Ανακατασκευή Προγραμματιστικών Διεπαφών Υπηρεσιών Διαδικτύου

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

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

από τον

Γεώργιο Μίσκο

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Ιούνιος 2011

ΑΦΙΕΡΩΣΗ

Αφιερώνεται στην οικογένεια μου

ΕΥΧΑΡΙΣΤΙΕΣ

Με την περάτωση της παρούσας διατριβής μου δίνεται η ευκαιρία να ευχαριστήσω τον επιβλέποντα καθηγητή μου Απόστολο Ζάρρα και τον υποψήφιο διδάκτορα Διονύση Αθανασόπουλο, για την καθοδήγηση και την πολύτιμη βοήθεια τους κατά την διάρκεια της προσπάθειας μου να διεκπεραιώσω την παρούσα μελέτη. Επίσης τους ευχαριστώ για την κατανόηση να υλοποιήσω μέρος της δουλειάς εξ αποστάσεως, πράγμα που δυσκόλευε την επικοινωνία μας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Η ποιότητα του λογισμικού αποτελεί ένα διαχρονικό αντικείμενο έρευνας. Στα συστήματα λογισμικού ένα σύνηθες κριτήριο που δείχνει την ποιότητα είναι η συνεκτικότητα. Σε αυτό το πλαίσιο, η διατριβή αυτή εξετάζει την ποιότητα του λογισμικού στο πεδίο των υπηρεσιών διαδικτύου. Η σχεδίαση τέτοιων υπηρεσιών αφορά την παροχή διεπαφών λογισμικού που περιέχουν σχετικές λειτουργίες. Πολλές φορές όμως, πολλές και άσχετες λειτουργίες μεταξύ τους είναι ομαδοποιημένες σε μια μόνο διεπαφή. Αυτό δημιουργεί προβλήματα στην χρήση των υπηρεσιών από τους προγραμματιστές, δυσκολία κατανόησης και συντήρησης. Για να αντιμετωπιστούν τα προβλήματα αυτά, η παρούσα εργασία προτείνει ένα μηχανισμό ανακατασκευής διεπαφών υπηρεσιών διαδικτύου. Εξετάζονται τα αποτελέσματα 5 αλγορίθμων ανακατασκευής για 3 μετρικές συνεκτικότητας. Ο μηχανισμός ανακατασκευάζει την αρχική διεπαφή σε άλλες διεπαφές περισσότερο συνεκτικές διατηρώντας την λειτουργικότητα της αρχικής διεπαφής. Για να εκτιμηθεί η προτεινόμενη προσέγγιση, εξετάσθηκε η εφαρμογή του μηχανισμού σε υπηρεσίες μεγάλων παρόχων, όπως η Amazon και η Yahoo. Επίσης τα αποτελέσματα εκτιμήθηκαν από ειδικούς υπηρεσιών διαδικτύου, εξετάζοντας αν μπορούν αυτά να βοηθήσουν στη σχεδίαση ποιοτικότερων διεπαφών.

EXTENDED ABSTRACT IN ENGLISH

Quality software design has always been an important issue in research. A software system's structure degrades over time. One common criterion about the quality of software is the cohesion. More particularly in object-oriented systems, classes may become very large ann less cohesive. Continuous requirements drifts the design in a dead end. Since the quality of the structure has major impact on the maintainability of a system, the structure has to be reconditioned from time to time.

In order to identify such problematic cases, existing approaches have proposed the use of cohesion metrics to restructure the code. Program restructuring is a key method to improve the quality of ill-structured programs and therefore to increase the understandability and reduce the maintenance cost. However, even a system with good quality can't solve the portability interoperability problems between different systems. Even if the service – oriented architecture faces the aforementioned problems, several problems in the quality of the produced software still exist.

In this vein, we studied the quality of software in the context of service-oriented paradigm. Towards offering cohesive services, providers should include in the service interfaces related operations. However, in most of the cases, many and unrelated operations are grouped together in one interface. This creates problems in understanding and using the functionality offered by an interface, and in maintaining it. To cope with these problems, the present master thesis proposes a cohesion-driven restructuring mechanism for service interfaces. The results of using five restructuring algorithms in combination with three cohesion metrics are examined. The mechanism restructures an initial service interface in more cohesive interfaces preserving the functionality offered by the initial interface. To assess this mechanism, we applied it

in service interfaces offered by real-world providers, such as Amazon and Yahoo. Finally, the results were also qualitatively assessed by experts with deep knowledge of the service-oriented paradigm and software evolution.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

Το κεφάλαιο αυτό αποτελεί μια εισαγωγή του θέματος που πραγματεύεται η παρούσα εργασία. Αρχικά περιγράφονται οι υπηρεσίες διαδικτύου (υπό-ενότητα 1.1), πως ορίζονται, τι επιτελούν και τα μειονεκτήματα που έχουν. Στη συνέχεια ορίζεται το πρόβλημα και ο στόχος της διατριβής (υπό-ενότητα 1.2) και τέλος παρουσιάζεται η δομή της διατριβής (υπό-ενότητα 1.3).

1.1. Υπηρεσίες Διαδικτύου

Η ποιότητα του λογισμικού αποτελεί διαχρονικά ένα πεδίο έρευνας στον χώρο της πληροφορικής, με ερωτήματα όπως η μείωση του κόστους αλλαγών στις εφαρμογές, η δυνατότητα μεταφοράς ανάμεσα σε διαφορετικά συστήματα και η εύκολη συντήρηση μεγάλων εφαρμογών. Οι συνεχώς αυξανόμενες απαιτήσεις οδηγούν στην προσαρμογή των συστημάτων και στη συνεχόμενη προσθήκη νέων λειτουργιών. Αυτό έχει ως αντίκτυπο την κακή ποιότητα του λογισμικού αφού η δομή του γίνεται χαώδης. Η συντήρηση αυτών των συστημάτων έχει μεγάλο κόστος και απαιτεί μεγάλη προσπάθεια και χρόνο από τους προγραμματιστές. Υπάρχει ανάγκη δημιουργίας ποιοτικού λογισμικού προσαρμόσιμο σε αλλαγές, με σαφείς εξαρτήσεις ανάμεσα στα συστατικά του.

Γενικά, στις μέρες μας, μειώνονται οι προσπάθειες ανάπτυξης εφαρμογών εξ' ολοκλήρου από την αρχή. Αντίθετα, γίνεται προσπάθεια οι εφαρμογές να επικοινωνούν μεταξύ τους, ώστε να χρησιμοποιούνται λειτουργίες που έχουν ήδη υλοποιηθεί από μια άλλη εφαρμογή, δηλαδή να καταναλώνονται η μία από την άλλη. Αυτό θα ήταν δύσκολο στο παραδοσιακό μοντέλο του πελάτη / διακομιστή γιατί εν γένει υπάρχουν προβλήματα μεταφερσιμότητας, και διαλειτουργικότητας.

Την απάντηση στα παραπάνω προβλήματα ήρθαν να δώσουν οι υπηρεσίες διαδικτύου. Μια τεχνολογία που επιτρέπει στις εφαρμογές διαδικτύου να επικοινωνούν μεταξύ τους. Μια υπηρεσία διαδικτύου είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μία λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Έτσι οι υπηρεσίες μοιράζονται κώδικα και δεδομένα μέσω αυτής της διεπαφής στο διαδίκτυο [10].

Οι υπηρεσίες διαδικτύου επιτρέπουν σε διαφορετικές εφαρμογές από διαφορετικές πηγές να επικοινωνούν μεταξύ τους χωρίς χρονοβόρα κωδικοποίηση. Τα δεδομένα μεταφέρονται χρησιμοποιώντας ένα πρότυπο δικτυακό πρωτόκολλο. Στις περισσότερες περιπτώσεις αυτό το πρωτόκολλο είναι το SOAP (Simple Object Access Protocol). Έτσι, είναι δυνατή η επικοινωνία εφαρμογών που έχουν υλοποιηθεί σε διαφορετικές γλώσσες προγραμματισμού και βασίζονται σε διαφορετικά λειτουργικά συστήματα. Αυτό δημιουργεί και συγκριτικά πλεονεκτήματα σε σχέση με παλαιότερες κατανεμημένες τεχνολογίες, όπως τον ευκολότερο χειρισμό δεδομένων, ευκολία στην επικοινωνία, διαλειτουργικότητα και ευκολία ανάπτυξης νέων εφαρμογών.

Το SOAP στην έκδοση 1.2 είναι ένα ελαφρύ πρωτόκολλο προορισμένο για την ανταλλαγή δομημένων πληροφοριών σε ένα κατανεμημένο περιβάλλον. Είναι ευέλικτο, γιατί χρησιμοποιεί πρότυπα πρωτόκολλα όπως το HTTP και το SMTP ως

μέσα μεταφοράς. Επιπλέον χρησιμοποιεί τεχνολογίες XML για να καθορίσει ένα επεκτάσιμο πλαίσιο παρέχοντας μια δομή μηνυμάτων η οποία μπορεί να ανταλλαχθεί πάνω από ποικίλα δικτυακά πρωτόκολλα. Το προαναφερθέν πλαίσιο έχει σχεδιαστεί να είναι ανεξάρτητο από οποιοδήποτε προγραμματιστικό μοντέλο και σημασιολογία υλοποίησης [18].

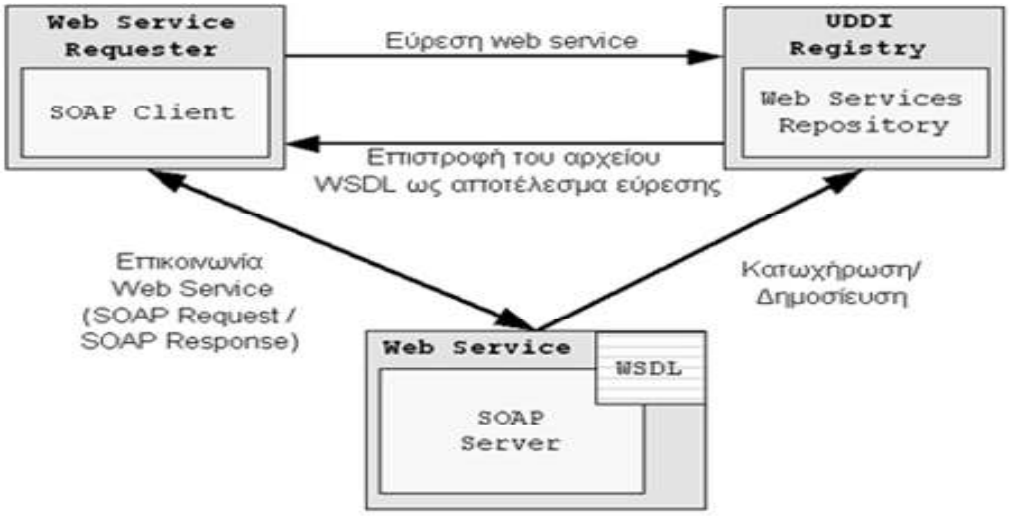
Οι υπηρεσίες παρέχουν έναν τρόπο να περιγράψουν τις διεπαφές τους με αρκετή λεπτομέρεια ώστε να επιτρέψουν στο χρήστη τους να χτίσει μια εφαρμογή πελάτη η οποία να επικοινωνήσει μαζί τους. Η περιγραφή συνήθως παρέχεται σε ένα έγγραφο XML το οποίο ακολουθεί το πρότυπο WSDL (Web Services Description Language). Η WSDL είναι ένα σχήμα XML για την περιγραφή δικτυακών υπηρεσιών σαν ένα σύνολο από τελικά σημεία που λειτουργούν με μηνύματα τα οποία περιέχουν πληροφορία είτε προσανατολισμένη στα έγγραφα είτε προσανατολισμένη στις διαδικασίες [22].

Με πιο απλά λόγια η WSDL μας βοηθάει να περιγράψουμε ένα σύνολο από μηνύματα και το πώς αυτά τα μηνύματα ανταλλάσσονται. Σκοπός της WSDL είναι να παρέχει ένα τρόπο στους παροχείς υπηρεσιών να περιγράψουν τη βασική μορφή των αιτήσεων και απαντήσεων των υπηρεσιών πάνω από διαφορετικά πρωτόκολλα και κωδικοποιήσεις. Συνοπτικά, η WSDL χρησιμοποιείται για να περιγράψει τι μπορεί να κάνει μια υπηρεσία, που βρίσκεται και πως να την καλέσει κανείς

Οι υπηρεσίες διαδικτύου πρέπει κάπου να καταχωρούνται ώστε οι χρήστες / καταναλωτές να μπορούν να τις βρουν εύκολα. Αυτό γίνεται με το UDDI (Universal Discovery Description and Integration). Το Universal Description Discovery & Integration (UDDI) εστιάζει στον καθορισμό ενός συνόλου από υπηρεσίες που θα υποστηρίζουν την περιγραφή και την εύρεση των εταιριών, οργανισμών και άλλων παρόχων υπηρεσιών, των υπηρεσιών που είναι διαθέσιμες και των τεχνικών διεπαφών οι οποίες μπορούν να χρησιμοποιηθούν ώστε να έχει κάποιος χρήστης πρόσβαση σε αυτές τις υπηρεσίες. Βασισμένο σε ένα κοινό σύνολο από βιομηχανικά πρότυπα, συμπεριλαμβανομένων των HTTP, XML, και SOAP το UDDI παρέχει μία

διαλειτουργική, θεμελιώδη υποδομή για ένα περιβάλλον λογισμικού προσανατολισμένο στις υπηρεσίες τόσο για δημόσια διαθέσιμες υπηρεσίες όσο και για υπηρεσίες που εκτίθενται μόνο εσωτερικά ενός οργανισμού. Όλα τα παραπάνω περιγράφονται στο σχήμα 1.1 που απεικονίζει το μοντέλο των υπηρεσιών διαδικτύου.

Συγκεντρωτικά, τα οφέλη από τις υπηρεσίες διαδικτύου είναι η αλληλεπίδραση μεταξύ υπηρεσιών σε οποιαδήποτε πλατφόρμα και οποιαδήποτε γλώσσα προγραμματισμού, η χαλαρή συνδεσιμότητα μεταξύ των εφαρμογών και η δυνατότητα προσαρμογής των ήδη υπαρχόντων εφαρμογών στις μεταβαλλόμενες επιχειρησιακές συνθήκες και ανάγκες των καταναλωτών. Ενώ λοιπόν οι υπηρεσίες προσπερνούν μερικά από τα προβλήματα που αναφέρθηκαν αρχικά μπορεί και σε αυτές να ανακύψουν προβλήματα που αφορούν το λογισμικό και την ποιότητα του.



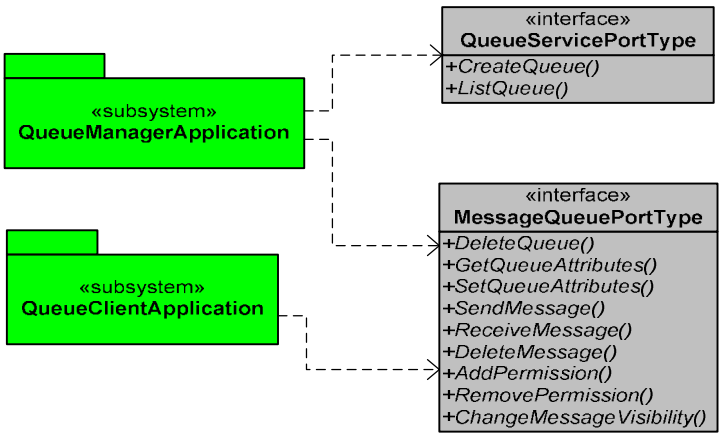
Σχήμα 1.1 Το Μοντέλο και η Αρχιτεκτονική των Υπηρεσιών Διαδικτύου [20]

1.2. Σκοπός της εργασίας

Όπως αναφέρθηκε, οι υπηρεσίες διαδικτύου παρέχουν ένα σύνολο από λειτουργίες. Οι λειτουργίες αυτές είναι ομαδοποιημένες σε διαφορετικές διεπαφές, ανάλογα με τον ρόλο που επιτελούν. Όμως, στην πράξη οι υπηρεσίες συχνά παρέχουν πολλές

λειτουργίες, συγκεντρωμένες σε μία μόνο διεπαφή. Αυτό πολλές φορές δημιουργεί πρόβλημα συνεκτικότητας. Η συνεκτικότητα αναφέρεται στο βαθμό εσωτερικής συνάφειας μεταξύ των μεθόδων και των χαρακτηριστικών μιας κλάσης [26], ενώ στην περίπτωση των υπηρεσιών διαδικτύου αναφέρεται στο βαθμό που οι λειτουργίες μιας υπηρεσίας είναι συναφείς. Η σχεδίαση μεγάλων, μη-συνεκτικών διεπαφών μπορεί να οφείλεται σε λόγους που αφορούν τις απαιτήσεις του οργανισμού που δημοσιεύει την υπηρεσία. Από την άλλη μεριά, η έλλειψη συνεκτικότητας έχει αποδειχτεί ότι έχει αρνητικά αποτελέσματα στη συντήρηση του λογισμικού. Επιπρόσθετα στον τομέα των υπηρεσιών αυτό μπορεί να έχει αντίκτυπο σε πολλές εφαρμογές που καταναλώνουν την υπηρεσία.

Επί παραδείγματι, η Amazon ένας από τους μεγαλύτερους παρόχους υπηρεσιών, προσφέρει μια μεγάλη ποικιλία υπηρεσιών μέσω του διαδικτύου. Ανάμεσα σε αυτές, η Amazon Simple Queue Service (SQS 2007) επιτρέπει επικοινωνία μέσα από ουρές μηνυμάτων. Η υπηρεσία αυτή αποτελείται από δύο διεπαφές που φαίνονται παρακάτω στο σχήμα 1.2.



Σχήμα 1.2 Οι Διεπαφές της Υπηρεσίας Amazon SQS2007

Η πρώτη διεπαφή που ονομάζεται QueueServicePortType, παρέχει 2 λειτουργίες που επιτρέπουν την δημιουργία νέας ουράς και την παρουσίαση των ήδη διαθέσιμων ουρών αντίστοιχα. Η δεύτερη διεπαφή, καλείται MessageQueuePortType, διαθέτει 9 λειτουργίες που ενεργοποιούν την διαγραφή ουράς, τον ορισμό χαρακτηριστικών σε

μια ουρά, την πρόσθεση ή αφαίρεση άδειας πρόσβασης σε μια ουρά, την αποστολή μηνυμάτων σε μια ουρά, την λήψη μηνυμάτων από ουρά και την αλλαγή ορατότητας μηνυμάτων που βρίσκονται σε μια ουρά.

Παρατηρείται, ότι διαφορετικές λειτουργίες είναι κατανεμημένες σε δύο διεπαφές. Δηλαδή, λειτουργίες που αφορούν την διαχείριση ουράς βρίσκονται και στις δύο διεπαφές. Επίσης, λειτουργίες που διαχειρίζονται την πρόσβαση σε μια ουρά είναι στην ίδια διεπαφή με λειτουργίες που αφορούν την ανταλλαγή μηνυμάτων. Αυτό με μια πρώτη ματιά δείχνει προβληματικό. Η λανθασμένη κατανομή λειτουργιών σε διεπαφές κάνει την υπηρεσία πολύπλοκη για τους προγραμματιστές των διαδικτυακών εφαρμογών. Ένας προγραμματιστής που υλοποιεί έναν διαχειριστή ουράς, θα έπρεπε να μελετήσει και τις δύο διεπαφές, δηλαδή ένα σύνολο από 11 διαφορετικές λειτουργίες, από όπου μόνο 5 είναι σχετικές με την διαχείριση ουράς. Όμοια, ένας προγραμματιστής που σκοπεύει να αναπτύξει έναν πελάτη ουράς που επικοινωνεί με άλλες ουρές στέλνοντας και λαμβάνοντας μηνύματα πρέπει να μελετήσει την διεπαφή MessageQueuePortType, που έχει 9 λειτουργίες. Από αυτές όμως, μόνο 4 είναι σχετικές με ανταλλαγή μηνυμάτων.

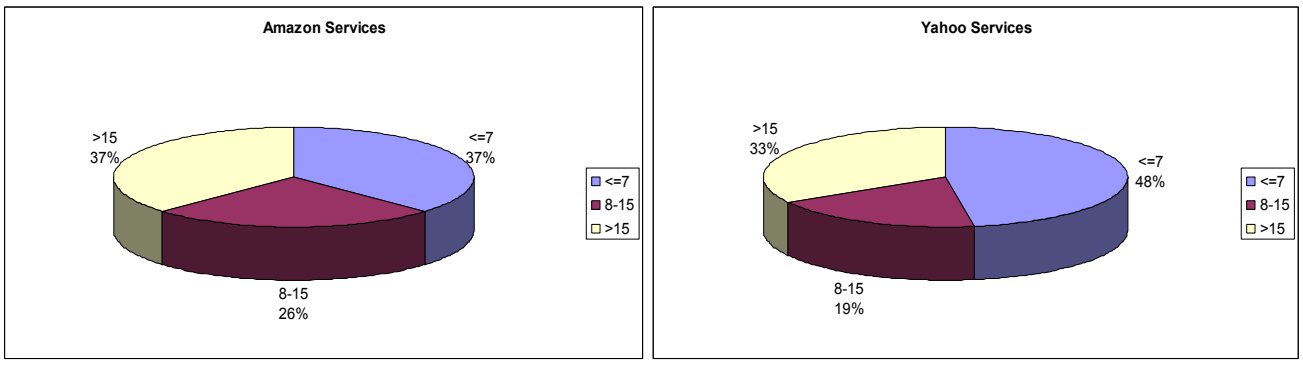
Άρα, η ποιότητα του λογισμικού σε εφαρμογές βασισμένες στις υπηρεσίες διαδικτύου είναι ένα θέμα που ακόμα εγείρει ερωτηματικά. Αν αναλογιστούμε ότι πρέπει να γίνουν αλλαγές στην διεπαφή MessageQueuePortType, τότε από αυτές τις αλλαγές θα επηρεαστούν φαινομενικά και οι δύο εφαρμογές που περιγράφηκαν παραπάνω, του διαχειριστή ουράς και του πελάτη ουράς. Οι προγραμματιστές θα πρέπει να ελέγξουν αν επηρεάζονται και ουσιαστικά οι εφαρμογές τους ή όχι. Για παράδειγμα, αν οι αλλαγές γίνουν στις λειτουργίες ανταλλαγής μηνυμάτων, τότε επί της ουσίας θα επηρεαστεί μόνο η εφαρμογή του πελάτη ουράς.

Γενικότερα μπορούμε να αναφερθούμε στον νόμο του Miller [10], που δείχνει ότι ο μέσος αριθμός αντικειμένων που μπορεί να θυμάται ένας άνθρωπος είναι 7. Αυτή η παρατήρηση, σε συνδυασμό με τα προβλήματα συνεκτικότητας υποδηλώνουν ότι μεγάλες και μη-συνεκτικές διεπαφές δεν είναι βολικές για τους προγραμματιστές

εφαρμογών που βασίζονται σε τέτοιες υπηρεσίες. Παρ' όλα αυτά στην πράξη και πιο συγκεκριμένα στο παράδειγμα της Amazon, το 63% των υπηρεσιών που παρέχονται, έχουν μεγάλες διεπαφές (Σχήμα 1.3). Ειδικά, στην χειρότερη περίπτωση μια υπηρεσία παρέχει διεπαφή με 87 λειτουργίες. Η κατάσταση είναι παρόμοια και στον άλλο μεγάλο πάροχο υπηρεσιών, την Yahoo.

Προβλήματα από μεγάλες και μη-συνεκτικές διεπαφές προκύπτουν και στη συντήρηση των εφαρμογών που τις χρησιμοποιούν. Για παράδειγμα μπορούμε να αναφερθούμε στις διαφορετικές εκδόσεις της SQS υπηρεσίας μετά το 2007. Στην έκδοση του 2008 οι λειτουργίες πρόσβασης για διαχείριση ουράς αφαιρέθηκαν, ενώ στην έκδοση του 2009 αυτές οι λειτουργίες άλλαξαν όνομα και προστέθηκαν ξανά στην διεπαφή της υπηρεσίας. Παρόμοιες καταστάσεις αφαίρεσης λειτουργιών από διεπαφές προέκυψαν για τις υπηρεσίες της Amazon, Flexible Payment Service και Mechanical Turk. Σε τέτοιες περιπτώσεις, μια πιο συνεκτική διάσπαση των σχετικών λειτουργιών σε ξεχωριστές διεπαφές, θα απλοποιούσε την προσπάθεια ενός προγραμματιστή να εντοπίσει αν μια αλλαγή επηρεάζει την εφαρμογή του και σε ποιο σημείο του κώδικα επηρεάζεται από αυτήν. Περαιτέρω έρευνα [12] έδειξε ότι η συνεκτικότητα στις διεπαφές των υπηρεσιών μπορεί να βελτιώσει την ανάλυση των σφαλμάτων που εμφανίζονται στις εφαρμογές που χρησιμοποιούν υπηρεσίες και συνεπώς να συμβάλει στην συντήρηση τέτοιων εφαρμογών.

