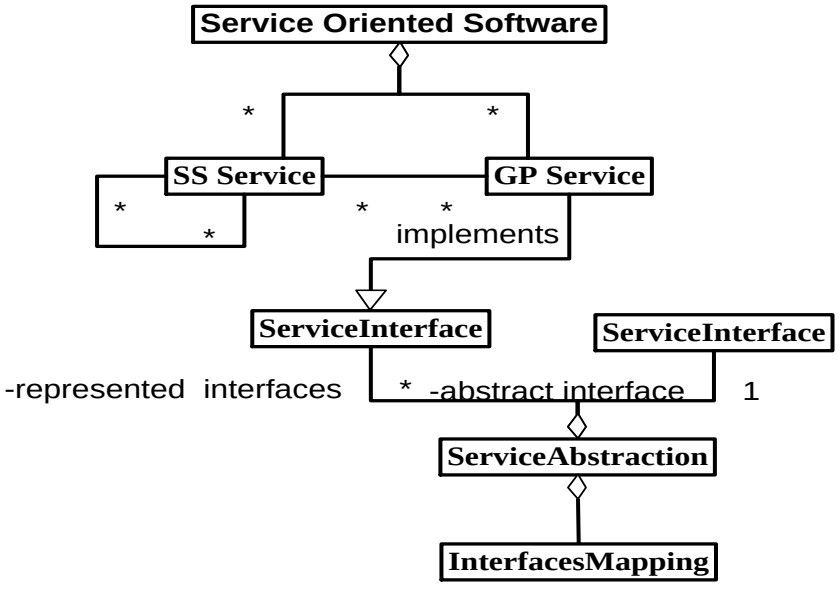
- Η πολυπλοκότητα των διάφορων εργασιών προσαρμογής που εκτελούνται από τον μηχανισμό θα πρέπει να κλιμακώνεται λογικά σε σχέση με τις οντότητες του σχετίζονται με τις εν λόγω εργασίες.

Έχοντας κατά νου τις παραπάνω ιδιότητες, σχεδιάσαμε ένα μηχανισμό που υλοποιεί τον πολυμορφισμό για υπηρεσίες που αντιπροσωπεύονται από μια αφηρημένη υπηρεσία. Καταρχάς, σε αυτή την ενότητα καθορίζουμε τις θεμελιώδης έννοιες σχετικά με τον προτεινόμενο μηχανισμό.

Σε ένα λογισμικό προσανατολισμένο στις υπηρεσίες, κάθε υπηρεσία σχετίζεται με άλλες υπηρεσίες με τις οποίες αλληλεπιδρά. Έχουμε μια περαιτέρω διάκριση ανάμεσα σε δυο είδη υπηρεσιών (Σχήμα 4.1). Πιο συγκεκριμένα, υποθέτουμε ένα σύνολο υπηρεσιών ***S*** οι οποίες έχουν αναπτυχθεί για τον σκοπό του λογισμικού και ένα σύνολο *GP* υπηρεσιών γενικού σκοπού που έχουν ανακαλυφθεί στο διαδίκτυο. Δεδομένου ότι οι υπηρεσίες του ***S*** έχουν αναπτυχθεί ειδικά για το συγκεκριμένο λογισμικό, οι προγραμματιστές του λογισμικού αυτού έχουν πρόσβαση στις υλοποιήσεις τους. Η πρόσβαση στην υλοποίηση των υπηρεσιών του ***S*** επιτρέπει την εύκολη προσαρμογή τους, υπό την έννοια ότι είναι σε θέση να συμβάλλουν στην προσαρμογή του λογισμικού. Οι υπηρεσίες του *GP* είναι υπηρεσίες γενικού σκοπού για τα οποίες οι προγραμματιστές δεν έχουν πρόσβαση στις υλοποιήσεις τους. Επιπλέον, οι υπηρεσίες του GP δεν παρέχουν καμία υποστήριξη για την προσαρμογή του λογισμικού. Ωστόσο, θεωρούμε ότι κάθε διεπαφή της υπηρεσίας του *GP* ανήκει σε μια αφηρημένη υπηρεσία, μαζί με άλλες διεπαφές υπηρεσιών που αντιπροσωπεύονται επίσης από την ίδια αφηρημένη υπηρεσία. Η αφηρημένη υπηρεσία περαιτέρω χαρακτηρίζεται από μια αφηρημένη διεπαφή και τις αντιστοιχίσεις ανάμεσα στην διεπαφή και τις διεπαφές των υπηρεσιών που εκπροσωπούνται (Κεφάλαιο 3).

Με βάση την προηγούμενη άποψη, το κύριο πρόβλημα για να επιτραπεί η διαδικασία της προσαρμογής του λογισμικού που είναι προσανατολισμένο στις υπηρεσίες είναι η αλλαγή των *GP* υπηρεσιών που χρησιμοποιούνται στο λογισμικό.

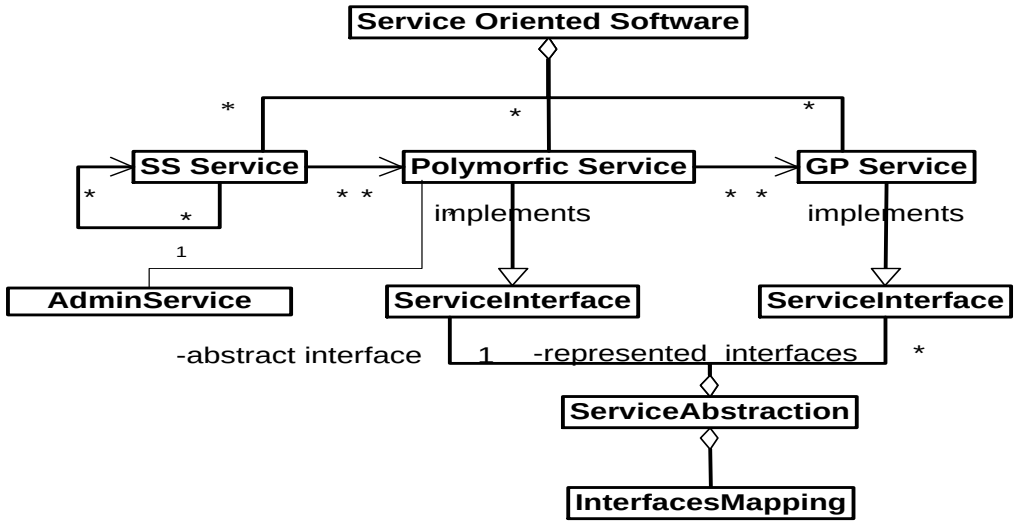
Για να αλλάξουμε μια υπηρεσία *GP* με μια άλλη πρέπει να αλλάξουμε την υλοποίηση όλων των υπηρεσιών ***S*** που χρησιμοποιούν αυτήν την υπηρεσία *GP*. Η κατάσταση αυτή δεν ακολουθεί την ιδιότητα της ελάχιστης διαταραχής του λογισμικού που αναφέραμε στο ξεκίνημα αυτής της ενότητας. Η αντιμετώπιση αυτών των θεμάτων είναι ο κύριος σκοπός των λεγόμενων *πολυμορφικών υπηρεσιών* που εισάγονται στην αρχιτεκτονική του λογισμικού (Σχήμα 4.2). Ειδικότερα, κάθε υπηρεσία *GP* που χρησιμοποιείται στο λογισμικό είναι κρυμμένη πίσω από την αντίστοιχη πολυμορφική υπηρεσία. Οι ***S*** υπηρεσίες που πρέπει να χρησιμοποιήσουν τις κρυμμένες *GP* υπηρεσίες καλούν την πολυμορφική υπηρεσία, αντί να καλούν την κρυμμένη υπηρεσία *GP*.



Σχήμα 4.1: Λογισμικό Προσανατολισμένο στις Υπηρεσίες

Οι κλήσεις εκτελούνται σε σχέση με τη διεπαφή της πολυμορφικής υπηρεσίας, η οποία είναι στην πραγματικότητα η αφηρημένη διεπαφή της αφηρημένης υπηρεσίας που αντιπροσωπεύει την κρυμμένη υπηρεσία *GP*. Οι κλήσεις που γίνονται στην

πολυμορφική υπηρεσία στην συνέχεια μεταφράζονται σε κλήσεις στην κρυμμένη υπηρεσία *GP*. Για την υλοποίηση αυτής της διαδικασίας μετάφρασης, η πολυμορφική υπηρεσία βασίζεται στον μηχανισμό πολυμορφισμού που προτείνουμε. Ο προτεινόμενος μηχανισμός πολυμορφισμού παίρνει ως είσοδο την αντιστοίχιση, μεταξύ της αφηρημένης διεπαφής και της διεπαφής της κρυμμένης υπηρεσίας *GP*, που προέκυψε κατά την εξόρυξη της αφηρημένης υπηρεσίας η οποία αντιπροσωπεύει την κρυφή υπηρεσία *GP*. Οπότε, για να αλλάξουμε την κρυφή υπηρεσία *GP* με μια άλλη υπηρεσία *GP*, που εκπροσωπείται από την ίδια αφηρημένη υπηρεσία μπορούμε απλά να αλλάξουμε την αντιστοίχιση που χρησιμοποιείται από τον προτεινόμενο μηχανισμό πολυμορφισμού.



Σχήμα 4.2: Επεκτάσιμο Λογισμικό Προσανατολισμένο στις Υπηρεσίες

Για να εξασφαλιστεί η συνεπής εκτέλεση του λογισμικού συνολικά κατά την διάρκεια και μετά την αλλαγή της υπηρεσίας *GP*, ο προτεινόμενος πολυμορφικός μηχανισμός είναι περαιτέρω υπεύθυνος για την γνωστοποίηση στις υπηρεσίες του ***S*** που

χρησιμοποιούν την αφηρημένη υπηρεσία σχετικά με την αλλαγή της υπηρεσίας *GP* που κρύβεται πίσω από την αφηρημένη υπηρεσία. Μετά από μια τέτοια ειδοποίηση, κάθε υπηρεσία *S* είναι υπεύθυνη για την διατήρηση της τοπικής συνοχής.

Η αρχικοποίηση για το ποια GP υπηρεσία κρύβεται πίσω από τον μηχανισμό πολυμορφισμού γίνεται μέσω του AdminService (Σχήμα 4.2). Το AdminService είναι επιπλέον υπεύθυνο και για την αντικατάσταση της GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία με κάποια άλλη που εκπροσωπείται από αυτήν καλώντας τις απαραίτητες διαδικασίες αναπροσαρμογής.

Με βάση αυτή την εννοιολογική επισκόπηση του προσαρμόσιμου λογισμικού προσανατολισμένου στις υπηρεσίες, στην ενότητα 4.2 δίνουμε περαιτέρω λεπτομέρειες σχετικά με την ανάπτυξη της πολυμορφικής υπηρεσίας και του προτεινόμενου μηχανισμού πολυμορφισμού.

4.2. Πολυμορφισμός Υπηρεσιών

Για την διευκόλυνση της ανάπτυξης πολυμορφικών υπηρεσιών αναπτύξαμε μια γεννήτρια που παράγει αυτόματα τέτοιες υπηρεσίες παίρνοντας ως είσοδο τις προδιαγραφές μιας αφηρημένης υπηρεσίας. Η υλοποίηση των πολυμορφικών υπηρεσιών βασίζεται στον προτεινόμενο πολυμορφικό μηχανισμό που περιγράφεται επίσης σε αυτήν την ενότητα.

4.2.1. Δημιουργία Πολυμορφικών Υπηρεσιών

Ο μηχανισμός παραγωγής πολυμορφικών υπηρεσιών που προτείνουμε σε αυτή την διατριβή παίρνει σαν είσοδο την αφηρημένη διεπαφή μιας αφηρημένης υπηρεσίας που προκύπτει από την διαδικασία εξόρυξης και παράγει την πολυμορφική υπηρεσία. Πιο συγκεκριμένα ο μηχανισμός παράγει ένα java interface με όνομα το όνομα της διεπαφής που παίρνει σαν είσοδο. Το interface αυτό περιέχει τόσες μεθόδους όσες και η αντίστοιχη αφηρημένη διεπαφή που παίρνει σαν είσοδο. Επίσης στην διεπαφή που προκύπτει εισάγουμε και όλα τα απαραίτητα annotations για να

μπορεί η αφηρημένη κλάση αργότερα να μετατραπεί σε υπηρεσία διαδικτύου. Έστω ότι έχουμε το παράδειγμα της Ενότητας 3 που περιέχει τις υπηρεσίες WeatherService και WeatherForecastService και την αφηρημένη υπηρεσία με όνομα AbstractWeatherForecastService το Java interface που παράγεται για την αφηρημένη υπηρεσία φαίνεται στο Σχήμα 4.3.

```
@WebService
public interface AbstractWeatherForecastService {
@WebMethod public GetDailyWeatherForecastResponseOut
getDailyWeatherForecast ( GetDailyWeatherForecastIn input );

}
```

Σχήμα 4.3: Java Interface της Πολυμορφικής Υπηρεσίας

Ακόμη παράγουμε μια κλάση που είναι η υλοποίηση του Java interface που φαίνεται στο Σχήμα 4.3 με όνομα το όνομα της διεπαφής και την προσθήκη στο τέλος της κατάληξης “Impl” η κλάση που παράγεται φαίνεται στο Σχήμα 4.4. Πιο συγκεκριμένα, το πεδίο interfaceMapping χρησιμοποιείται για να αποθηκεύουμε τις αντιστοιχίσεις που χρησιμοποιούνται από την συγκεκριμένη αφηρημένη διεπαφή, το πεδίο serviceInstance χρησιμοποιείται για την αποθήκευση του URL το οποίο αντιστοιχεί στο στιγμιότυπο της GP υπηρεσίας που χρησιμοποιείται, το πεδίο serviceInterface χρησιμοποιείται για την αποθήκευση του ονόματος της διεπαφής της εκάστοτε GP υπηρεσίας που κρύβεται πίσω από την αφηρημένη διεπαφή και τέλος υπάρχει και το πεδίο abstractionName χρησιμοποιείται για την αποθήκευση του ονόματος της αφηρημένης διεπαφής. Επίσης πρέπει να σημειωθεί πως η κλάση BaseAbstractionServiceImpl είναι ανεξάρτητη από την εκάστοτε αφηρημένη υπηρεσία και για αυτό λόγο αυτό υλοποιήθηκε με βάση το μηχανισμό Reflection της Java.

```
@WebService()
public class AbstractWeatherForecastServiceImpl extends BaseAbstractionServiceImpl
implements AbstractWeatherForecastService
{
public WeatherForecastServiceImpl ()
{
        super("WeatherForecastService");
}

publicGetDailyWeatherForecastResponseOut                    getDailyWeatherForecast(
GetDailyWeatherForecastIn input )
{

        GetDailyWeatherForecastResponseOut          result          =          new
GetDailyWeatherForecastResponseOut();


        LoadVariables();

        ServiceInterface                abstractionServiceInterface                =
interfaceMapping.getServiceInterface(abstractionName);

        Object    output    =    invokeConcreteService(input,"getDailyWeatherForecast",
abstractionServiceInterface);

        setOutPut  (output,  abstractionServiceInterface,  "getDailyWeatherForecast"  ,
result);

        return result;

        }

}
```

Σχήμα 4.4 Η Υλοποίηση του Java Interface για την Πολυμορφική Υπηρεσία

Ακόμη, κατά την δημιουργία του πολυμορφικού μηχανισμού αποθηκεύονται με χρήση του μηχανισμού Serialization της Java οι αντιστοιχίσεις μεταξύ της πολυμορφικής υπηρεσίας και των υπηρεσιών που εκπροσωπούνται από αυτήν σε ένα αρχείο που έχει την κατάληξη *.ser*.

Επιπλέον, για κάθε μέθοδο της προαναφερθείσας διεπαφής παράγουμε μια κλάση με όνομα το όνομα της μεθόδου και την προσθήκη της κατάληξης *"In"* για να ομαδοποιήσουμε τα ορίσματα εισόδου της μεθόδου. Αντίστοιχα με την είσοδο

παράγεται και μια κλάση για την έξοδο με όνομα το όνομα της μεθόδου και την κατάληξη *"Out"* για την ομαδοποίηση των εξόδων της μεθόδου. Στο παράδειγμα της Ενότητας 3, η μέθοδος getDailyWeatherForecast παίρνει ένα αντικείμενο WeatherForecastRequest ως όρισμα η κλάση που παράγεται από τον μηχανισμό πολυμορφισμού φαίνεται στο Σχήμα 4.5.

```
public class GetDailyWeatherForecastIn
{
        public WeatherForecastRequest inRequest;
}
```

Σχήμα 4.5: Η Κλάση που Παράγεται για την Είσοδο

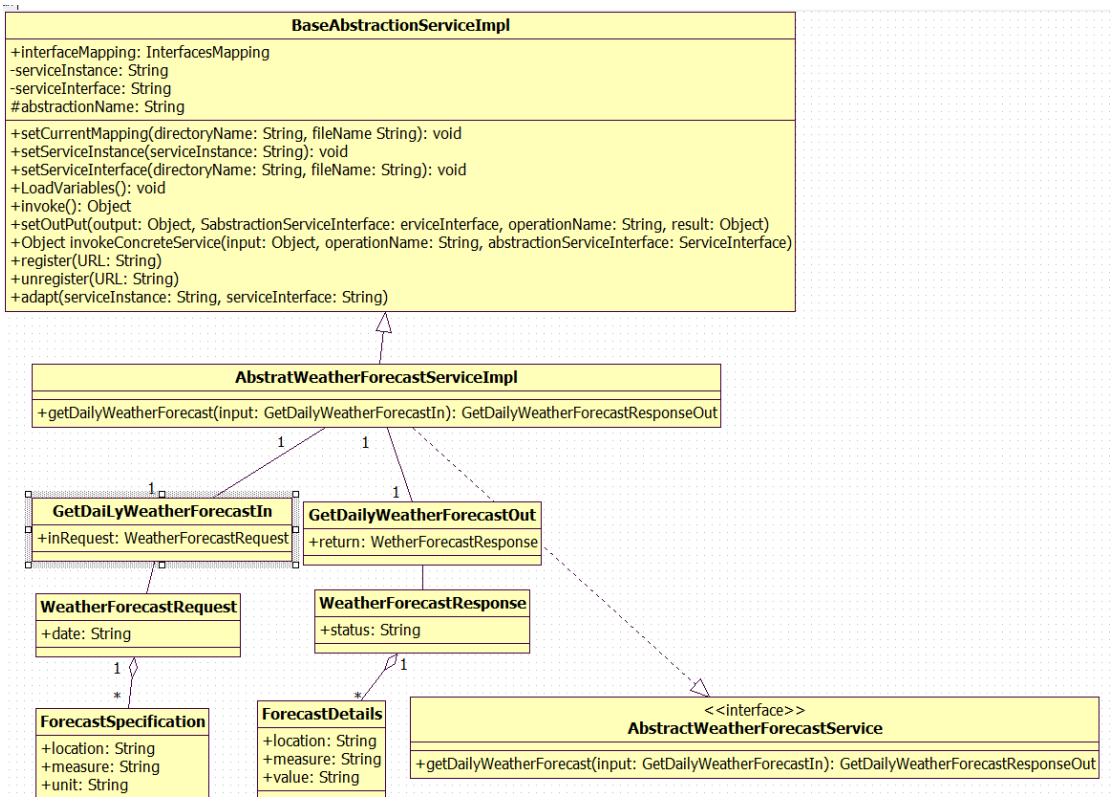
Αντίστοιχα, η έξοδος της μεθόδου επιστρέφει ένα αντικείμενο WeatherForecastResponse η κλάση που παράγεται φαίνεται στο Σχήμα 4.6.

```
public class GetDailyWeatherForecastResponseOut
{
        public WeatherForecastResponse return;
}
```

Σχήμα 4.6: Η Κλάση που Παράγεται για την Έξοδο

Τέλος, ο μηχανισμός παραγωγής της αφηρημένης υπηρεσίας παράγει και όλες τις απαραίτητες κλάσεις που πιθανόν να περιέχονται τόσο στα ορίσματα των μεθόδων όσο και στις εξόδους τους χρησιμοποιώντας το εργαλείο wsimport και το wsdl έγγραφο που αντιστοιχεί στην διεπαφή της αφηρημένης υπηρεσίας. Πιο συγκεκριμένα, για το παράδειγμα μας θα παράγουμε τις κλάσεις WeatherForecastRequest και ForecastSpecification για την είσοδο της αφηρημένης λειτουργίας καθώς και τις κλάσεις WeatherForecastResponse και ForecastDetails.