Πίνακας 1.1 Κατανομές των Υπηρεσιών Amazon και Yahoo με βάση το Μέγεθος των Διεπαφών



Η ανακατασκευή στα αντικειμενοστρεφή συστήματα γίνεται διατηρώντας την συμπεριφορά του κώδικα. Η ανακατασκευή επιτρέπει την αυτόματη ανακατανομή κομματιών κώδικα, ανάμεσα στην ιεραρχία των κλάσεων. Ο σκοπός είναι να βελτιωθεί η ποιότητα του συστήματος. Μια μετρική που δείχνει την ποιότητα του συστήματος είναι η συνεκτικότητα μεταξύ των οντοτήτων που διέπουν το σύστημα. Μερικές λειτουργίες που βελτιώνουν την σχεδίαση του λογισμικού είναι η διάσπαση ή συγχώνευση κλάσεων, η μεταφορά ή η προσθήκη μεθόδων σε μια κλάση ή ακόμα και διαγραφή κλάσεων.

Προσαρμόζοντας όλα τα παραπάνω στο πεδίο των υπηρεσιών διαδικτύου, η εργασία αυτή εστιάζει στην ανακατασκευή των διεπαφών με κριτήριο την συνεκτικότητα μεταξύ των λειτουργιών. Η ανακατασκευή λογισμικού περιλαμβάνει διάφορες τεχνικές, όμως η ανακατασκευή διεπαφών στις υπηρεσίες δεν έχει διερευνηθεί σε βάθος. Γίνεται προσπάθεια να δοθούν απαντήσεις σε θέματα όπως:

- επιλογή μετρικής συνεκτικότητας που είναι καλύτερη για ανακατασκευή διεπαφών
- αυτοματοποίηση της διαδικασίας ανακατασκευής για τους παρόχους υπηρεσιών
- σύγκριση των καλύτερων τεχνικών ανακατασκευής.

Για να αντιμετωπιστεί το πρώτο θέμα, προτείνεται μια μετρική που εκτιμά την σχετικότητα των λειτουργιών βάσει κοινών εννοιών στα ονόματα τους. Ακόμα εξετάζονται και συγκρίνονται με την προηγούμενη 2 μετρικές έλλειψης συνεκτικότητας που υπολογίζουν την ομοιότητα των λειτουργιών μιας διεπαφής, με βάση την ομοιότητα των δεδομένων που χρησιμοποιούν σαν είσοδο και έξοδο [2].

Για το δεύτερο και τρίτο ερώτημα, προτείνονται και συγκρίνονται μηχανισμοί που αυτοματοποιούν τη διαδικασία ανακατασκευής των διεπαφών. Δέχονται ως είσοδο διεπαφές υπηρεσιών και παράγουν ως έξοδο έναν διαφορετικό τρόπο σχεδίασης της αρχικής διεπαφής, που είναι πιο συνεκτικός. Η μια εκδοχή ξεκινά με την αρχική διεπαφή και την αποσυνθέτει σε απλούστερες, προτείνοντας πιο συνεκτικά σύνολα

διεπαφών σε μια από πάνω προς τα κάτω προσέγγιση (top – down), ενώ η δεύτερη, θεωρεί τις λειτουργίες μιας διεπαφής ως ανεξάρτητες διεπαφές με μοναδικές λειτουργίες, ομαδοποιώντας τις πιο όμοιες, με προσέγγιση από κάτω προς τα πάνω (bottom - up). Ο πυρήνας αυτών των μηχανισμών είναι η ιεραρχική συσταδοποίηση (Hierarchical Clustering), προσαρμοσμένη στις ιδιαιτερότητες του προβλήματος μας.

Για να εκτιμηθεί η προτεινόμενη προσέγγιση, εξετάσθηκαν υπηρεσίες της Amazon και Yahoo. Συγκεκριμένα, συγκρίνονται διαφορετικές εκδοχές του μηχανισμού που προκύπτουν σε συνδυασμό με τις 3 μετρικές που αναφέρθηκαν από ποσοτικής πλευράς, για παράδειγμα αν μια διεπαφή ανακατασκευάζεται σε έναν λογικό αριθμό από άλλες διεπαφές που αποτελούνται από λογικό αριθμό λειτουργιών η κάθε μια. Επίσης, τα αποτελέσματα της ανακατασκευής διεπαφών από τον μηχανισμό εξετάζονται από ποιοτικής πλευράς, δηλαδή αν κατά πόσο τα αποτελέσματα που παράγονται θα ήταν χρήσιμα για τους σχεδιαστές των υπηρεσιών.

1.3. Δομή της Διατριβής.

Η διατριβή έχει δομηθεί ως εξής. Στο Κεφάλαιο 2 παρουσιάζεται το θεωρητικό υπόβαθρο της εργασίας, καθώς και σχετικές εργασίες που έχουν πραγματοποιηθεί και αφορούν ανακατασκευή λογισμικού. Στο Κεφάλαιο 3 παρουσιάζονται αναλυτικά οι μετρικές συνεκτικότητας που χρησιμοποιούνται για την ανακατασκευή των διεπαφών υπηρεσιών διαδικτύου. Ακολούθως, το Κεφάλαιο 4, περιγράφει λεπτομερώς τους μηχανισμούς ανακατασκευής διεπαφών. Στο Κεφάλαιο 5 παρουσιάζεται η πειραματική μελέτη με την οποία παρουσιάζονται τα οφέλη της ανακατασκευής διεπαφών υπηρεσιών και τέλος στο Κεφάλαιο 6 περιγράφονται τα συμπεράσματα της διατριβής και οι μελλοντικές κατευθύνσεις που δίνονται.

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ ΚΑΙ ΣΧΕΤΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

Στο κεφάλαιο αυτό παρουσιάζεται το θεωρητικό υπόβαθρο στο οποίο βασίστηκε η διατριβή. Αρχικά περιγράφεται πότε ένα λογισμικό θεωρείται ποιοτικό (υπό-ενότητα 2.1) και από τι εξαρτάται η ποιότητα ενός λογισμικού, ακολούθως ορίζονται μετρικές συνεκτικότητας του λογισμικού και πως αυτές συμμετέχουν ως κριτήριο ανακατασκευής του λογισμικού (υπό-ενότητα 2.2). Επίσης γίνεται μια εισαγωγή σε μετρικές συνεκτικότητας για υπηρεσίες διαδικτύου. Τέλος, περιγράφονται τεχνικές ανακατασκευής λογισμικού (υπό-ενότητα 2.3)

2.1. Ποιότητα Λογισμικού

Μια βασική ιδιότητα του λογισμικού στις μέρες μας είναι η ανάγκη εξέλιξης του. Στον κύκλο ζωής του λογισμικού, η εξέλιξη του στις νέες απαιτήσεις περιλαμβάνει το 60% του συνολικού του κόστους. Όσο το λογισμικό ή οι γραμμές κώδικα αυξάνονται, τροποποιείται και προσαρμόζεται σε νέες απαιτήσεις, ο κώδικας γίνεται πολύπλοκος, διαφοροποιείται από την αρχική σχεδίαση, με αποτέλεσμα την κακή ποιότητα του. Υπολογίζεται ότι περίπου 50-90% της δουλειάς που γίνεται για την εξέλιξη του λογισμικού, επικεντρώνεται πρώτα στην κατανόηση του, προκειμένου να γίνουν οι απαραίτητες αλλαγές [23].

Ένα κακής ποιότητας λογισμικό συνήθως περιλαμβάνει μεθόδους και συναρτήσεις που υλοποιούν πολύπλοκο έργο, κάτι που κάνει ένα πρόγραμμα δύσκολο στην κατανόηση. Συνήθως η ανάπτυξη ενός συστήματος γίνεται με αυστηρό χρονοδιάγραμμα με αποτέλεσμα οι προγραμματιστές να δίνουν βάρος στην υλοποίηση των λειτουργιών παρά στα ποιοτικά χαρακτηριστικά που διέπουν τον προγραμματισμό. Είναι δύσκολο να διατηρηθεί μια ποιοτική σχεδίαση καθ' όλη τη διάρκεια της προγραμματιστικής διαδικασίας. Ακόμα και όταν ένα λογισμικό είναι καλά σχεδιασμένο, στο πέρασμα του χρόνου ο κώδικας τροποποιείται ώστε να ανταποκριθεί στις συνεχόμενες ανάγκες των πελατών και των τεχνολογιών. Η αρχική δομή σταδιακά αλλάζει προς το χειρότερο. Ο κώδικας γίνεται δύσκολος στην κατανόηση και αυξάνεται το κόστος συντήρησης.

Για να αντιμετωπιστεί το πρόβλημα της κακής ποιότητας του λογισμικού αναπτύχθηκαν τεχνικές που προτείνουν την βελτίωση του. Το πεδίο της έρευνας που αντιμετωπίζει αυτά τα ζητήματα ονομάζεται ανακατασκευή / αναδόμηση. Η ανακατασκευή του κώδικα ορίζεται ως εξής:

Ο μετασχηματισμός από μια σχεδίαση σε μια άλλη, στο ίδιο συγκριτικό επίπεδο αφαίρεσης, ενώ διατηρείται η εξωτερική συμπεριφορά του συστήματος (λειτουργικότητα και σημασιολογία) [12].

Ένας μετασχηματισμός ανακατασκευής στοχεύει στην αλλαγή της δομής του κώδικα με απώτερο σκοπό την βελτίωση της ποιότητας του λογισμικού. Η ανακατασκευή δημιουργεί νέες εκδόσεις που προτείνουν αλλαγές στο σύστημα, στην ουσία δεν περιλαμβάνει αλλαγές εξαιτίας των νέων απαιτήσεων. Παρ' όλα αυτά, μπορεί να οδηγήσει σε καλύτερη οπτική του συστήματος που θα ευνοεί αλλαγές οι οποίες θα βελτιώνουν πτυχές του συστήματος.

Ποιοτικό είναι το λογισμικό που ικανοποιεί τις άμεσες και έμμεσες απαιτήσεις του χρήστη, είναι καλά τεκμηριωμένο και εκτελείται αποτελεσματικά στο υλικό για το οποίο έχει αναπτυχθεί. Ένα κριτήριο που δείχνει πότε το λογισμικό έχει καλή

ποιότητα είναι η συνεκτικότητα. Ένα ποιοτικό λογισμικό όταν έχει υψηλή συνεκτικότητα τα συστατικά του είναι σαφώς διαχωρισμένα και ομαδοποιημένα σε ανεξάρτητες οντότητες (κλάσεις) σύμφωνα με την λειτουργία που επιτελούν. Η έλλειψη συνεκτικότητας δείχνει ότι το λογισμικό είναι προβληματικό και πρέπει να αντιμετωπιστεί με τις μεθόδους ανακατασκευής. Η συνεκτικότητα είναι το κριτήριο που επικεντρώνεται αυτή η εργασία προκειμένου να βελτιωθεί η ποιότητα των υπηρεσιών διαδικτύου.

2.2. Μετρικές Συνεκτικότητας

Η συνεκτικότητα παραδοσιακά ορίζεται ως ο βαθμός που τα στοιχεία μιας οντότητας σχετίζονται το ένα με το άλλο, δηλαδή χρησιμοποιεί το ένα το άλλο. Για να οριστεί μια μετρική συνεκτικότητας γενικά πρέπει να πληρούνται κάποια κριτήρια [5]. Πρώτη προϋπόθεση είναι η αναγνώριση των εννοιών, δηλαδή ποιες θα είναι οι οντότητες που θα εκτιμηθούν. Στη συνέχεια ακολουθεί η επιλογή των χαρακτηριστικών τα οποία θα μετρηθούν για τις οντότητες και τέλος γίνεται η επιλογή μονάδων για την μέτρηση των χαρακτηριστικών και ο τύπος κλίμακας.

Υπάρχουν διάφορα είδη συνεκτικότητας [25]. Οι τύποι συνεκτικότητας από την χειρότερη προς την καλύτερη είναι:

Συμπτωματική συνεκτικότητα (Coincidental cohesion), προκύπτει όταν στοιχεία μιας οντότητας ομαδοποιούνται αυθαίρετα, η μόνη σχέση ανάμεσα στα μέρη της οντότητας είναι ότι βρίσκονται μαζί.

Λογική συνεκτικότητα (Logical cohesion), προκύπτει όταν τα στοιχεία μιας οντότητας έχουν ομαδοποιηθεί έτσι επειδή έχουν κατηγοριοποιηθεί λογικά να κάνουν το ίδιο έργο, ακόμα και αν είναι διαφορετικά ως προς τη φύση τους. Δηλαδή ομαδοποίηση όλων των συναρτήσεων που χειρίζονται είσοδο από το πληκτρολόγιο (keyboard) και το ποντίκι (mouse).

Διαδικασιακή συνεκτικότητα (Procedural cohesion), προκύπτει όταν στοιχεία μιας οντότητας ομαδοποιούνται επειδή πάντα ακολουθούν μια συγκεκριμένη σειρά εκτέλεσης (π.χ. συνάρτηση που πρώτα ελέγχει την πρόσβαση σε αρχεία και μετά τα ανοίγει).

Επικοινωνιακή συνεκτικότητα (Communicational Cohesion), προκύπτει όταν τα στοιχεία μιας οντότητας ομαδοποιούνται επειδή χρησιμοποιούν τα ίδια δεδομένα (π.χ. μια μέθοδος που χρησιμοποιεί την ίδια εγγραφή πληροφορίας).

Ακολουθιακή συνεκτικότητα (Sequential Cohesion), προκύπτει όταν τα στοιχεία της οντότητας ομαδοποιούνται μαζί, επειδή η έξοδος από ένα, αποτελεί είσοδο για άλλα στοιχεία, υπάρχει δηλαδή μια σειριακή συνδεσμολογίας μεταξύ των στοιχείων.

Λειτουργική συνεκτικότητα (Functional Cohesion), προκύπτει όταν τα στοιχεία μιας οντότητας ομαδοποιούνται μαζί, επειδή συνεισφέρουν στην υλοποίηση ενός καλά καθορισμένου σκοπού που επιτελεί η οντότητα.

Έχει αποδειχθεί ότι οι τρεις τελευταίες κατηγορίες δίνουν καλά αποτελέσματα και θεωρούνται οι καλύτερες μορφές συνεκτικότητας [26]. Αυτές οι μορφές συνεκτικότητας χρησιμοποιήθηκαν και στην παρούσα εργασία προσαρμοσμένες στις ανάγκες των υπηρεσιών διαδικτύου. Όλα αυτά περιγράφονται αναλυτικότερα στο κεφάλαιο 3.

2.2.1. Μετρικές Αντικειμενοστρεφούς Σχεδίασης

Στα αντικειμενοστρεφή συστήματα, οι κλάσεις μπορεί να γίνουν αρκετά μεγάλες και μη-συνεκτικές σε αναλογία με αυτά που περιγράφτηκαν προηγουμένως. Στον αντικειμενοστρεφή προγραμματισμό, η βασική αρχή είναι ότι μια κλάση πρέπει να υλοποιεί μια μόνο διαδικασία. Από την σκοπιά συντήρησης λογισμικού αυτό μπορεί να σημαίνει, ότι για να τροποποιηθεί μια κλάση πρέπει να υπάρχει μια μόνο αιτία. Η παραβίαση αυτής της αρχής οδηγεί σε μεγάλες και πολύπλοκες κλάσεις. Αυτό μπορεί

να προκύψει με δυο τρόπους, είτε μια κλάση χρησιμοποιεί πολλά δεδομένα (χαρακτηριστικά), είτε εμπεριέχει μεγάλο μέρος της λειτουργικότητας του συστήματος, δηλαδή πολλές και πολύπλοκες μεθόδους Στην πρώτη περίπτωση, η ανακατανομή των χαρακτηριστικών ή η μεταφορά μεθόδων από άλλες κλάσεις που χρησιμοποιούν αυτά τα χαρακτηριστικά μπορεί να δώσει λύση. Στην δεύτερη περίπτωση, η μεταφορά μεθόδων από την μεγάλη κλάση προς άλλες ανάλογα με τα χαρακτηριστικά που χρησιμοποιούνται ή ο διαχωρισμός της κλάσης απομακρύνοντας ανεξάρτητες μεθόδους σε νέες κλάσεις δίνει λύση στο πρόβλημα [5, 8].

Στα αντικειμενοστρεφή συστήματα μια από τις πιο κλασικές μετρικές είναι η *έλλειψη συνεκτικότητας μεταξύ των Μεθόδων* (Lack Of Cohesion Of Methods) που προτάθηκε από τους Chidamber και Kemmerer [3]. Σύμφωνα με αυτή κάθε μέθοδος σε μια κλάση θεωρείται ότι προσπελαύνει ένα ή περισσότερα κοινά μέλη δεδομένων. Δύο μέθοδοι είναι συνεκτικές εάν τα σύνολα των μελών δεδομένων που χρησιμοποιούν έχουν κοινά στοιχεία. Έστω μια κλάση *C* που περιλαμβάνει ένα σύνολο $M=\{m_1, m_2, \ldots, m_n\}$ μεθόδων. Για την κλάση *C* σχηματίζονται δύο σύνολα από ζεύγη μεθόδων της κλάσης: έστω *P*, το σύνολο των μη συνεκτικών ζευγών μεθόδων (δηλαδή τα ζεύγη μεθόδων που δεν χρησιμοποιούν κανένα κοινό πεδίο) και *Q*, το σύνολο των συνεκτικών ζευγών μεθόδων της κλάσης *C* (δηλαδή τα ζεύγη μεθόδων που χρησιμοποιούν τουλάχιστον ένα κοινό πεδίο), η μετρική *LCOM* ορίζεται ως εξής:

$$LCOM(C) = \{ |P| - |Q|, \ \alpha\nu \ |P| > |Q| \ \ \acute{\eta} \ 0, \ \alpha\nu \ |P| \leq |Q| \qquad \text{Εξ. 2.1}$$

Υπάρχουν και δύο παραπλήσιες μετρικές της *LCOM*. Έστω *a* το πλήθος των χαρακτηριστικών της κλάσης *C* και *mA*, ο αριθμός των μεθόδων που έχουν πρόσβαση σε κάποιο χαρακτηριστικό *A*, τότε οι μετρικές *LCOM2* και *LCOM3* ορίζονται ως εξής:

$$LCOM2(C) = 1 - \frac{\sum_{\forall A \in C} mA}{M * a} \qquad \text{Εξ. 2.2}$$

$$LCOM3(C) = \frac{M - \frac{\sum_{\forall A \in C} mA}{a}}{M - 1}$$ Εξ. 2.3

Η μετρική *LCOM2* ισούται με το ποσοστό των μεθόδων που δεν έχουν πρόσβαση σε κάποιο χαρακτηριστικό προς όλα τα χαρακτηριστικά της κλάσης. Αν ο αριθμός των μεθόδων ή χαρακτηριστικών είναι μηδενικός, τότε η τιμή της *LCOM2* δεν ορίζεται. Για την LCOM3 οι τιμές 1 ως 2 θεωρούνται ανησυχητικές. Σε μια καλά σχεδιασμένη κλάση που οι μέθοδοι έχουν πρόσβαση σε όλα τα μέλη δεδομένων της κλάσης, η τιμή της *LCOM3* είναι 0 (υψηλή συνεκτικότητα). Ενώ, όταν *LCOM3*=1 τότε υπάρχει μεγάλη έλλειψη συνεκτικότητας και η κλάση πρέπει να διασπαστεί. Όταν υπάρχουν μέλη δεδομένων που δεν χρησιμοποιούνται από καμιά μέθοδο της κλάσης τότε $1 < LCOM3 < 2$.

Στην έρευνα τους οι Simon, Steinbruckner και Lewerentz [17] δείχνουν πως οι μετρικές όπως και η συνεκτικότητα μπορούν να βοηθήσουν στον εντοπισμό ανωμαλιών που χρήζουν ανακατασκευής. Τονίζοντας, πως ο τελευταίος υπεύθυνος για το που θα γίνει ανακατασκευή είναι ο προγραμματιστής. Αυτός θα πάρει τις αποφάσεις με βάση τις ενδείξεις των μετρικών. Στην εν λόγω εργασία χρησιμοποιήθηκαν 4 λειτουργίες για να ανακατασκευαστεί ο κώδικας. Αυτές ήταν η *μεταφορά μεθόδων* από μια κλάση Α σε μια κλάση Β που χρησιμοποιεί περισσότερο την εκάστοτε μέθοδο. Η αρχική μέθοδος της κλάσης Α θα πρέπει να αφαιρεθεί. Το κίνητρο για αυτήν την ανακατασκευή είναι ότι μια μέθοδος χρησιμοποιεί ή χρησιμοποιείται από περισσότερα χαρακτηριστικά μιας άλλης κλάσης, παρά στην κλάση στην οποία ορίζεται. Η δεύτερη λειτουργία αφορά την *μεταφορά χαρακτηριστικών*. Για παράδειγμα μεταφορά ενός χαρακτηριστικού από μια κλάση Α σε μια άλλη κλάση Β που χρησιμοποιεί το χαρακτηριστικό περισσότερο. Όλοι οι χρήστες αυτού του χαρακτηριστικού (κλάσεις, μέθοδοι) πρέπει να τροποποιηθούν. Τρίτη λειτουργία είναι η *εξαγωγή κλάσης*, δηλαδή δημιουργείται νέα κλάση όπου συνεκτικά, εξαρτώμενα χαρακτηριστικά και μέθοδοι από μια προηγούμενη κλάση τοποθετούνται στη νέα κλάση. Το κίνητρο για αυτήν την ανακατασκευή είναι ότι μια

κλάση παρέχει πολύπλοκη λειτουργικότητα, δηλαδή οι λόγοι για να τροποποιηθεί είναι παραπάνω από ένας. Έτσι πρέπει να διασπαστεί το λιγότερο σε δυο νέες κλάσεις. Και τέλος, μια άλλη μέθοδος ανακατασκευής είναι η *εισαγωγή κλάσης*, που περιλαμβάνει την μεταφορά όλων των μελών μιας κλάσης σε μια άλλη και διαγραφή της πρώτης γιατί δεν υλοποιεί κάποια λειτουργία και δεν έχει λόγο ύπαρξης.

2.2.2. Μετρικές Συνεκτικότητας για Υπηρεσίες Διαδικτύου

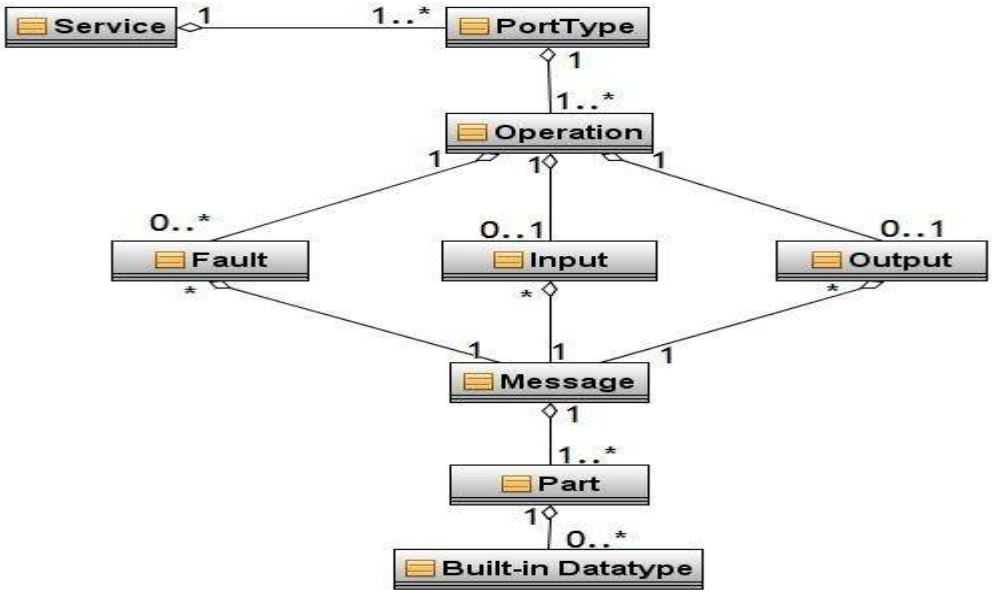
Στον τομέα των υπηρεσιών διαδικτύου έχουν οριστεί μετρικές συνεκτικότητας [2] με βάση ένα μοντέλο που περιγράφεται στο Σχήμα 2.1. Συγκεκριμένα, μια υπηρεσία παρέχει ένα σύνολο από διεπαφές. Μια διεπαφή αποτελείται από ένα σύνολο λειτουργιών. Κάθε λειτουργία επιτελεί ένα έργο και η εκτέλεση της απαιτεί το πολύ ένα μήνυμα εισόδου και παράγει το πολύ ένα μήνυμα εξόδου, τα οποία μπορεί να είναι κοινά σε πολλές λειτουργίες.

Ένα μήνυμα μπορεί να αποτελείται από ένα διακριτό σύνολο από μέρη (parts). Κάθε μήνυμα αποτελείται από μηδέν έως περισσότερα μέρη (parts), ενώ τα συστατικά των μηνυμάτων συνήθως καθορίζονται στο XML Schema που συνοδεύει το WSDL αρχείο, το οποίο περιγράφει την υπηρεσία. Στο XML Schema λοιπόν, ορίζονται τα δεδομένα των συστατικών, άρα τα δεδομένα που περιέχονται στα μηνύματα (Σχήμα 2.1).

Για να εκτιμηθεί η συνεκτικότητα μιας διεπαφής υπηρεσίας ελέγχονται οι σχέσεις μεταξύ των λειτουργιών της διεπαφής. Με βάση τις έννοιες της επικοινωνιακής και ακολουθιακής συνεκτικότητας που ορίστηκαν στο [26], μπορούμε να συσχετίσουμε δυο λειτουργίες με τους ακόλουθους τρόπους αντίστοιχα:

- Αν οι λειτουργίες χρησιμοποιούν ίδια είσοδο και / ή έξοδο.
- Αν η έξοδος μιας λειτουργίας μπορεί να αποτελεί είσοδο για μια άλλη.

Στην παρούσα εργασία χρησιμοποιήθηκαν οι μετρικές που προτάθηκαν στην εργασία [2] προσαρμοσμένες στις συσχετίσεις εισόδου / εξόδου ανάμεσα στις λειτουργίες μιας διεπαφής (θα αναλυθούν περαιτέρω το κεφάλαιο 3).



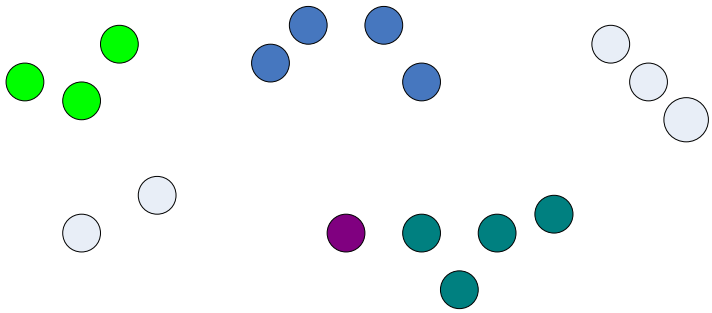
Σχήμα 2.1 Μοντέλο Υπηρεσιών Διαδικτύου κατά το Πρότυπο W3C

2.3. Τεχνικές Ανακατασκευής

Η πιο συνηθισμένη τεχνική ανακατασκευής συστημάτων είναι η διαίρεση του αρχικού σχεδιασμού σε μικρότερα πιο συνεκτικά υποσυστήματα, είτε η συνένωση υποσυστημάτων σε μεγαλύτερα υποσυστήματα με τον ίδιο σκοπό, να είναι συνεκτικά.

Η διαίρεση και η συνένωση οντοτήτων είναι τεχνικές που χρησιμοποιούνται σε διάφορους τομείς. Μερικές από αυτές είναι η θεωρία γραφημάτων, επιχειρηματική ανάλυση, ανάκτηση πληροφορίας, σχεδίαση κυκλωμάτων, οικονομία και βιολογία. Με γενικούς όρους μπορούμε να ορίσουμε την ανακατασκευή ως την δημιουργία ομάδων από στοιχεία (κλάσεις) με τέτοιον τρόπο ώστε τα στοιχεία μιας ομάδος να είναι στενά αλληλοεξαρτώμενα σε σχέση με την αλληλεξάρτηση που θα έχουν τα στοιχεία άλλων ομάδων. Οι ομάδες αυτές ονομάζονται συστάδες. Ένας ορισμός για

τις συστάδες είναι, μια συνεχόμενη περιοχή που περιλαμβάνει μια μεγάλη πυκνότητα σημείων εμφανώς ξεχωριστή από άλλες τέτοιες περιοχές, με περιοχές που δεν έχουν μεγάλη πυκνότητα σημείων (Σχήμα 2.2) [19].



Σχήμα 2.2 Γενική Ιδέα των Συστάδων

Οι μέθοδοι συσταδοποίησης χρησιμοποιούνται συχνά ως τεχνική ανακατασκευής. Γενικά, μπορεί να προκύψουν διαφορετικές ομαδοποιήσεις από διαφορετικές τεχνικές συσταδοποίησης. Έτσι ένας αλγόριθμος μπορεί να επιβάλλει μια δομή, παρά να βρει μια υπάρχουσα. Ακόμα μπορεί να δημιουργήσει μια δομή που δεν είναι λογική για τα δεδομένα. Αυτό κυρίως συμβαίνει σε διάσπαρτα δεδομένα που δεν υπάρχει καμιά λογική ομαδοποίηση.

Στην παρούσα εργασία, η συσταδοποίηση αποτέλεσε τη βασική τεχνική για τη βελτίωση των διεπαφών υπηρεσιών διαδικτύου, οπότε έπρεπε να διερευνηθούν τα παρακάτω θέματα:

- Σε ποιες οντότητες θα εφαρμοστούν οι τεχνικές ομαδοποίησης;
- Πότε είναι όμοιες δυο τέτοιες οντότητες;
- Ποιος αλγόριθμος θα εφαρμοστεί;

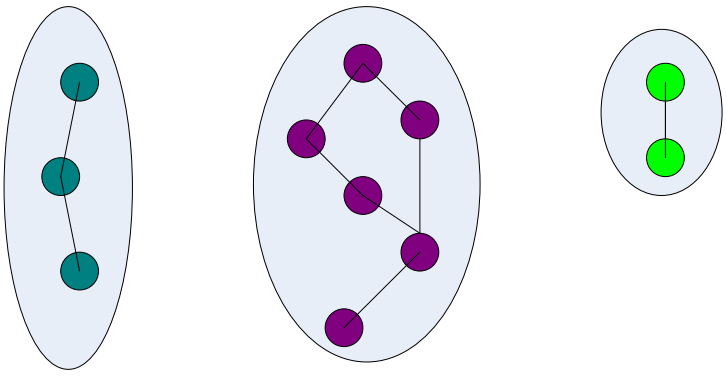
Γενικά, υπάρχουν αρκετές κατηγορίες αλγορίθμων συσταδοποίησης [4, 6, 19]:

- Αλγόριθμοι θεωρίας γραφημάτων
- Αλγόριθμοι κατασκευής
- Αλγόριθμοι βελτιστοποίησης

- Ιεραρχικοί Αλγόριθμοι

Οι αλγόριθμοι της θεωρίας γραφημάτων εφαρμόζονται σε γράφους. Οι κόμβοι αναπαριστούν τις οντότητες και οι ακμές τις σχέσεις μεταξύ τους (σχήμα 2.3). Αναζητούνται συνεκτικοί υπογράφοι που αντιπροσωπεύουν τις ενότητες ενός συστήματος (αρχικός γράφος). Οι αλγόριθμοι κατασκευής χρησιμοποιούν *γεωγραφικές τεχνικές* ή *τεχνικές αναζήτησης πυκνότητας* σε ένα επίπεδο δυο διαστάσεων προσπαθώντας να κάνουν ομαδοποιήσεις με βάση τις περιοχές που εμφανίζουν μεγάλη πυκνότητα από στοιχεία.

Οι αλγόριθμοι βελτιστοποίησης χρησιμοποιούν ευρετικές μεθόδους που προσπαθούν να βελτιώσουν μια αρχική τμηματοποίηση. Ξεκινάνε με μια αρχική ομαδοποίηση από συστάδες και οι οντότητες μετακινούνται ανάμεσα στις συστάδες ανάλογα με κάποιο κριτήριο μέχρι να σχηματιστεί η καλύτερη ομαδοποίηση. Οι πιο απλές μέθοδοι είναι η επαναληπτική τμηματοποίηση ή πιο απλά μη-ιεραρχικές μέθοδοι συσταδοποίησης. Τέλος, οι ιεραρχικοί αλγόριθμοι που ήταν ο κεντρικός μοχλός ανακατασκευής των διεπαφών υπηρεσιών διαδικτύου χωρίζονται σε δυο κατηγορίες. Στους διαιρετικούς και ενωτικούς. Και οι δυο κατηγορίες χτίζουν μια ιεραρχία από συστάδες με τέτοιο τρόπο όπου σε κάθε επίπεδο υπάρχουν οι ίδιες συστάδες με το προηγούμενο επίπεδο εκτός από 2 συστάδες που συνενώνονται για να σχηματίσουν μια νέα συστάδα.

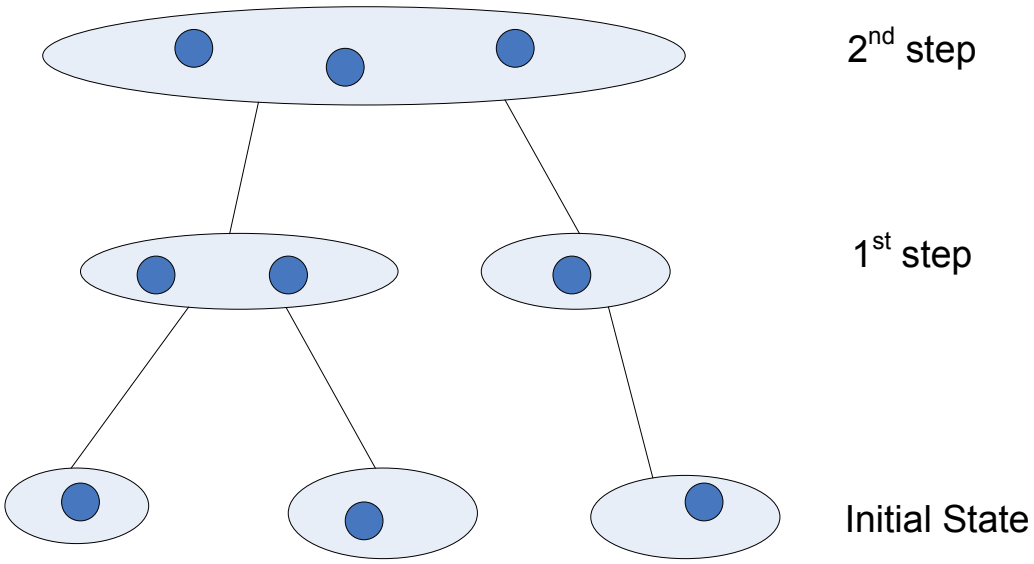


Σχήμα 2.3 Συνδεσιμότητα μεταξύ Κόμβων και Ανάλογη Συσταδοποίηση

2.3.1. Ιεραρχικοί Ενωτικοί

Οι ιεραρχικοί ενωτικοί αλγόριθμοι ξεκινούν από το αρχικό στάδιο της ιεραρχίας όπου υπάρχουν τόσες συστάδες, όσες και οι οντότητες. Κάθε οντότητα αποτελεί μια ομάδα. Σε κάθε βήμα δυο συστάδες συνενώνονται. Κάθε επίπεδο συνιστά μια διαφορετική προτεινόμενη συσταδοποίηση. Στόχος είναι να βρεθεί ένα σημείο τομής. Οι συστάδες σε εκείνο το επίπεδο είναι η έξοδος του αλγορίθμου. Οι ενωτικοί αλγόριθμοι συμφωνούν με την παρακάτω ακολουθία βημάτων (σχήμα 2.4):

1. Ξεκίνα με N συστάδες που η κάθε μια περιέχει μια οντότητα και υπολόγισε τους βαθμούς ομοιότητας μεταξύ όλων των οντοτήτων (συστάδες).
2. Όσο υπάρχουν περισσότερες από μια συστάδες:
   α. Βρες το πιο όμοιο ζευγάρι από συστάδες
   β. Ένωσε αυτές τις συστάδες
   γ. Ανανέωσε τους βαθμούς ομοιότητας ανάμεσα στις συστάδες

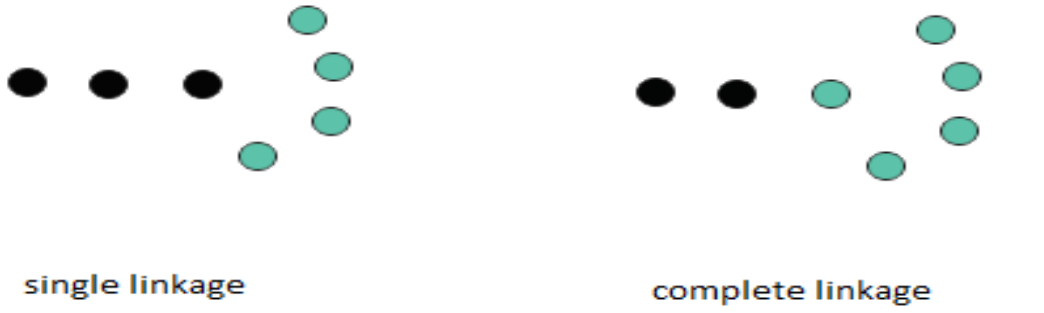


Σχήμα 2.4 Ιεραρχία Συστάδων για 3 Οντότητες

Γενικά, υπάρχουν διάφορες μέθοδοι υπολογισμού της ομοιότητας δύο συστάδων. Οι πιο συνήθεις είναι οι ακόλουθες οι οποίες και χρησιμοποιήθηκαν στα πλαίσια της παρούσας εργασίας:

- Ο πρώτος τρόπος βασίζεται στον κανόνα του *κοντινότερου γείτονα (single linkage)*. Με βάση αυτόν τον κανόνα η ομοιότητα δύο συστάδων Α, Β ισούται με την μέγιστη ομοιότητα μεταξύ των στοιχείων τους. Αντίστοιχα, η απόσταση των Α, Β με βάση αυτόν τον κανόνα ισούται με την ελάχιστη απόσταση μεταξύ των στοιχείων των δύο συστάδων.
- Ο δεύτερος τρόπος βασίζεται στον κανόνα του *μακρινότερου γείτονα (complete linkage)*. Με βάση αυτόν τον κανόνα η ομοιότητα δύο συστάδων Α, Β ισούται με την ελάχιστη ομοιότητα μεταξύ των στοιχείων τους. Αντίστοιχα, η απόσταση των Α, Β με βάση αυτόν τον κανόνα ισούται με την μέγιστη απόσταση μεταξύ των στοιχείων των δύο συστάδων.
- Ο τελευταίος κανόνας βασίζεται *στο μέσο όρο των ομοιοτήτων (average linkage)*. Συγκεκριμένα, με βάση αυτόν τον η ομοιότητα δύο συστάδων Α, Β ισούται με το μέσο όρο των ομοιοτήτων μεταξύ των στοιχείων τους. Αντίστοιχα, η απόσταση των Α, Β με βάση αυτόν τον κανόνα ισούται με το μέσο όρο των αποστάσεων μεταξύ των στοιχείων των δύο συστάδων.

Διαφορετικοί κανόνες ανανέωσης, μπορεί να έχουν ως αποτέλεσμα διαφορετικές ομαδοποιήσεις (Σχήμα 2.5).



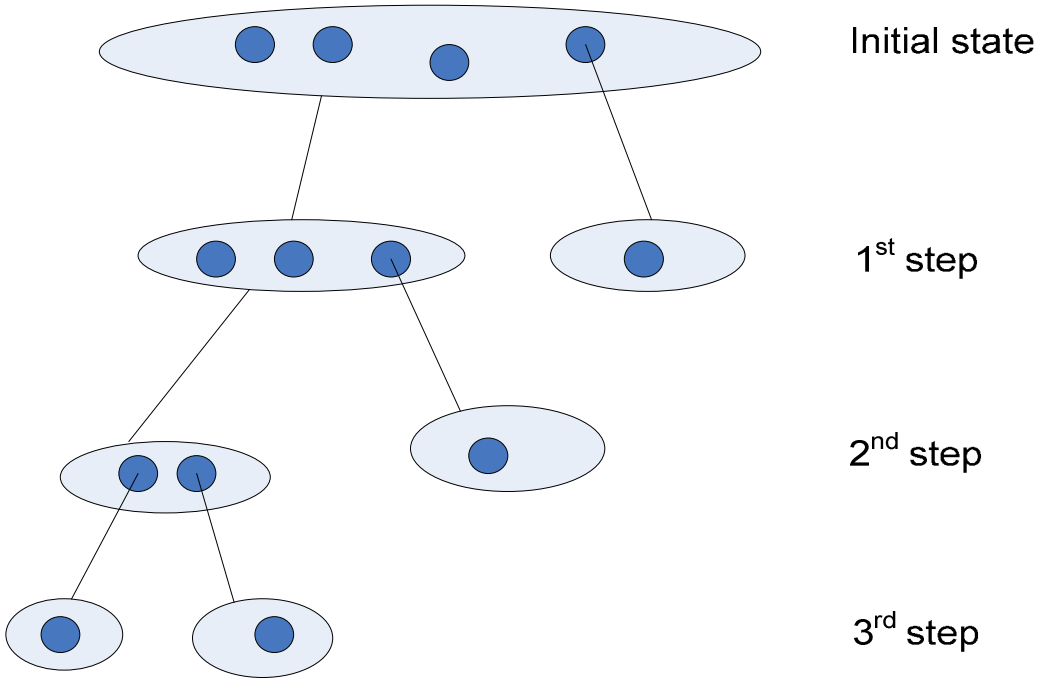
Σχήμα 2.5 Διαφορετικές Ομαδοποιήσεις από διαφορετικούς Κανόνες Ανανέωσης

Ο κανόνας του κοντινότερου γείτονα με βάση τον οποίο δυο σημεία από δυο ομάδες είναι κοντινά μεταξύ τους χωρίς να λαμβάνονται υπόψη τα άλλα, μπορεί να οδηγήσει στην δημιουργία "αλυσίδων". Όπου το ένα σημείο είναι κοντινό στο διπλανό του, αλλά μέσα στην συστάδα θα υπάρχουν και σημεία που μεταξύ τους δεν είναι τόσο όμοια. Αυτός ο κανόνας δημιουργεί μεγάλες συστάδες, παρότι κάποιες οντότητες των συστάδων είναι μακριά μεταξύ τους. Από την άλλη, ο κανόνας του μακρινότερου γείτονα λειτουργεί αντίθετα. Για να ενωθούν δυο συστάδες, κάθε οντότητα στις συστάδες πρέπει να είναι κοντινή σε κάθε άλλη οντότητα των δυο συστάδων. Ένα ζήτημα με τους αλγόριθμους ιεραρχίας είναι το κριτήριο τερματισμού. Αν δεν οριστεί τέτοιο κριτήριο οι ενωτικοί αλγόριθμοι φτάνουν στο τελευταίο επίπεδο ιεραρχίας, δηλαδή τοποθετούν όλες τις οντότητες σε μια ομάδα.

2.3.2. Ιεραρχικοί Διαιρετικοί

Οι διαιρετικές μέθοδοι επιβαρύνονται από το γεγονός της υπολογιστικής πολυπλοκότητας. Διάφορες μέθοδοι διαχειρίζονται αυτό το γεγονός λαμβάνοντας υπόψη μόνο ένα υποσύνολο από όλους τις πιθανούς συνδυασμούς διαίρεσης. Σε αυτή την μέθοδο αρχικά η συστάδα περιέχει όλες τις οντότητες και απομακρύνεται στο πρώτο επίπεδο η οντότητα που είναι η πιο ανόμοια από όλες τις άλλες. Έστω η ομάδα *A* και *α* μια οντότητα αυτής που είναι η πιο ανόμοια με όλες τις άλλες. Τότε, αυτή απομακρύνεται σχηματίζοντας μια αυτόνομη συστάδα, έστω *S*.

Σε κάθε επανάληψη, η οντότητα που είναι περισσότερο όμοια με τις οντότητες της συστάδας *S* από ότι με τις οντότητες της συστάδας *A*, απομακρύνεται από την ομάδα *A* προς την ομάδα *S* και οι ομοιότητες ανανεώνονται. Οι συστάδες *A* και *S* υπόκεινται στην ίδια διαδικασία στο επόμενο βήμα του αλγορίθμου (Σχήμα 2.6).



Σχήμα 2.6 Διαιρετικός Αλγόριθμος Τμηματοποίησης

ΚΕΦΑΛΑΙΟ 3. ΜΕΤΡΙΚΕΣ ΣΥΝΕΚΤΙΚΟΤΗΤΑΣ ΓΙΑ ΥΠΗΡΕΣΙΕΣ ΔΙΑΔΙΚΤΥΟΥ

Στο κεφάλαιο αυτό, αναλύονται οι μετρικές που χρησιμοποιήθηκαν στην εργασία για τον υπολογισμό της έλλειψης συνεκτικότητας στις υπηρεσίες διαδικτύου. Στην υπό-ενότητα 3.1 περιγράφονται η επικοινωνιακή και ακολουθιακή συνεκτικότητα, που βασίζονται στη συσχέτιση που υπάρχει μεταξύ των μηνυμάτων εισόδου / εξόδου που χρησιμοποιούν οι λειτουργίες των διεπαφών. Στη συνέχεια, στην ενότητα 3.2 ορίζεται η εννοιολογική συνεκτικότητα και τέλος δίνεται ένα παράδειγμα έλλειψης συνεκτικότητας με βάση αυτές τις μετρικές για την καλύτερη κατανόηση του προβλήματος έλλειψης συνεκτικότητας (υπό-ενότητα 3.3)

3.1. Επικοινωνιακή (Communicational) και Ακολουθιακή (Sequential) Συνεκτικότητα

Για να ποσοτικοποιηθούν οι επικοινωνιακή και ακολουθιακή συνεκτικότητα, χρησιμοποιήθηκαν μετρικές αντιστοιχίας που βασίζονται στην παραδοχή ότι οι λειτουργίες μιας διεπαφής υπηρεσίας σχετίζονται αν ταιριάζουν οι παράμετροι εισόδου / εξόδου. Σύμφωνα με το πρότυπο του W3C για την αρχιτεκτονική των υπηρεσιών [20] (Πίνακας 3.1), μία διεπαφή υπηρεσίας *I (Interface)*, χαρακτηρίζεται από ένα όνομα και ένα σύνολο από λειτουργίες (Πίνακας 3.1(1)). Με τη σειρά της, μια λειτουργία (Operation) καθορίζεται από το όνομα της, ένα μήνυμα εισόδου και ένα μήνυμα εξόδου (Πίνακας 3.1(3)). Γενικά, ένα μήνυμα μπορεί να αναπαρασταθεί

ως ένα δέντρο χωρίς διάταξη (Σχήμα 3.1). Η ρίζα του δέντρου αναπαριστά το μήνυμα. Οι εσωτερικοί κόμβοι αντιστοιχούν σε πολύπλοκα στοιχεία, όπου τα στοιχεία χαρακτηρίζονται από XML τύπους (complex type), που με τη σειρά τους αποτελούνται από άλλα συστατικά στοιχεία. Τα φύλλα του δέντρου αναπαριστούν βασικά στοιχεία. Αυτά τα βασικά στοιχεία χαρακτηρίζονται από βασικούς τύπους XML (built-in types).

Πίνακας 3.1 Το μοντέλο των υπηρεσιών

(1) $Interface = (name : string, O)$

(2) $O = \{op : Operation\}$

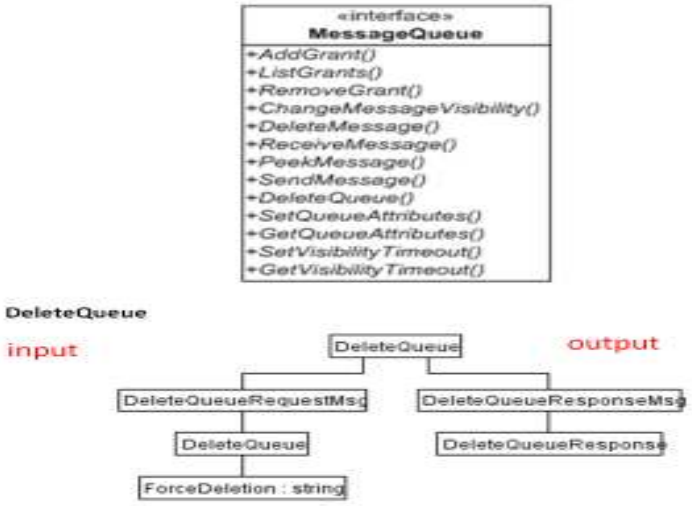
(3) $Operation = (name : string, in : Message, out : Message)$

(4) $Message = (V, E)$

(5) $V = \{u, element\}$

(6) $Element = (name : string, type : anyType)$

(7) $E = \{(u_i, u_j) \in V \times V \mid i \neq j\}$



Σχήμα 3.1 Παράδειγμα Λειτουργίας (Amazon SQS2007)

Έτσι, η ομοιότητα $MS(m_i,m_j)$ ανάμεσα σε δύο μηνύματα μπορεί να οριστεί χρησιμοποιώντας την μετρική Jaccard για τα μηνύματα $m_i.V$ και $m_j.V$ (Πίνακας 3.2(1)).