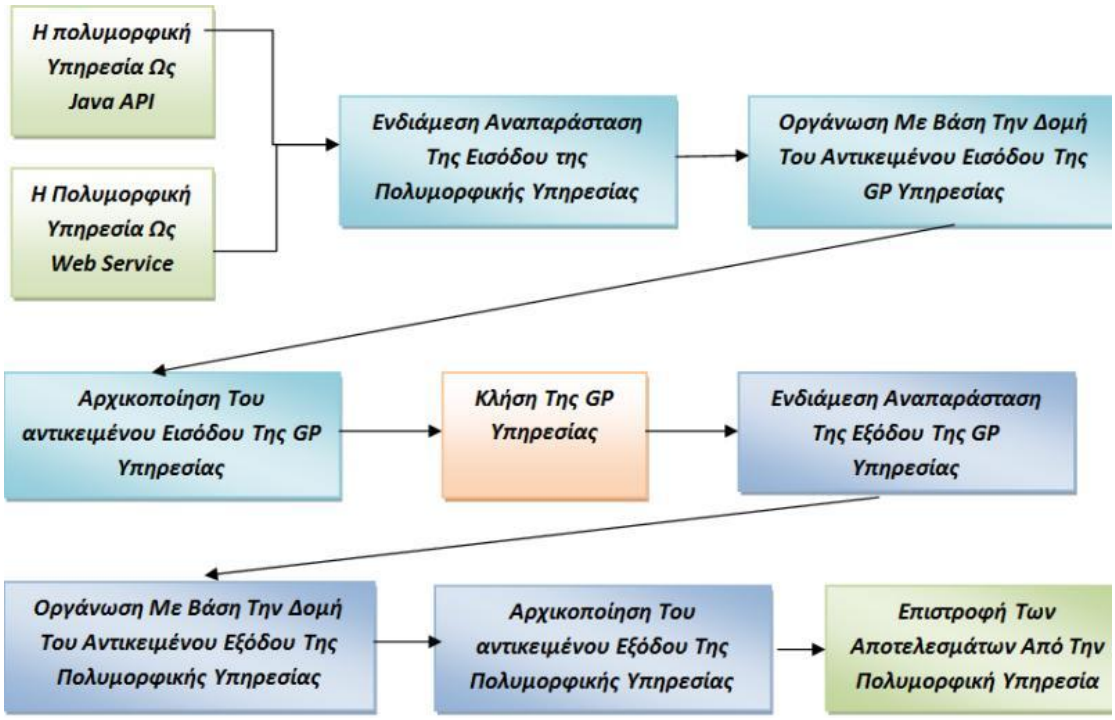
Στο Σχήμα 4.7 φαίνεται το διάγραμμα κλάσεων που παράγονται από τον μηχανισμό παραγωγής πολυμορφικών υπηρεσιών καθώς και οι συσχετίσεις τους με βασικές κλάσεις του πολυμορφικού μηχανισμού.



Σχήμα 4.7: Διάγραμμα Κλάσεων

4.2.2. Μηχανισμός Πολυμορφισμού

Σ’ αυτήν την ενότητα θα περιγράψουμε τον μηχανισμό πολυμορφισμού που προτείνεται από αυτήν την διατριβή. Ο μηχανισμός που προτείνουμε βασίζεται σε μια τεχνική που προτάθηκε στο [35] για XML μετασχηματισμούς και εμείς την υλοποιήσαμε για Java αντικείμενα. Πιο συγκεκριμένα, θα περιγράψουμε τα βήματα που χρειάζονται για να καλέσουμε την GP υπηρεσία που κρύβεται πίσω από την πολυμορφική υπηρεσία, καθώς και τα βήματα που πρέπει να κάνει ο μηχανισμός πολυμορφισμού ώστε να πάρουμε πίσω τα αποτελέσματα των κλήσεων όπως φαίνεται στο Σχήμα 4.8. Στην ενότητα 4.2.2.1 περιγράφεται πως γίνεται η κλήση της GP υπηρεσίας και στην ενότητα 4.2.2.2 περιγράφεται πως γίνεται η επιστροφή των αποτελεσμάτων των κλήσεων της πολυμορφικής υπηρεσίας.



Σχήμα 4.8: Ο Μηχανισμός Πολυμορφισμού

4.2.2.1. Κλήση της GP υπηρεσίας

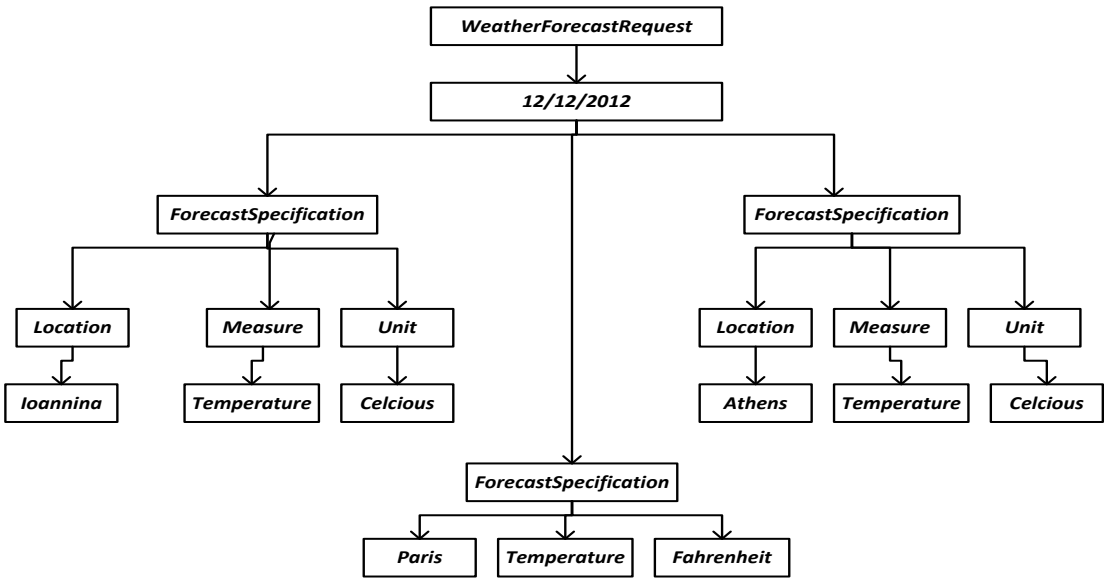
Σε αυτή την ενότητα θα περιγράψουμε την διαδικασία που απαιτείται για να κληθεί η GP υπηρεσία που κρύβεται πίσω από την πολυμορφική υπηρεσία. Πιο συγκεκριμένα η διαδικασία κάνει τα εξής βήματα:

1 Δημιουργία ενδιάμεσης αναπαράστασης της εισόδου της πολυμορφικής υπηρεσίας.
2 Οργάνωση με βάση την δομή του αντικειμένου εισόδου της GP υπηρεσίας.
3 Αρχικοποίηση του αντικειμένου εισόδου της GP υπηρεσίας.

4.2.2.1.1. Ενδιάμεση Αναπαράσταση

Το στάδιο αυτό είναι υπεύθυνο για την μετατροπή των δεδομένων της εισόδου της πολυμορφικής υπηρεσίας σε ένα πίνακα από πλειάδες. Οι πλειάδες αυτές χρησιμοποιούνται σαν μια ενδιάμεση μορφή αναπαράστασης από την οποία θα προκύψουν οι παράμετροι εισόδου για την υπηρεσία GP που κρύβεται πίσω από την πολυμορφική υπηρεσία.

Έστω ότι έχουμε την πολυμορφική υπηρεσία AbstractWeatherForecastService του παραδείγματος του κεφαλαίου 3. Έστω ότι έχουμε την μέθοδο abstractGetDailyWeatherForecast της πολυμορφικής υπηρεσίας που παίρνει σαν είσοδο ένα αντικείμενο WeatherForecastRequest. Έστω ότι έχουμε την GP υπηρεσία WeatherService και την μέθοδο getWeather της GP που παίρνει σαν είσοδο ένα αντικείμενο WeatherIn. Σκοπός του σταδίου αυτού είναι παράγουμε μια ενδιάμεση αναπαράσταση των δεδομένων της αφηρημένης υπηρεσίας. Έστω ότι έχουμε την είσοδο της πολυμορφικής μεθόδου που φαίνεται στο Σχήμα 4.9. Το αντικείμενο WeatherForecastRequest που φαίνεται στο Σχήμα 4.9 έχει στο πεδίο date την τιμή 12/12/12 και μια λίστα από τρία αντικείμενα ForecastSpecification. Το πρώτο αντικείμενο ForecastSpecification έχει στο πεδίο location τιμή Ioannina, στο πεδίο measure την τιμή Temperature και το πεδίο unit έχει την τιμή Celcious. Αντίστοιχα το δεύτερο αντικείμενο ForecastSpecification έχει στο πεδίο location τιμή Paris, στο πεδίο measure την τιμή Temperature και το πεδίο unit έχει την τιμή Fahrenheit. Τέλος το τρίτο αντικείμενο ForecastSpecification έχει στο πεδίο location τιμή Athens, στο πεδίο measure την τιμή Temperature και το πεδίο unit έχει την τιμή Celcious.



Σχήμα 4.9: Είσοδος της Αφηρημένης Μεθόδου

Η διαδικασία διασχίζει αναδρομικά το αντικείμενο weatherForecast που παίρνει ως είσοδο η αφηρημένη μέθοδος abstractGetDailyWeatherForecast. Αρχικά βρίσκει το πεδίο date που έχει την τιμή 12/12/2012 και βάζουμε την τιμή σε ένα υπό-πίνακα της ενδιάμεσης αναπαράστασης όπως φαίνεται στον Πίνακα 4.1.

Πίνακας 4.1: Ο Υπό-Πίνακας που Προκύπτει από το Πεδίο Date

	Date
T1	12/12/2012

Στην συνέχεια βάζουμε τις τιμές που προκύπτουν από κάθε στοιχείο της λίστας σε ένα υπό πίνακα όπως φαίνεται στον Πίνακα 4.2.

Πίνακας 4.2: Ο Υπό-Πίνακας που Προκύπτει από την Λίστα

	Location	Measure	Unit
T1	Ioannina	Temperature	Celcious
T2	Paris	Temperature	Fahrenheit
T3	Athens	Temperature	Celcious

Τέλος παίρνουμε όλους τους δυνατούς συνδυασμούς των γραμμών του Πίνακα 4.1 με τις γραμμές του Πίνακα 4.2. Η ενδιάμεση αναπαράσταση που προκύπτει για την είσοδο του Σχήματος 4.9 φαίνεται στον Πίνακα 4.3. Οι πολλαπλές πλειάδες που φαίνονται στον πίνακα προκύπτουν όταν κατά την αναδρομική διαδικασία συναντήσουμε ως πεδίο μια λίστα είτε βασικού τύπου είτε λίστα αντικειμένων.

Πίνακας 4.3: Τελικός Πίνακας Ενδιάμεσης Αναπαράστασης

	Date	Location	Measure	Unit
T1	12/12/2012	Ioannina	Temperature	Celcious
T2	12/12/2012	Paris	Temperature	Fahrenheit
T3	12/12/2012	Athens	Temperature	Celcious

Για να είναι η μέθοδος που διασχίζει το αντικείμενο της εισόδου ανεξάρτητη από το αντικείμενο της εισόδου η μέθοδος που δημιουργεί τον πίνακα της ενδιάμεσης

αναπαράστασης έχει υλοποιηθεί με την βοήθεια του μηχανισμού Reflection της Java. Στο πιο τεχνικό κομμάτι ο πίνακας ενδιάμεσης αναπαράστασης έχει υλοποιηθεί με την χρήση δισδιάστατων λιστών από Object. Πιο συγκεκριμένα έχουμε τόσες λίστες όσοι και βασικοί τύποι της εισόδου και το μέγεθος κάθε λίστας είναι όσες και οι διαφορετικές πλειάδες που προκύπτουν από τον συνδυασμό των επιμέρους υπό λιστών.

4.2.2.1.2. Οργάνωση με Βάση το Αντικείμενο της GP Υπηρεσίας

Σκοπός αυτού του σταδίου είναι να οργανώσουμε τα δεδομένα του πίνακα ενδιάμεσης αναπαράστασης (Πίνακας 4.3) που προκύπτει από το προηγούμενο βήμα της διαδικασίας σύμφωνα με την δομή του αντικειμένου που παίρνει η μέθοδος της GP υπηρεσίας.

Έστω ο πίνακας ενδιάμεσης αναπαράστασης που φαίνεται στον Πίνακα 4.3 και η μέθοδος getWeather της υπηρεσίας WeatherService που κρύβεται πίσω από την πολυμορφική υπηρεσία που παίρνει ως είσοδο το αντικείμενου WeatherIn. Η διαδικασία διασχίζει αναδρομικά τη δομή του αντικείμενο WeatherIn. Για τους βασικούς τύπους της κλάσης βρίσκουμε με την χρήση των αντιστοιχίσεων μεταξύ της εισόδου της πολυμορφικής υπηρεσίας και της εισόδου της GP υπηρεσίας (interface Mappings Σχήμα 4.7) σε ποια στήλη της ενδιάμεσης αναπαράστασης αντιστοιχίζονται. Το πεδίο date της κλάσης WeatherIn αντιστοιχίζεται με την στήλη date του Πίνακα 4.3 αντίστοιχα το πεδίο location αντιστοιχίζεται με την στήλη location που Πίνακα 4.3. Στον Πίνακα 4.4 φαίνονται οι τιμές των πλειάδων της ενδιάμεσης αναπαράστασης που αντιστοιχούν σ’ αυτούς τους βασικούς τύπους.

Πίνακας 4.4: Ποιό Κομμάτι του Πίνακα Χρησιμοποιείται

	Date	Location	Measure	Unit
T1	12/12/2012	Ioannina	Temperature	Celcious
T2	12/12/2012	Paris	Temperature	Fahrenheit
T3	12/12/2012	Athens	Temperature	Celcious

Κάθε διαφορετικός συνδυασμός τιμών των στηλών που αντιστοιχούν σε πεδία βασικού τύπου του αντικειμένου εισόδου αντιστοιχεί σε ένα διαφορετικό αντικείμενο WeatherIn. Τα δεδομένα κάθε διαφορετικού αντικειμένου WeatherIn αποθηκεύονται σε ένα Hash - Table όπου κλειδί κάθε στοιχείου του HashTable είναι οι τιμές των βασικών τύπων των διαφορετικών αντικειμένων WeatherIn όπως φαίνεται στο Σχήμα 4.10.

Σχήμα 4.10: Τα HashTable που Δημιουργούνται για τα Διαφορετικά WeatherIn

Κάθε ένα από τα κλειδιά μπορεί να συσχετίζεται με κανένα ή περισσότερα HashTable που αντιστοιχούν στα επιμέρους αντικείμενα που συνθέτουν το αντικείμενο WeatherIn. Πιο συγκεκριμένα στο παράδειγμα μας θα συσχετίζεται μόνο με ένα HashTable καθώς το αντικείμενο WeatherIn έχει μόνο ένα επιμέρους αντικείμενο την λίστα με αντικείμενα ForecastSpecification. Συγκεκριμένα όπως και στο αρχικό βήμα αποθηκεύουμε πάλι τις τιμές των βασικών τύπων του αντικειμένου ForecastSpecification στα αντίστοιχα HashTable. Κάθε ένα από τα επιμέρους σχετίζεται μόνο με το κομμάτι του πίνακα ενδιάμεσης αναπαράστασης που αντιστοιχεί στα διαφορετικά αντικείμενα WeatherIn. Για το HashTable του Σχήματος 4.9 έστω ότι παίρνουμε το στοιχείο του HashTable το οποίο έχει για τον βασικό τύπο date την τιμή 12/12/2012 και για τον βασικό τύπο location την τιμή Ioannina τότε οι τιμές από τον πίνακα ενδιάμεσης αναπαράστασης (Πίνακας 4.3) φαίνονται στον Πίνακα 4.5.

Πίνακας 4.5: Ποια Δεδομένα Χρησιμοποιούνται από την Ενδιάμεση Αναπαράσταση

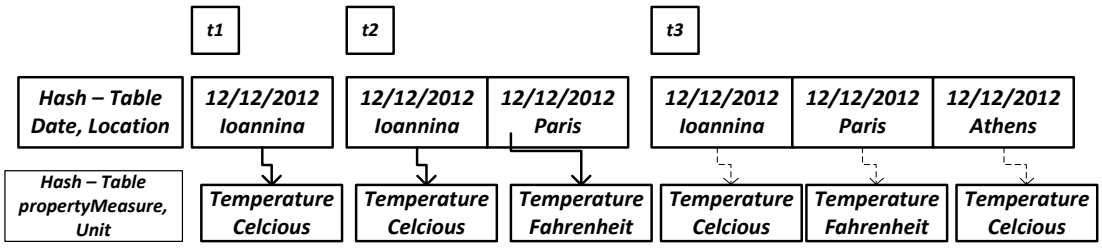
	Date	Location	Measure	Unit
T1	12/12/2012	Ioannina	Temperature	Celcious
T2	12/12/2012	Paris	Temperature	Fahrenheit
T3	12/12/2012	Athens	Temperature	Celcious

Το πεδίο propertyMeasure του αντικείμενου ForecastSpecification αντιστοιχίζεται στην στήλη measure της ενδιάμεσης αναπαράστασης και το πεδίο unit αντιστοιχίζεται στην στήλη unit ενδιάμεσης αναπαράστασης. Οι τιμές των βασικών τύπων φαίνονται στον Πίνακα 4.6.

Πίνακας 4.6: Ποιο Κομμάτι του Πίνακα Ενδιάμεσης Αναπαράστασης Χρησιμοποιείται

	Date	Location	Measure	Unit
T1	12/12/2012	Ioannina	Temperature	Celcious
T2	12/12/2012	Paris	Temperature	Fahrenheit
T3	12/12/2012	Athens	Temperature	Celcious

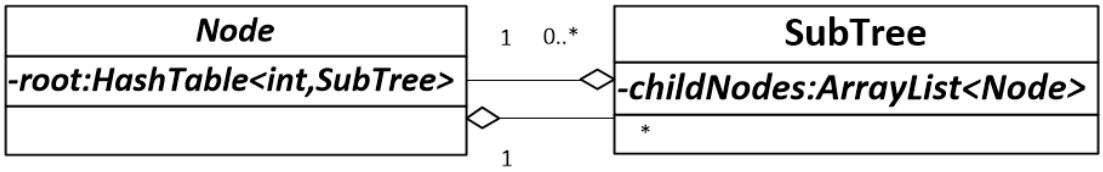
Η ίδια διαδικασία ακολουθείται και για όλα τα διαφορετικά αντικείμενα WeatherIn που δημιουργήθηκαν αρχικά, κάθε φορά με το αντίστοιχο κομμάτι του πίνακα ενδιάμεσης αναπαράστασης. Τα HashTable που δημιουργούνται φαίνονται στο Σχήμα 4.11.