Η επικοινωνιακή ομοιότητα OpS_{com} δυο λειτουργιών op_i $op_j \in si.O$ μιας διεπαφής υπηρεσίας *si* ορίζεται ως ο μέσος όρος των ομοιοτήτων των μηνυμάτων εισόδου και των μηνυμάτων εξόδου αντίστοιχα (Πίνακας 3.2(2)). Όμοια η ακολουθιακή ομοιότητα OpS_{seq} δύο λειτουργιών op_i $op_j \in si$ μιας διεπαφής ορίζεται ως ο μέσος όρος της ομοιότητας του μηνύματος εισόδου της op_i και του μηνύματος εξόδου της op_j και της ομοιότητας του μηνύματος εξόδου op_i και μηνύματος εισόδου op_j (Πίνακας 3.2(3)). Τρίτον, η εννοιολογική ομοιότητα OpS_{con} δυο λειτουργίων op_i $op_j \in s_i.O$ μιας διεπαφής υπηρεσίας *si* ορίζεται χρησιμοποιώντας την μετρική *Jaccard* για τα σύνολα των εννοιολογικών όρων Top_i, Top_j που περιέχονται στα ονόματα των λειτουργιών op_i $op_j \in si.O$ αντίστοιχα.

Τέλος, η έλλειψη επικοινωνιακής συνεκτικότητας, $LoC_{CM}(si)$, για μια διεπαφή *si* ορίζεται ως το συμπλήρωμα του μέσου όρου της επικοινωνιακής ομοιότητας των λειτουργιών της διεπαφής *si* (Πίνακας 3.2(6)), ενώ η έλλειψη ακολουθιακής συνεκτικότητας, $LoC_{SM}(si)$ (Πίνακας 3.2(7)), ορίζεται ως το συμπλήρωμα του μέσου όρου της σειριακής ομοιότητας των λειτουργιών της *si*. Ακόμα, η έλλειψη εννοιλογικής συνεκτικότητας, $LoC_{CON}(si)$ (Πίνακας 3.2(8)), ορίζεται ως το συμπλήρωμα του μέσου όρου της εννοιολογικής ομοιότητας των λειτουργιών της *si*.

Πίνακας 3.2 Ορισμοί Μετρικών

(1) $$MS(m_i, m_j) = \frac{|\, m_i.V \cap m_j.V \,|}{|\, m_i.V \cup m_j.V \,|}$$

(2) $$OpS_{com}(op_i, op_j) = \frac{MS(op_i.in, op_j.in)}{2} + \frac{MS(op_i.out, op_j.out)}{2}$$

(3) $$OpS_{seq}(op_i, op_j) = \frac{MS(op_i.in, op_j.out)}{2} + \frac{MS(op_i.out, op_j.in)}{2}$$

(4) $$OpS_{con}(op_i, op_j) = \frac{|\, Top_i \cap Top_j \,|}{|\, Top_i \cup Top_j \,|}$$

(5) $$Csi - \{(op_i, op_j) \in si.O \times si.O \,|\, (op_i \neq op_j) \wedge (op_i, op_j) \notin C_{si}\}$$

(6) $$LoC_{CM}(si) = 1 - \frac{\sum_{\forall (op_i, op_j) \in C_{si}} OpS_{com}(op_i, op_j)}{\frac{|\, si.O \,| * (|\, si.O \,| - 1)}{2}}$$

(7) $$LoC_{SQ}(si) = 1 - \frac{\sum_{\forall (op_i, op_j) \in C_{si}} OpS_{seq}(op_i, op_j)}{\frac{|\, si.O \,| * (|\, si.O \,| - 1)}{2}}$$

(8) $$LoC_{CN}(si) = 1 - \frac{\sum_{\forall (op_i, op_j) \in C_{si}} OpS_{con}(op_i, op_j)}{\frac{|\, si.O \,| * (|\, si.O \,| - 1)}{2}}$$

3.2. Εννοιολογική (Conceptual) Συνεκτικότητα

Η ποσοτικοποίηση της εννοιολογικής / σημασιολογικής συνεκτικότητας θα ήταν εύκολη κάτω από την προϋπόθεση ότι μια διεπαφή υπηρεσίας περιγράφεται με

έννοιες που χαρακτηρίζουν ένα συγκεκριμένο πεδίο εφαρμογών. Υπάρχουν πολλά πρότυπα σημασιολογικής περιγραφής υπηρεσιών (π.χ. OWL-S [11], SAWSDL [15], WSDL-S [22]) που προτείνονται για αυτόν το σκοπό. Δυστυχώς, πρακτικά η προϋπόθεση αυτή είναι σπάνια έγκυρη. Οι περιγραφές των υπηρεσιών που προέρχονται από μεγάλους παρόχους όπως η Amazon και η Yahoo δεν έχουν σχολιαστεί σημασιολογικά. Αυτό ισχύει για περιγραφές υπηρεσιών που υπάρχουν και σε γνωστούς καταλόγους (π.χ. ServiceFinder [16], WebServices-List [21], RemoteMethods [14]).

Παρ' όλα αυτά, ανάμεσα στις λειτουργίες διεπαφών μπορούν να βρεθούν σημασιολογικές σχέσεις . Παρατηρήθηκε ότι ονόματα λειτουργιών περιέχουν έννοιες που αναφέρονται στο πεδίο εφαρμογής τους. Αυτές οι σχέσεις κρύβονται στα ονόματα τους. Για να ποσοτικοποιηθεί η εννοιολογική συνεκτικότητα προτείνεται η παρακάτω μέθοδος.

Αρχικά, το όνομα κάθε λειτουργίας op_i μιας διεπαφής υπηρεσίας *si* αναλύεται σε ένα σύνολο από όρους Top_i. Για να δημιουργηθεί αυτό το σύνολο για κάθε λειτουργία op_i, συγκεντρώνονται όλοι οι όροι που έχουν μέγεθος πάνω από 2 χαρακτήρες και το πολύ το μέγεθος του ονόματος της λειτουργίας. Δηλαδή, όλες οι υπολέξεις μεγέθους $q=2.....|op_i.name|$.

Εναλλακτικά, τα ονόματα των λειτουργιών μπορούν να αναλυθούν με βάση κανόνες που ακολουθούνται από τους παρόχους. Οι Amazon και Yahoo χρησιμοποιούν το πρότυπο PascalCase για κωδικοποίηση, έτσι τα ονόματα των λειτουργιών είναι ακολουθίες όρων με το πρώτο γράμμα κάθε όρου να είναι κεφαλαίο. Οι όροι που αντιστοιχούν σε ρήματα αφαιρούνται από το σύνολο Top_i, γιατί αναφέρονται σε πράξεις που δεν προσδιορίζουν το πεδίο εφαρμογής της λειτουργίας. Επίσης, οι όροι αποκόπτονται στην ρίζα από την οποία απορρέουν χρησιμοποιώντας έναν τυπικό αλγόριθμο [13]. Επίσης αφαιρούνται διακόπτουσες λέξεις (π.χ. αντωνυμίες, άρθρα, σύνδεσμοι κτλ) από το σύνολο Top_i. Τέλος, δεδομένου των συνόλων Top_i και Top_j που προκύπτουν από την ανάλυση ονομάτων δύο λειτουργιών op_i, op_j, υπολογίζεται η

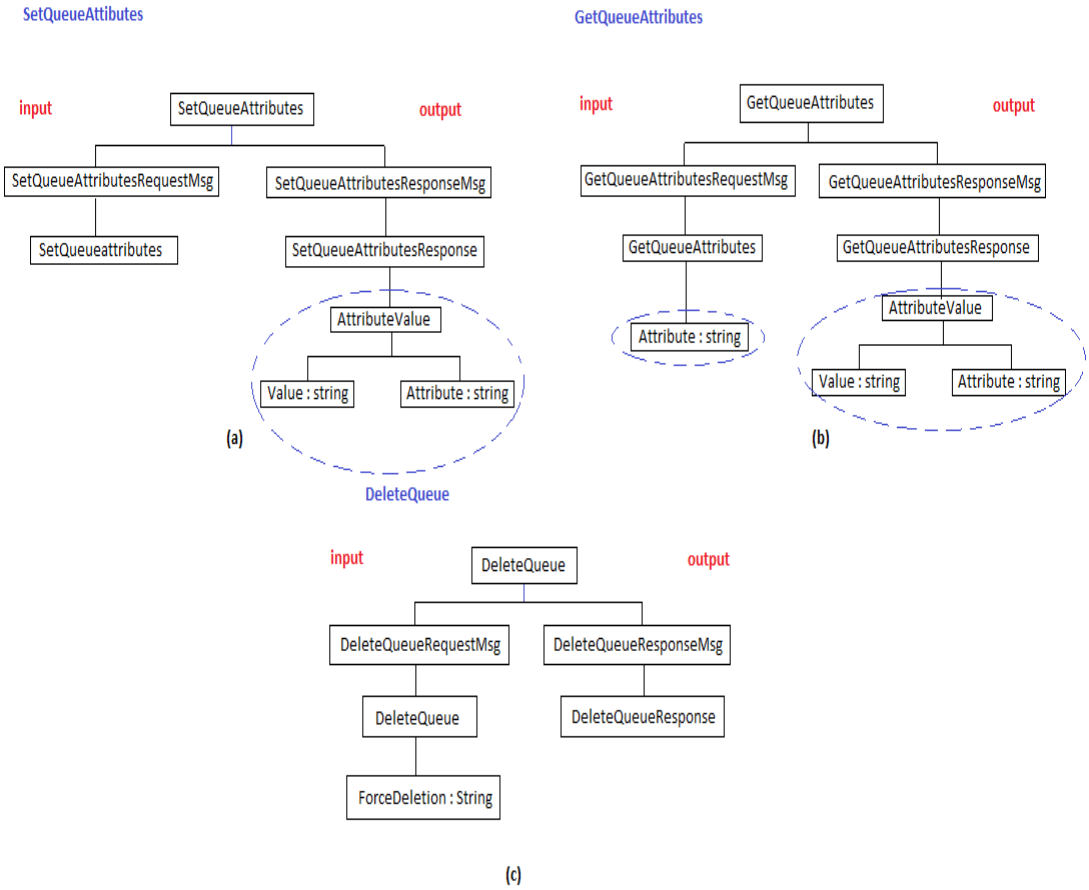
εννοιολογική ομοιότητα, με τη μετρική Jaccard (Πίνακας 3.2(4)). Η έλλειψη της εννοιολογικής συνεκτικότητας $LoC_{CN}(s_i)$ ορίζεται ως το συμπλήρωμα της μέσης εννοιολογικής ομοιότητας των ζευγαριών των λειτουργιών της διεπαφής s_i (Πίνακας 3.2(8)).

3.3. Παράδειγμα Έλλειψης Συνεκτικότητας

Παίρνοντας το παράδειγμα της υπηρεσίας Amazon SQS2007 (Σχήμα 3.2), φαίνονται τρεις λειτουργίες που περιέχονται στην διεπαφή της υπηρεσίας, που ονομάζεται MessageQueue. Στο σχήμα 3.2 φαίνεται η δομή τριών από τις λειτουργίες της διεπαφής. Οι πρώτες δυο λειτουργίες επιτρέπουν την ανάκτηση και τον ορισμό τιμών σε συγκεκριμένα χαρακτηριστικά ουράς, ενώ η τρίτη λειτουργία μπορεί να χρησιμοποιηθεί για να διαγράψει μια ήδη υπάρχουσα ουρά.

Σύμφωνα με την προσέγγιση που ακολουθείται στην παρούσα διατριβή, τα μηνύματα εισόδου των δύο πρώτων λειτουργιών φαίνεται να έχουν κοινά εσωτερικά χαρακτηριστικά (π.χ. το στοιχείο Attribute που έχει μαρκαριστεί). Η ένωση αυτών των μηνυμάτων εισόδου περιέχει 7 στοιχεία, έτσι η ομοιότητα είναι 1/7. Ενώ τα μηνύματα εξόδου είναι εντελώς άσχετα μεταξύ τους. Έτσι η επικοινωνιακή ομοιότητα (communicational similarity) (Πίνακας 3(2)) είναι (1/7)/2. Συνολικά, για την διεπαφή MessageQueue υπάρχουν 78 ζευγάρια από λειτουργίες που συνεισφέρουν στην τιμή της LoC_{CM} (Πίνακας 3(6)). Συγκεκριμένα η τιμή της είναι LoC_{CM}(MessageQueue)=0.98. Αντίστοιχα για την ακολουθιακή συνεκτικότητα, παρατηρείται ότι, ανάμεσα στο μήνυμα εισόδου της SetQueueAttributes και το μήνυμα εξόδου της GetQueueAttributes, υπάρχουν 3 κοινά στοιχεία, έτσι η ομοιότητα είναι 3/7, ενώ δεν υπάρχει κανένα κοινό στοιχείο ανάμεσα στο μήνυμα εισόδου της GetQueueAttributes και μήνυμα εξόδου της SetQueueAttributes. Άρα η συνολική σειριακή ομοιότητα των δύο είναι (3/7)/2. Ενώ η συνολική Σειριακή έλλειψη συνεκτικότητας LoC_{SQ}(MessageQueue)=0.98.

Τέλος, παρ'ότι τα μηνύματα των δυο πρώτων λειτουργιών δεν έχουν κανένα κοινό στοιχείο με τα μηνύματα της λειτουργίας DeleteQueue, και οι 3 λειτουργίες αναφέρονται σε ένα κοινό εννοιολογικό στοιχείο, αυτό της ουράς. Για αυτό και η εννοιολογική ομοιότητα ανάμεσα στις λειτουργίες GetQueueAttributes και SetQueueAttributes είναι 1 επειδή χαρακτηρίζονται από 2 κοινές έννοιες, Queue και Attribute, ενώ η λειτουργία DeleteQueue με κάθε μια από τις άλλες 2 λειτουργίες, έχει εννοιολογική ομοιότητα $1/2$ Συνολικά, η έλλειψη εννοιολογικής συνεκτικότητας LoC_{CN}(MessageQueue)=0.81. Συγκεντρώνοντας όλες τις περιπτώσεις, οι τιμές των μετρικών για την διεπαφή MessageQueue είναι κοντά στο 1, κάτι που δείχνει την ανάγκη για ανακατασκευή της διεπαφής MessageQueue.



Σχήμα 3.2 Παράδειγμα Λειτουργιών (Amazon SQS2007)

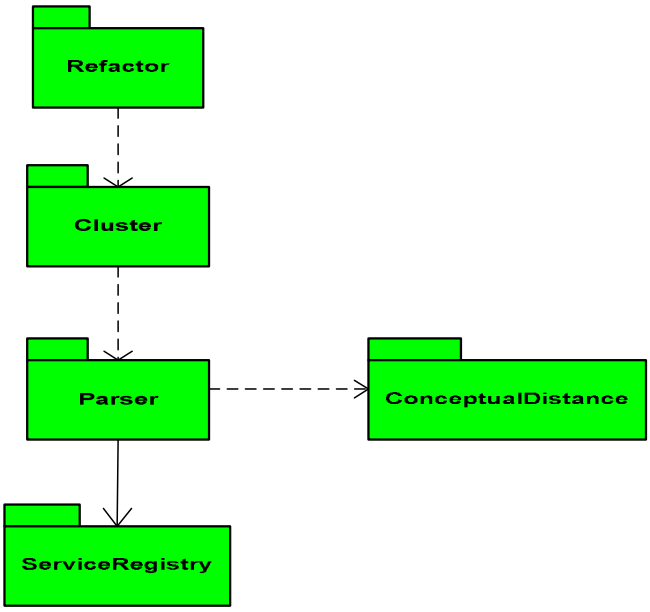
ΚΕΦΑΛΑΙΟ 4. ΕΡΓΑΛΕΙΟ ΑΝΑΚΑΣΚΕΥΗΣ ΔΙΕΠΑΦΩΝ

Στο κεφάλαιο αυτό παρουσιάζεται η αρχιτεκτονική του συστήματος ανακατασκευής (υπό-ενότητα 4.1) και οι μέθοδοι ομαδοποίησης που χρησιμοποιούνται στην διατριβή. Περιγράφονται 2 διαφορετικοί αλγόριθμοι (υπό-ενότητα 4.2), ένας ιεραρχικός ενωτικός και ένας ιεραρχικός διαιρετικός. Ακόμα αναλύονται 4 εκδοχές του ιεραρχικού ενωτικού αλγορίθμου

4.1. Αρχιτεκτονική

Το σύστημα που περιγράφεται στην διατριβή αποτελείται από 5 βασικά υποσυστήματα, εκ των οποίων τα 3 είναι τα σημαντικότερα και επιτελούν το μεγαλύτερο μέρος της διαδικασίας. Αυτά είναι τα υποσυστήματα Cluster, Parser και ConceptualDistance. Οι υπηρεσίες προς ανακατασκευή βρίσκονται σε έναν κατάλογο (ServiceRegistry). Ο κατάλληλος αλγόριθμος επιλέγεται από το υποσύστημα Refactor δημιουργώντας αντίστοιχα αντικείμενα κλάσεων που βρίσκονται στο υποσύστημα Cluster. Τέλος δημιουργούνται τα δέντρα κόμβων που αποτελούν τα μηνύματα των λειτουργιών των εκάστοτε διεπαφών υπηρεσιών. Τα δέντρα των μηνυμάτων εισόδου-εξόδου των λειτουργιών δημιουργούνται από το υποσύστημα Parser. Όλα τα

παραπάνω περιγράφονται συνοπτικά στο σχήμα 4.1 και αναλύονται περισσότερο παρακάτω.



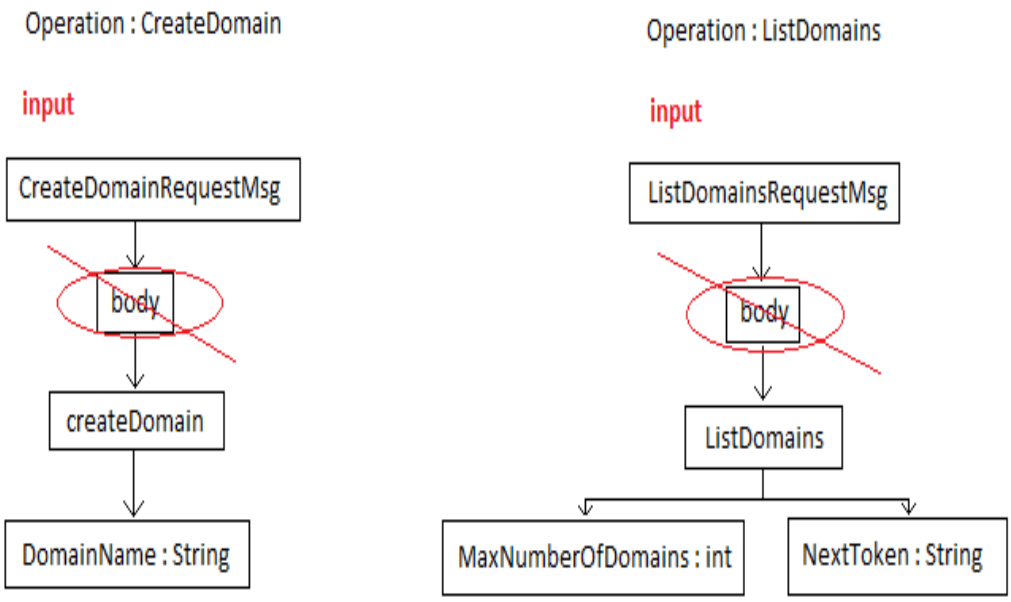
Σχήμα 4.1 Το Περιγραφόμενο από τη Διατριβή Σύστημα

Το σύστημα Refactor προσφέρει μια απλή διεπαφή, δίνοντας στον χρήστη το δικαίωμα να επιλέξει μια από τις 5 εκδοχές των αλγορίθμων που εξετάζονται στην εργασία αυτή, 4 εκδοχές ενός ιεραρχικού ενωτικού αλγορίθμου και έναν ιεραρχικό διαιρετικό. Οι 4 εκδοχές του ιεραρχικού ενωτικού αλγορίθμου, βασίζονται σε 4 διαφορετικές μεθόδους υπολογισμού των αποστάσεων μεταξύ των λειτουργιών στις διεπαφές των υπηρεσιών. Αυτές οι 4 μέθοδοι είναι:

1. Του κοντινότερου γείτονα (*Single Linkage*)
2. Του μακρινότερου γείτονα (*Complete Linkage*)
3. Του μέσου όρου των αποστάσεων (*Avg Linkage*)
4. Της πιθανής Έλλειψης Συνεκτικότητας (*LoC*$_*$)

Οι 3 πρώτες παρουσιάστηκαν στο κεφάλαιο 2, ενώ η τέταρτη αναλύεται παρακάτω στο κεφάλαιο. Το σύστημα Parser είναι αυτό που συγκεντρώνει τα χαρακτηριστικά για τον υπολογισμό των μετρικών δημιουργώντας τα δέντρα των μηνυμάτων εισόδου-εξόδου. Διαχειρίζεται το XMLSchema που συνοδεύει το WSDL έγγραφο και τα βασικά συστατικά ενός WSDL εγγράφου που περιγράφουν τις υπηρεσίες, όπως για παράδειγμα τις διεπαφές (PortType) τα μηνύματα εισόδου-εξόδου (Messages), τα μέρη των μηνυμάτων (Parts).

Αφού συγκεντρωθούν τα δέντρα όλων των μηνυμάτων των λειτουργιών μιας διεπαφής, απομακρύνονται οι κόμβοι που είναι κοινοί και υπάρχουν σε όλα τα δέντρα, όπως φαίνεται και στο σχήμα 4.2. Ο κόμβος body υπάρχει σε όλα τα δέντρα μηνυμάτων εισόδου-εξόδου των λειτουργιών τις διεπαφής AmazonSDBPortType, οπότε απομακρύνεται. Αυτή η ενέργεια, η εκκαθάριση δηλαδή των κόμβων που βρίσκονται σε όλα τα δέντρα μηνυμάτων, βοηθά στο να μη λαμβάνονται υπόψη συσχετίσεις που δεν παίζουν σημαντικό ρόλο.



Σχήμα 4.2 Παράδειγμα Δέντρων των Μηνυμάτων Εισόδου από 2 Λειτουργίες

Αφού συγκεντρωθούν όλα τα δέντρα μηνυμάτων μιας διεπαφής , υπολογίζονται η επικοινωνιακή η ακολουθιακή και η εννοιολογική συνεκτικότητα. Αφού γίνει αυτό κρίνεται αν η διεπαφή χρήζει ανακατασκευής. Τα αποτελέσματα από τις συγκρίσεις των αλγορίθμων και των μετρικών αναλύονται και συγκρίνονται στο επόμενο κεφάλαιο.

4.2. Μέθοδοι Ομαδοποίησης

4.2.1. Ενωτικές Μέθοδοι Ομαδοποίησης

Οι ιεραρχικοί ενωτικοί μέθοδοι ομαδοποίησης ξεκινούν θεωρώντας κάθε λειτουργία μια ξεχωριστή οντότητα δηλαδή μια αυτόνομη ξεχωριστή διεπαφή και σε κάθε βήμα ενώνονται οι δύο πιο όμοιες διεπαφές. Ο προτεινόμενος ιεραρχικός ενωτικός αλγόριθμος με τις 4 διαφορετικές μεθόδους υπολογισμού των αποστάσεων μεταξύ των λειτουργιών δέχεται ως είσοδο μια λίστα *Lo* από λειτουργίες, που προηγουμένως άνηκαν σε μια διεπαφή *si* προς ανακατασκευή και εξάγει μια λίστα από διεπαφές *Lp*, με λειτουργίες ομαδοποιημένες ως προς το κριτήριο που επιλέγεται (επικοινωνιακή, ακολουθιακή, εννοιολογική συνεκτικότητα). Στη συνέχεια οι όροι OpS* και LoC* αναφέρονται στις αποστάσεις λειτουργιών και στην έλλειψης συνεκτικότητας όπως αυτά υπολογίζονται με βάση το εκάστοτε κριτήριο (Πίνακας 3.2)

Ο αλγόριθμος διατηρεί την λειτουργικότητα της αρχικής υπηρεσίας και οι ανακατασκευασμένες διεπαφές, έχουν μικρότερη έλλειψη συνεκτικότητας από την αρχική διεπαφή και λιγότερες λειτουργίες. Ο αλγόριθμος αρχικά διατηρεί μια λίστα από διεπαφές όσες και οι λειτουργίες της αρχικής διεπαφής (γραμμές 1-4, σχήμα 4.3). Στη συνέχεια ο αλγόριθμος επαναληπτικά για κάθε ζευγάρι διεπαφών prt_i, $prti_j$ (γραμμές 8-11, σχήμα 4.3) βρίσκει όλες τις αποστάσεις μεταξύ των λειτουργιών που ανήκουν σε αυτές τις διεπαφές και τις αποθηκεύει στη λίστα *Ld* (γραμμές 12-15).

Ανάλογα με την μέθοδο υπολογισμού αποστάσεων που επιλέχθηκε, υπολογίζεται η απόσταση μεταξύ prt_i, $prti_j$. Για παράδειγμα, αν επιλέχθηκε η μέθοδος του κοντινότερου γείτονα (*Single Linkage*), τότε η συνάρτηση *Selected_Method* (γραμμή 16, σχήμα 4.3) επιλέγει την μικρότερη απόσταση από τη *Ld* ή ανάλογα αν επιλεγεί η μέθοδος του μέσου όρου, τότε υπολογίζεται ο μέσος όρος των αποστάσεων που υπάρχουν στη λίστα *Ld*.

Πιο συγκεκριμένα η συνάρτηση Selected_Method καθορίζει την απόσταση των prt_i, prt_j. Αυτό γίνεται με 4 διαφορετικές εκδοχές που διαφοροποιούν τις προτεινόμενες ομαδοποιήσεις:

- Η πρώτη (*Average Linkage*) υπολογίζει το μέσο όρο των αποστάσεων της λίστας *Ld*: $dist(prt_i, prt_j) = \frac{\sum_{\forall d_i \in Ld} d_i}{|Ld|}$
- Η δεύτερη (*Single Linkage*) υπολογίζει την ελάχιστη των αποστάσεων της λίστας *Ld* : $dist(prt_i, prt_j) = MIN(d_i \mid \forall d_i \in Ld)$
- Η τρίτη (*Complete Linkage*) υπολογίζει τη μέγιστη των αποστάσεων της λίστας *Ld*: $dist(prt_i, prt_j) = MAX(d_i \mid \forall d_i \in Ld)$
- Τέλος η τέταρτη εκδοχή θεωρεί ως κριτήριο συνένωσης την έλλειψη συνεκτικότητας της διεπαφής που θα προκύψει αν συνενωθούν οι prt_i, prt_j : $dist(prt_i, prt_j) = LoC_*(prt_i, prt_j)$

Εν συνεχεία ελέγχεται για το τρέχον ζεύγος διεπαφών prt_i, $prti_j$ αν η απόσταση είναι μικρότερη από τις αποστάσεις που υπολογίστηκαν για προηγούμενα ζεύγη (γραμμές 15-20, σχήμα 4.3). Τελικά, συγχωνεύεται το ζεύγος διεπαφών που χαρακτηρίζεται από τη μικρότερη απόσταση και πληροί 2 επιπλέον περιορισμούς. Οι διεπαφές που συγχωνεύτηκαν απομακρύνονται από την λίστα ,ενώ η νέα εισάγεται στη λίστα *Lp* (γραμμές 25-37, σχήμα 4.3).

Algorithm Agglomerative

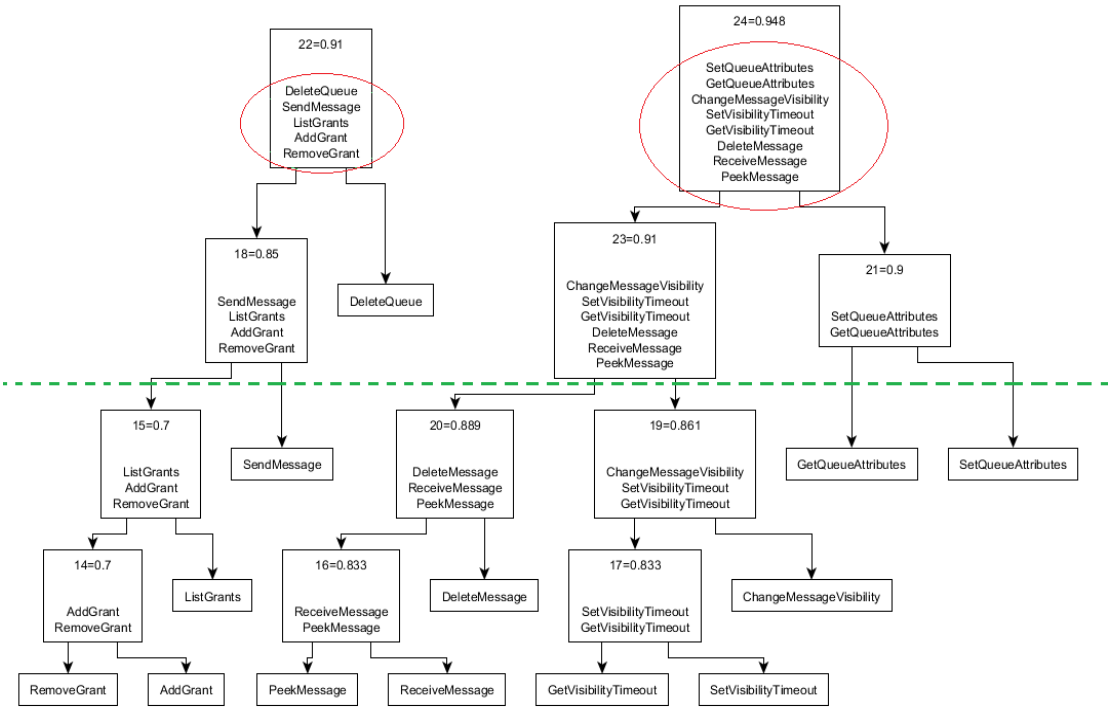
```
Input : Lo : List < Operation >
Output : Lp : List < Interfaces >
//phase 1: Creation of singleton interfaces
 1. for all op_i ∈ Lo do
 2.     tempPrt = Create_PortType.add(op_i)
 3.     Lp.add (tempPrt)
 4. end for

//phase 2: Creation of clusters
 5. Distance ← 1, found ← true ,
 6. while | Lp.size |>1 & & found = true do
 7.    found = false
 8.    for all prti ∈ Lp do
 9.       for all prtj ∈ Lp do
10.          Ld ← ∅  // Ld list for distance values
11.          if prti ≠ prtj then
12.             for all (op_i, opj) ∈ prt_i.getOperations() × prt_j.getOperations()
13.                d ← OpS-(op_i, op_j)
14.                Ld ← Ld ∪ {d}
15.             end for
16.             dist ← Selected _Method(Ld)
17.             if dist < Distance && Restrictions () = true then
16.                found = true
18.                Distance=dist
19.                I = Lp.indexOf (prt_i)
20.                J = Lp.indexOf (prt_j)
21.             end if
22.          end if
23.       end for
24.    end for
25.    if found = true then
26.       tempPrt = Create_PortType
27.       for all opi ∈ Lp.get(I).getOperations
28.          tempPrt.add (opi)
29.       end for
30.       for all opj ∈ Lp.get(J).getOperations
31.          tempPrt.add (opj)
32.       end for
33.       Lp.add (tempPrt)
34.       Lp.remove (I)
35.       Lp.remove (J)
36.       Distance=1
37.    end if
38. end while
39. return Lp
```

Σχήμα 4.3 Ιεραρχικός Ενωτικός Αλγόριθμος

Στην συνάρτηση Restrictions της γραμμής 15 (σχήμα 4.3) ελέγχονται οι 2 επιπλέον περιορισμοί που πρέπει να ισχύουν προκειμένου να γίνει η συγχώνευση. Οι περιορισμοί είναι :

1. Κάθε νέα διεπαφή που δημιουργείται να έχει μικρότερη έλλειψη συνεκτικότητας από την αρχική διεπαφή *si*.
2. Κάθε νέα διεπαφή που δημιουργείται πρέπει να έχει μικρότερη έλλειψη συνεκτικότητας τουλάχιστον από μια από τις δύο προηγούμενες

Συνήθως οι ιεραρχικοί αλγόριθμοι συνεχίζονται μέχρι το υψηλότερο επίπεδο ομαδοποίησης, δηλαδή μέχρι οι οντότητες να ομαδοποιηθούν σε μια ομάδα. Με αυτόν τον τρόπο η κατάληξη θα ήταν οι λειτουργίες να δημιουργήσουν ξανά την αρχική διεπαφή. Για να αποφευχθεί αυτό υπήρχε η ανάγκη των προηγούμενων περιορισμών, που θα αναγκάζουν την διαδικασία να σταματά νωρίτερα. Ο πρώτος περιορισμός έχει ως συνέπεια η διαδικασία να προχωρά όσο οι ομάδες που θα δημιουργούνται να έχουν μικρότερη έλλειψη συνεκτικότητας από την αρχική διεπαφή. Από μόνος του αυτός ο περιορισμός είναι αρκετός, γιατί ο αλγόριθμος απλά θα σταματούσε ένα βήμα πριν την τελική ομαδοποίηση (σχήμα 4.4).

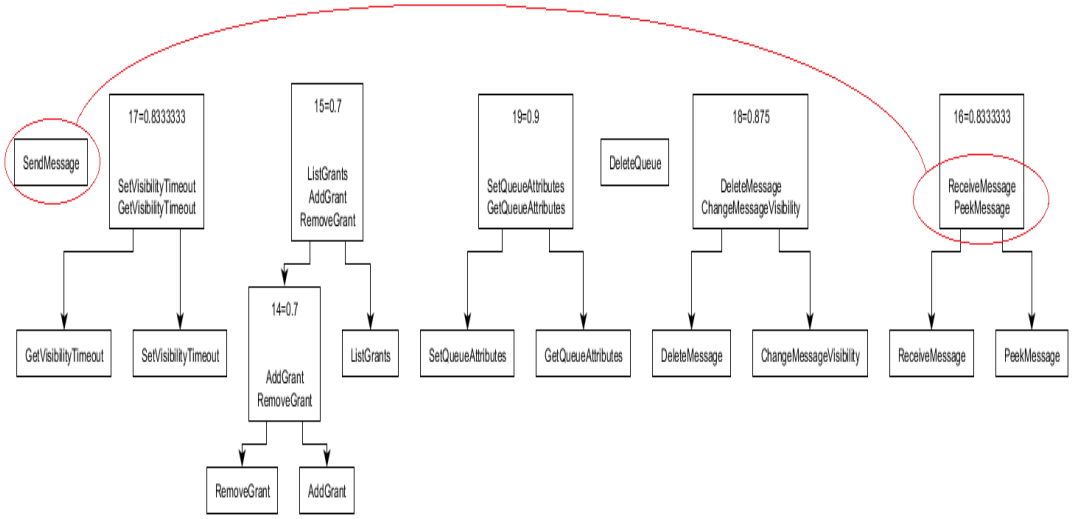


Σχήμα 4.4 Ομαδοποίηση της Διεπαφής Amazon MessageQueueuPortType

Στο παράδειγμα του σχήματος 4.4 φαίνεται ότι δεν έχει προκύψει καλή ομαδοποίηση αφού στις δυο τελικές διεπαφές (μαρκαρισμένες με γραμμή, σχήμα 4.4) έχουν συγχωνευτεί μεταξύ τους άσχετες λειτουργίες, όπως λειτουργίες διαχείρισης μηνυμάτων με λειτουργίες διαχείρισης ουράς. Φαίνεται στα παρακάτω επίπεδα

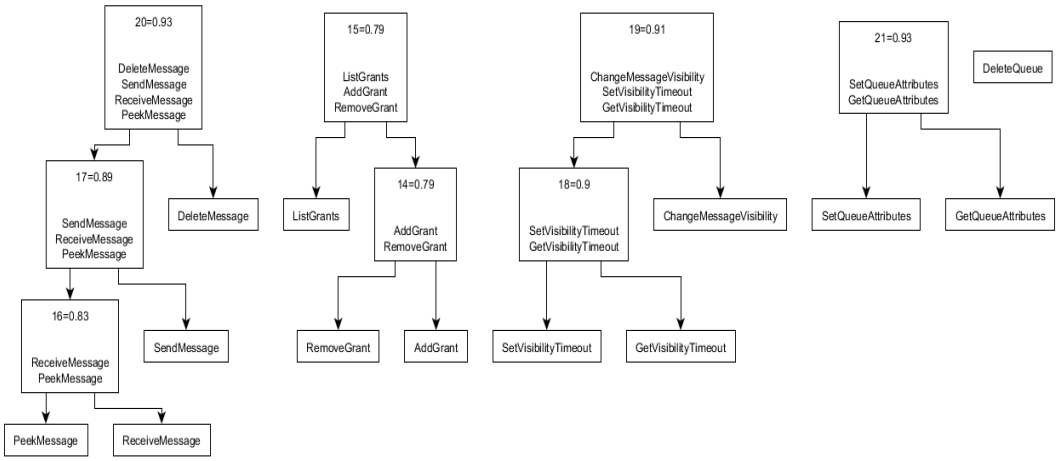
μαρκαρισμένα με πράσινη διακεκομμένη γραμμή ότι υπάρχουν πιο ιδανικές ομαδοποιήσεις. Οπότε μόνο ο πρώτος περιορισμός, δεν ήταν αρκετός.

Στην βιβλιογραφία έχουν προταθεί αυστηρότεροι περιορισμοί [7, 19], όπως για παράδειγμα η νέα οντότητα να έχει χαμηλότερη έλλειψη συνεκτικότητας και από τις δύο προηγούμενες. Αυτό οδηγεί μεν σε πολύ συνεκτικές διεπαφές, οι οποίες όμως έχουν λίγες λειτουργίες, είναι πολλές στον αριθμό και δεν εξυπηρετούν τον σκοπό των τεχνικών ομαδοποίησης, αφού σχετικές διεπαφές παραμένουν ξένες μεταξύ τους (σχήμα 4.5)



Σχήμα 4.5 Ομαδοποίηση της Διεπαφής Amazon MessageQueuePortType με Αυστηρό Περιορισμό

Με βάση τα παραπάνω ο δεύτερος περιορισμός που επιλέχθηκε είναι πιο χαλαρός, επιτρέπει στη νέα διεπαφή να έχει μικρότερη έλλειψη συνεκτικότητας από μία από τις δύο προηγούμενες. Τα αποτελέσματα έδειξαν ότι αυτοί οι περιορισμοί αυτοματοποιούν την διαδικασία με τα καλύτερα δυνατά αποτελέσματα και προτείνουν καλύτερες ομαδοποιήσεις (σχήμα 4.6).



Σχήμα 4.6 Ομαδοποίηση της Διεπαφής Amazon MessageQueuePortType με τον Προτεινόμενο Περιορισμό

4.2.2. Διαιρετικές Μέθοδοι Ομαδοποίησης

Ο διαιρετικός αλγόριθμος σχήμα 4.8 δέχεται ως είσοδο μια διεπαφή *si* και η έξοδος του αλγορίθμου είναι ένα σύνολο από διεπαφές R_I = *{r: Interface}*. Αρχικά διατηρείται μια ουρά που περιέχει την αρχική διεπαφή, *Q={si}*. Στη συνέχεια, ο αλγόριθμος επαναληπτικά, εξάγει μια διεπαφή από την ουρά. Στόχος κάθε επανάληψης είναι να μειωθεί η έλλειψη συνεκτικότητας της διεπαφής *r* απομακρύνοντας ένα σύνολο λειτουργιών που είναι πιο όμοιες μεταξύ τους, από ότι με τις λειτουργίες που παραμένουν στην διεπαφή. Η λειτουργίες που απομακρύνονται δημιουργούν μια νέα διεπαφή r_s. Η διεπαφές αυτές εισάγονται στην ουρά. Η επαναληπτική διαδικασία τερματίζεται, όταν στην ουρά δεν θα υπάρχουν διεπαφές που μπορεί να βελτιωθεί η έλλειψη συνεκτικότητας.

Συγκεκριμένα, κάθε επανάληψη *I*, αποτελείται από 2 φάσεις. Η πρώτη φάση δημιουργεί την αποσχισθείσα διεπαφή (splinter interface), που στο εξής θα αναφέρεται ως *splinter*, που εξάγεται από την αρχική και ούτω καθ'εξής, ενώ η δεύτερη φάση γεμίζει την διεπαφή *splinter* με λειτουργίες:

1. Δημιουργία της διεπαφής *splinter*: Η δημιουργία της διεπαφής r_s, αποτελείται από 3 βήματα (γραμμές 5-19):

a) Κατά την διάρκεια του πρώτου βήματος, ο αλγόριθμος κατασκευάζει ένα σύνολο από λειτουργίες R_{O_I} που είναι υποψήφιες προς απομάκρυνση από το σύνολο των λειτουργιών $r.O$. Το κριτήριο για την απομάκρυνση μιας λειτουργίας είναι να μειωθεί η έλλειψη συνεκτικότητας της διεπαφής r (γραμμές 6-13). Αν $R_{O_I} = \varnothing$, τότε η r δεν μπορεί να βελτιωθεί περαιτέρω, έτσι εισάγεται στο σύνολο των αποτελεσμάτων R_I και η επανάληψη I σταματά (γραμμές 14-16). Διαφορετικά, ακολουθούν τα επόμενα 2 βήματα.

b) Ο αλγόριθμος επιλέγει από το σύνολο R_{O_I} μια λειτουργία op_s, τέτοια ώστε η απομάκρυνση από την διεπαφή r να μειώνει στο μέγιστο την έλλειψη συνεκτικότητας αυτής (γραμμή 17).

c) Τελικά, η op_s απομακρύνεται από την r και δημιουργείται η διεπαφή *splinter* $r_s.O = \{op_s\}$ μαζί με την διεπαφή r_c που προκύπτει από την απομάκρυνση της op_s από την r (γραμμές 18-19).

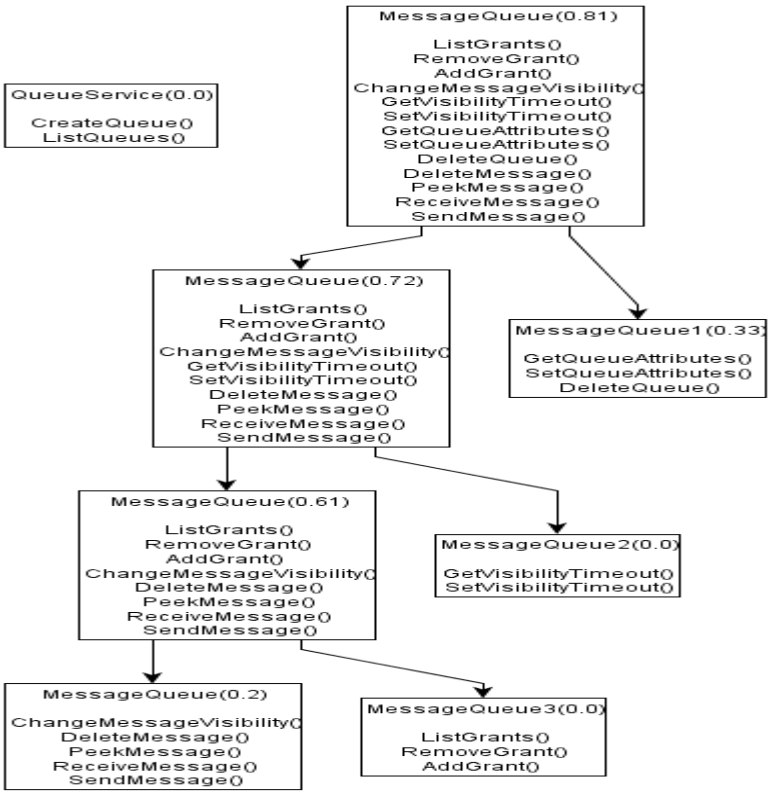
2. Πρόσθεση λειτουργιών στην διεπαφή *splinter*: σκοπός αυτής της φάσης είναι η περαιτέρω βελτίωση της συνεκτικότητας των r_c και r_s μετακινώντας λειτουργίες από την r_c στην r_s. Η επαναληπτική διαδικασία σταματά όταν δεν μπορεί να συνεχιστεί η βελτίωση. Κάθε επανάληψη της δεύτερης φάσης ακολουθεί τα παρακάτω βήματα (γραμμές 20-36) :

a) Όμοια με το πρώτο βήμα της προηγούμενης φάσης, ο αλγόριθμος κατασκευάζει ένα σύνολο από λειτουργίες $R_{O_{II}}$, που είναι υποψήφιες προς μετακίνηση από την r_c στην r_s. Όμως, η μεταφορά μιας λειτουργίας $op_i \in R_{O_{II}}$ πρέπει να μειώνει την έλλειψη συνεκτικότητας και της r_c και της r_s. Επιπλέον, η έλλειψη συνεκτικότητας της r_s πρέπει να είναι μικρότερη από την έλλειψη συνεκτικότητας της αρχικής διεπαφής r που διαιρέθηκε στην πρώτη φάση (γραμμές 22-30).

b) Στη συνέχεια, ο αλγόριθμος επιλέγει από το σύνολο $R_{O_{II}}$ μια λειτουργία op_s προς μετακίνηση από την r_c στην r_s. Η επιλογή γίνεται έτσι ώστε να επιτυγχάνεται η μεγαλύτερη μείωση της έλλειψης συνεκτικότητας (γραμμή 32).

c) Τελικά η επιλεγμένη λειτουργία αφαιρείται από την r_c και προστίθενται στην r_s (γραμμές 33-34).

Μετά το τέλος του αλγόριθμου, το σύνολο R_I περιέχει έναν αριθμό διεπαφών, που η έλλειψη συνεκτικότητας τους δεν μπορεί να μειωθεί περισσότερο. Αυτές οι διεπαφές είναι και το αποτέλεσμα του αλγορίθμου. Στο παρακάτω σχήμα φαίνεται μια ένα αποτέλεσμα του αλγορίθμου.



Σχήμα 4.7 Ομαδοποίηση της Διεπαφής Amazon MessageQueuePortType από τον Διαιρετικό Αλγόριθμο

Algorithm 1: $A(LoC_*)$

```
Input: si : Interface
Output: R_I = {r : Interface}

 1. R_I ← ∅
 2. Q.Enqueue(si)
 3. repeat
 4.     r ← Q.Dequeue()

        //Phase 1: Creation of splinter interface r_s.
 5.     R_OI ← ∅
 6.     for all op_i ∈ r.O do
 7.         if Is_Marked(op_i, r) = false then
 8.             r_i.O ← r.O − {op_i} //op_i isn't marked (line 48).
 9.             if LoC_*(r_i) < LoC_*(r) then
10.                 R_OI ← R_OI ∪ {op_i}
11.             end if
12.         end if
13.     end for
14.     if R_OI = ∅ then
15.         R_I ← R_I ∪ {r} //LoC(r) can not be further improved.
16.     else
17.         op_s ← Find_MAX_ΔLoC_*(r, R_OI)
18.         r_s.O ← {op_s}
19.         r_c.O ← r.O − {op_s}

            //Phase 2: Population of splinter interface r_s.
20.         repeat
21.             R_OII ← ∅
22.             for all op_i ∈ r_c.O do
23.                 r_ci.O ← r_c.O − {op_i}
24.                 r_si.O ← r_s.O ∪ {op_i}
25.                 if LoC_*(r_ci) < LoC_*(r_c) then
26.                     if LoC_*(r_si) < LoC_*(r_s) ∧ LoC_*(r_si) < LoC_*(r) then
27.                         R_OII ← R_OII ∪ {op_i}
28.                     end if
29.                 end if
30.             end for
31.             if R_OII ≠ ∅ then
32.                 op_s ← Find_MAX_Overall_ΔLoC_*(r_c, r_s, R_OII)
33.                 r_s.O ← r_s.O ∪ {op_s}
34.                 r_c.O ← r_c.O − {op_s}
35.             end if
36.         until R_OII = ∅

37.         if |r_s.O| = 1 then
38.             irrelevant ← true
39.             for all op_j ∈ r_c.O do
40.                 if OpS_*(r_s.O.op, op_j) > 0 then
41.                     irrelevant ← false
42.                 end if
43.             end for
44.             if irrelevant = true then
45.                 R_I ← R_I ∪ {r_s} //r_s is a singleton interface.
46.                 Q.Enqueue(r_c)
47.             else
48.                 Mark_Operation_Incapable_for_Splinter(r_s.O.op, r)
49.                 Q.Enqueue(r)
50.             end if
51.         else
52.             Q.Enqueue(r_s)
53.             Q.Enqueue(r_c)
54.         end if
55.     end if
56. until Q.IsEmpty
```

Σχήμα 4.8 Διαιρετικός Αλγόριθμος Ομαδοποίησης

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΑ

Για να εκτιμηθεί η προσέγγιση της διατριβής εκτελέστηκε ένας αριθμός πειραμάτων που επικεντρώνονται στην ανακατασκευή πραγματικών διεπαφών υπηρεσιών που προσφέρονται προς κατανάλωση από τους 2 μεγαλύτερους παρόχους υπηρεσιών, την Amazon και την Yahoo (υπό-ενότητα 5.1). Για την ανακατασκευή των διεπαφών αυτών των υπηρεσιών χρησιμοποιήθηκαν οι 4 εκδοχές του ιεραρχικού ενωτικού αλγορίθμου και ο ιεραρχικός διαιρετικός αλγόριθμος. Κάθε εκδοχή εξετάστηκε για τις 3 μετρικές που εκτιμούν την επικοινωνιακή, ακολουθιακή και εννοιολογική έλλειψη συνεκτικότητας.

Συγκεκριμένα, έγινε ποσοτική ανάλυση, όπου μελετήθηκαν και συγκρίθηκαν τα αποτελέσματα των διαφορετικών εκδοχών που εξετάστηκαν (υπό–ενότητα 5.2). Ακόμα, διεξήχθη ποιοτική αξιολόγηση (υπό–ενότητα 5.3), όπου εκτιμήθηκε κατά πόσο είναι όντως χρήσιμα τα αποτελέσματα για έναν κατασκευαστή υπηρεσιών. Η εκτίμηση βασίστηκε σε ειδικούς, που είχαν γνώση του τομέα υπηρεσιών διαδικτύου.