Σχήμα 4.11: Τα Συνολικά HashTable που Δημιουργούνται για τα Διαφορετικά WeatherIn

Η διαδικασία η οποία είναι υπεύθυνη για την οργάνωση του πίνακα ενδιάμεσης αναπαράστασης σύμφωνα με το αντικείμενο της εισόδου της GP υπηρεσίας για να είναι ανεξάρτητη της εισόδου έχει υλοποιηθεί με βάση τον μηχανισμό Reflection της Java. Οι δομές που χρησιμοποιούνται για την αναπαράσταση του πίνακα ενδιάμεσης αναπαράστασης με βάση το αντικείμενο της εισόδου είναι ένα HashTable

<int,SubTree>. Για το κλειδί του HashTable χρησιμοποιούμε έναν ακέραιο που είναι μοναδικός και σχετίζεται με την τιμή των εκάστοτε βασικών τύπων. Το SubTree είναι η δομή που κρατάει μια λίστα με τα επιμέρους αντικείμενα που απαρτίζουν το αντικείμενο της εισόδου. Η δομή φαίνεται στο Σχήμα 4.12.

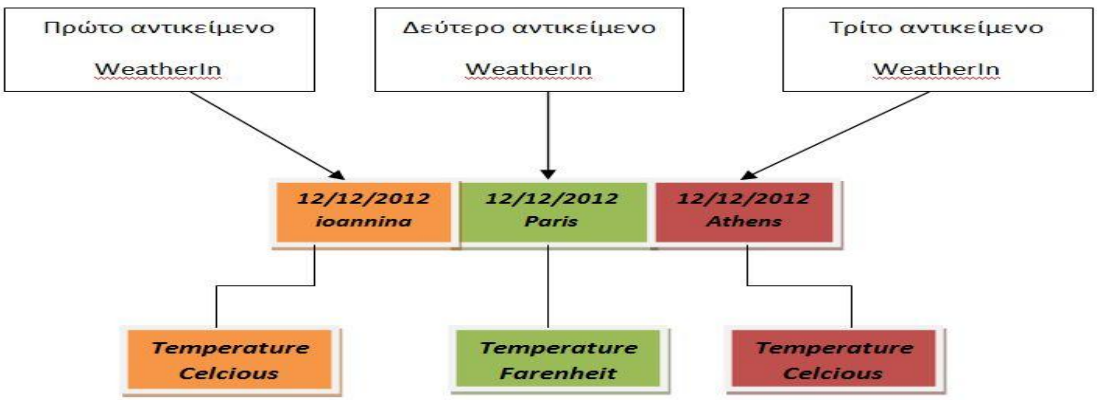


Σχήμα 4.12: Το Διάγραμμα Κλάσεων των Δομών που Χρησιμοποιούνται

4.2.2.1.3. Αρχικοποίηση του Αντικειμένου της GP Υπηρεσίας

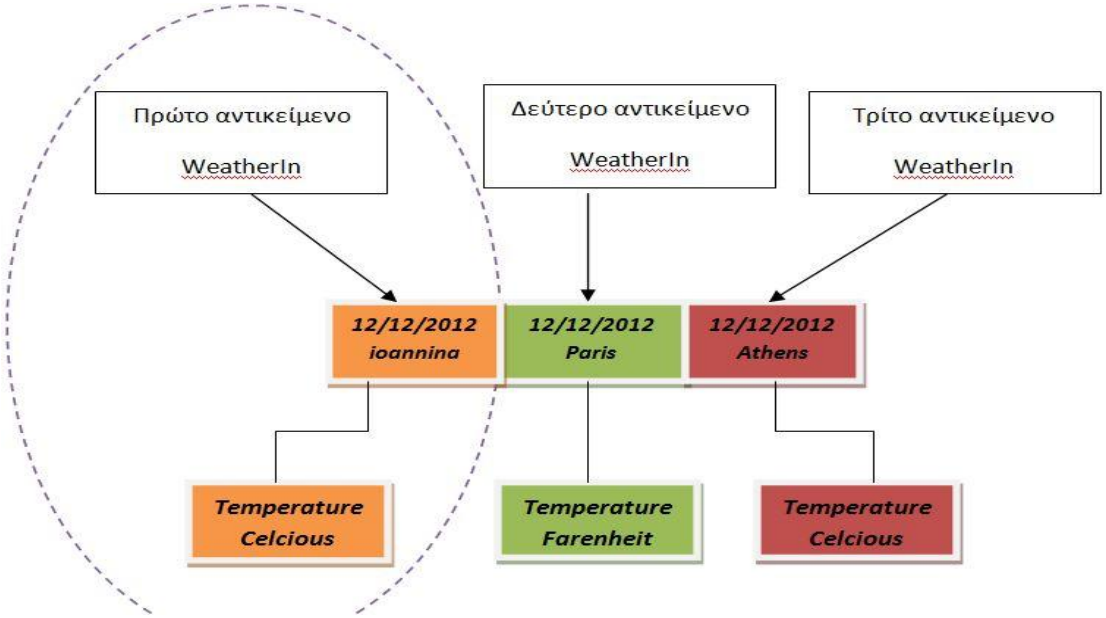
Στόχος αυτού του σταδίου είναι να χρησιμοποιηθεί η δομή που προκύπτει από το προηγούμενο στάδιο (Σχήμα 4.10) και να αρχικοποιηθεί το αντικείμενο που παίρνει ως είσοδο η GP υπηρεσία που κρύβεται πίσω από την πολυμορφική υπηρεσία.

Έστω ότι έχουμε την δομή του αντικειμένου WeatherIn που φαίνεται στο Σχήμα 4.10 που παίρνει ως είσοδο η μέθοδος getWeather(). Κάθε στοιχείο του HashTable αντιστοιχεί σε διαφορετικά αντικείμενα WeatherIn. Στο Σχήμα 4.10 που προέκυψε από την δομή του αντικειμένου προκύπτουν τρία διαφορετικά αντικείμενα όπως φαίνεται στο Σχήμα 4.13.



Σχήμα 4.13: Τα Διαφορετικά Αντικείμενα WeatherIn

Ο λόγος για τον οποίο προκύπτουν διαφορετικά αντικείμενα είναι γιατί μπορεί ένας βασικός τύπος της εισόδου της GP υπηρεσίας να αντιστοιχιστεί σε μια λίστα στο αντικείμενο εισόδου της πολυμορφικής υπηρεσίας λόγω αυτής της αντιστοίχισης για τον βασικό τύπο της GP προκύπτουν διαφορετικές τιμές άρα και διαφορετικά αντικείμενα. Κατά την αρχικοποίηση του αντικειμένου WeatherIn επιλέγουμε με τυχαίο τρόπο ανάμεσα στα στοιχεία του HashTable που προέκυψαν από το προηγούμενο βήμα έστω ότι επιλέγουμε το πρώτο από τα τρία αντικείμενα όπως φαίνεται στο Σχήμα 4.14.



Σχήμα 4.14: Επιλογή Αντικειμένου

Το πεδίο date του αντικείμενου WeatherIn παίρνει την τιμή 12/12/2012 και το πεδίο location την τιμή Ioannina. Στην συνέχεια αρχικοποιούμε το αντικείμενο ForecastSpecification. Το πεδίο propertyMeasure παίρνει την τιμή Temperature και το πεδίο unit παίρνει την τιμή Celcious.

Η διαδικασία της αρχικοποίησης του αντικειμένου της εισόδου βασίζεται στον μηχανισμό Reflection της Java ώστε η διαδικασία να είναι ανεξάρτητη από το εκάστοτε αντικείμενο της εισόδου της GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία.

4.2.2.2. Αποτελέσματα Κλήσεων της Πολυμορφικής Υπηρεσίας

Σ’ αυτήν την ενότητα θα περιγράψουμε τα βήματα που κάνει ο μηχανισμός πολυμορφισμού προκειμένου να επιστρέψουμε τα αποτελέσματα της κλήσης της πολυμορφικής υπηρεσίας. Πιο συγκεκριμένα τα βήματα που κάνει ο μηχανισμός πολυμορφισμού είναι:

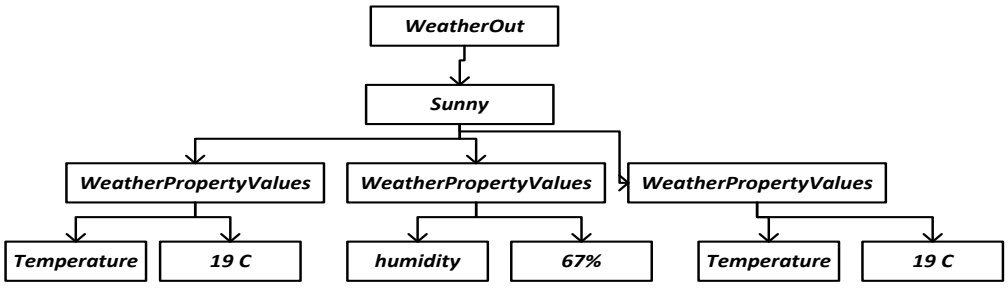
1. Δημιουργία ενδιάμεσης αναπαράστασης της εξόδου της πολυμορφικής υπηρεσίας
2. Οργάνωση των δεδομένων του πίνακα ενδιάμεσης αναπαράστασης με βάση την δομή του αντικειμένου εξόδου της πολυμορφικής υπηρεσίας.
3. Αρχικοποίηση του αντικειμένου εξόδου της πολυμορφικής υπηρεσίας

4.2.2.2.1. Ενδιάμεση αναπαράσταση

Το στάδιο αυτό είναι υπεύθυνο για την μετατροπή των δεδομένων της εξόδου της GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία σε ένα πίνακα από πλειάδες. Οι πλειάδες αυτές χρησιμοποιούνται σαν μια ενδιάμεση μορφή αναπαράστασης από το οποίο θα προκύψει το αντικείμενο εξόδου της πολυμορφικής υπηρεσίας.

Έστω ότι έχουμε την λειτουργία της GP υπηρεσίας getWeather που κρύβεται πίσω από την πολυμορφική υπηρεσία του παραδείγματος του Κεφαλαίου 3 η οποία έχει ως έξοδο το αντικείμενο WeatherOut. Σκοπός αυτού του σταδίου είναι να παράγουμε μια ενδιάμεση αναπαράσταση των δεδομένων εξόδου της GP υπηρεσίας. Έστω ότι έχουμε την έξοδο της GP υπηρεσίας που φαίνεται στο Σχήμα 4.15. Το αντικείμενο εξόδου WeatherOut του Σχήματος 4.15 έχει στο πεδίο status την τιμή “sunny” και μία λίστα από τρία αντικείμενα WeatherPropertyValues. Το πρώτο αντικείμενο WeatherPropertyValues έχει στο πεδίο propertyMeasure την τιμή Temperature και στο πεδίο propertyValue την τιμή 19 C. Αντίστοιχα το δεύτερο αντικείμενο WeatherPropertyValues έχει στο πεδίο propertyMeasure την τιμή Humidity και στο πεδίο propertyValue έχει την τιμή 67%. Τέλος το τρίτο αντικείμενο

WeatherPropertyValues έχει στο πεδίο propertyMeasure την τιμή Temperature και το πεδίο propertyValue έχει την τιμή 32 C.



.

Σχήμα 4.15: Έξοδος της GP Υπηρεσίας

Η διαδικασία διασχίζει αναδρομικά το αντικείμενο WeatherOut που έχει ως έξοδο η GP υπηρεσία που κρύβεται πίσω από την πολυμορφική υπηρεσία. Αρχικά βρίσκουμε το πεδίο status που έχει την τιμή "sunny" και βάζουμε την τιμή σε ένα υπό-πίνακα της ενδιάμεσης αναπαράστασης όπως φαίνεται στον Πίνακα 4.7.

Πίνακας 4.7: Ο Υπό Πίνακας που Προκύπτει από το Πεδίο Status

	Status
T1	Sunny

Στην συνέχεια βάζουμε όλες τις τιμές της λίστας σε ένα υπό-πίνακα όπως φαίνεται στον Πίνακα 4.8 όπου κάθε γραμμή του πίνακα αντιστοιχεί σε ένα στοιχείο της λίστας.

Πίνακας 4.8: Ο Υπό Πίνακας που Προκύπτει από την Λίστα

	PropertyMeasure	PropertyValue
T1	Temperature	19 C
T2	Humidity	67%
T3	Temperature	32 C

Τέλος παίρνουμε όλους τους δυνατούς συνδυασμούς των γραμμών του Πίνακα 4.7 με τις γραμμές του Πίνακα 4.8. η ενδιάμεση αναπαράσταση που προκύπτει για την έξοδο της GP υπηρεσίας που φαίνεται στο Σχήμα 4.15 φαίνεται στον Πίνακα 4.9. Οι πολλαπλές πλειάδες που φαίνονται στον πίνακα προκύπτουν όταν κατά την αναδρομική διαδικασία συναντήσουμε μια λίστα είτε βασικού είτε λίστα αντικειμένων.

Πίνακας 4.9: Τελικός Πίνακας Ενδιάμεσης Αναπαράστασης

	Status	PropertyMeasure	PropertyValue
T1	Sunny	Temperature	19 C
T2	Sunny	Humidity	67%
T3	Sunny	Temperature	32 C

Η διαδικασία υλοποιείται με τον ίδιο τρόπο που περιγράφτηκε για την είσοδο στην ενότητα 4.2.2.1.1. και ο τρόπος υλοποίησης των πινάκων είναι ίδιος με αυτόν που περιγράφτηκε στην ενότητα 4.2.2.1.1.

4.2.2.2.2. Οργάνωση Με Βάση Την Δομή Της Εξόδου Της Πολυμορφικής Υπηρεσίας

Σκοπός του σταδίου αυτού είναι η οργάνωση των δεδομένων του πίνακα ενδιάμεσης αναπαράστασης της εξόδου της GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία που προκύπτει από το προηγούμενο βήμα με βάση την δομή του αντικειμένου εξόδου της πολυμορφικής υπηρεσίας.

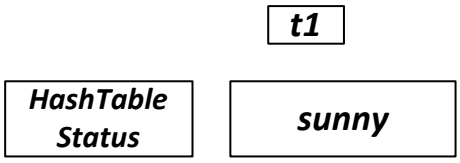
Έστω ο πίνακας ενδιάμεσης αναπαράστασης της εξόδου που φαίνεται στον Πίνακα 4.9 και η μέθοδος getDailyWeatherForecast της πολυμορφικής υπηρεσίας που έχει ως έξοδο ένα αντικείμενο WeatherForecastResponse. Η διαδικασία διασχίζει αναδρομικά το αντικείμενο WeatherForecastResponse. Για τους βασικούς τύπους της κλάσης βρίσκουμε μέσω των αντιστοιχίσεων μεταξύ των εξόδων της GP υπηρεσίας και των εξόδων της πολυμορφικής υπηρεσίας (interfaceMappings Σχήμα 4.7) σε ποια στήλη του πίνακα ενδιάμεσης αναπαράστασης αντιστοιχίζονται. Το πεδίο status της κλάσης WeatherForecastResponse αντιστοιχίζεται στην στήλη status του Πίνακα 4.9.

Στον Πίνακα 4.10 φαίνονται οι τιμές των πλειάδων της ενδιάμεσης αναπαράστασης της εξόδου της GP υπηρεσίας που αντιστοιχεί στο πεδίο status.

Πίνακας 4.10: Ποιο Κομμάτι του Πίνακα Χρησιμοποιείται

	Status	PropertyMeasure	PropertyValue
T1	Sunny	Temperature	19 C
T2	Sunny	Humidity	67%
T3	Sunny	Temperature	32 C

Κάθε διαφορετικός συνδυασμός τιμών των στηλών που αντιστοιχούν σε πεδία βασικού τύπου του αντικείμενου εξόδου αντιστοιχεί σε διαφορετικό αντικείμενο WeatherForecastRequest. Στο συγκεκριμένο παράδειγμα επειδή η τιμή “sunny” επαναλαμβάνεται έχουμε ένα μόνο αντικείμενο WeatherForecastRequest. Τα δεδομένα αποθηκεύονται σε ένα HashTable όπου κλειδί είναι η τιμή του βασικού τύπου όπως φαίνεται στο Σχήμα 4.16.



Σχήμα 4.16: Το HashTable του Αντικειμένου WeatherForecastResponse

Κάθε ένα από τα κλειδιά μπορεί να συσχετίζεται με κανένα ή περισσότερα HashTable που αντιστοιχούν στα επιμέρους αντικείμενα που συνθέτουν το αντικείμενο WeatherForecastResponse. Πιο Συγκεκριμένα στο παράδειγμα μας θα συσχετίζεται μόνο με ένα HashTable καθώς το αντικείμενο WeatherOut έχει μόνο ένα επιμέρους αντικείμενο την λίστα με αντικείμενα ForecastDetails. Συγκεκριμένα όπως και στο αρχικό βήμα αποθηκεύουμε πάλι τις τιμές των βασικών τύπων του αντικειμένου ForecastDetails σε αντίστοιχα HashTable. Κάθε ένα από τα επιμέρους αντικείμενα σχετίζεται μόνο με το κομμάτι του πίνακα ενδιάμεσης αναπαράστασης που αντιστοιχεί στα διαφορετικά αντικείμενα WeatherForecastResponse. Στο παράδειγμα μας έχουμε μόνο ένα αντικείμενο WeatherForecastResponse που σχετίζεται με το

κομμάτι του πίνακα ενδιάμεσης αναπαράστασης (Πίνακας 4.9) που φαίνεται στον Πίνακα 4.11.

Πίνακας 4.11: Ποια Δεδομένα Χρησιμοποιούνται από την Ενδιάμεση Αναπαράσταση

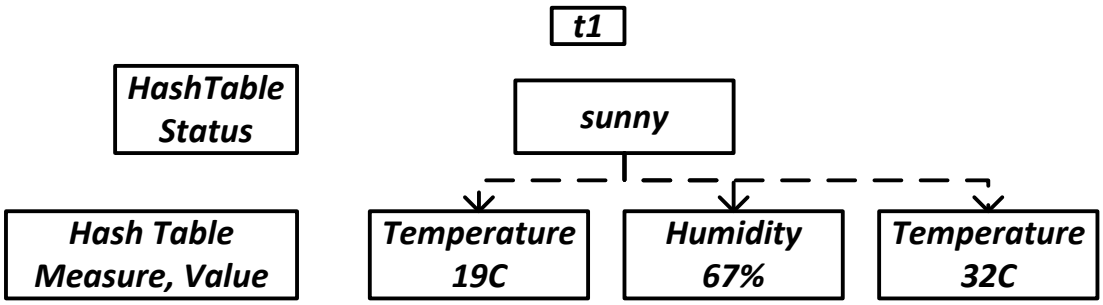
	Status	PropertyMeasure	PropertyValue
T1	Sunny	Temperature	19 C
T2	Sunny	Humidity	67%
T3	Sunny	Temperature	32 C

Το πεδίο measure του αντικειμένου ForecastDeails αντιστοιχίζεται στην στήλη propertyMeasure της ενδιάμεσης αναπαράστασης και το πεδίο value στην στήλη propertyValue της ενδιάμεσης αναπαράστασης. Οι τιμές των βασικών τύπων φαίνονται στον Πίνακα 4.12.

Πίνακας 4.12: Ποιο Κομμάτι του Πίνακα Χρησιμοποιείται

	Status	PropertyMeasure	PropertyValue
T1	Sunny	Temperature	19 C
T2	Sunny	Humidity	67%
T3	Sunny	Temperature	32 C

Επομένως θα προκύψουν τρείς εγγραφές στο HashTable της λίστας όπως φαίνεται στο Σχήμα 4.17.



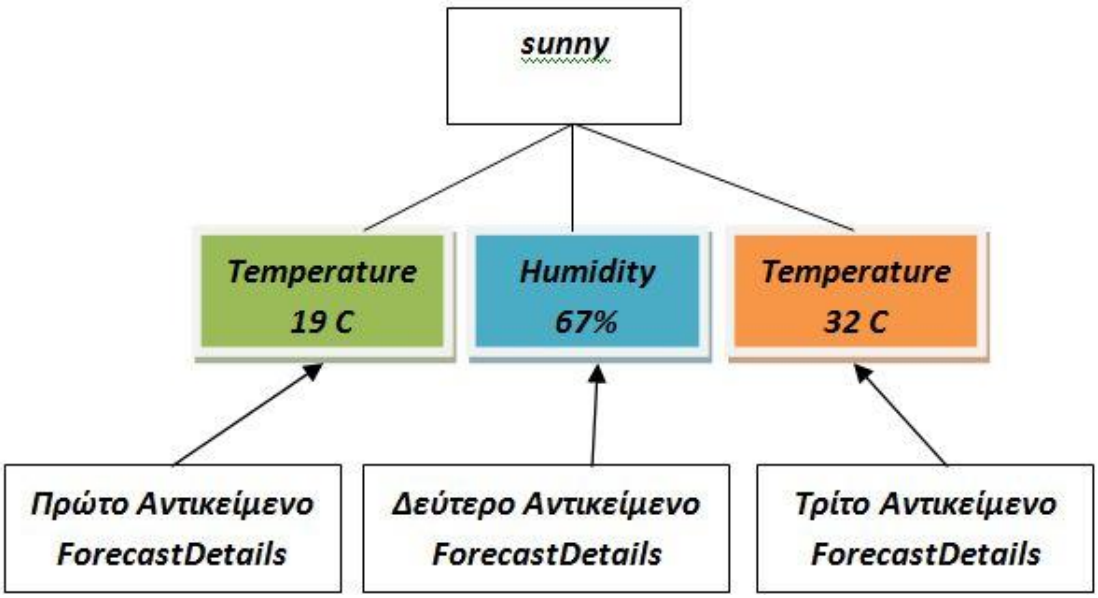
Σχήμα 4.17: Το Σύνολο των HashTable που Δημιουργούνται

Τόσο η δομές που χρησιμοποιούνται για την αποθήκευση των δεδομένων όσο και ο τρόπος που υλοποιείται η διαδικασία είναι ίδιες με αυτές που περιγράφονται στην ενότητα 4.2.2.1.2.

4.2.2.2.3. Αρχικοποίηση Του Αντικειμένου Εξόδου Της Πολυμορφικής Υπηρεσίας

Σκοπός αυτού του σταδίου είναι χρησιμοποιώντας την δομή του αντικείμενου εξόδου που δημιουργήθηκε στο προηγούμενο στάδιο να αρχικοποιήσει το αντικείμενο εξόδου της πολυμορφικής υπηρεσίας.

Έστω ότι έχουμε την δομή του αντικειμένου WeatherForecastResponse που φαίνεται στο Σχήμα 4.17. Η διαδικασία διασχίζει αναδρομικά το αντικείμενο WeatherForecastResponse. Επιλέγουμε από τα HashTable ένα από τα πιθανά αντικείμενα που μπορούν να υπάρξουν. Στο παράδειγμα όπως φαίνεται και στο Σχήμα 4.17 έχουμε ένα μόνο πιθανό αντικείμενο που μπορεί να προκύψει. Το πεδίο Status παίρνει την τιμή “sunny” που είναι αποθηκευμένη στο HashTable. Για την λίστα από το Σχήμα 4.17 προκύπτει ότι θα έχει τρία στοιχεία όπως φαίνεται στο Σχήμα 4.18.



Σχήμα 4.18: Τα Αντικείμενα της Λίστας

Η διαδικασία δημιουργεί ένα αντικείμενο ForecastDetails που στο πεδίο measure έχει την τιμή Temperature και στο πεδίο value έχει την τιμή 19 C και στην συνέχεια το εισάγει στην λίστα. Η ίδια διαδικασία επαναλαμβάνεται και για τα άλλα δύο αντικείμενα που φαίνονται στο Σχήμα 4.18.

Η διαδικασία της αρχικοποίησης του αντικειμένου της εξόδου βασίζεται στον μηχανισμό Reflection της Java ώστε η διαδικασία να είναι ανεξάρτητη από το εκάστοτε αντικείμενο εξόδου της πολυμορφικής υπηρεσίας.

4.2.3. Μηχανισμός Αντικατάστασης

Ο μηχανισμός αντικατάστασης είναι υπεύθυνος για την αντικατάσταση της GP υπηρεσίας που χρησιμοποιείται με κάποια άλλη GP υπηρεσία που εκπροσωπείται από την πολυμορφική υπηρεσία. Γ’ αυτό τον σκοπό προσφέρονται τρεις λειτουργίες που είναι υπεύθυνες για την αντικατάσταση της GP υπηρεσίας . Πιο συγκεκριμένα μέσω της κλάσης BaseAbstractionServiceImpl (Σχήμα 4.7) οι λειτουργίες register, unregister και η λειτουργία adapt. Οι λειτουργίες θα περιγραφούν παρακάτω:

- Η λειτουργία register (Σχήμα 4.7) χρησιμοποιείται από τις ***S*** υπηρεσίες για να δηλώσουν ότι χρησιμοποιούν την πολυμορφική υπηρεσία ώστε να μπορούν να ενημερωθούν ότι η GP υπηρεσία που χρησιμοποιείται πρόκειται να αλλάξει.

- Η λειτουργία unregister χρησιμοποιείται από τις ***S*** υπηρεσίες για να δηλώσουν ότι πλέον δεν χρησιμοποιούν την πολυμορφική υπηρεσία άρα δεν θέλουν να ενημερώνονται για τυχόν αλλαγές στην GP υπηρεσία που χρησιμοποιείται.

- Τέλος η λειτουργία adapt (Σχήμα 4.7) είναι η λειτουργία του μηχανισμού αναπροσαρμογής που είναι υπεύθυνη για την αντικατάσταση μιας GP υπηρεσίας με κάποια άλλη GP υπηρεσία που εκπροσωπείται από τον μηχανισμό πολυμορφισμού. Πιο συγκεκριμένα η λειτουργία αυτή παίρνει ως είσοδο το στιγμιότυπο της GP υπηρεσίας που θέλουμε να χρησιμοποιήσουμε καθώς και το όνομα της διεπαφής της GP υπηρεσίας και στην συνέχεια κάνει όλες τις απαραίτητες ενέργειες για την αντικατάσταση της τρέχουσας GP υπηρεσίας με αυτήν που δίνεται ως είσοδο. Η διαδικασία της αντικατάστασης περιλαμβάνει τρία στάδια:

    1. Το πρώτο στάδιο είναι υπεύθυνο για την ενημέρωση όλων των ***S*** υπηρεσιών που χρησιμοποιούν την πολυμορφική υπηρεσία για την επικείμενη αντικατάσταση της GP υπηρεσίας ώστε αυτή να γίνει με συνεπή τρόπο.
    2. Το δεύτερο στάδιο είναι υπεύθυνο για την αντικατάσταση της GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία με αυτήν που έχει πάρει ως είσοδο η διαδικασία. Πιο συγκεκριμένα το στάδιο αυτό ενημερώνει τα πεδία serviceInstance και serviceInterface (Σχήμα 4.7) της κλάσης BaseAbstractionServiceImpl. Επίσης σ’ αυτό το σημείο γίνεται και η παραγωγή των απαραίτητων stubs για την κλήση της νέας GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία. Η παραπάνω αλλαγές γίνονται με την κλήση των λειτουργιών της κλάσης BaseAbstractionServiceImpl setServiceInstance και setServiceInterface.

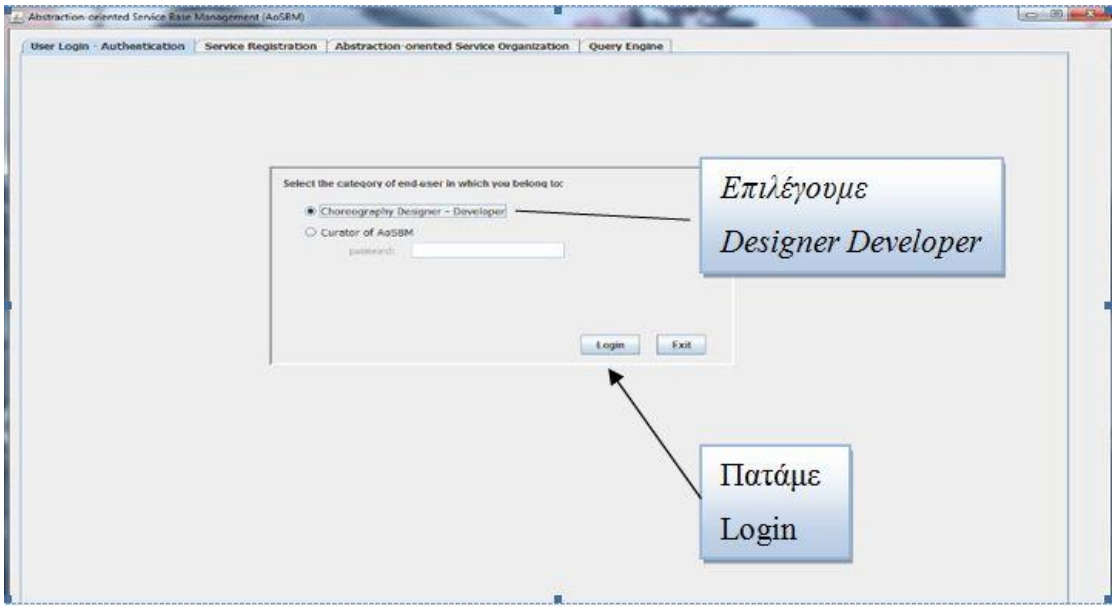
3. Το τελευταίο στάδιο είναι υπεύθυνο για την ειδοποίηση όλων των ***S*** υπηρεσιών που χρησιμοποιούν των μηχανισμό πολυμορφισμού ότι η διαδικασία προσαρμογής τελείωσε.

Σ’ αυτό το σημείο ας δώσουμε ένα παράδειγμα έστω ότι έχουμε την πολυμορφική υπηρεσία AbstractWeatherForecastService (Κεφάλαιο 3) και τις GP υπηρεσίες WeatherService και WeatherForecastService. Έστω ότι με την χρήση της λειτουργίας register (Σχήμα 4.7) χρησιμοποιούν την πολυμορφική υπηρεσία 10 S υπηρεσίες και η GP υπηρεσία που κρύβεται πίσω από την πολυμορφική υπηρεσία είναι η GP υπηρεσία WeatherService. Έστω λοιπόν ότι κάποια στιγμή θέλουμε να αντικαταστήσουμε την GP υπηρεσία που κρύβεται πίσω από τον μηχανισμό πολυμορφισμού με την GP υπηρεσία WeatherForecastService που εκπροσωπείται από την αφηρημένη υπηρεσία. Το AdminService (Σχήμα 4.2) καλεί την διαδικασία αναπροσαρμογής (adapt Σχήμα 4.7). Πιο συγκεκριμένα αρχικά ειδοποιούνται οι 10 S υπηρεσίες για την επικείμενη αλλαγή. Στην συνέχεια γίνεται η αντικατάσταση της GP υπηρεσίας WeatherService με την GP υπηρεσία WeatherForecastService. Για να γίνει αυτό χρησιμοποιούνται οι λειτουργίες setServiceInstance και setServiceInterface για να αλλάξουν οι τιμές των serviceInstance και serviceInterface της κλάσης BaseAbstractionServiceImpl (Σχήμα 4.7). Τέλος η μέθοδος ειδοποιεί τις 10 S υπηρεσίες πως η διαδικασία αναπροσαρμογής έχει τελειώσει.

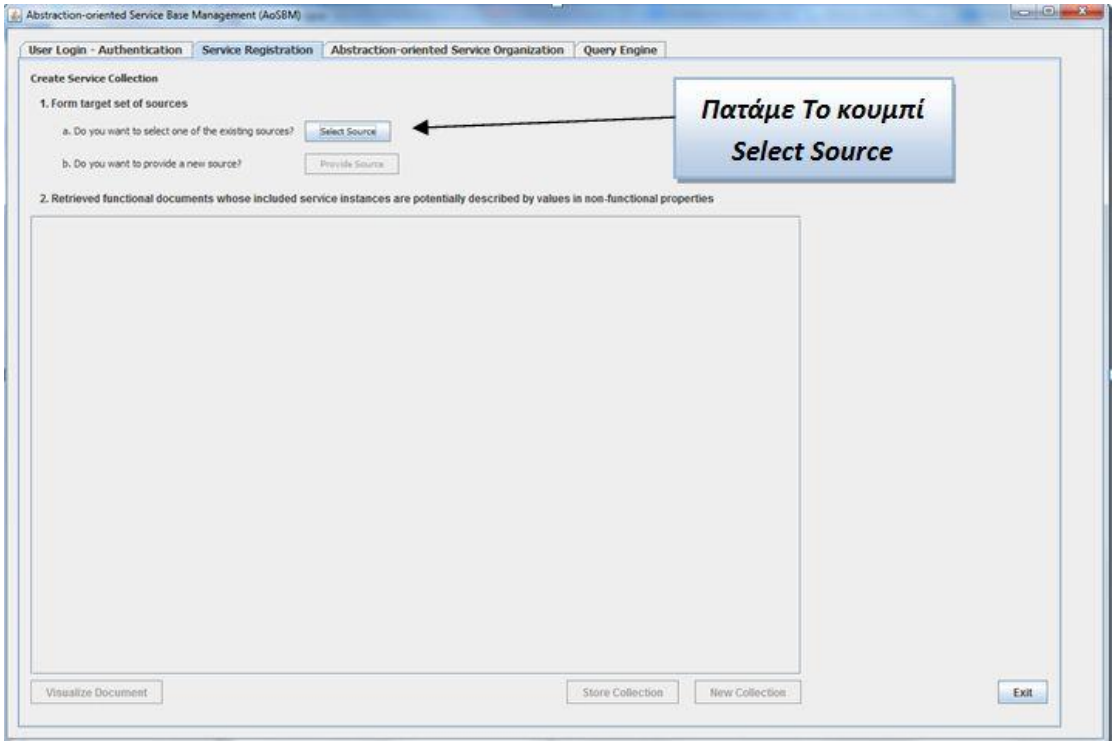
4.3. Tool Support

Για να καταστεί δυνατή η δημιουργία πολυμορφικών υπηρεσιών επεκτείναμε ένα υπάρχον εργαλείο που κατασκευάζει αφηρημένες υπηρεσίες σύμφωνα με την προσέγγιση που προτάθηκε στο [5]. Πιο συγκεκριμένα η διαδικασία που ακολουθείται για την δημιουργία της πολυμορφικής υπηρεσίας περιγράφεται παρακάτω:

- Στο αρχικό πλαίσιο του εργαλείου εξόρυξης αφηρημένων υπηρεσιών επιλέγουμε Choreography Designer-Developer και πατάμε το κουμπί Login όπως φαίνεται στο Σχήμα 4.19. και μεταφερόμαστε στο πλαίσιο του εργαλείου που φαίνεται στο Σχήμα 4.20.

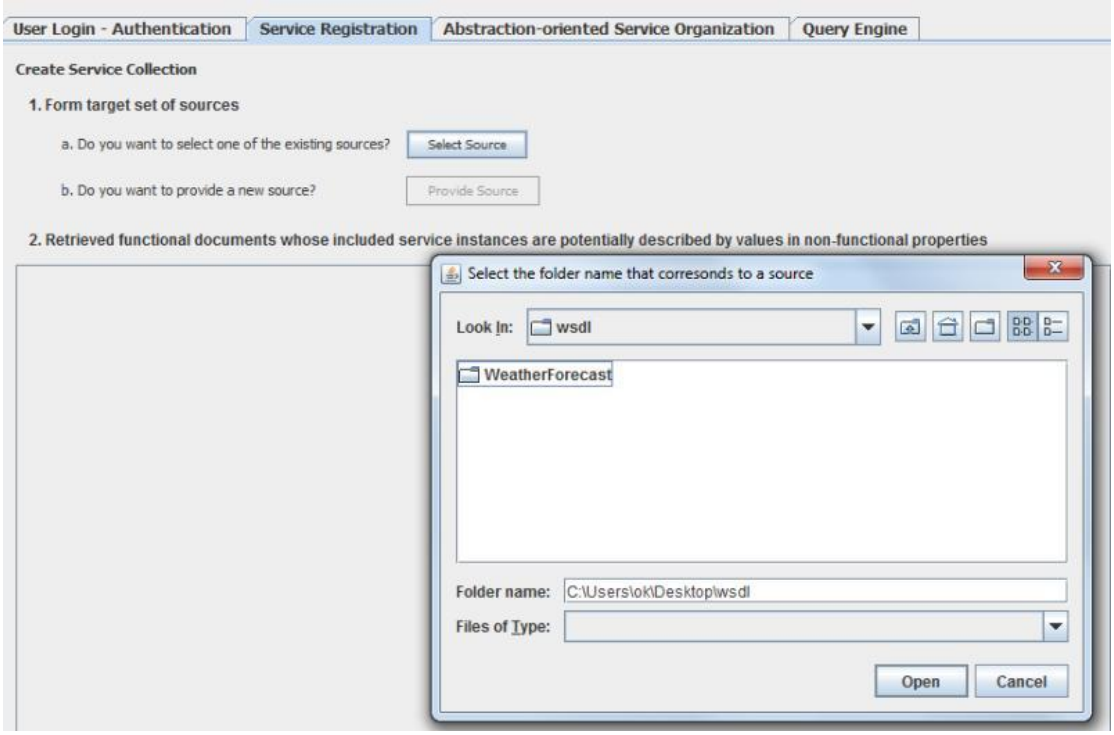


Σχήμα 4.19: Αρχικό Πλαίσιο Εργαλείου



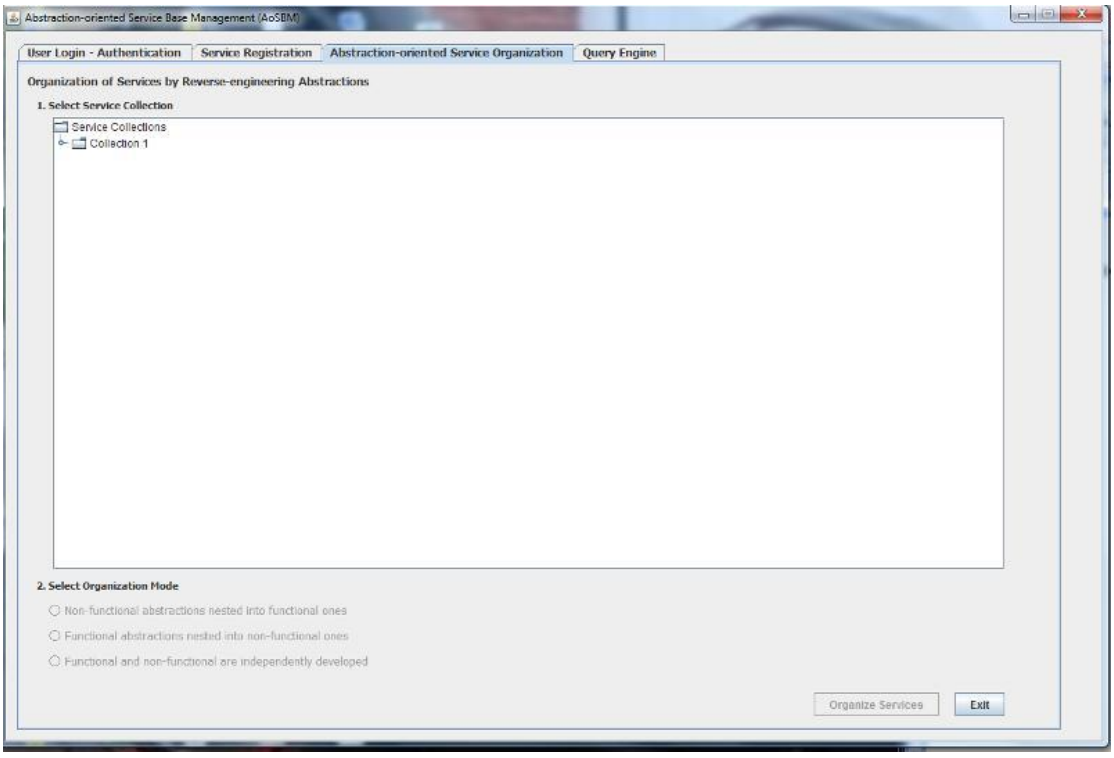
Σχήμα 4.20: Πλαίσιο Επιλογής των WSDL των Υπηρεσιών

- Στο πλαίσιο που εμφανίζεται (Σχήμα 4.20) πατάμε του κουμπί Select Source και εμφανίζεται ένας File chooser (Σχήμα 4.21) για να επιλέξουμε τον φάκελο στον οποίο βρίσκονται τα WSDL των υπηρεσιών που θέλουμε να δημιουργήσουμε την αφηρημένη υπηρεσία.