5.1. Περιγραφή των Συνόλων Δεδομένων

Η Amazon παρέχει 21 υπηρεσίες που είναι διαθέσιμες μέσω του διαδικτύου [1] . Από αυτές 16 χρησιμοποιήθηκαν για τον σκοπό των πειραμάτων. Οι 5 υπηρεσίες που εξαιρέθηκαν δεν κάλυπταν τις προδιαγραφές του εργαλείου ανακατασκευής. Το

εργαλείο δέχεται ως είσοδο υπηρεσίες που περιγράφονται με έγγραφα WSDL, κάτι που δεν είχαν οι υπηρεσίες που εξαιρέθηκαν. Οι 16 υπηρεσίες της Amazon, παρέχουν 19 διεπαφές, 14 υπηρεσίες παρέχουν μια μόνο διεπαφή, ενώ άλλες 2 υπηρεσίες (Fulfillment Web Service (FWS) και Simple Queue Service (SQS)) παρέχουν 3 και 2 διεπαφές αντίστοιχα. Από τις 19 διεπαφές στην ποσοτική αξιολόγηση δεν συμμετείχαν αυτές που παρείχαν λιγότερες από 7 λειτουργίες. Το όριο των 7 λειτουργιών υιοθετήθηκε με βάση τον κανόνα του *Miller[13]*. Επομένως 13 διεπαφές αξιολογήθηκαν ποσοτικά, ενώ ποιοτικά και οι 19 διεπαφές.

Η Yahoo παρέχει 21 υπηρεσίες όπου κάθε μια έχει μια διεπαφή [24], ενώ και σε αυτό το σύνολο δεν λαμβάνονται υπόψη για την ποσοτική αξιολόγηση διεπαφές που περιέχουν λιγότερες από 7 λειτουργίες. Από δω και στο εξής για λόγους απλότητας οι υπηρεσίες θα αναφέρονται με αναγνωριστικά. Για τις υπηρεσίες της Amazon θα χρησιμοποιηθούν τα αναγνωριστικά A1-A19, ενώ για τις υπηρεσίες της Yahoo τα αναγνωριστικά Y1-Y21 (Πίνακας 5.1).

Ο λόγος που δεν συμπεριλαμβάνονται οι διεπαφές που έχουν λιγότερες από 7 λειτουργίες στην ποσοτική αξιολόγηση είναι ότι το μέγεθος τους θεωρήθηκε μικρό και κατά συνέπεια δεν προσφέρονται για την εξαγωγή χρήσιμων συμπερασμάτων.

Πίνακας 5.1 Τα Σύνολα Υπηρεσιών της Amazon και Yahoo.

names AMAZON	ID	Size
CloudWatchPortType	A1	2
ElasticMapReducePortType	A2	4
AmazonFBAOutboundPortType	A3	7
AmazonSNSPortType	A4	13
MechanicalTurkRequesterPortType	A5	27
ElasticLoadBalancingPortType	A6	13
AmazonFPSPortType	A7	27
AmazonImportExportPortPortType	A8	5
QueueService	A9	2
AmazonFBAInventoryPortType	A10	4

AmazonLSPortType	A11	6
AmazonSDBPortType	A12	9
MessageQueue	A13	13
AutoScalingPortType	A14	13
AmazonFWSInboundPortType	A15	18
AmazonVPCPortType	A16	21
AmazonRDSv2PortType	A17	23
AmazonEC2PortType	A18	87
AmazonS3	A19	16
names YAHOO	ID	Size
AccountService	Y1	20
AdGroupService	Y2	28
AdService	Y3	20
BasicReportService	Y4	12
BidInformationService	Y5	4
BudgetingService	Y6	7
BulkService	Y7	7
CampaignService	Y8	19
ConverterService	Y9	4
ExcludedWordsService	Y10	10
ForecastService	Y11	5
GeographicalDictionaryService	Y12	10
KeywordResearchService	Y13	6
KeywordService	Y14	34
MasterAccountService	Y15	7
MobileDictionaryService	Y16	3
TargetingConverterService	Y17	12
TargetingDictionaryService	Y18	4
TargetingService	Y19	23
VaultService	Y20	5
UserManagementService	Y21	28

5.2. Ποσοτική Αξιολόγηση

Μια πρώτη ένδειξη για την ανάγκη ανακατασκευής των αρχικών διεπαφών είναι οι αρχικές τιμές για τις μετρικές τις επικοινωνιακής, ακολουθιακής και εννοιολογικής

συνεκτικότητας. Οι τιμές αυτές ήταν υψηλές όπως φαίνεται και στον πίνακα 5.3, κάτι που δείχνει ότι χρήζουν ανακατασκευής.

Πίνακας 5.2 Αρχικές Τιμές των Μετρικών Έλλειψης Συνεκτικότητας.

ID	Size	Communicational	Sequential	Conceptual
A3	7	0.94	0.94	0.51
A4	13	0.96	0.97	0.84
A5	27	0.92	0.84	0.83
A6	13	0.93	0.97	0.72
A7	27	0.92	0.97	0.96
A12	9	0.94	0.97	0.79
A13	13	0.98	0.98	0.81
A14	13	0.96	0.98	0.79
A15	18	0.93	0.96	0.73
A16	21	0.95	0.98	0.82
A17	23	0.91	0.96	0.56
A18	87	0.98	0.99	0.94
A19	16	0.89	0.97	0.75
Y1	20	0.92	0.98	0.88
Y2	28	0.84	0.94	0.65
Y3	20	0.79	0.89	0.88
Y4	12	0.91	0.99	0.92
Y6	7	0.93	0.94	0.14
Y7	7	0.94	1.00	0.93
Y8	19	0.83	0.91	0.91
Y10	10	0.72	0.81	0.54
Y12	10	0.79	0.99	0.65
Y14	34	0.84	0.93	0.91
Y15	7	0.83	0.88	0.50
Y19	23	0.74	0.96	0.74
Y21	28	0.96	0.97	0.91

Για να συγκριθούν οι διαφορετικές εκδοχές που εξετάστηκαν μετρήθηκαν:

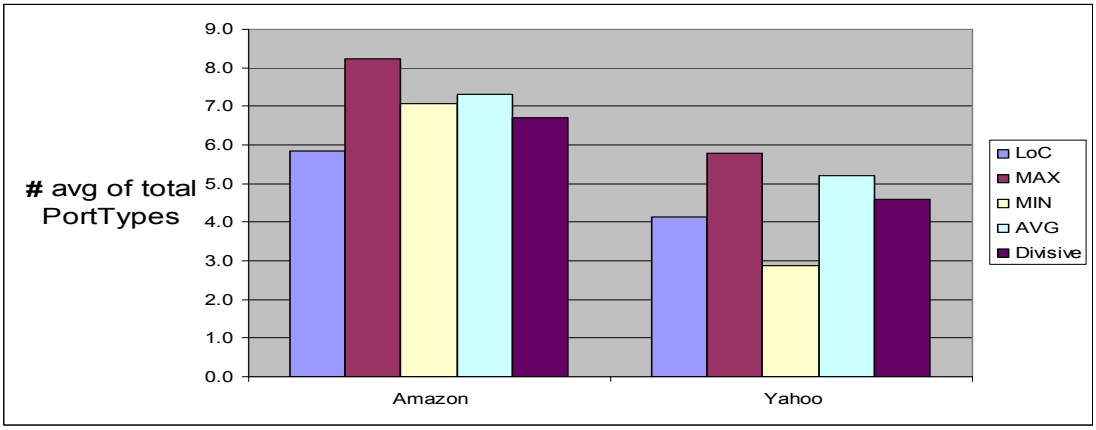
1. Ο συνολικός αριθμός των διεπαφών που παράγονται από την ανακατασκευή.

2. Ο μέσος όρος του μεγέθους των διεπαφών που προέκυψαν (αριθμός λειτουργιών).
3. Το ποσοστό μείωσης της έλλειψης συνεκτικότητας που προέκυψε από τις 3 μετρικές και για τις 5 εκδοχές των αλγορίθμων.

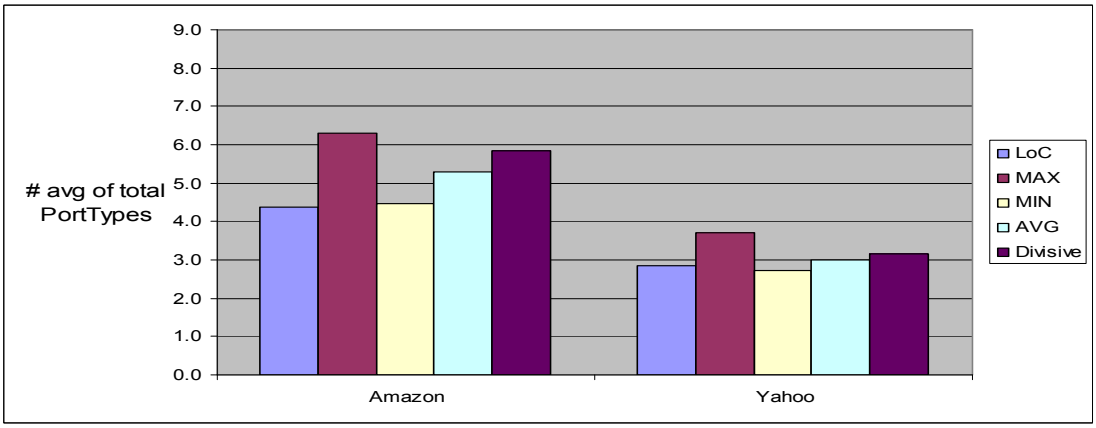
Στη μέτρηση των παραπάνω χαρακτηριστικών, δεν λαμβάνονται υπόψη διεπαφές που έχουν μόνο μία λειτουργία. Εν γένει αυτές οι διεπαφές αντιπροσωπεύουν λειτουργίες, που δεν ήταν δυνατόν να συσχετιστούν με τις υπόλοιπες και κατά μια έννοια αποτελούν θόρυβο που υπήρχε στην αρχική διεπαφή. Αναλυτικά τα αποτελέσματα για κάθε αλγόριθμο και για τα 3 κριτήρια συνεκτικότητας βρίσκονται στο παράρτημα Α.

Όσον αφορά στο μέγεθος των διεπαφών που δημιουργούνται από τον μηχανισμό ανακατασκευής, στα σχήματα του Πίνακα 5.3 δίνονται οι μέσοι όροι των παραγόμενων διεπαφών μετά την ανακατασκευή. Μια 1η παρατήρηση είναι ότι η συμπεριφορά κάθε εκδοχής σε σχέση με τις υπόλοιπες που μελετήθηκαν είναι παρόμοια και για τα δύο σύνολα δεδομένων. Μια $2^{η}$ παρατήρηση είναι ότι γενικά το πλήθος των διεπαφών που παράγεται είναι σχετικά μικρό, πράγμα που είναι επιθυμητό για να διευκολύνεται η δουλειά των προγραμματιστών που θα χρησιμοποιήσουν τις διεπαφές αυτές (γεγονός που προέκυψε και από την ποιοτική αξιολόγηση που ακολουθεί). Ανεξάρτητα του είδους συνεκτικότητας που χρησιμοποιείται, οι εκδοχές του ενωτικού αλγορίθμου που βασίζονται στον κανόνα του μακρινότερου γείτονα (MAX) παράγουν τις περισσότερες διεπαφές. Από την άλλη πλευρά τις λιγότερες διεπαφές παράγουν οι εκδοχές που βασίζονται είτε στον διαιρετικό αλγόριθμο ή στον ενωτικό αλγόριθμο όταν αυτός συνδυάζεται με τα κριτήρια MIN ή LOC. Ανεξάρτητα από τον αλγόριθμο που χρησιμοποιείται, οι εκδοχές που βασίζονται στην ακολουθιακή συνεκτικότητα παράγουν το μικρότερο πλήθος διεπαφών, ενώ για την επικοινωνιακή και εννοιολογική συνεκτικότητα το πλήθος των διεπαφών που παράγεται είναι συγκρίσιμο.

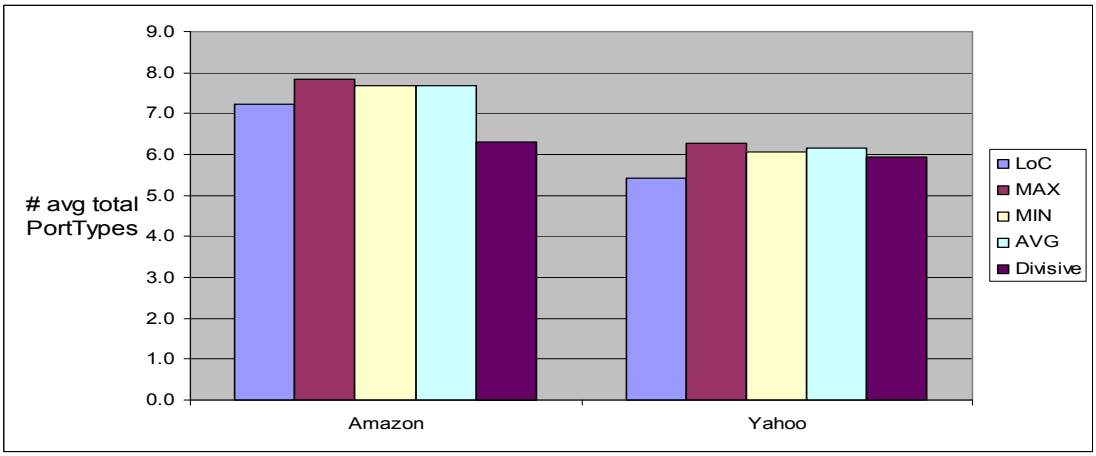
Πίνακας 5.3 Μέσος Όρος Παραγόμενων Διεπαφών για τις 3 Μετρικές ανά Αλγόριθμο.



(a) communicational



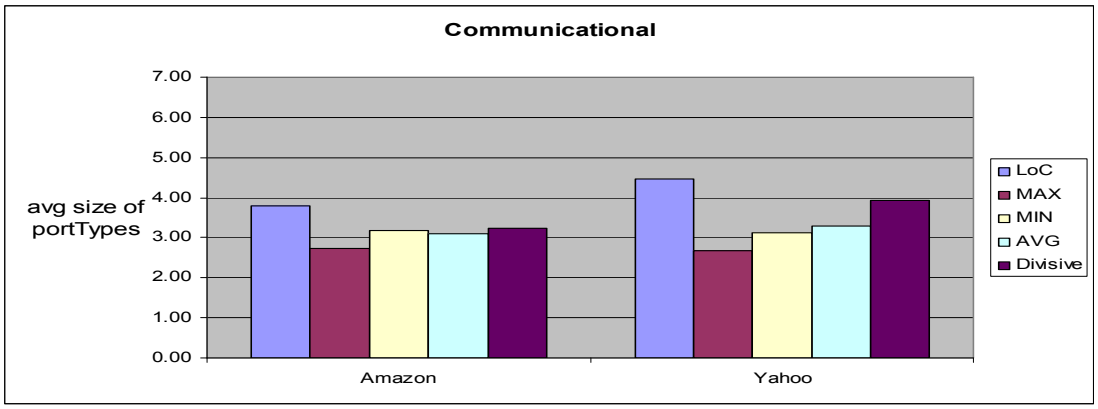
(b) sequential



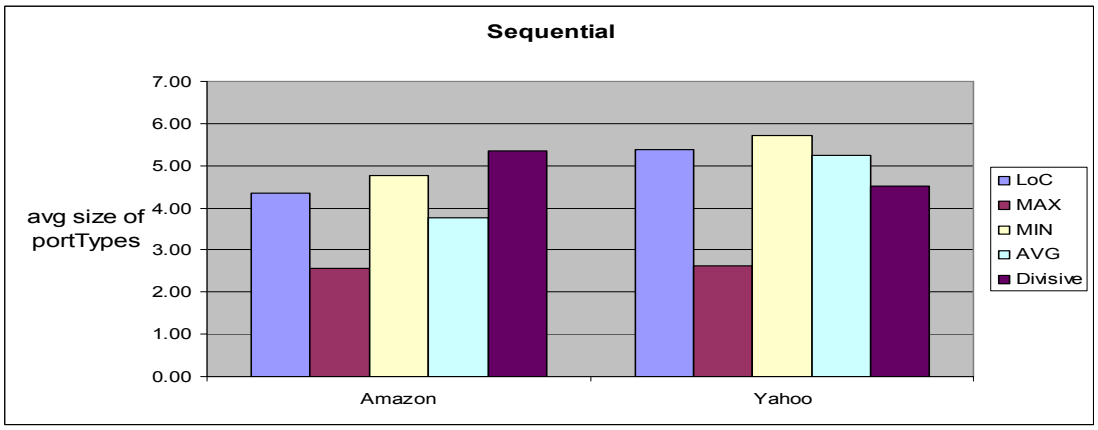
(c) conceptual

Σχετικά με το μέγεθος των διεπαφών που δημιουργούνται, υπολογίστηκε ο μέσος όρος του μέσου μεγέθους των διεπαφών που παράγονται για τις διεπαφές που εξετάστηκαν (Πινάκας 5.4). Όπως και στην περίπτωση του πλήθους των διεπαφών που παράγονται παρατηρούμε ότι η συμπεριφορά κάθε εκδοχής σε σχέση με τις υπόλοιπες που μελετήθηκαν είναι παρόμοια και για τα δύο σύνολα δεδομένων. Γενικά το μέγεθος των διεπαφών που παράγεται είναι σε λογικά πλαίσια (συνήθως κυμαίνεται από 3 μέχρι 6). Ανεξάρτητα του είδους συνεκτικότητας που χρησιμοποιείται, οι εκδοχές του ενωτικού αλγορίθμου που βασίζονται στον κανόνα του μακρινότερου γείτονα (MAX) παράγουν τις μικρότερες διεπαφές. Από την άλλη πλευρά στις περισσότερες περιπτώσεις οι εκδοχές που βασίζονται στον διαιρετικό αλγόριθμο ή στον ενωτικό και στο κριτήριο LOC παράγουν τις μεγαλύτερες διεπαφές πράγμα που είναι γενικά επιθυμητό μια που οι πολύ μικρές διεπαφές εν γένει δεν είναι βολικές για τους προγραμματιστές (γεγονός που προέκυψε και από την ποιοτική αξιολόγηση που ακολουθεί). Ανεξάρτητα από τον αλγόριθμο που χρησιμοποιείται, οι εκδοχές που βασίζονται στην εννοιολογική συνεκτικότητα παράγουν τις μικρότερες διεπαφές, ενώ για την επικοινωνιακή και ακολουθιακή συνεκτικότητα το μέγεθος των διεπαφών που παράγονται είναι συγκρίσιμο.

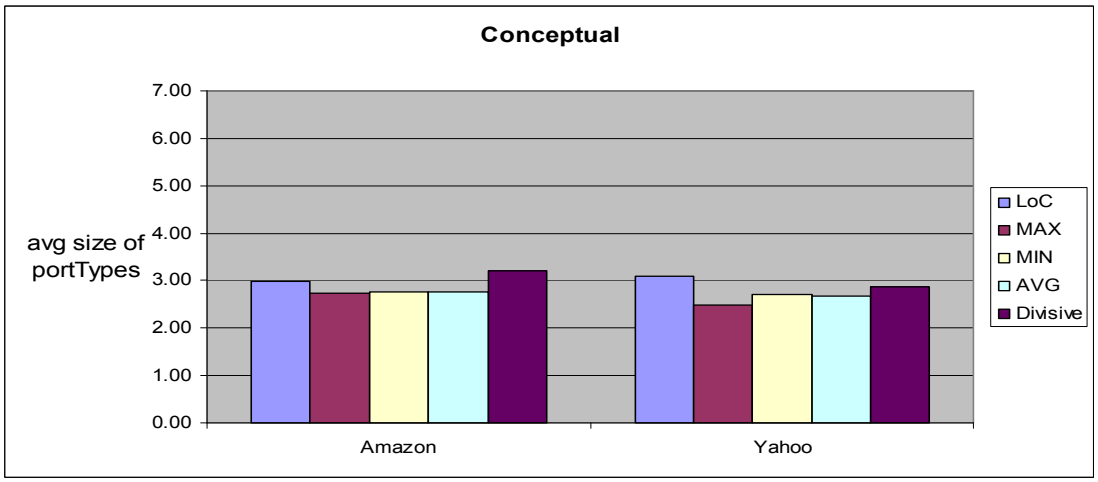
Πίνακας 5.4 Μέσος Όρος του Μέσου Μεγέθους των Διεπαφών για τις 3 Μετρικές ανά Αλγόριθμο.



(a) communicational



(b) sequential

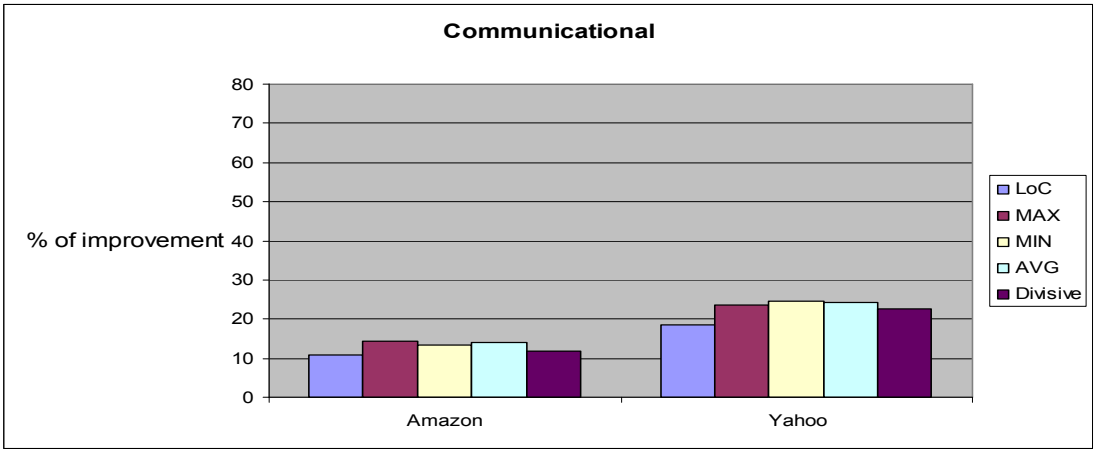


(c) conceptual

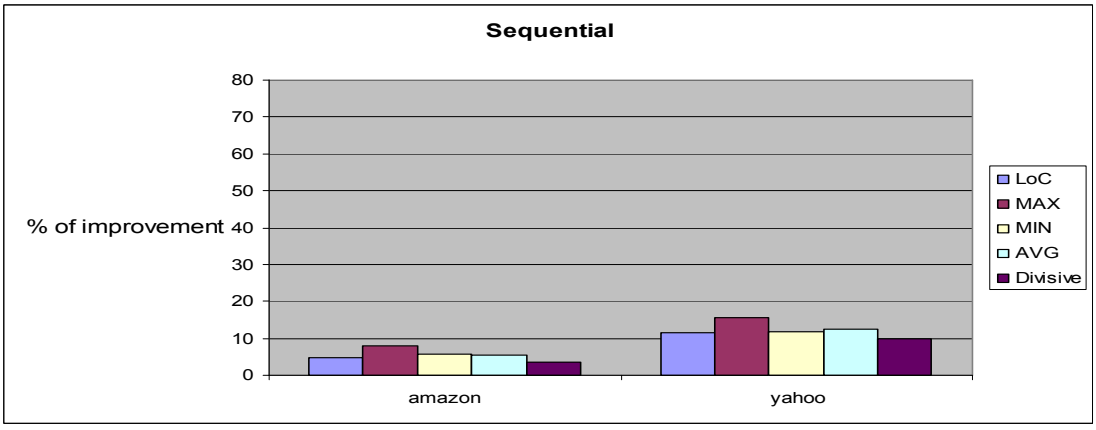
Τέλος για να ποσοτικοποιηθεί η μείωση της έλλειψης συνεκτικότητας, που επετεύχθη με βάση μια μετρική LoC_* για μια διεπαφή *si* , μετρήθηκε ο μέσος όρος την έλλειψης συνεκτικότητας των τελικών προτεινόμενων διεπαφών *D* μετά την ανακατασκευή της *si*., $\overline{LoC_*}(D) = \frac{\sum_{\forall r \in D}(LoC_*(r))}{|D|}$. Έτσι, το ποσοστό μείωσης της έλλειψης συνεκτικότητας δίνεται από τον τύπο: $decrease(si) = \frac{LoC_*(si) - \overline{LoC_*}(D)}{LoC_*(si)} * 100\%$, όπου $LoC_*(si)$ είναι η αρχική τιμή της έλλειψης συνεκτικότητας για την *si*.