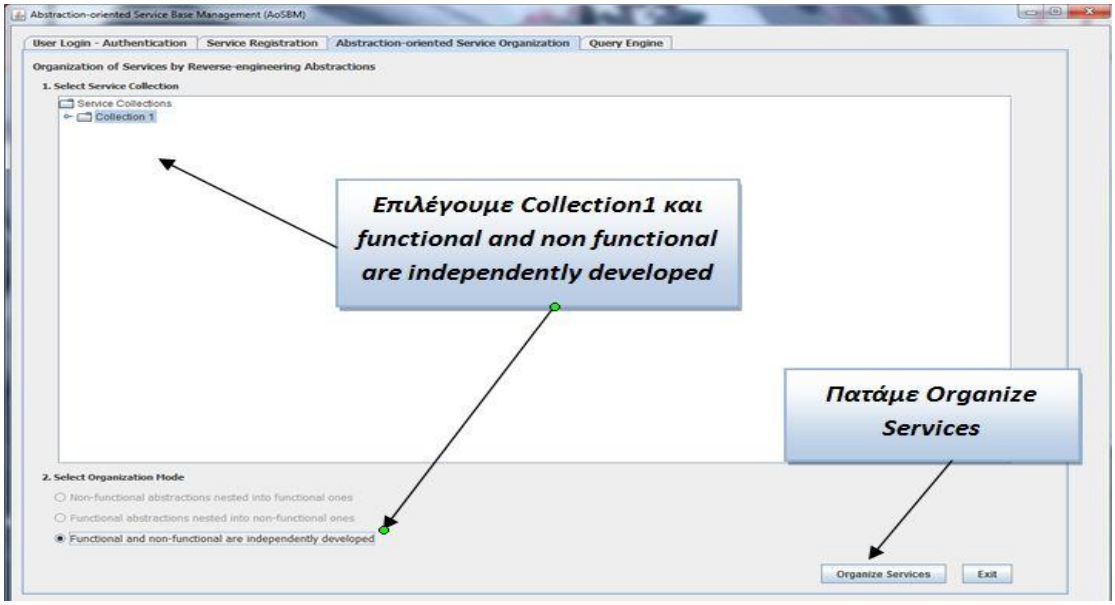
Σχήμα 4.21: Επιλογή Φακέλου που Περιέχει το WSDL

- Επιλέγουμε την καρτέλα με όνομα Abstraction-oriented Service Organization και μεταφερόμαστε στο πλαίσιο που φαίνεται στο Σχήμα 4.22.



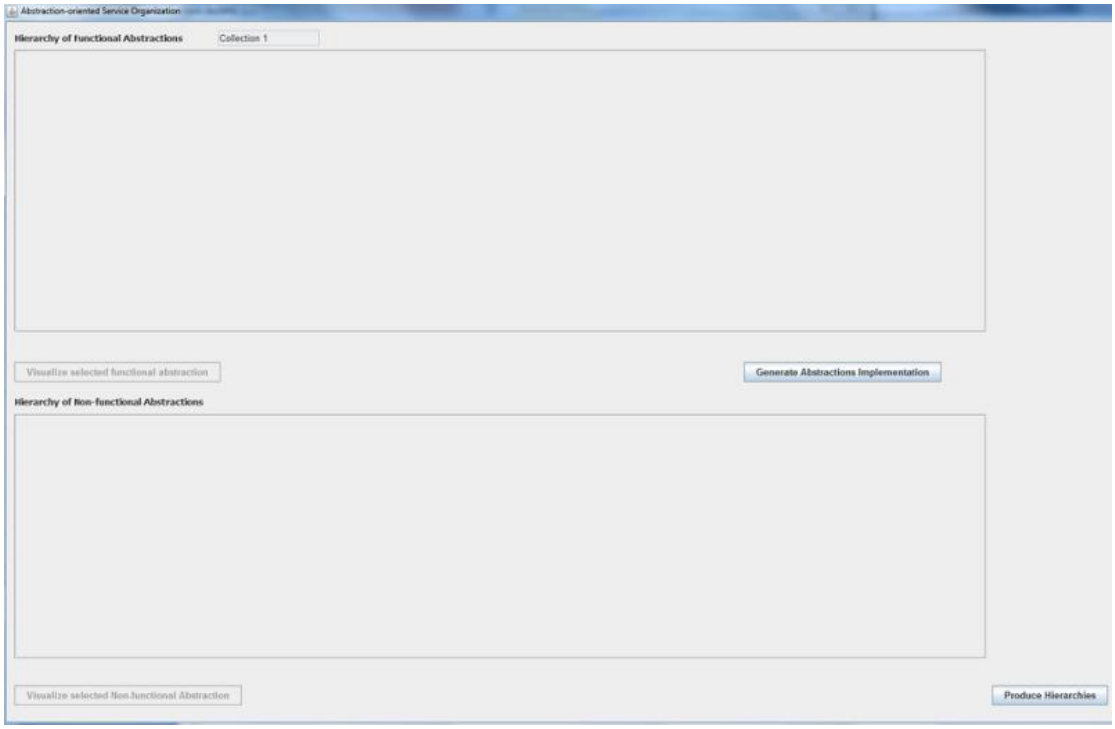
Σχήμα 4.22: Καρτέλα Abstraction-oriented Service Organization

- Στην καρτέλα Abstraction-oriented Service Organization επιλέγουμε *Collection1* και *functional and non functional are independently developed* όπως φαίνεται στο Σχήμα 4.22 και πατάμε το κουμπί *Organize Services.*



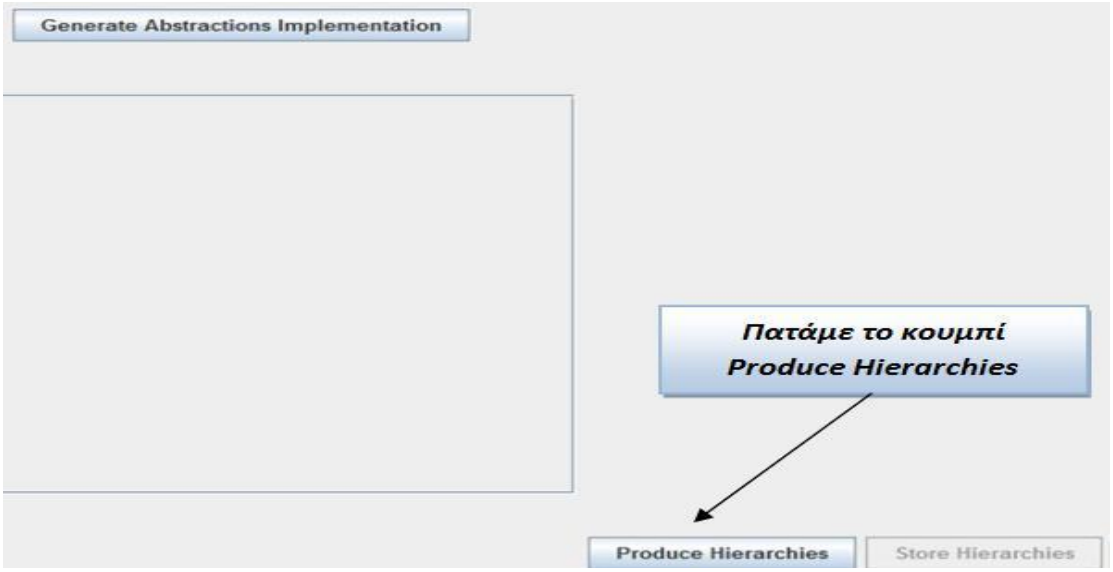
Σχήμα 4.23: Organize Services

- Στο πλαίσιο του εργαλείου που ανοίγει Σχήμα 4.24 γίνεται η δημιουργία της αφηρημένης υπηρεσίας όσο και της πολυμορφικής υπηρεσίας.



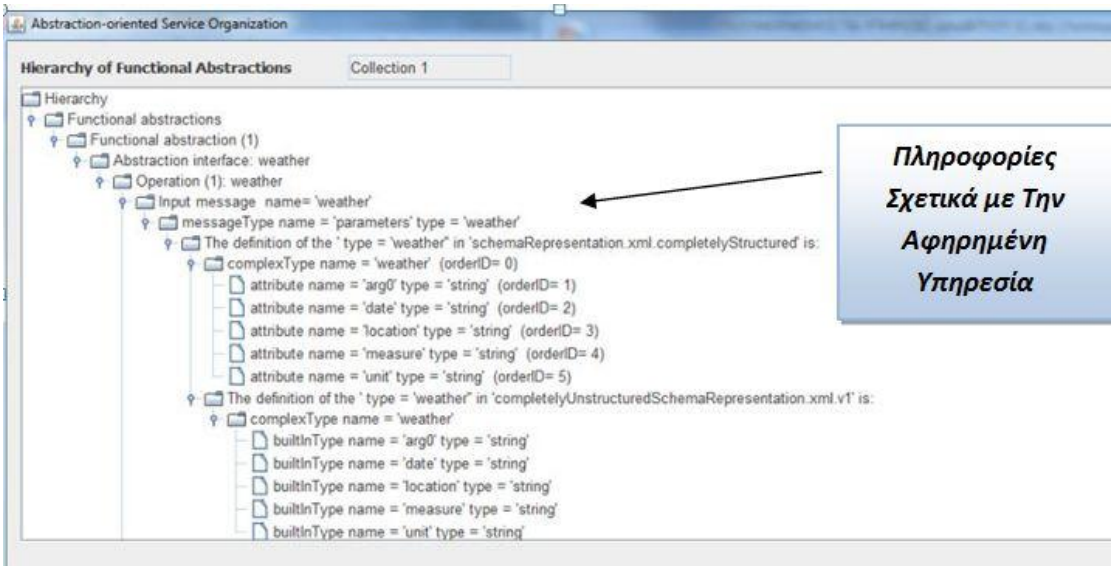
Σχήμα 4.24: Πλαίσιο που Γίνεται η Δημιουργία των Αφηρημένων Υπηρεσιών

- Στο πλαίσιο που φαίνεται στο Σχήμα 4.25 πατάμε το κουμπί Produce Hierarchies και αρχίζει να τρέχει ο αλγόριθμος δημιουργίας αφηρημένων υπηρεσιών.



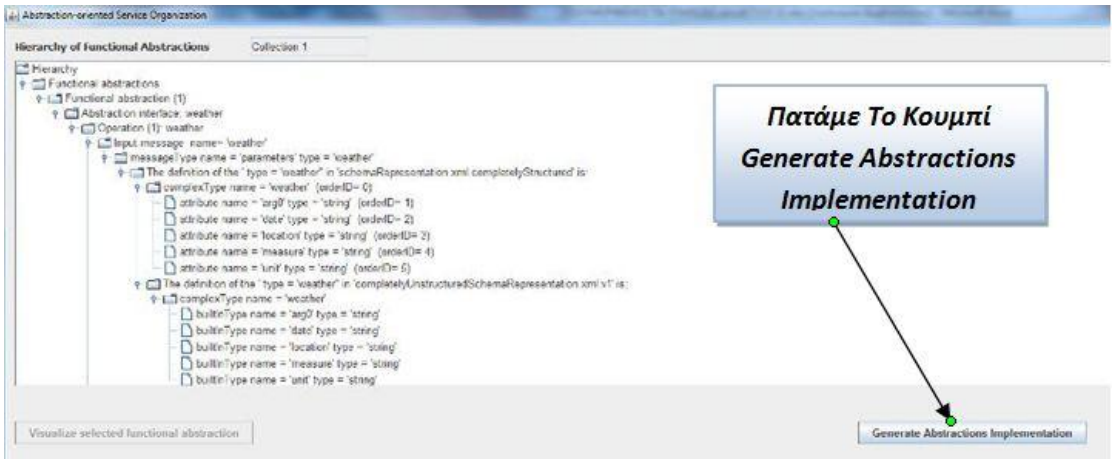
Σχήμα 4.25: Δημιουργία Πολυμορφικής Υπηρεσίας

- Μετά την δημιουργία της αφηρημένης υπηρεσίας (Σχήμα 4.26)



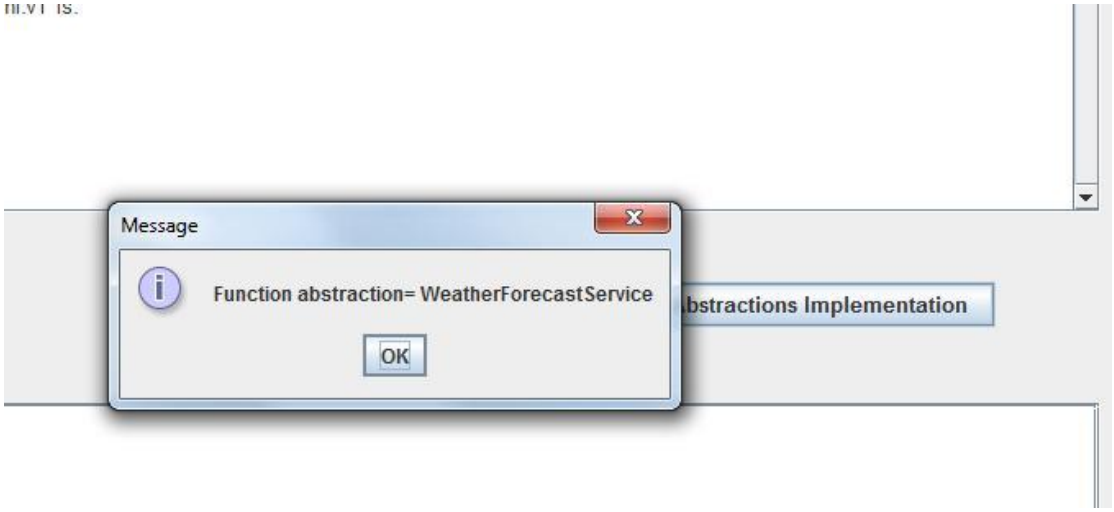
Σχήμα 4.26: Πληροφορίες Σχετικά με την Αφηρημένη Υπηρεσία

- Πατάμε το κουμπί Generate Abstractions Implementation Σχήμα 4.27.



Σχήμα 4.27: Δημιουργία Πολυμορφικής Υπηρεσίας

- Τέλος η πολυμορφική υπηρεσία έχει δημιουργηθεί Σχήμα 4.28.



Σχήμα 4.28: Η πολυμορφική Υπηρεσία Δημιουργήθηκε

4.4. Θέματα Σχετικά με την Υλοποίηση

Σε αυτή την ενότητα θα ασχοληθούμε με κάποια εργαλεία που χρησιμοποιούνται από τον μηχανισμό πολυμορφισμού και κάποιες παραδοχές που έχουμε κάνει κατά την υλοποίηση του μηχανισμού πολυμορφισμού. Πιο συγκεκριμένα:

Για την δημιουργία των απαραίτητων stubs για την κλήση των GP υπηρεσιών χρησιμοποιείται από τον μηχανισμό πολυμορφισμού το εργαλείο wsimport το οποίο παίρνει σαν όρισμα το URL όπου έχει δημοσιευτεί το WSDL έγγραφο της GP υπηρεσίας και παράγει όλες τις απαραίτητες κλάσεις για την κλήση της GP υπηρεσίας μέσω του πρότυπου JAX-WS.

Επιπλέον κατά την υλοποίηση του μηχανισμού πολυμορφισμού έχουμε πως κάθε Complex type που συναντάμε είτε στην είσοδο είτε στην έξοδο της GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία περιέχει ένα τουλάχιστον ένα βασικό τύπο που θα παίξει τον ρόλο του κλειδιού για τα Hash-Tables που δημιουργούνται από την ενδιάμεση αναπαράσταση της είσόδου/εξόδου.

Τέλος όταν λόγω των αντιστοιχίσεων μεταξύ της πολυμορφικής υπηρεσίας και της GP υπηρεσίας προκύψουν περισσότερα από ένα πιθανά αντικείμενα που μπορούν να δημιουργηθούν είτε για την είσοδο είτε για την έξοδο τότε η επιλογή γίνεται με τυχαίο τρόπο. Εδώ πιθανόν θα μπορούσαν χρησιμοποιηθούν διαφορετικές τεχνικές για την επιλογή του αντικειμένου π.χ., επιλογή του πρώτου ή του τελευταίου από τα υποψήφια αντικείμενα.

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ

5.1. Σκοπός της Αξιολόγησης

Σκοπός της αξιολόγησης είναι να μελετήσουμε την απόδοση του μηχανισμού πολυμορφισμού που προτείνεται σ' αυτήν την διατριβή. Πιο συγκεκριμένα, θέλουμε να μελετήσουμε τα εξής ερωτήματα:

- Πώς κατανέμεται ο χρόνος/κόστος εκτέλεσης μιας λειτουργίας στα επιμέρους βήματα του μηχανισμού πολυμορφισμού ?
- Πώς επηρεάζεται ο χρόνος εκτέλεσης μιας λειτουργίας από το πλήθος των αντιστοιχίσεων μεταξύ των αντικείμενων εισόδου/εξόδου της πολυμορφικής υπηρεσίας και των αντικειμένων εισόδου/εξόδου της GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία ?
- Πώς επηρεάζεται ο συνολικός χρόνος εκτέλεσης από το πλήθος των δεδομένων εισόδου/εξόδου της πολυμορφικής υπηρεσίας ?
- Πώς επηρεάζεται ο μηχανισμός αναπροσαρμογής μιας GP υπηρεσίας με το πλήθος των ***S*** υπηρεσιών που χρησιμοποιούν την πολυμορφική υπηρεσία ?

5.2. Μεθοδολογία της Αξιολόγησης

5.2.1. Μεθοδολογία Αξιολόγησης του Μηχανισμού Πολυμορφισμού

Για την αξιολόγηση του μηχανισμού πολυμορφισμού χρησιμοποιήθηκαν οι παρακάτω 6 πολυμορφικές υπηρεσίες που αναπτύχθηκαν στα πλαίσια του ερευνητικού προγράμματος CHOReOS (http://www.choreos.eu/bin/view/Main/) και σχετίζονται με διαδικασίες που λαμβάνουν χώρα σε ένα διεθνές αεροδρόμιο:

- Η 1η πολυμορφική υπηρεσία ομαδοποιεί τις GP υπηρεσίες πρόγνωσης καιρού του Κεφαλαίου 3 (AbstractWeatherForecastService).
- Η 2η πολυμορφική υπηρεσία ομαδοποιεί GP υπηρεσίες διαφορετικών ξενοδοχείων (AbstractHotel).
- Η 3η πολυμορφική υπηρεσία ομαδοποιεί διαφορετικές GP υπηρεσίες ασφάλειας (AbstractSecurityCompany).
- Η 4η πολυμορφική υπηρεσία ομαδοποιεί διαφορετικές GP υπηρεσίες αεροπορικών εταιριών (AbstractAirplane).
- Η 5η πολυμορφική υπηρεσία ομαδοποιεί διαφορετικές GP υπηρεσίες ενημέρωσης του αεροδρομίου (AbstractDisplaysManagement).
- Η 6η πολυμορφική υπηρεσία ομαδοποιεί διαφορετικές GP υπηρεσίες μεταφοράς επιβατών (AbstractAirportBusCompany).

Η αξιολόγηση του μηχανισμού πολυμορφισμού γίνεται χρησιμοποιώντας την κάθε πολυμορφική υπηρεσία είτε σαν μια απλή Java κλάση ή μια υπηρεσία διαδικτύου. Για την αξιολόγηση του μηχανισμού πολυμορφισμού ως υπηρεσία διαδικτύου η διάταξη που χρησιμοποιήθηκε είναι η εξής: οι GP υπηρεσίες που καλούνται καθώς και ο πολυμορφικός μηχανισμός έχουν εγκατασταθεί σε ένα υπολογιστή και ο client που καλεί τον μηχανισμό πολυμορφισμού βρίσκεται σε ένα δεύτερο υπολογιστή. Η σύνδεση των δυο υπολογιστών γίνεται απευθείας με χρήση καλωδίου Ethernet.

Για την μέτρηση του χρόνου/κόστους εκτέλεσης των επιμέρους βημάτων του μηχανισμού πολυμορφισμού, υποθέτουμε διαφορετικά σενάρια για κάθε

πολυμορφική υπηρεσία που μελετήθηκε. Σε κάθε σενάριο η κλήση της πολυμορφικής υπηρεσίας εκτελείται 10 φορές και στην συνέχεια παίρνουμε το μέσο όρο των χρόνων που προκύπτουν από τις εκτελέσεις αυτές. Το πιο πολύπλοκο σενάριο αφορά στην πολυμορφική υπηρεσία AbstractWeatherForecastService. Στην περίπτωση αυτή μελετήθηκε με περισσότερη λεπτομέρεια το πώς επηρεάζεται ο μηχανισμός πολυμορφισμού από το πλήθος των δεδομένων. Για το λόγο αυτό η πολυμορφική υπηρεσία καλείται με διαφορετικά μεγέθη μηνυμάτων εισόδου και παράγουν αντίστοιχα μεγέθη μηνυμάτων εξόδου. Το αντικείμενο GetDailyWeatherIn της εισόδου της πολυμορφικής υπηρεσίας έχει σαν πεδίο μία λίστα. Καλούμε την πολυμορφική υπηρεσία με μέγεθος λίστας, 100, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000 στοιχεία. Για να μελετήσουμε το πώς επηρεάζεται ο χρόνος από τις αντιστοιχίσεις μεταξύ της εισόδου/εξόδου της πολυμορφικής υπηρεσίας και της GP υπηρεσίας εκτελούμε την παραπάνω μελέτη για τις διαφορετικές GP υπηρεσίες που εκπροσωπούνται από τον μηχανισμό πολυμορφισμού.

Οι υλοποιήσεις των πολυμορφικών υπηρεσιών και των GP υπηρεσιών που χρησιμοποιήθηκαν κατά την πειραματική αξιολόγηση του μηχανισμού πολυμορφισμού βρίσκονται στο:
http://gatepc73.cs.uoi.gr:9880/middlewareLab/CHOReOS/SourceCode/AoSBM/

5.2.2. Μεθοδολογία Αξιολόγησης του Μηχανισμού Αναπροσαρμογής

Για την μελέτη του χρόνου που απαιτείται για την αναπροσαρμογή της GP υπηρεσίας του κρύβεται πίσω από μια πολυμορφική υπηρεσία καλούμε την διαδικασία αναπροσαρμογής για διαφορετικό πλήθος ***S*** υπηρεσιών που χρησιμοποιούν την πολυμορφική υπηρεσία. Πιο συγκεκριμένα καλούμε την διαδικασία αναπροσαρμογής για 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 ***S*** υπηρεσίες που χρησιμοποιούν την πολυμορφική υπηρεσία και επηρεάζονται από την αναπροσαρμογή.

5.3. Αποτελέσματα

Στην ενότητα αυτή θα δώσουμε τα αποτελέσματα της πειραματικής αξιολόγησης τόσο του μηχανισμού πολυμορφισμού όσο και του μηχανισμού αναπροσαρμογής.

5.3.1. Αξιολόγηση του Μηχανισμού Πολυμορφισμού

Στην ενότητα 5.3.1.1 δίνονται τα αποτελέσματα που σχετίζονται με την 1η πολυμορφική υπηρεσία. Στην ενότητα 5.3.1.2 δίνονται τα αποτελέσματα που σχετίζονται με τις πολυμορφική υπηρεσίες 2 - 6.

5.3.1.1. 1η Πολυμορφική Υπηρεσία (AbstractWeatherForecastService)

Τα αποτελέσματα των πειραμάτων για την κλήση της GP υπηρεσίας WeatherService παράδειγμα Κεφάλαιο 3 που κρύβεται πίσω από τον πολυμορφικό μηχανισμό παρουσιάζονται στους Πίνακες 5.1 - 5.2 και τα αποτελέσματα της κλήσης της GP υπηρεσίας WeatherForecastService παρουσιάζονται στους Πίνακες 5.3 – 5.4. Οι παρατηρήσεις μας είναι ανάλογες και για τις δύο περιπτώσεις. Πιο συγκεκριμένα παρατηρούμε ότι:

- Ο συνολικός χρόνος που απαιτείται για την κλήση της GP υπηρεσίας (Πίνακας 5.5(A) - 5.5(B)) αυξάνεται όσο αυξάνεται και το πλήθος των δεδομένων εισόδου / εξόδου αυξάνει σε μέγεθος.

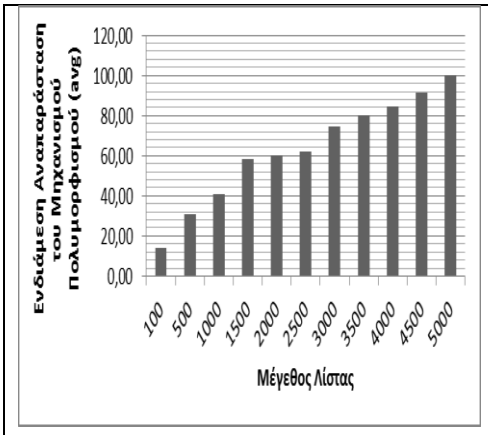
- Όταν η πολυμορφική υπηρεσία είναι μια υπηρεσία διαδικτύου (Πίνακας 5.5(A) – 5.5(B)) και όχι απλά μια Java εφαρμογή έχουμε ένα επιπλέον κόστος που οφείλεται στον επιπλέον χρόνο που προστίθεται λόγω της κλήσης της υπηρεσίας διαδικτύου.

- Όσο αυξάνεται το πλήθος των αντιστοιχίσεων αυξάνεται και ο συνολικός χρόνος για την κλήση της πολυμορφικής υπηρεσίας αυτό προκύπτει

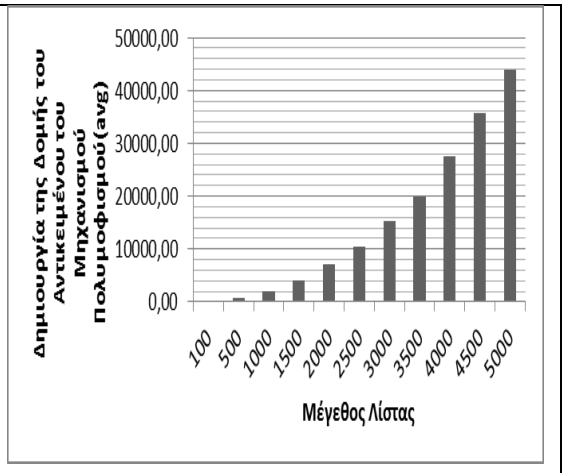
συγκρίνοντας τους χρόνους που φαίνονται Πίνακας 5.5(Α) – 5.5(Β). Αυτό συμβαίνει γιατί όταν έχουμε περισσότερες αντιστοιχίσεις ψάχνουμε περισσότερο χρόνο στις αντιστοιχίσεις που έχουν υλοποιηθεί με λίστες και το να βρούμε ένα στοιχείο σε αυτές είναι αρκετά χρονοβόρο. Επίσης λόγω των περισσότερων αντιστοιχίσεων δημιουργούνται και περισσότερα αντικείμενα με αποτέλεσμα να αυξάνει κι' άλλο ο συνολικός χρόνος που απαιτείται.

- Όπως προκύπτει από την πειραματική αξιολόγηση του μηχανισμού πολυμορφισμού το στάδιο της οργάνωσης με βάση την δομή του αντικειμένου εισόδου /εξόδου (Πίνακας 5.1 (Β), (Ε) - Πίνακας 5.2 (Β), (Ε) - Πίνακας 5.3 (Β), (Ε) - Πίνακας 5.4 (Β), (Ε)) παίρνει περισσότερο χρόνο σε σχέση με την δημιουργία της ενδιάμεσης αναπαράστασης και της αρχικοποίησης του αντικειμένου είτε της εισόδου είτε της εξόδου. Αυτό προκύπτει γιατί σε αυτό το στάδιο ο μηχανισμός όχι μόνο διασχίζει το αντικείμενο εισόδου / εξόδου με την χρήση του μηχανισμού Reflection της Java αλλά ταυτόχρονα ψάχνει και στις αντιστοιχίσεις οι οποίες έχουν υλοποιηθεί με την χρήση λιστών και η εύρεση ενός στοιχείου απαιτεί την διάσχιση της λίστας μέχρις ότου βρεθεί το στοιχείο που ψάχνουμε μια διαδικασία αρκετά χρονοβόρα.
- Ο χρόνος που χρειάζεται για την αρχικοποίηση του αντικείμενου εισόδου/εξόδου αυξάνεται καθώς αυξάνεται το πλήθος των δεδομένων Πίνακας 5.1 (Γ), (Στ) - Πίνακας 5.2 (Γ), (Στ) - Πίνακας 5.3 (Γ), (Στ) - Πίνακας 5.4 (Γ), (Στ).
- Η σημαντική διαφορά που προκύπτει στους χρόνους που απαιτούνται για την ενδιάμεση αναπαράσταση της εσόδου και της εξόδου Πίνακα 5.1 (Α), (Δ) – Πίνακα 5.2 (Α), (Δ) – Πίνακα 5.3 (Α), (Δ) – Πίνακα 5.4 (Α), (Δ) προκύπτει επειδή στην είσοδο έχουμε περισσότερα επίπεδα έχουμε τρία επίπεδα αναδρομής στα οποία χρησιμοποιούμε τον μηχανισμό Reflection της Java ενώ για την έξοδο έχουμε μόνο ένα επίπεδο αναδρομής που χρησιμοποιούμε τον μηχανισμό Reflection της Java.

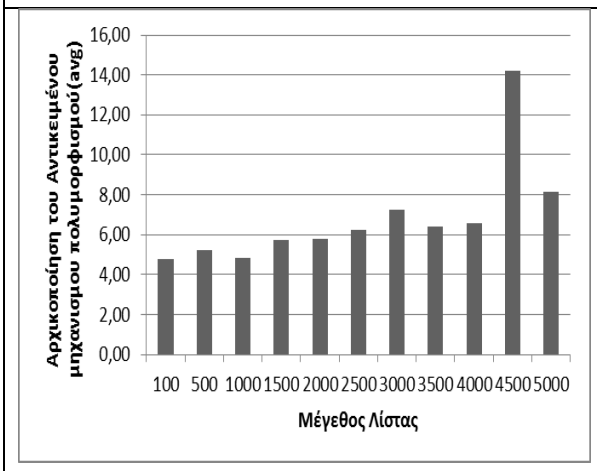
Πίνακας 5.1: Κλήση της WeatherService ως Java API (Millisecond)



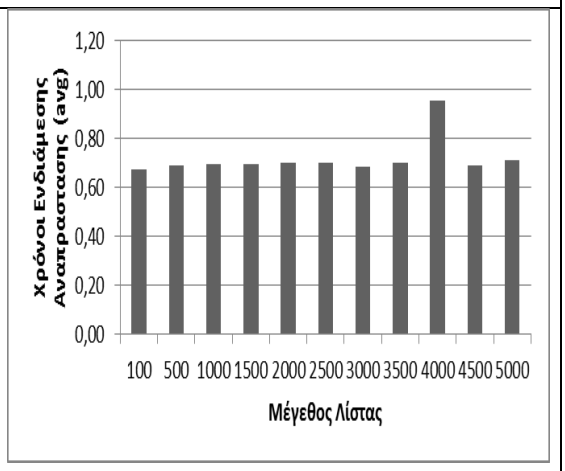
Α) Ενδιάμεση Αναπαράσταση της Εισόδου της Πολυμορφικής υπηρεσίας



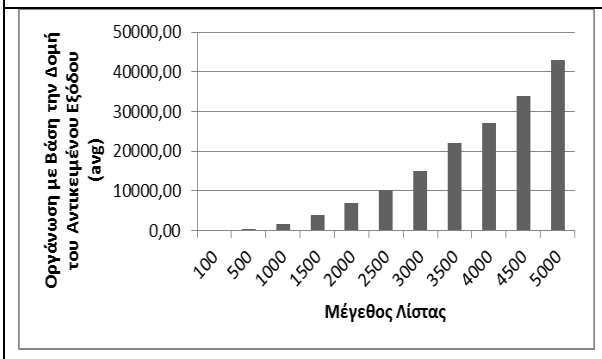
Β) Οργάνωση με Βάση τη Δομή του Αντικειμένου Εισόδου της GP Υπηρεσίας



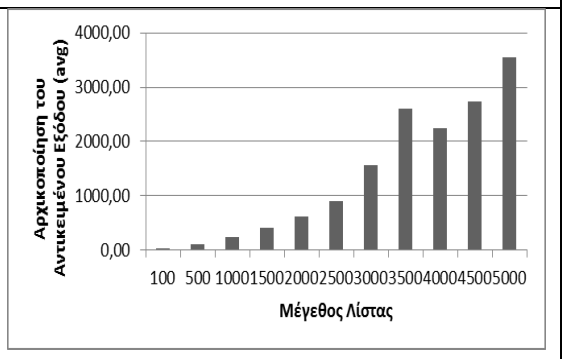
Γ) Αρχικοποίηση του Αντικειμένου Εισόδου της GP Υπηρεσίας



Δ) Ενδιάμεση Αναπαράσταση της Εξόδου της GP Υπηρεσίας

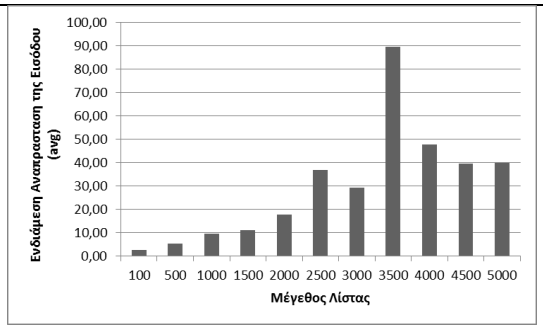


Ε) Οργάνωση με Βάση το Αντικείμενο Εξόδου

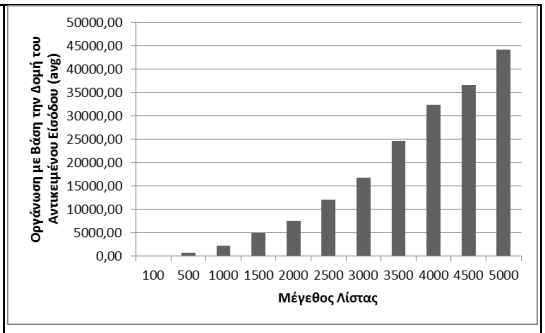


Στ) Αρχικοποίηση του Αντικειμένου Εξόδου

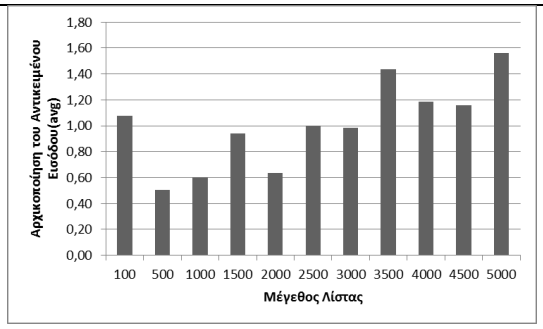
Πίνακας 5.2: Κλήση του WeatherService ως Υπηρεσία Διαδικτύου (Millisecond)



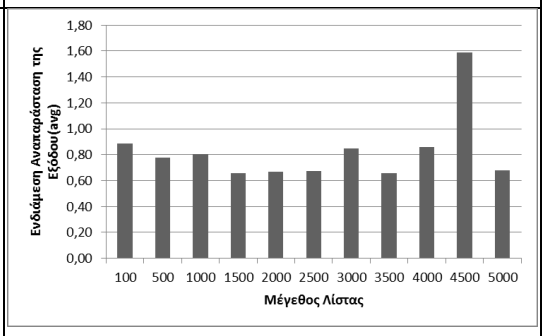
Α) Ενδιάμεση Αναπαράσταση της Εισόδου της Πολυμορφικής Υπηρεσίας



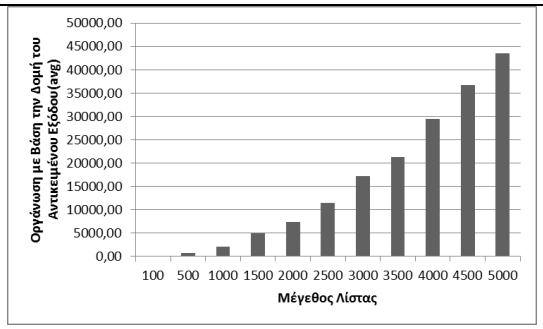
Β) Οργάνωση με Βάση το Αντικείμενο Εισόδου της GP Υπηρεσίας



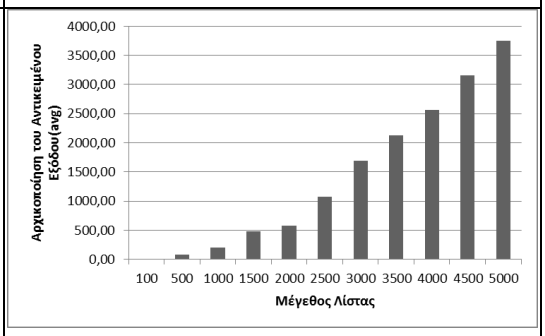
Γ) Αρχικοποίηση της Εισόδου της GP Υπηρεσίας



Δ) Αρχικοποίηση της Εισόδου της GP Υπηρεσίας

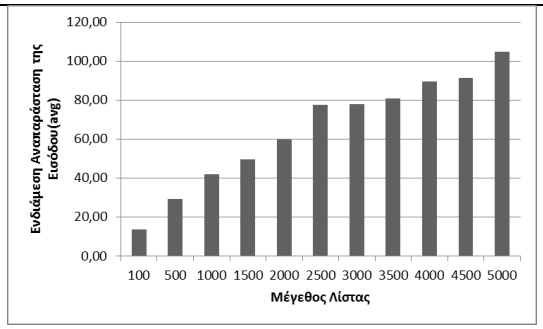


Ε) Οργάνωση με Βάση την Δομή του Αντικειμένου Εξόδου

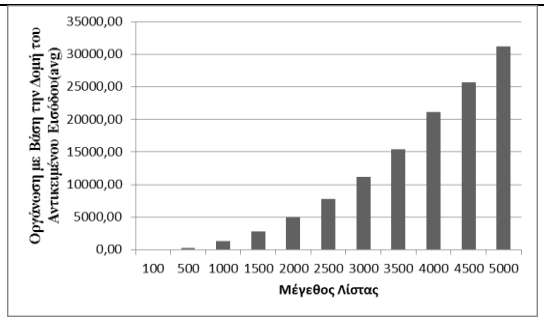


Στ) Οργάνωση με Βάση την Δομή του Αντικειμένου Εξόδου

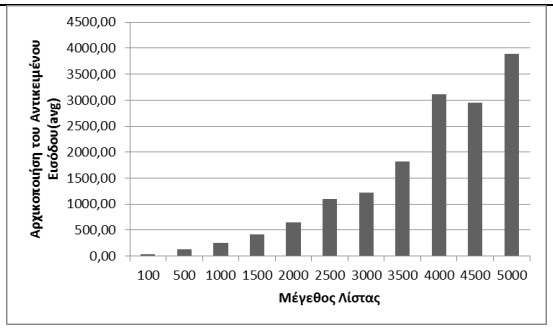
Πίνακας 5.3: Κλήση της WeatherForecastService ως Java API (Millisecond)