Στα σχήματα του πίνακα 5.5 φαίνεται το ποσοστό μείωσης της έλλειψης συνεκτικότητας για κάθε εκδοχή. Τα ποσοστά αυτά προέκυψαν από το μέσο όρο των ποσοστών βελτίωσης των διεπαφών για κάθε μετρική ανά αλγόριθμο. Όπως προηγουμένως, η συμπεριφορά κάθε εκδοχής που μελετήθηκε σε σχέση με τις υπόλοιπες είναι παρόμοια και για τα δύο σύνολα δεδομένων. Γενικά οι εκδοχές που βασίζονται στην εννοιολογική συνεκτικότητα επιτυγχάνουν μεγάλα ποσοστά μείωσης πράγμα που αποτελεί ένδειξη ότι οι σχέσεις εννοιολογικής συνεκτικότητας μεταξύ των λειτουργιών είναι συχνές και ισχυρές. Αντίθετα, στις εκδοχές που βασίζονται στην επικοινωνιακή και ακολουθιακή συνεκτικότητα τα ποσοστά μείωσης είναι μικρά, γεγονός που αποτελεί ένδειξη ότι οι αντίστοιχες σχέσεις συνεκτικότητας μεταξύ των λειτουργιών είναι λίγες και ασθενείς. Ανεξάρτητα του είδους συνεκτικότητας που χρησιμοποιείται, οι εκδοχές του ενωτικού αλγορίθμου που βασίζονται στον κανόνα του μακρινότερου γείτονα (MAX) παράγουν τις πιο συνεκτικές διεπαφές. Από την άλλη πλευρά, οι εκδοχές που βασίζονται είτε στον διαιρετικό αλγόριθμο ή στον ενωτικό και στο κριτήριο LOC παράγουν τις μικρότερες μειώσεις.

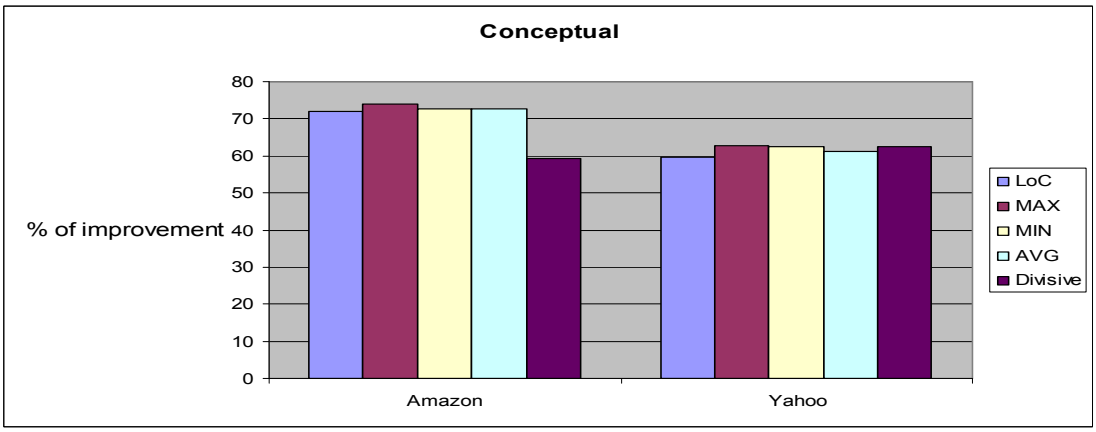
Πίνακας 5.5 Ποσοστό Μείωσης της Έλλειψης Συνεκτικότητας για τις 3 Μετρικές ανά Αλγόριθμος.



(a) communicational



(b) sequential



(c) conceptual

Συνοψίζοντας οι εκδοχές που μελετήθηκαν δεν παρουσιάζουν σημαντικές διαφορές στη γενικότερη συμπεριφορά τους. Παρόλα αυτά προέκυψαν κάποια διαφορετικά χαρακτηριστικά στα αποτελέσματα που παράγονται τα οποία μπορούν να αποτελέσουν κριτήριο επιλογής κάποιας εκδοχής ανάλογα με το αποτέλεσμα που είναι περισσότερο επιθυμητό. Συγκεκριμένα, αν στόχος είναι η δημιουργία όσο το δυνατόν πιο συνεκτικών διεπαφών, τότε καταλληλότερες είναι οι εκδοχές που βασίζονται στον κανόνα του μακρινότερου γείτονα. Αυτές οι εκδοχές όμως δημιουργούν πολλές στον αριθμό και μικρές διεπαφές. Από την άλλη αν στόχος είναι η δημιουργία μικρότερου πλήθους και μεγαλύτερου μεγέθους διεπαφών, καταλληλότερες είναι οι εκδοχές που βασίζονται είτε στο διαιρετικού αλγόριθμο ή στον ενωτικό αλγόριθμο και στο κριτήριο LOC. Όσον αφορά στο είδος της συνεκτικότητας (επικοινωνιακή, ακολουθιακή, εννοιολογική) το γεγονός ότι οι σχέσεις εννοιολογικής συνεκτικότητας είναι γενικότερα πιο συχνές και ισχυρές είναι μια ένδειξη ότι και οι διεπαφές που προκύπτουν με βάση αυτό το είδος συνεκτικότητας πιθανόν να είναι προτιμότερες. Παρόλα αυτά ένα πιο ασφαλές συμπέρασμα σχετικά με τη σύγκριση των 3 ειδών συνεκτικότητας προκύπτει από την ποιοτική αξιολόγηση των αποτελεσμάτων στην επόμενη ενότητα.

5.3. Ποιοτική Αξιολόγηση

Για την ποιοτική αξιολόγηση, δόθηκε σε 2 ειδικούς ένα κείμενο αξιολόγησης που περιείχε τα αποτελέσματα και ένα ερωτηματολόγιο προς συμπλήρωση (παράδειγμα υπηρεσίας στο Παράρτημα Β). Τα αποτελέσματα που δόθηκαν αφορούσαν τον διαιρετικό αλγόριθμο. Πιο συγκεκριμένα, για κάθε μια από τις 19 διεπαφές της Amazon το κείμενο περιείχε την αρχική διεπαφή και τα σύνολα διεπαφών που προέκυψαν για κάθε μετρική έλλειψης συνεκτικότητας με βάση τον διαιρετικό αλγόριθμο. Επιλέχθηκε ο διαιρετικός αλγόριθμος γιατί δόθηκε βάρος στη δημιουργία λίγων διεπαφών με ικανοποιητικό μέγεθος.

Ως προς τη σειρά βημάτων της αξιολόγησης, αρχικά οι ειδικοί κλήθηκαν να απαντήσουν αν οι αρχικές διεπαφές έπρεπε να διαιρεθούν. Στη συνέχεια για κάθε διεπαφή που έπρεπε να διαιρεθεί, ρωτήθηκαν να επιλέξουν ένα από τα 3 προτεινόμενα σύνολα διεπαφών το οποίο θεωρούν καλύτερο. Τελικώς, αν κανένα από τα προτεινόμενα σύνολα δεν κάλυπταν τις απαιτήσεις τους, κλήθηκαν να προτείνουν μια δική τους ανακατασκευή (μεταφορά λειτουργιών από μια διεπαφή σε μια άλλη, συγχωνεύσεις / διαχωρισμοί διεπαφών) που να καλύπτει τις απαιτήσεις τους.

Αφού συγκεντρώθηκαν τα αποτελέσματα, οι ειδικοί γενικά ανέφεραν ότι το κριτήριο επιλογής ενός συνόλου διεπαφών ήταν αν οι διεπαφές ομαδοποιούν λειτουργίες που είναι σχετικές ως προς λειτουργικότητα τους. Αν υπήρχαν 2 ή περισσότερα τέτοια σύνολα συνήθως επέλεγαν αυτό με τις λιγότερες διεπαφές.

Στο πρώτο στάδιο, ο πρώτος ειδικός απάντησε ότι δεν πρέπει να υποδιαιρεθούν οι διεπαφές A1, A2, A3, A8, A9, A10 και A11. Ο δεύτερος ειδικός επέλεξε τις διεπαφές A1, A2 και A9. Το κριτήριο και στις 2 περιπτώσεις ήταν ότι οι διεπαφές ήταν μικρές και είχαν σχετικές λειτουργίες.

Στη συνέχεια, ο πρώτος ειδικός επέλεξε το κριτήριο της έλλειψης εννοιολογικής συνεκτικότητας (LoC_{CN}) σε 9 εκ των 12 περιπτώσεων. Σε μία περίπτωση (A5) επέλεξε το σύνολο που προέκυψε από το κριτήριο έλλειψης επικοινωνιακής συνεκτικότητας (LoC_{CM}) επειδή αποτελούνταν από λιγότερες διεπαφές, σε σχέση με τα άλλα 2 σύνολα που τα θεώρησε εξίσου λογικά. Σε δύο διεπαφές (A15, A19), δεν επέλεξε κανένα από τα προτεινόμενα σύνολα επειδή θα προτιμούσε λιγότερες διεπαφές. Για αυτές τις περιπτώσεις, περιέγραψε την επιθυμητή ομαδοποίηση. Παρατηρήθηκε ότι η ομαδοποίηση αυτή θα μπορούσε να προκύψει από την συγχώνευση συγκεκριμένων διεπαφών που είχαν προκύψει στο σύνολο που είχε κατασκευαστεί με βάση το κριτήριο της εννοιολογικής συνεκτικότητας (LoC_{CN}).

Ο δεύτερος ειδικός, επέλεξε τα σύνολα που προέκυψαν με βάση την εννοιολογική συνεκτικότητα (LoC_{CN}) σε 15 από τις 16 περιπτώσεις. Σε μία περίπτωση επέλεξε το κριτήριο της ακολουθιακής συνεκτικότητας (LoC_{SQ}).

Στο τρίτο βήμα, ο πρώτος ειδικός πρότεινε αλλαγές σε 6 από τα 10 επιλεγμένα σύνολα, ενώ ο δεύτερος πρότεινε αλλαγές σε 8 από τα 16 επιλεγμένα σύνολα. Γενικά, σε κάθε περίπτωση οι ειδικοί δεν πρότειναν περισσότερες από 4 αλλαγές (μετακινήσεις λειτουργιών από μια διεπαφή σε μια άλλη, συγχωνεύσεις / διαχωρισμούς διεπαφών).

Συγκεντρωτικά, από τις απαντήσεις των ειδικών εξάγονται τα παρακάτω συμπεράσματα:

1. Και οι δυο ειδικοί έκριναν πως δεν πρέπει να διαιρεθούν μικρές διεπαφές, παρ' όλα αυτά υπάρχει ασάφεια στο τι θεωρείται μικρή διεπαφή (μέχρι 7 λειτουργίες για τον πρώτο ειδικό, μέχρι 4 λειτουργίες για τον δεύτερο ειδικό).
2. Και οι δυο ειδικοί έκριναν ότι η διαίρεση ήταν απαραίτητη σε πολλές περιπτώσεις. Ειδικά, θεώρησαν πολύ μεγάλο το μέγεθος και την πολυπλοκότητα συγκεκριμένων διεπαφών όπως η A5, A7 και A18.
3. Στις περισσότερες των περιπτώσεων, οι ειδικοί προτίμησαν τα σύνολα διεπαφών που βασίστηκαν στο κριτήριο της εννοιολογικής συνεκτικότητας. Σε μερικές περιπτώσεις τα σύνολα που βασίστηκαν στο κριτήριο της επικοινωνιακής και ακολουθιακής συνεκτικότητας θεωρήθηκαν εξίσου χρήσιμα.
4. Οι αλλαγές που έγιναν από τους ειδικούς στα επιλεγμένα σύνολα ήταν σχετικά λίγα, αποδεικνύοντας ότι ο μηχανισμός μπορεί να παρέχει χρήσιμα αποτελέσματα.
5. Τέλος, το γεγονός ότι σε μερικές περιπτώσεις οι ειδικοί βρήκαν τα σύνολα μεγάλα σε μέγεθος (μεγάλος αριθμός διεπαφών), δείχνει ότι η προτεινόμενος μηχανισμός θα μπορούσε να ενισχυθεί λαμβάνοντας υπόψη

την άποψη σχεδιαστών σχετικά με το πλήθος και το μέγεθος των διεπαφών.

ΚΕΦΑΛΑΙΟ 6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα εργασία προτάθηκε μια προσέγγιση ανακατασκευής διεπαφών υπηρεσιών διαδικτύου βασισμένη στη συνεκτικότητα. Για να ποσοτικοποιηθεί η έλλειψη συνεκτικότητας μελετήθηκαν 3 εναλλακτικές μετρικές για επικοινωνιακή, ακολουθιακή και εννοιολογική συνεκτικότητα.

Παράλληλα, προτάθηκε ένας μηχανισμός που βασισμένος στις παραπάνω μετρικές, ανακατασκευάζει μια διεπαφή υπηρεσίας διαδικτύου, σε ένα σύνολο συνεκτικότερων διεπαφών. Ο μηχανισμός στηρίζεται σε 5 εκδοχές αλγορίθμων, προσαρμοσμένους στις ιδιαιτερότητες του προβλήματος. Οι 4 από αυτές τις εκδοχές, προέκυψαν από έναν ιεραρχικό ενωτικό αλγόριθμο και η άλλη εκδοχή βασίζεται σε ιεραρχικό διαιρετικό αλγόριθμο.

Για να εκτιμηθεί η προτεινόμενη προσέγγιση εξετάστηκαν δυο διαφορετικά σύνολα διεπαφών υπηρεσιών, προερχόμενα από πραγματικές υπηρεσίες, δυο μεγάλων παρόχων υπηρεσιών, προερχόμενα από την Amazon και Yahoo αντίστοιχα. Αποδεικνύεται, ότι κάθε αλγόριθμος και για τα 3 κριτήρια συνεκτικότητας συμπεριφέρεται παρόμοια για τα δυο σύνολα δεδομένων που εξετάζονται, πράγμα που ισχυροποιεί την γενίκευση των συμπερασμάτων.

Γενικά, οι 5 αλγόριθμοι παράγουν μικρό πλήθος διεπαφών, κάτι που είναι επιθυμητό από τους προγραμματιστές όπως φάνηκε και από την ποιοτική αξιολόγηση. Ο ενωτικός αλγόριθμος που χρησιμοποιεί τον κανόνα του μακρινότερου γείτονα, παράγει τις περισσότερες διεπαφές, ανεξαρτήτου κριτηρίου συνεκτικότητας, ενώ ο ενωτικός αλγόριθμος που χρησιμοποιεί τον κανόνα του κοντινότερου γείτονα και ο διαιρετικός αλγόριθμος δημιουργούν τις λιγότερες.

Όσων αφορά στο μέγεθος των διεπαφών που παράγονται, ανεξάρτητα από την μετρική που χρησιμοποιείται, ο ενωτικός αλγόριθμος που βασίζεται στον κανόνα του μακρινότερου γείτονα παράγει τις μικρότερες διεπαφές, ενώ ο ενωτικός αλγόριθμος που βασίζεται στο κριτήριο της έλλειψης συνεκτικότητας και ο διαιρετικός αλγόριθμος παράγουν μεγαλύτερες σε μέγεθος διεπαφές. Από την μεριά των κριτηρίων συνεκτικότητας, η εννοιολογική συνεκτικότητα δημιουργεί τις μικρότερες διεπαφές, ενώ το μέγεθος που προκύπτει από την επικοινωνιακή και ακολουθιακή συνεκτικότητα, είναι συγκρίσιμο.

Από την σκοπιά της βελτίωσης συνεκτικότητας των διεπαφών, παρατηρήθηκε ότι ανεξαρτήτου μετρικής, ο ενωτικός αλγόριθμος που βασίζεται στον κανόνας του μακρινότερου γείτονα, δημιουργεί πιο συνεκτικές διεπαφές, αντίθετα ο ενωτικός αλγόριθμος που βασίζεται στον κανόνα έλλειψης συνεκτικότητας και ο διαιρετικός αλγόριθμος σημειώνουν μικρότερα ποσοστά βελτίωσης της συνεκτικότητας. Από την πλευρά των κριτηρίων συνεκτικότητας, το κριτήριο της έλλειψης εννοιολογικής συνεκτικότητας σημείωσε μεγάλα ποσοστά μείωσης σε σχέση με τις άλλες 2 μετρικές.

Από ποιοτικής άποψης, εξετάσθηκαν οι 3 μετρικές για τον διαιρετικό αλγόριθμο. Οι ειδικοί απάντησαν σε ερωτηματολόγια αξιολογώντας τα σύνολα διεπαφών που προέκυπταν μετά την ανακατασκευή μιας ήδη υπάρχουσας διεπαφής. Παρατηρήθηκε ότι οι ειδικοί στις περισσότερες περιπτώσεις επέλεξαν τα σύνολα που προέκυψαν χρησιμοποιώντας σαν κριτήριο την έλλειψη εννοιολογικής συνεκτικότητας. Ενώ όταν τα σύνολα που προέκυπταν, ήταν χρήσιμα για παραπάνω από μια περίπτωση, τότε κριτήριο επιλογής για τους ειδικούς ήταν το μικρό πλήθος διεπαφών.

Γενικά, για την δημιουργία συνεκτικών διεπαφών, ανεξαρτήτου πλήθους και μεγέθους, συστήνεται η χρήση του ενωτικού αλγορίθμου που βασίζεται στον κανόνα του μακρινότερου γείτονα, που δημιουργεί μεν συνεκτικές διεπαφές, είναι όμως δε πολλές και μικρές. Αντίθετα, όταν κριτήριο για τους προγραμματιστές είναι το λογικό

πλήθος και μέγεθος των διεπαφών, σε συνδυασμό με συνεκτικές διεπαφές συστήνεται η χρήση του ενωτικού αλγορίθμου που βασίζεται στον κανόνα έλλειψης συνεκτικότητας και του διαιρετικού αλγορίθμου. Ενώ φαίνεται ξεκάθαρα από τα ποιοτικά αποτελέσματα, ότι το κριτήριο της έλλειψης εννοιολογικής συνεκτικότητας προτιμάται από τους ειδικούς.

Εμπνευσμένοι, από την αξιολόγηση των ειδικών, σκοπεύουμε να ενισχύσουμε τον προτεινόμενο μηχανισμό ώστε να είναι προσαρμόσιμος στις προτιμήσεις των ειδικών, όσων αφορά στο πλήθος και στο μέγεθος των διεπαφών. Ακόμα, σκοπεύουμε να επεκτείνουμε τον προτεινόμενο μηχανισμό με μέσα που θα βοηθήσουν τους σχεδιαστές υπηρεσιών στην κατασκευή των διεπαφών.

ΑΝΑΦΟΡΕΣ

[1] Amazon Services, http://aws.amazon.com/

[2] Dionysis Athanasopoulos, Apostolos Zarras. Fine-grained Metrics of Cohesion Lack for Service Interfaces (Industry Track). 9th International Conference on Web Services (ICWS), Washington DC, USA, July 2011

[3] Shyam R. Chidamer, Chris F. Kemerer. "A metrics suite for object - oriented desing". IEEE Trans. Software Eng., vol.20, no.6, 476-493, June 1994.

[4] J. Davey and E. Burd, "Evaluating the suitability of Data Clustering for Software Remodularization", Proc. Seventh Working Conf. Reverse Eng., 2000.

[5] S. Demeyer, S. Ducasse and O.M. Nierstrasz. "Object Oriented Reengineering Patterns". Morgan Kaufman Publishers, 2002.

[6] B. Everitt. "Cluster Analysis". Heineman Education Books, London, 1974.

[7] M. Fokaefs, N. Tsiantalis, A. Chatzigeorgiou and j. Sander. "Decomposing Object – Oriented Class Modules Using Agglomerative clustering Technique. In Proceedings of the IEEE International Conference on Software Maintenance (ICSM), pages 93-101, 2009.

[8] M. Fowler, K Beck, J. Brant, W. Opdyke and D. Roberts. "Refactoring Improving the Design of Existing Code. Addison Wesley, Boston, MA, 1999.

[9] IBM www.ibm.com

[10] G.A. Miller. "The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information". Psychological Review, 63(2):343U355, 1956.

[11] OWL-S, http://www.w3.org/submission/OWL-S/

[12] M. Perepletchikov, C. Ryan, and Z. Tari. "The Impact of Service Cohesion on the analyzability of Service – oriented Software". IEEE Transactions on Services Computing, 3(2):89-103, 2010.

[13] Porter Stemmer, http://tartarus.org/martin/PorterStemmer/

[14] Remote Methods, http://www.remotemethods.com/

[15] SAWSDL, http://www.w3.org/2002/ws/sawsdl/spec/

[16] Service Finder, http://www.demo.service-finder.eu/index

[17] F. Simon. F. Steinbruckner and C. Lewerentz. "Metrics Based Refactoring". In proc. Of the 5th IEEE European Conference on Software Maintenance and Reengineering (CSMR), p. 30-39, 2001.

[18] UDDI, http://uddi.xml.org/

[19] T. A. Wiggerts. "Using Clustering Algorithms in Legacy Systems Remodularization". IEEE, 1997.

[20] Web Service Architecture, http://www.w3.org/TR/ws-arch/

[21] Web Service List, http://www.webservicelist.com/

[22] WSDL specification, http://www.w3.org/submission/wsdl-s/

[23] Xu, X., Lung, C., Zaman, M., and Srinivasan, A. Program Restructuring Through Clustering Techniques. In Proceedings of SCAM. pp 75-84, 2004.

[24] Yahoo Services, http://developer.searchmarketing.yahoo.com/

[25] H. Yao, A. Mark Orme and L. Etzkorn, "Cohesion Metrics for Ontology Design and Application", Journal of Computer Science 1(1): 107-113, 2005. ISSN 1549-3626, 2005.

[26] E. Yourdon and L. Constantine, Structured Design: "Fundamentals of a Discipline of Computer Programs and Systems Design". N.J., USA: Prentice – Hall, Inc., 1979.

ΠΑΡΑΡΤΗΜΑ Α

Πίνακας 6.1 Αποτελέσματα Ενωτικού Αλγορίθμου με βάση των Μέσο Όρο (AVG – επικοινωνιακή, ακολουθιακή και εννοιολογική συνεκτικότητα αντίστοιχα)

	Average Linkage (Communicational						
	Output interfaces						
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.83	11.62	1	3	1	4	2.33
A4	0.89	7.10	1	5	1	4	2.60
A5	0.79	14.38	0	8	2	5	3.38
A6	0.83	10.50	0	4	2	4	3.25
A7	0.75	18.37	2	9	1	8	3.22
A12	0.87	8.16	0	3	3	3	3.00
A13	0.89	9.05	3	7	1	4	2.60
A14	0.88	9.18	0	4	2	6	3.25
A15	0.74	20.63	1	7	1	4	2.57
A16	0.80	15.83	0	8	2	6	2.63
A17	0.61	32.55	0	8	2	5	2.88
A18	0.81	17.49	0	32	2	8	2.72
A19	0.68	24.28	0	5	2	5	3.20
Y1	0.83	9.71	0	4	3	9	5.00
Y2	0.58	30.94	0	10	2	4	2.80
Y3	0.51	36.11	2	8	1	6	2.50
Y4	0.67	26.74	1	5	1	3	2.40
Y6	0.77	17.02	0	3	2	3	2.33
Y7	0.87	7.86	0	2	3	4	3.50
Y8	0.49	40.90	0	7	2	4	2.71
Y10	0.42	41.92	0	4	2	4	2.50
Y12	0.62	21.63	1	5	1	3	2.00
Y14	0.61	27.98	0	9	2	8	3.78
Y15	0.78	6.34	1	2	1	6	3.50

Y17	0.72	15.06	0	3	2	6	4.00
Y19	0.49	33.46	0	8	2	6	2.88
Y21	0.74	23.50	2	13	1	4	2.15

Avereage Linkage (Sequential)

Output interfaces

ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.88	5.83	1	3	1	4	2.33
A4	0.93	4.01	0	4	2	5	3.25
A5	0.96	3.51	24	25	1	3	1.08
A6	0.86	11.92	2	4	1	8	3.25
A7	0.92	5.71	2	7	1	14	4.14
A12	0.94	2.92	2	4	1	4	2.25
A13	0.87	11.15	3	7	1	5	2.60
A14	0.94	3.87	0	3	4	5	4.33
A15	0.91	5.54	1	5	1	6	3.60
A16	0.93	5.07	0	6	3	5	3.50
A17	0.86	10.13	0	6	2	7	3.83
A18	0.92	7.81	1	27	1	10	3.22
A19	0.90	7.01	4	8	1	3	2.00
Y1	0.72	25.91	9	14	1	3	1.43
Y2	0.86	9.39	10	12	1	14	2.33
Y3	0.86	3.04	2	3	1	18	6.67
Y4	0.96	3.39	2	5	1	4	2.40
Y6	0.78	16.55	2	4	1	3	1.75
Y7	-	-	7	7	1	1	1.00
Y8	0.82	10.51	4	6	1	12	3.17
Y10	0.63	22.00	2	5	1	4	2.00
Y12	0.98	2.31	7	8	1	3	1.25
Y14	0.88	5.12	2	5	1	24	6.80
Y15	0.76	14.39	2	3	1	5	2.33
Y17	0.47	41.73	0	6	2	2	2.00
Y19	0.81	15.68	5	9	1	6	2.56
Y21	0.80	17.57	2	11	1	5	2.55

Average Linkage (Conceptual)

Output interfaces

ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.33	34.88	1	3	1	3	2.33
A4	0.08	90.13	3	7	1	3	1.86