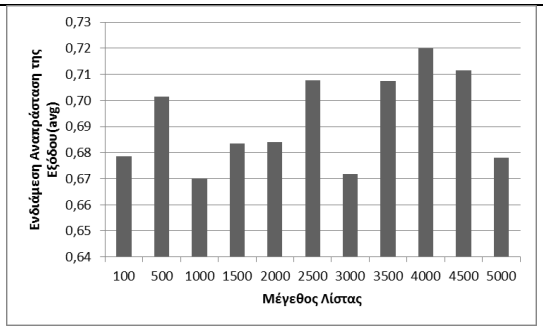
Α) Ενδιάμεση Αναπαράσταση της Εισόδου της Πολυμορφικής Υπηρεσίας



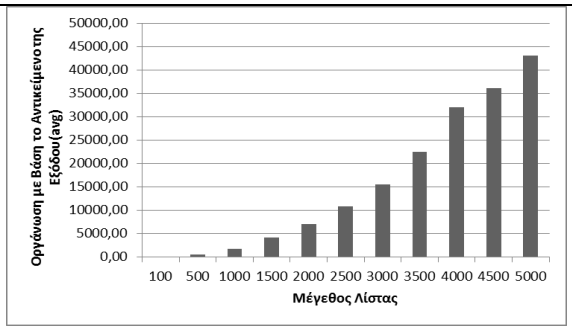
Β) Οργάνωση με Βάση το Αντικείμενο Εισόδου της GP Υπηρεσίας



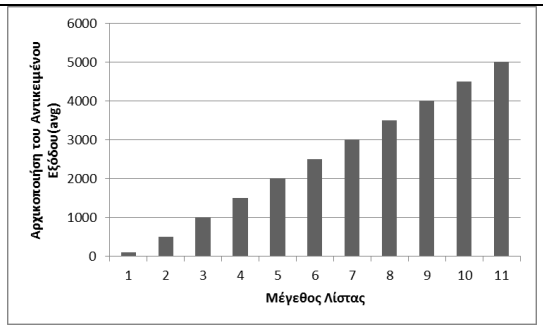
Γ) Αρχικοποίηση της Εισόδου της GP Υπηρεσίας



Δ) Ενδιάμεση Αναπαράσταση της Εξόδου της GP Υπηρεσίας

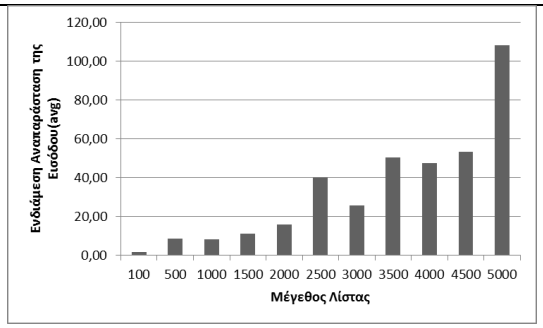
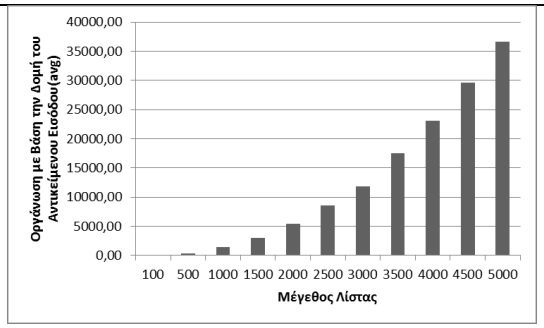
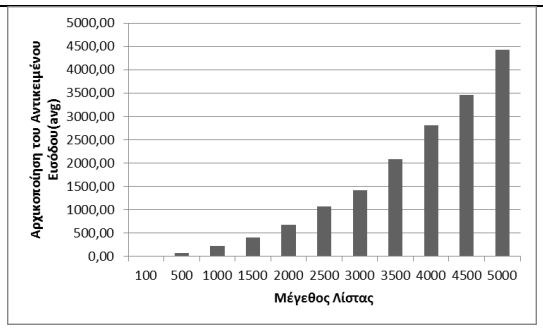
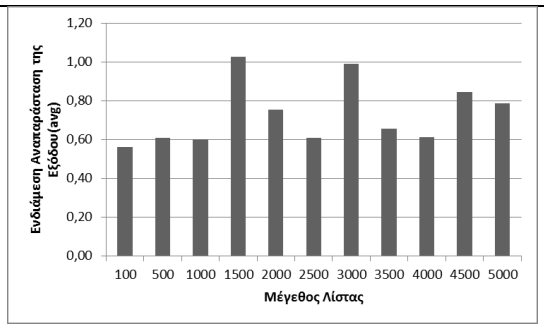
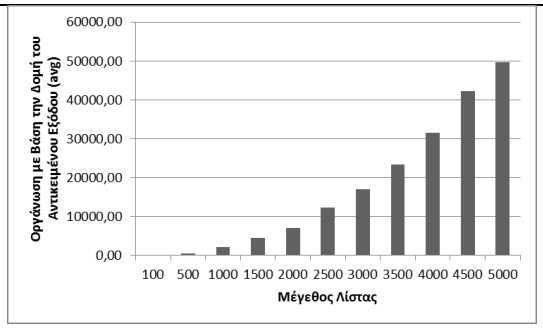
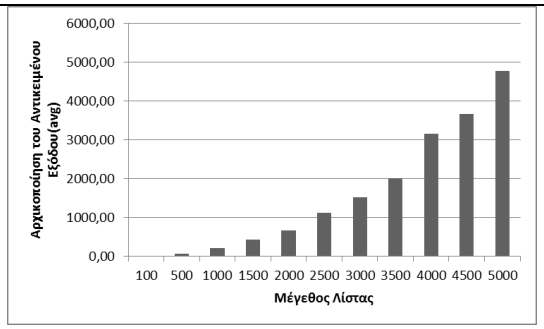


Ε) Οργάνωση με Βάση την Δομή του Αντικειμένου Εξόδου

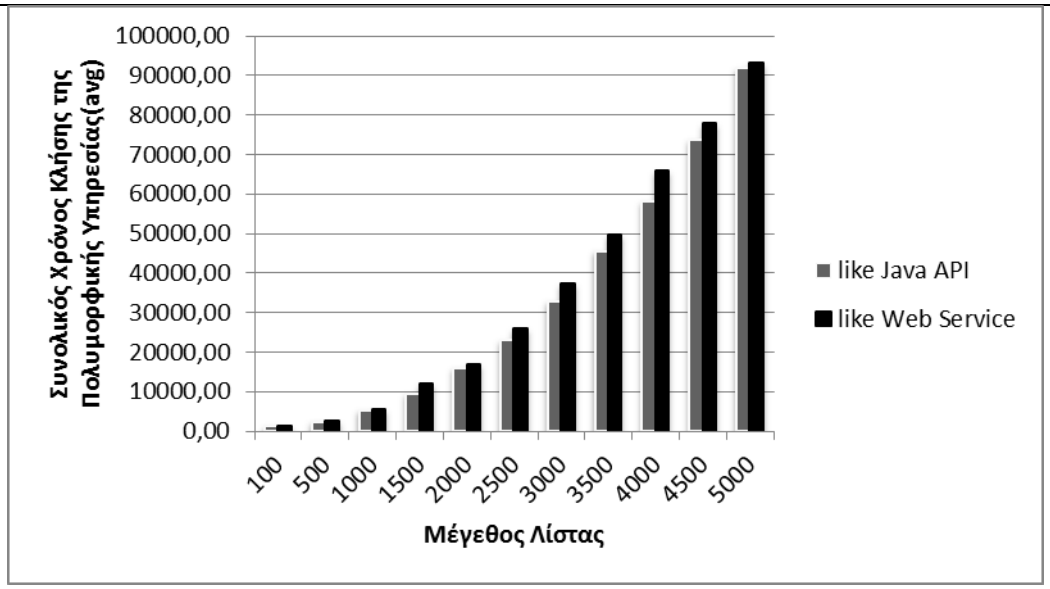


Στ) Αρχικοποίηση της Εξόδου της Πολυμορφικής Υπηρεσίας

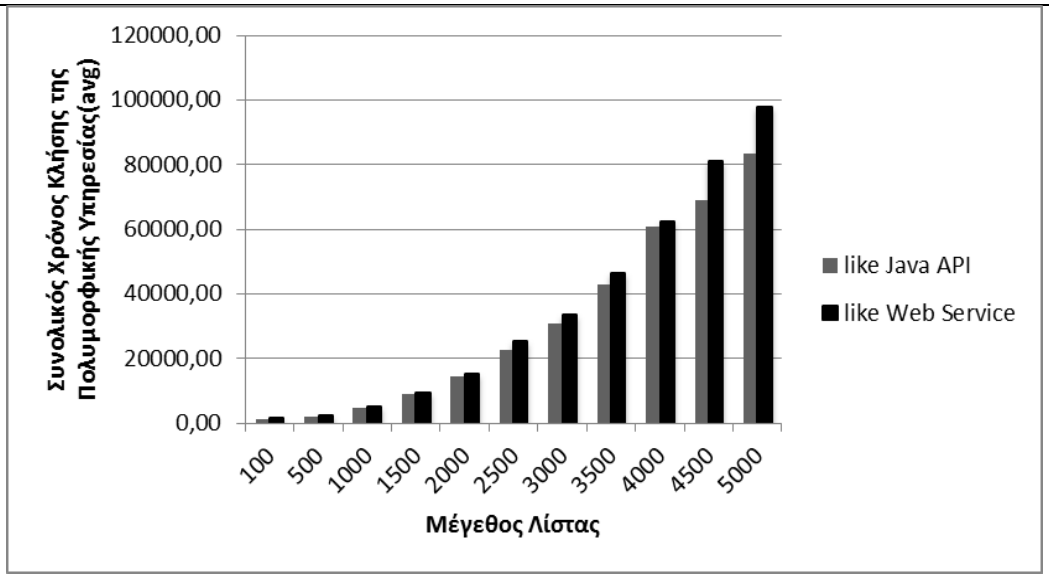
Πίνακας 5.4: Κλήση του WeatherForecastService ως Υπηρεσία Διαδικτύου (Millisecond)

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Α) Ενδιάμεση Αναπαράσταση της Εισόδου της Πολυμορφικής Υπηρεσίας</td><td>Β) Οργάνωση με Βάση το Αντικείμενο Εισόδου της GP Υπηρεσίας</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Γ) Αρχικοποίηση της Εισόδου της GP Υπηρεσίας</td><td>Δ) Ενδιάμεση Αναπαράσταση της Εξόδου της GP Υπηρεσίας</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Ε) Οργάνωση με Βάση την Δομή του Αντικειμένου Εξόδου</td><td>Στ) Αρχικοποίηση της Εξόδου της Πολυμορφικής Υπηρεσίας</td></tr>
</table>

Πίνακας 5.5: Συνολικός Χρόνος για την Κλήση των Υπηρεσιών Διαδικτύου (Millisecond)



A) Συνολικός Χρόνος για την Κλήση της GP Υπηρεσίας WeatherService



B) Συνολικός Χρόνος για την Κλήση της GP Υπηρεσίας WeatherForecastService

5.3.1.2. Πολυμορφικές Υπηρεσίες 2 - 6

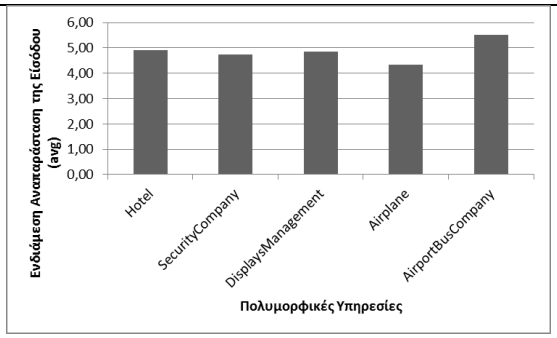
Τα αποτελέσματα των πειραμάτων για την κλήση των πολυμορφικών υπηρεσιών 2 - 6 φαίνονται στους Πίνακες 5.6 - 5.7 – 5.8 . Οι παρατηρήσεις είναι παρόμοιες για όλες τις πολυμορφικές υπηρεσίες που εξετάζουμε:

- Όταν η πολυμορφική υπηρεσία είναι μια υπηρεσία διαδικτύου (Πίνακας 5.8(Α) και όχι απλά μια Java εφαρμογή έχουμε ένα επιπλέον κόστος που οφείλεται στον επιπλέον χρόνο που προστίθεται λόγω της κλήσης της υπηρεσίας διαδικτύου.

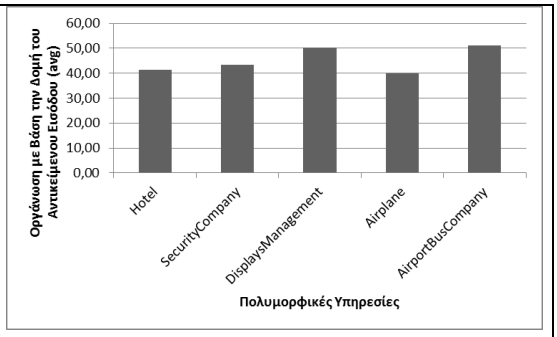
- Ο χρόνος που απαιτείται για την ενδιάμεση αναπαράσταση της εισόδου Πίνακας 5.6 (Β) – 5.7 (Β) παίρνει περισσότερο χρόνο σε σχέση με την δημιουργία της ενδιάμεσης αναπαράστασης και της αρχικοποίησης του αντικειμένου είτε της εισόδου.

- Ο χρόνος που απαιτείται για την ενδιάμεση αναπαράσταση της εξόδου Πίνακας 5.6 (Ε) – 5.7 (Ε) δεν παίρνει πάντα περισσότερο χρόνο από ότι η ενδιάμεση αναπαράσταση της εξόδου και η αρχικοποίηση της εξόδου. Αυτό οφείλεται στο γεγονός ότι τα αντικείμενα των εξόδων των πολυμορφικών υπηρεσιών είναι πολύ απλά.

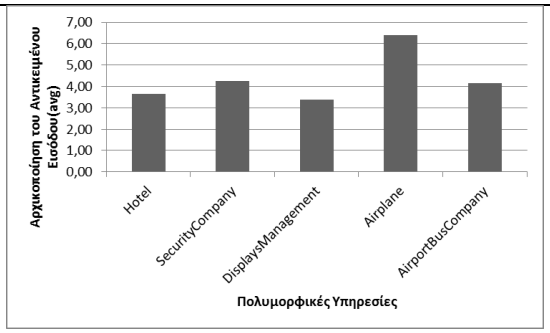
Πίνακας 5.6: Κλήση των Πολυμορφικών Υπηρεσιών 2 – 6 ως Java API (Millisecond)



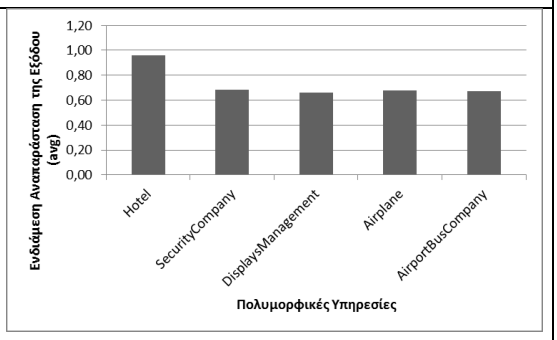
Α) Ενδιάμεση Αναπαράσταση της Εισόδου της Πολυμορφικής Υπηρεσίας



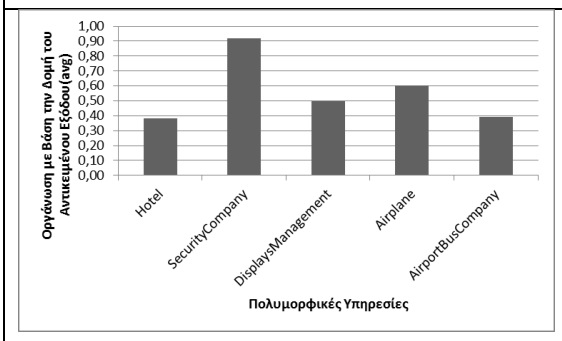
Β) Οργάνωση με Βάση το Αντικείμενο Εισόδου της GP Υπηρεσίας



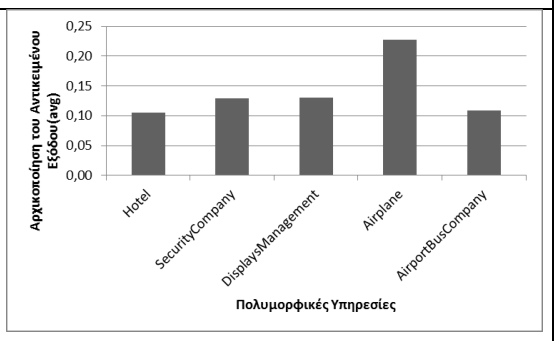
Γ) Αρχικοποίηση της Εισόδου της GP Υπηρεσίας



Δ) Ενδιάμεση Αναπαράσταση της Εξόδου της GP Υπηρεσίας

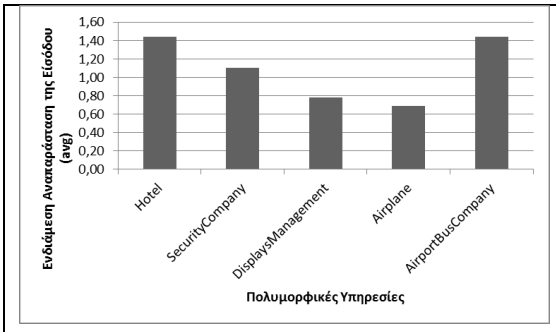
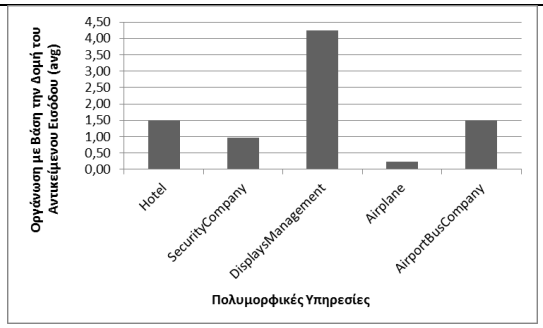
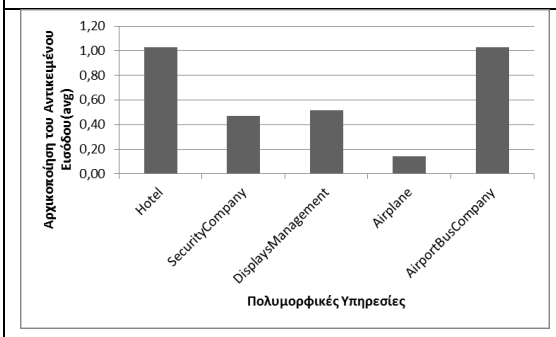
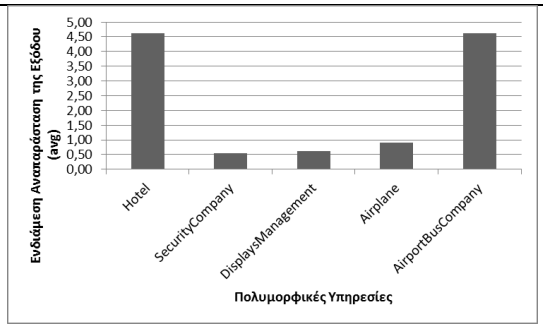
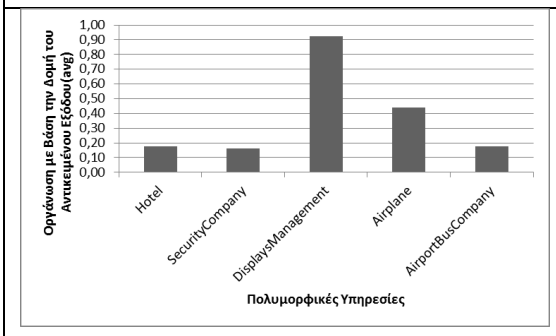
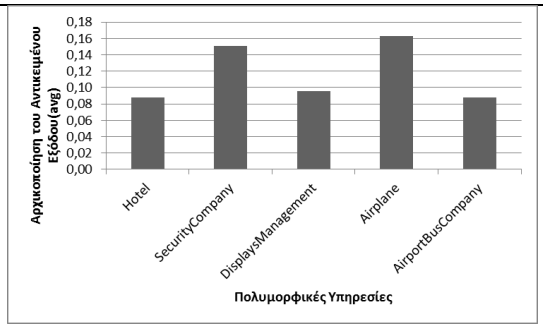