A5	0.23	72.24	3	12	1	5	2.25
A6	0.28	61.36	0	6	2	3	2.17
A7	0.34	64.25	4	13	1	5	2.23
A12	0.13	84.21	2	4	1	4	2.25
A13	0.13	83.58	2	7	2	3	2.60
A14	0.28	65.01	1	5	1	3	2.60
A15	0.30	58.85	0	6	2	5	3.00
A16	0.00	100.00	0	8	2	3	2.63
A17	0.12	77.90	1	8	1	4	2.88
A18	0.03	96.38	4	35	1	4	2.49
A19	0.10	87.32	0	7	2	3	2.29
Y1	0.00	100.00	5	11	1	3	1.82
Y2	0.14	77.98	0	12	2	4	2.33
Y3	0.16	82.42	1	9	1	4	2.22
Y4	0.10	89.09	2	7	1	2	1.71
Y6	0.00	100.00	0	3	2	3	2.33
Y7	0.75	19.23	3	4	1	4	1.75
Y8	0.22	75.52	0	8	2	4	2.38
Y10	0.15	72.07	0	4	2	3	2.50
Y12	0.40	38.39	0	3	3	4	3.33
Y14	0.26	71.14	3	15	1	4	2.27
Y15	0.23	54.43	0	2	2	5	3.50
Y17	0.38	28.81	0	4	2	5	3.00
Y19	0.24	67.62	0	9	2	3	2.56
Y21	0.07	92.21	7	16	1	4	1.75

Πίνακας 6.2 Αποτελέσματα Ενωτικού Αλγορίθμου με βάση τον Κανόνα του Κοντινότερου Γείτονα (MIN – επικοινωνιακή, ακολουθιακή και εννοιολογική συνεκτικότητα αντίστοιχα)

Single Linkage (Communicational)							
Output interfaces							
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.83	11.62	1	3	1	4	2.33
A4	0.89	7.10	1	5	1	4	2.60
A5	0.82	11.08	0	8	2	6	3.38
A6	0.83	10.29	0	4	2	4	3.25

A7	0.76	17.80	2	10	1	6	2.90
A12	0.87	8.16	0	3	3	3	3.00
A13	0.89	9.05	3	7	1	4	2.60
A14	0.87	9.65	0	3	2	6	4.33
A15	0.76	18.73	1	7	1	4	2.57
A16	0.85	11.07	0	7	2	4	3.00
A17	0.59	35.36	0	7	2	6	3.29
A18	0.83	15.01	0	31	2	7	2.81
A19	0.68	24.28	0	5	2	5	3.20
Y1	0.88	4.82	0	5	2	7	4.00
Y2	0.54	35.46	0	10	2	5	2.80
Y3	0.48	39.91	2	8	1	5	2.50
Y4	0.60	34.25	1	5	1	4	2.40
Y6	0.80	13.69	0	3	2	3	2.33
Y7	0.87	7.86	0	2	3	4	3.50
Y8	0.50	39.80	0	7	2	5	2.71
Y10	0.42	41.92	0	4	2	4	2.50
Y12	0.62	21.63	1	5	1	3	2.00
Y14	0.63	25.09	0	9	2	6	3.78
Y15	0.78	6.34	1	2	1	6	3.50
Y17	0.70	16.73	0	4	2	4	3.00
Y19	0.49	33.31	0	8	2	5	2.88
Y21	0.74	23.50	2	13	1	4	2.15

Single Linkage (Sequential)

Output interfaces

ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.91	2.71	1	2	1	6	3.50
A4	0.93	4.06	0	4	2	5	3.25
A5	0.96	3.51	24	25	1	3	1.08
A6	0.82	15.76	4	7	1	4	1.86
A7	0.92	5.15	2	7	1	10	4.14
A12	0.96	0.83	2	3	1	7	3.00
A13	0.87	10.63	3	7	1	4	2.60
A14	0.94	3.87	0	3	4	5	4.33
A15	0.93	3.53	1	4	1	7	4.50
A16	0.90	7.70	0	4	2	9	5.25
A17	0.89	7.10	0	4	3	10	5.75
A18	0.92	7.07	1	24	1	9	3.63

A19	0.83	14.79	3	5	1	11	3.20
Y1	0.72	25.91	9	14	1	3	1.43
Y2	0.85	9.42	10	12	1	14	2.33
Y3	0.86	3.04	2	3	1	18	6.67
Y4	0.96	3.39	2	5	1	4	2.40
Y6	0.78	16.55	2	4	1	3	1.75
Y7	-	-	7	7	1	1	1.00
Y8	0.82	10.51	4	6	1	12	3.17
Y10	0.65	20.05	2	5	1	3	2.00
Y12	0.98	2.31	7	8	1	3	1.25
Y14	0.91	2.36	2	4	1	26	8.50
Y15	0.76	14.39	2	3	1	5	2.33
Y17	0.47	41.73	0	6	2	2	2.00
Y19	0.81	15.68	5	9	1	6	2.56
Y21	0.86	11.12	1	7	1	7	4.00

Single Linkage (Conceptual)

Output interfaces

ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.33	34.88	1	3	1	3	2.33
A4	0.08	90.13	3	7	1	3	1.86
A5	0.23	72.06	3	12	1	4	2.25
A6	0.28	61.36	0	6	2	3	2.17
A7	0.34	64.25	4	13	1	5	2.23
A12	0.13	84.21	2	4	1	4	2.25
A13	0.13	83.58	2	7	2	3	2.60
A14	0.28	65.01	1	5	1	3	2.60
A15	0.30	58.85	0	6	2	5	3.00
A16	0.00	100.00	0	8	2	3	2.63
A17	0.12	77.90	1	8	1	4	2.88
A18	0.03	96.38	4	35	1	4	2.49
A19	0.10	87.32	0	7	2	3	2.29
Y1	0.00	100.00	5	11	1	3	1.82
Y2	0.13	79.85	0	11	2	6	2.55
Y3	0.16	82.42	1	9	1	4	2.22
Y4	0.10	89.09	2	7	1	2	1.71
Y6	0.00	100.00	0	3	2	3	2.33

Y7	0.75	19.23	3	4	1	4	1.75
Y8	0.22	75.52	0	8	2	4	2.38
Y10	0.15	72.07	0	4	2	3	2.50
Y12	0.28	57.64	0	3	2	5	3.33
Y14	0.24	74.05	3	15	1	4	2.27
Y15	0.23	54.43	0	2	2	5	3.50
Y17	0.37	30.59	0	4	2	6	3.00
Y19	0.24	67.62	0	9	2	3	2.56
Y21	0.07	92.21	7	16	1	4	1.75

Πίνακας 6.3 Αποτελέσματα Ενωτικού Αλγορίθμου με βάση τον Κανόνα του Μακρινότερου Γείτονα (MAX – επικοινωνιακή, ακολουθιακή και εννοιολογική συνεκτικότητα αντίστοιχα)

Complete Linkage (Communicational)							
			Output interfaces				
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.77	17.81	2	4	1	3	1.75
A4	0.90	6.05	1	5	1	4	2.60
A5	0.79	14.34	0	8	2	6	3.38
A6	0.86	7.96	0	4	2	4	3.25
A7	0.74	19.34	1	10	1	8	2.90
A12	0.84	11.32	1	4	1	3	2.25
A13	0.89	9.22	1	7	1	3	2.17
A14	0.88	8.75	1	6	1	3	2.17
A15	0.74	20.63	1	7	1	4	2.57
A16	0.82	13.73	0	10	2	3	2.10
A17	0.61	32.55	0	8	2	5	2.88
A18	0.83	15.50	1	38	1	4	2.29
A19	0.66	26.24	1	6	1	4	2.67
Y1	0.83	10.08	1	7	1	4	2.86
Y2	0.56	33.43	1	11	1	3	2.55
Y3	0.55	29.95	1	8	1	4	2.50
Y4	0.67	26.45	1	5	1	3	2.40
Y6	0.77	17.02	0	3	2	3	2.33
Y7	0.78	17.18	1	3	1	4	2.33
Y8	0.49	40.90	0	7	2	4	2.71

Y10	0.42	41.92	0	4	2	4	2.50
Y12	0.62	21.63	1	5	1	3	2.00
Y14	0.67	20.81	0	10	2	7	3.40
Y15	0.71	14.77	1	3	1	4	2.33
Y17	0.69	17.84	0	6	2	2	2.00
Y19	0.47	37.15	0	9	2	5	2.56
Y21	0.74	23.50	2	13	1	4	2.15

	Complete Linkage (Sequential)						
	Output interfaces						
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.81	13.16	2	4	1	3	1.75
A4	0.91	6.31	4	8	1	3	1.63
A5	0.96	3.51	24	25	1	3	1.08
A6	0.84	13.73	5	8	1	3	1.63
A7	0.92	5.82	5	12	1	5	2.42
A12	0.90	7.02	4	6	1	3	1.50
A13	0.86	12.04	4	9	1	3	1.86
A14	0.73	25.93	7	10	1	2	1.30
A15	0.88	8.25	2	9	1	3	2.00
A16	0.92	5.63	1	8	1	3	2.63
A17	0.84	12.37	0	7	2	5	3.29
A18	0.91	8.61	9	38	1	5	2.29
A19	0.84	13.57	6	11	1	2	1.45
Y1	0.72	25.91	9	14	1	3	1.43
Y2	0.77	18.49	15	18	1	7	1.56
Y3	0.86	3.71	5	9	1	9	2.22
Y4	0.95	4.55	4	7	1	3	1.71
Y6	0.68	27.08	3	5	1	2	1.40
Y7	-	-	7	7	1	1	1.00
Y8	0.84	7.88	4	8	1	8	2.38
Y10	0.65	20.05	2	5	1	3	2.00
Y12	0.96	3.69	8	9	1	2	1.11
Y14	0.89	4.81	7	14	1	11	2.43
Y15	0.76	14.39	2	3	1	5	2.33
Y17	0.47	41.73	0	6	2	2	2.00
Y19	0.61	36.70	15	19	1	2	1.21
Y21	0.72	26.21	9	18	1	3	1.56

Complete Linkage (Conceptual) Output interfaces							
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.33	34.88	1	3	1	3	2.33
A4	0.08	90.13	3	7	1	3	1.86
A5	0.15	81.55	5	14	1	4	1.93
A6	0.28	61.36	0	6	2	3	2.17
A7	0.33	65.51	5	15	1	4	1.93
A12	0.13	84.21	2	4	1	4	2.25
A13	0.11	86.32	0	6	2	3	2.60
A14	0.28	65.01	1	5	1	3	2.60
A15	0.30	58.91	1	7	1	3	2.57
A16	0.00	100.00	0	8	2	3	2.63
A17	0.14	75.62	1	8	1	4	2.88
A18	0.04	95.97	4	35	1	4	2.49
A19	0.10	87.32	0	7	2	3	2.29
Y1	0.00	100.00	5	11	1	3	1.82
Y2	0.14	77.98	0	12	2	4	2.33
Y3	0.12	86.35	2	10	1	3	2.00
Y4	0.10	89.09	2	7	1	2	1.71
Y6	0.00	100.00	0	3	2	3	2.33
Y7	0.54	41.67	3	5	1	2	1.40
Y8	0.19	79.70	1	9	1	3	2.11
Y10	0.15	72.07	0	4	2	3	2.50
Y12	0.40	38.39	0	3	3	4	3.33
Y14	0.24	73.67	4	16	1	4	2.13
Y15	0.23	54.43	0	2	2	5	3.50
Y17	0.34	35.51	0	5	2	3	2.40
Y19	0.26	65.27	0	9	2	4	2.56
Y21	0.04	95.55	8	17	1	3	1.65

Πίνακας 6.4 Αποτελέσματα Ενωτικού Αλγορίθμου με βάση τον Κανόνα της Έλλειψης Συνεκτικότητας (LoC – επικοινωνιακή, ακολουθιακή και εννοιολογική συνεκτικότητα αντίστοιχα)

	Lack of Cohesion (Communicational)						
	Output interfaces						
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.83	11.62	1	3	1	4	2.33
A4	0.91	4.44	1	4	1	7	3.25
A5	0.85	7.28	0	5	2	11	5.40
A6	0.88	5.72	0	3	2	6	4.33
A7	0.74	19.71	2	8	1	9	3.63
A12	0.89	5.39	0	3	2	4	3.00
A13	0.89	9.05	3	7	1	4	2.60
A14	0.93	3.87	0	3	4	5	4.33
A15	0.78	16.43	1	6	1	5	3.00
A16	0.84	11.73	0	8	2	3	2.63
A17	0.69	23.45	0	5	2	11	4.60
A18	0.87	10.99	0	25	2	8	3.48
A19	0.71	20.51	0	4	2	7	4.00
Y1	0.90	2.69	0	2	9	11	10.00
Y2	0.64	24.27	0	8	2	6	3.50
Y3	0.70	10.98	1	4	1	8	5.00
Y4	0.67	26.74	1	5	1	3	2.40
Y6	0.80	13.69	0	3	2	3	2.33
Y7	0.87	7.86	0	2	3	4	3.50
Y8	0.61	25.94	0	6	2	5	3.17
Y10	0.61	16.20	0	4	2	3	2.50
Y12	0.63	19.81	1	4	1	4	2.50
Y14	0.73	13.77	0	6	2	13	5.67
Y15	0.78	6.34	1	2	1	6	3.50
Y17	0.70	16.59	0	2	6	6	6.00
Y19	0.61	17.74	0	5	2	7	4.60
Y21	0.80	17.34	2	11	1	4	2.55

Lack of Cohesion (Sequential)
Output interfaces

ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.88	5.91	1	3	1	4	2.33
A4	0.94	2.65	0	4	2	4	3.25
A5	0.96	3.51	24	25	1	3	1.08
A6	0.86	11.87	2	4	1	8	3.25
A7	0.93	4.97	2	6	1	12	4.83
A12	0.94	2.92	2	4	1	4	2.25
A13	0.87	11.15	3	7	1	5	2.60
A14	0.94	3.87	0	3	4	5	4.33
A15	0.93	3.15	1	4	1	7	4.50
A16	0.94	3.88	0	5	3	5	4.20
A17	0.90	6.28	0	4	4	10	5.75
A18	0.94	5.31	1	21	1	9	4.14
A19	0.94	3.43	3	6	1	5	2.67
Y1	0.80	18.51	9	13	1	5	1.54
Y2	0.85	9.41	10	12	1	14	2.33
Y3	0.86	3.04	2	3	1	18	6.67
Y4	0.96	3.39	2	5	1	4	2.40
Y6	0.78	16.55	2	4	1	3	1.75
Y7	-	-	7	7	1	1	1.00
Y8	0.82	10.51	4	6	1	12	3.17
Y10	0.65	19.97	2	4	1	6	2.50
Y12	0.98	2.31	7	8	1	3	1.25
Y14	0.88	5.89	2	5	1	26	6.80
Y15	0.76	14.39	2	3	1	5	2.33
Y17	0.47	41.73	0	6	2	2	2.00
Y19	0.87	9.54	5	9	1	5	2.56
Y21	0.80	17.57	2	11	1	5	2.55

Lack of Cohesion (Conceptual)

Output interfaces

ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.33	34.88	1	3	1	3	2.33
A4	0.06	92.59	3	7	1	4	1.86
A5	0.29	65.33	3	10	1	7	2.70
A6	0.34	52.44	0	4	2	7	3.25
A7	0.35	63.22	4	12	1	5	2.42
A12	0.13	84.21	2	4	1	4	2.25

A13	0.13	83.58	2	7	2	3	2.60
A14	0.23	71.31	1	5	1	4	2.60
A15	0.36	50.62	0	5	2	5	3.60
A16	0.00	100.00	0	8	2	3	2.63
A17	0.09	83.32	1	8	1	5	2.88
A18	0.04	96.26	4	35	1	4	2.49
A19	0.10	87.32	0	7	2	3	2.29
Y1	0.00	100.00	5	11	1	3	1.82
Y2	0.17	73.26	0	10	2	5	2.80
Y3	0.16	82.42	1	9	1	4	2.22
Y4	0.10	89.09	2	7	1	2	1.71
Y6	0.00	100.00	0	3	2	3	2.33
Y7	0.75	19.23	3	4	1	4	1.75
Y8	0.22	75.52	0	8	2	4	2.38
Y10	0.15	72.07	0	4	2	3	2.50
Y12	0.47	28.17	0	3	3	4	3.33
Y14	0.27	70.61	3	13	1	6	2.62
Y15	0.23	54.43	0	2	2	5	3.50
Y17	0.35	33.39	1	3	1	9	4.00
Y19	0.38	48.47	0	5	2	6	4.60
Y21	0.07	92.21	7	16	1	4	1.75

Πίνακας 6.5 Αποτελέσματα Διαιρετικού Αλγορίθμου (Divisive – επικοινωνιακή, ακολουθιακή και εννοιολογική συνεκτικότητα αντίστοιχα)

Divisive (Communicational)							
			Output interfaces				
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.83	11.50	1	3	1	3	2.33
A4	0.91	5.28	1	5	1	4	2.60
A5	0.81	11.13	0	5	2	11	5.40
A6	0.83	10.50	0	4	2	4	3.25
A7	0.77	15.08	5	12	1	8	2.42
A12	0.89	6.19	0	3	2	4	3.00
A13	0.88	10.20	2	6	1	3	2.17
A14	0.86	11.07	1	6	1	3	2.17

A15	0.79	15.31	1	6	1	5	3.00
A16	0.81	15.11	0	8	2	6	2.63
A17	0.69	23.57	0	5	2	9	4.60
A18	0.84	14.29	1	31	1	10	2.81
A19	0.67	23.78	0	5	2	4	3.20
Y1	0.80	13.77	0	4	2	8	5.00
Y2	0.50	40.92	0	10	2	6	2.00
Y3	0.62	21.51	1	3	1	14	6.00
Y4	0.80	13.23	1	4	1	5	3.00
Y6	0.75	19.01	0	3	2	3	2.00
Y7	0.89	6.48	0	2	3	4	3.00
Y8	0.50	39.59	0	7	2	4	2.00
Y10	0.42	41.92	0	4	2	4	2.00
Y12	0.63	19.44	1	4	1	4	2.00
Y14	0.59	30.42	1	12	1	8	2.00
Y15	0.76	9.10	0	2	3	4	3.00
Y17	0.71	16.96	0	2	6	6	6.00
Y19	0.45	39.74	0	5	3	7	4.00
Y21	0.71	26.03	2	11	1	6	2.00

	Divisive (Sequential)						
			Output interfaces				
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.91	2.77	2	3	1	5	2.33
A4	0.94	3.01	0	2	2	11	6.50
A5	0.77	7.00	8	17	1	3	1.59
A6	0.97	0.00	0	1	13	13	13.00
A7	0.93	4.28	4	11	1	11	2.64
A12	0.97	0.00	0	1	9	9	9.00
A13	0.87	11.20	5	9	1	2	1.44
A14	0.98	0.00	0	1	13	13	13.00
A15	0.92	4.13	1	5	1	6	3.60
A16	0.94	3.91	0	7	2	5	3.00
A17	0.89	6.53	0	6	2	7	3.83
A18	0.93	6.23	2	30	1	6	2.90
A19	0.89	7.90	2	7	1	5	2.29
Y1	0.72	25.81	9	14	1	3	1.00
Y2	0.84	11.26	11	15	1	8	1.00

Y3	0.86	2.80	2	7	1	6	2.00
Y4	0.96	3.20	3	6	1	4	2.00
Y6	0.86	7.64	2	3	1	5	2.00
Y7	-	-	7	7	1	1	1.00
Y8	0.82	9.98	6	10	1	6	1.00
Y10	0.78	3.81	2	4	1	5	2.00
Y12	0.96	2.94	5	7	1	3	1.00
Y14	0.91	2.01	2	9	1	9	3.00
Y15	0.62	30.41	5	6	1	2	1.00
Y17	0.47	42.09	0	6	2	2	2.00
Y19	0.94	2.66	5	6	1	18	3.00
Y21	0.75	22.68	1	4	1	23	7.00

Divisive (Conceptual)							
			Output interfaces				
ID	avg LoC	% of LoC impr.	#singl.	#total	min #op	max #op	avg operations
A3	0.32	38.14	1	2	1	6	3.50
A4	0.11	86.84	3	7	1	3	1.86
A5	0.25	70.01	3	10	1	6	2.70
A6	0.28	61.36	0	6	2	3	2.17
A7	0.33	65.48	5	16	1	3	1.81
A12	0.13	84.21	2	4	1	2	1.33
A13	0.13	83.90	0	4	2	5	3.25
A14	0.21	73.23	1	5	1	5	2.60
A15	0.45	38.09	0	4	2	6	4.50
A16	0.00	100.00	0	6	3	5	3.50
A17	0.13	76.89	1	6	1	6	3.83
A18	0.03	96.46	4	26	1	15	3.35
A19	0.11	85.21	0	6	2	4	2.67
Y1	0.00	100.00	5	11	1	3	1.10
Y2	0.13	80.26	0	9	2	9	2.90
Y3	0.19	78.34	1	9	1	5	2.05
Y4	0.10	89.09	2	7	1	2	1.05
Y6	0.00	100.00	0	2	3	4	2.90
Y7	0.54	41.67	3	5	1	2	1.25
Y8	0.22	75.87	0	9	2	3	2.20
Y10	0.13	76.72	0	4	2	4	1.95
Y12	0.37	42.50	0	4	2	3	1.90
Y14	0.27	70.33	3	15	1	4	2.10

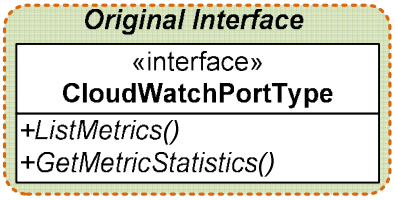
Y15	0.50	0.00	0	1	7	7	6.60
Y17	0.31	42.42	0	6	2	2	1.90
Y19	0.26	64.75	0	8	2	5	2.10
Y21	0.09	89.98	7	14	1	5	2.00

ΠΑΡΑΡΤΗΜΑ Β

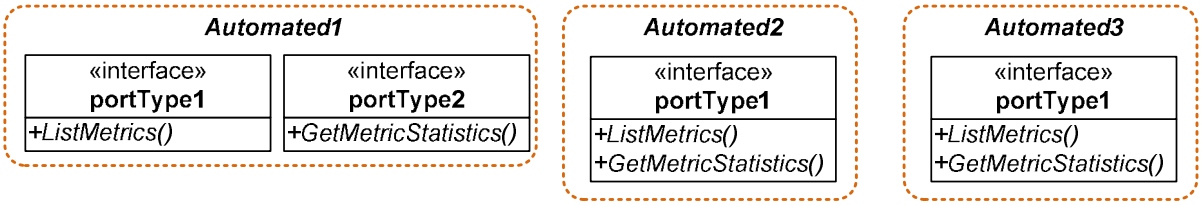
Στο παράρτημα Β υπάρχουν 2 παραδείγματα υπηρεσίων για το πως παρατέθηκαν στο ερωτηματολογιο. Το πλήρες ερωτηματολόγιο βρίσκεται διεύθυνση http://www.cs.uoi.gr/~dathanas/software/Template.pdf

Expert Evaluation

1. CLOUDWATCHPORTTYPE



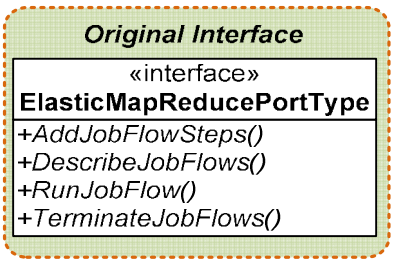
1.1. AUTOMATED DECOMPOSITIONS



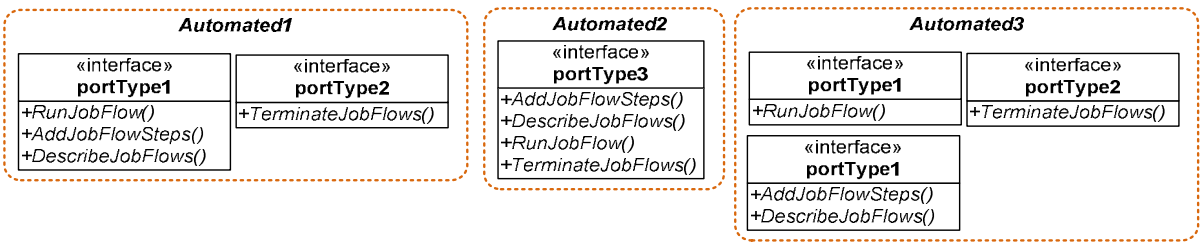
1.2. EVALUATION

Question	*Answer*		*Explanation*
Does the original interface need to be decomposed?	*Yes* ____	*No* ____	
Which automated decomposition do you prefer?	*A1* ____ *A2* ____ *A3* ____	*None* ____	
Which corrections in the selected decomposition(s) do you want to make?			

2. ELASTICMAPREDUCEPORTTYPE



2.1. AUTOMATED DECOMPOSITIONS



2.2. EVALUATION

Question	*Answer*		*Explanation*
Does the original interface need to be decomposed?	*Yes* ____	*No* ____	
6.1.1. Which automated decomposition do you prefer?	*A1* ____ *A2* ____ *A3* ____	*None* ____	
6.1.2. Which corrections in the selected decomposition(s) do you want to make?			