Ε) Οργάνωση με Βάση την Δομή του Αντικειμένου Εξόδου

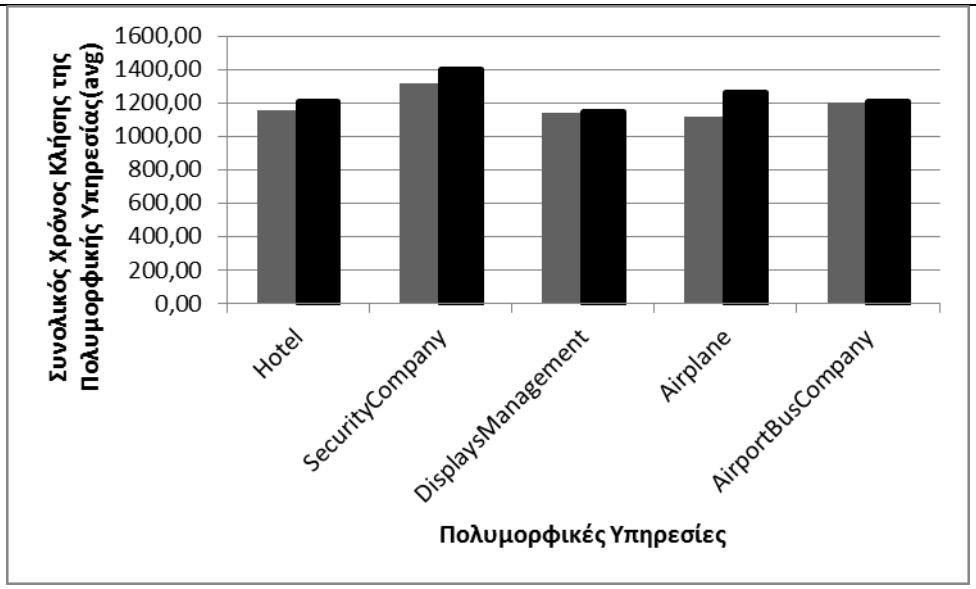


Στ) Αρχικοποίηση της Εξόδου της Πολυμορφικής Υπηρεσίας

Πίνακας 5.7: Κλήση των Πολυμορφικών Υπηρεσιών 2 – 6 ως Υπηρεσίες Διαδικτύου (Millisecond)

	
Α) Ενδιάμεση Αναπαράσταση της Εισόδου της Πολυμορφικής Υπηρεσίας	Β) Οργάνωση με Βάση το Αντικείμενο Εισόδου της GP Υπηρεσίας
	
Γ) Αρχικοποίηση της Εισόδου της GP Υπηρεσίας	Δ) Ενδιάμεση Αναπαράσταση της Εξόδου της GP Υπηρεσίας
	
Ε) Οργάνωση με Βάση την Δομή του Αντικειμένου Εξόδου	Στ) Αρχικοποίηση της Εξόδου της Πολυμορφικής Υπηρεσίας

Πίνακας 5.8: Συνολικός Χρόνος για την Κλήση των GP Υπηρεσιών (Millisecond)



Α) Συνολικός Χρόνος για την Κλήση Διαφορετικών GP Υπηρεσιών

5.3.2. Αξιολόγηση του Μηχανισμού Αναπροσαρμογής

Κατά την πειραματική αξιολόγηση του μηχανισμού αναπροσαρμογής παρατηρήσαμε τα εξής:

- Ο μέσος όρος του χρόνου (Πίνακας 5.9(Α)) που απαιτείται για την ειδοποίηση των S υπηρεσιών που χρησιμοποιούν τον πολυμορφικό μηχανισμό αυξάνεται καθώς αυξάνεται το πλήθος των S υπηρεσιών, αυτό συμβαίνει γιατί όσο αυξάνει το πλήθος των S υπηρεσιών γίνονται περισσότερες κλήσεις στην διαδικασία που είναι υπεύθυνη για την ειδοποίηση των S υπηρεσιών για την έναρξη της διαδικασίας αναπροσαρμογής.

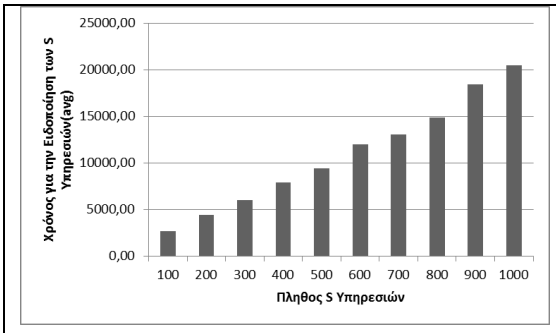
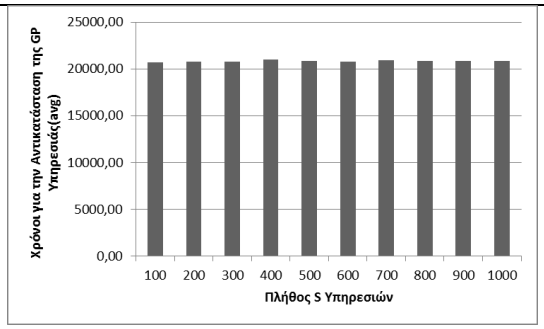
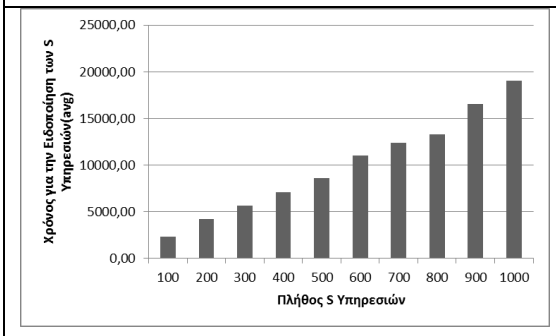
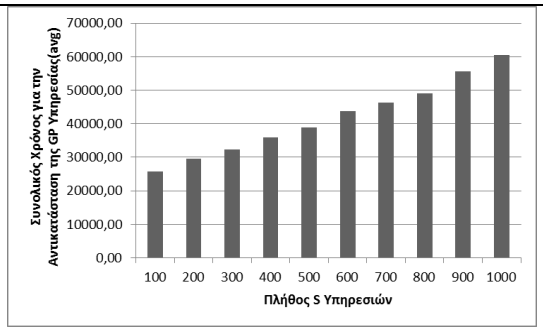
- Ο μέσος όρος του χρόνου (Πίνακας 5.9(Β)) που απαιτείται για την αναπροσαρμογή της υπηρεσίας είναι ανεξάρτητος από το πλήθος των υπηρεσιών που χρησιμοποιούν την πολυμορφική υπηρεσία καθώς παρατηρήσαμε παρόμοιες τιμές ανεξάρτητα από το πλήθος το πλήθος των S υπηρεσιών.

- Ο μέσος όρος του χρόνου (Πίνακας 5.9(Γ)) που απαιτείται για την ειδοποίηση των S υπηρεσιών που χρησιμοποιούν τον πολυμορφικό μηχανισμό για το τέλος της διαδικασίας αναπροσαρμογής αυξάνεται καθώς αυξάνεται το πλήθος των S υπηρεσιών, αυτό συμβαίνει γιατί όσο αυξάνει το πλήθος των S υπηρεσιών γίνονται περισσότερες κλήσεις στην διαδικασία που είναι υπεύθυνη για την ειδοποίηση των S υπηρεσιών για το τέλος της διαδικασίας αναπροσαρμογής.

- Ο μέσος όρος του συνολικού χρόνου (Πίνακας 5.9 (Δ)) που απαιτείται για την αναπροσαρμογή της GP υπηρεσίας που κρύβεται πίσω από την πολυμορφική υπηρεσία αυξάνεται καθώς αυξάνει το πλήθος των S υπηρεσιών που χρησιμοποιούν την πολυμορφική υπηρεσία αυτό συμβαίνει γιατί αυξάνει ο χρόνος που απαιτείται για την ειδοποίηση για την έναρξη της διαδικασίας

αναπροσαρμογής και του χρόνου που απαιτείται για την ειδοποίηση για το τέλος της διαδικασίας αναπροσαρμογής.

Πίνακας 5.9: Πειραματική Αξιολόγηση του Μηχανισμού Αναπροσαρμογής (Millisecond)

	
Α) Χρόνος που Απαιτείται για την Ειδοποίηση των S Υπηρεσιών για την Έναρξη της Αντικατάστασης	Β) Χρόνος που Απαιτείται για την Αντικατάσταση της GP Υπηρεσίας
	
Γ) Χρόνος που Απαιτείται για την Ειδοποίηση των S Υπηρεσιών για το Τέλος της Αντικατάστασης	Δ) Συνολικός Χρόνος που Απαίτειται από τον Μηχανισμό Αναπροσαρμογής για την Αντικατάσταση της GP Υπηρεσίας

ΚΕΦΑΛΑΙΟ 6. ΕΠΙΛΟΓΟΣ

Στην εργασία αυτή επεκτείναμε μια υπάρχουσα εργασία που παράγει αφηρημένες υπηρεσίες από ένα σύνολο υπηρεσιών που είναι διαθέσιμές μέσω διαδικτύου με ένα μηχανισμό πολυμορφισμού. Πιο συγκεκριμένα δέχεται ως είσοδο τις αντιστοιχίσεις ανάμεσα στην διεπαφή μιας αφηρημένης υπηρεσίας και των υπηρεσιών που εκπροσωπούνται από αυτή και με βάση αυτές τις αντιστοιχίσεις μεταφράζει δυναμικά τις κλήσεις που γίνονται με βάση την διεπαφή της αφηρημένης υπηρεσίας σε κλήσεις αντίστοιχων λειτουργιών που προσφέρονται από τις συγκεκριμένες υπηρεσίες που εκπροσωπούνται από την αφηρημένη υπηρεσία.

Ακόμη υλοποιήσαμε έναν μηχανισμό αναπροσαρμογής ο οποίος κάνει την αλλαγή της υπηρεσίας που καλείται από την αφηρημένη υπηρεσία με κάποια άλλη που εκπροσωπείται από αυτήν και επίσης ο μηχανισμός είναι υπεύθυνος για την γνωστοποίηση της αλλαγής σ’ αυτούς που χρησιμοποιούν την αφηρημένη υπηρεσία.

Τέλος κάναμε και πειραματική αξιολόγηση του μηχανισμού πολυμορφισμού και δείξαμε πως όσο αυξάνονται τα δεδομένα της εισόδου/εξόδου τόσο αυξάνεται και ο συνολικός χρόνος που απαιτείται για την κλήση της GP υπηρεσίας. Επίσης δείξαμε πως ο συνολικός χρόνος αυξάνεται όσο αυξάνονται και οι αντιστοιχίσεις μεταξύ της αφηρημένης υπηρεσίας και των υπηρεσιών που εκπροσωπούνται. Κατά την πειραματική αξιολόγηση του μηχανισμού αναπροσαρμογής δείξαμε πως ό χρόνος που απαιτείται αυξάνεται ανάλογα με το πλήθος των S υπηρεσιών που χρησιμοποιούν την αφηρημένη υπηρεσία.

ΚΕΦΑΛΑΙΟ 7. ΑΝΑΦΟΡΕΣ

[1] D. Ardagna and M. Comuzzi and E. Mussi and B. Pernici and P. Plebani, "PAWS: A Framework for Executing Adaptive Web-Service Processes", *IEEE Software*, vol. 24, no. 6, pp. 39-46, 2007.

[2] V. Agarwal and P.Jalote, "From Specification to Adaptation: An Integrated QoS-driven Approach for Dynamic Adaptation of Web Service Compositions", in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2010.

[3] D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes", *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369-384, 2007.

[4] D. Athanasopoulos and A. Zarras and V. Issarny, "Towards the Maintenance of Service Oriented Software", in *Proceedings of the 3rd CSMR Workshop on Software Quality and Maintenance (SQM)*, 2009.

[5] D. Athanasopoulos and A. Zarras and P. Vassiliadis and V. Issarny, "Mining service abstractions", in *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, 2011.

[6] A. Bucchiarone and C. Cappiello and E. Di Nitto and R. Kazhamiakin and V. Mazza and M. Pistore, "Design for Adaptation of Service-Based Applications: Main Issues and Requirements", in *Proceedings of the 2009 ICSOC/ServiceWave Workshops, LNCS 6275*, 2010.

[7] V. Cardellini and E. Casalicchio and V. Grassi and F. Lo Presti and R.Mirandola, "Qos-driven Runtime Adaptation of Service Oriented Architectures", in *Proceedings of the 7th joint European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2009.

[8] V. Cardellini and S. Iannucci, "Designing a Broker for QoS-driven Runtime Adaptation of SOA Applications", in *Proceedings of the IEEE International*

Conference on Web Services (ICWS), 2010.

[9] L. Cavallaro and E. Di Nitto, "An Approach to Adapt Service Requests to Actual Service Interfaces", in *Proceedings of the International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, 2008.

[10] L. Cavallaro and E. Di Nitto and C. A. Furia and M. Pradella, "A Tile-Based Approach for Self-Assembling Service Compositions", in *Proceedings of the 15th IEEE International Conference on Engineering of Complex Computer Systems*, 2010.

[11] L. Cavallaro and E. Di Nitto and M. Pradella, "An Automatic Approach to Enable Replacement of Conversational Services", in *Proceedings of the 7th International Joint Conference on Service-Oriented Computing (ICSOC-ServiceWave)*, 2009.

[12] G. Canfora and M. Di Penta and R. Esposito and M. L. Villani, "A Framework for QoS-aware Binding and Re-binding of Composite Web Services", *Journal of Systems and Software*, vol. 81, no. 10, pp. 1754-1769, 2008.

[13] J. Cardoso and A. Sheth, Semantic Web Services, Processes and Applications, Springer, 2006.

[14] K. Christos and C. Vassilakis and E. Rouvas and P. Georgiadis, "QoS-Driven Adaptation of BPEL Scenario Execution", in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2009.

[15] M. Fredj and N. Georgantas and V. Issarny and A. Zarras, "Dynamic Service Substitution in Service-Oriented Architectures", in *Proceedings of the IEEE International Conference on Services Computing (SCC)*, 2008.

[16] M. Fredj and A. Zarras and N. Georgantas and Valerie Issarny, "Service Intelligence and Service Science: Evolutionary Technologies and Challenges", in *Dynamic Maintenance of Service Orchestrations*, IGI, 2010.

[17] E. Gamma and R. Helm and R. Johnson and J.M. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1994.

[18] L. Kuang and S. Deng and J. Wu and Y. Li, "Towards Adaptation of Service Interface Semantics", in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2009.

[19] J. Kramer and J. Magee, "The Evolving Philosophers Problem", *IEEE Transactions on Software Engineering*, vol. 15, no. 1, pp. 1293-1306, 1990.

[20] W. Kongdenfha and H. R. Motahari Nezhad and B. Benatallah and F. Casati and R. Saint-Paul, "Ismatch Patterns and Adaptation Aspects: A Foundation for Rapid Development of Web Service Adapters", *IEEE Transactions on Services Computing (TSC)*, vol. 2, no. 2, pp. 94-107, 2009.

[21] X. Liu and C. Liu and M. Rege and A. Bouguettaya, "Semantic Support for Adaptive Long Term Composed Services", *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2010.

[22] B. Liskov and J.M. Wing, "A Behavioral Notion of Subtyping", *ACM Transactions on Programming Languages and Systems (ACM TOPLAS)*, vol. 16, no. 6, pp. 1811-1841, 1994.

[23] O. Maqbool and H. Babri, "Hierarchical Clustering for Software Architecture Recovery", *IEEE Transactions on Software Engineering*, τόμ. 33, αρ. 11, pp. 759-780, 2007.

[24] N. Ben Mabrouk and S. Beauche and E. Kuznetsova and N. Georgantas and V. Issarny, "QoS-aware Service Composition in Dynamic Service Oriented Environments", in *Proceedings of Middleware 2009 - ACM/IFIP/USENIX, 10th International Conference*, 2009.

[25] O. Moser and F. Rosenberg and S. Dustdar, "Non-Intrusive Monitoring and Service Adaptation for WS-BPEL", in *Proceedings of the 17th International World Wide Web Conference (WWW)*, 2008, pp. 815-824.

[26] J. Munkres, "Algorithms for the Assignment and Transportation Problems", *Journal of the Society for Industrial and Applied Mathematics*, pp. 32-58, 1957.

[27] H. R. Motahari Nezhad and B. Benatallah and A. Martens and F. Curbera and F. Casati, "Semi Automated Adaptation of Service Interactions", in *Proceedings of the International World Wide Web Conference (WWW)*, 2007, pp. 993-1002.

[28] H. R. Motahari Nezhad and G-Y. Xu and B. Benatallah, "Protocol-aware Matching of Web Service Interfaces for Adapter Development", in *Proceedings of the International World Wide Web Conference (WWW)*, 2010.

[29] S. R. Ponnekanti and A. Fox, "Interoperability Among Independently Evolving

Web Services", in *Proceedings of the 5th ACM/IFIP/USENIX International Middleware Conference (MIDDLEWARE)*, 2004, pp. 331-351.

[30] E. Thomas, "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall, 2005.

[31] W3C, "Web Services Architecture", W3C, http://www.w3c.org/TR/ws-arch.

[32] L. Zeng and B. Benatallah and A.H.H. Ngu and M. Dumas and J. Kalagnanam and H. Chang, "QoS-Aware Middleware for Web Services Composition", *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311-327, 2004.

[33] A. Zarras and M. Fredj and N. Georgantas and V. Issarny, "Engineering Reconfigurable Distributed Software Systems: Issues Arising for Pervasive Computing", Springer, 2006.

[34] Y. Zhai and J. Zhang and K-J. Lin, "SOA Middleware Support for Service Process Reconfiguration with End-to-End QoS Constraints", in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2009.

[35] H. Jiang, H. Ho, L. Popa και W.-S. Han, "Mapping - Driven XML Transformation", Proceedings of the 16th international conference on World Wide Web,2007.

ΚΕΦΑΛΑΙΟ 8. ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Ηλίας Μάκης γεννήθηκε στην Πρέβεζα το 1986. Αποφοίτησε το 2004 από το 2^{ο} Ενιαίο Λύκειο Πρέβεζας. Οι βασικές σπουδές πραγματοποιήθηκαν στο Τμήμα πληροφορικής του Πανεπιστημίου Ιωαννίνων, όπου εισήχθη το 2005 και αποφοίτησε το 2010. Συνέχισε για Μεταπτυχιακές Σπουδές στο ίδιο ίδρυμα και εξειδικεύτηκε στο Λογισμικό. Τα ενδιαφέροντά του περιλαμβάνουν την τεχνολογία λογισμικού και τις τεχνολογίες διαδικτύου.