Πανεπιστήμιο Ιωαννίνων
Σχολή Θετικών Επιστημών
Τμήμα Πληροφορικής
Ιούνιος 2005

Μεταπτυχιακή Εργασία Ειδίκευσης

Επιλογή της καλύτερης Σύνθετης Υπηρεσίας Διαδικτύου

Φοιτήτρια: Μαρία Παπαφώτη
Επιβλέπουσα Καθηγήτρια: κα. Ευαγγελία Πιτουρά

ΕΠΙΛΟΓΗ ΤΗΣ ΚΑΛΥΤΕΡΗΣ ΣΥΝΘΕΤΗΣ ΥΠΗΡΕΣΙΑΣ ΔΙΑΔΙΚΤΥΟΥ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

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

από την

Μαρία Παπαφώτη

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΝΟΕΜΒΡΙΟΣ 2005

ΕΥΧΑΡΙΣΤΙΕΣ

Δόξα τω Θεώ ήρθε η ώρα να γράψω κι αυτό το σημείωμα. Δόξα τω Θεώ τελειώσε καλά κι ο αγώνας για την απόκτηση του μεταπτυχιακού.

Το σημείωμα αυτό αποτελεί την τελευταία πινελιά στη μεταπτυχιακή μου εργασία και σηματοδοτεί το τέλος αυτής της εργασίας και ταυτόχρονα το τέλος των σπουδών μου ως φοιτήτρια στο μεταπτυχιακό πρόγραμμα σπουδών του τμήματος πληροφορικής του πανεπιστημίου Ιωαννίνων. Τελειώνοντας, λοιπόν, θα ήθελα να ευχαριστήσω όλους εκείνους που στη διάρκεια των τριών τελευταίων χρόνων με βοήθησαν, με οποιοδήποτε τρόπο, να πάρω το μεταπτυχιακό δίπλωμα ειδίκευσης.

Ιδιαιτέρα ευχαριστώ:

Την επιβλέπουσα καθηγήτρια μου, κυρία Ευαγγελία Πιτούρα, για την βοήθεια της στην εκπόνηση αυτής της εργασίας καθώς και για την προσφορά εργασίας και χώρου εργασίας στο εργαστήριο της από το πρώτο κιόλας έτος των μεταπτυχιακών σπουδών μου.

Τον κύριο Παναγιώτη Βασιλειάδη για τη πολύτιμη βοήθεια που προσέφερε στη εκπόνηση της εργασίας. Ο κύριος Βασιλειαδής, αν και δεν ήταν ο επιβλέπων καθηγητής μου, ήταν παρών στις περισσότερες συναντήσεις που αφορούσαν στην εξέλιξη της μεταπτυχιακής μου εργασίας και η συμβολή του ήταν σημαντική. Επιπλέον ευχαριστώ τον κύριο Βασιλειάδη για την ειλικρινή αλλά ταυτόχρονα θετική στάση που πάντα είχε απέναντι μου, και για το ότι, σε διάφορες φάσεις, με άκουσε και με ενεθάρρυνε να συνεχίσω.

Τους φίλους Γεωργία, Αγγελική, Στάμω, Μάχη και Νίκο για την συμπαράσταση, τη βοήθεια και την ενθάρρυνση που μου προσέφεραν τρία χρόνια τώρα.

Τους γονείς και τα αδέρφια μου που έζησαν όλο τον αγώνα μου τα τελευταία τρία χρόνια και ήταν πάντα δίπλα μου παρακινώντας με και βοηθώντας με να φτάσω στο τέλος.

Τους κυρίους Βασίλειο Δημακόπουλο και Απόστολο Ζάρα που ως μέλη της τριμελούς επιτροπής συνέβαλλαν με τις παρατηρήσεις τους στην βελτίωση και ολοκλήρωση της εργασίας.

Ευχαριστώ!

Μαρία

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Μαρία Παπαφώτη του Γεωργίου και της Φρειδερίκης. MSc , Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Νοέμβριος, 2005. Επιλογή της καλύτερης σύνθετης υπηρεσίας Διαδικτύου. Επιβλέπουσα: Ευαγγελία Πιτουρά.

Η συνεχής ανάπτυξη της ηλεκτρονικής τεχνολογίας επηρεάζει όλο και περισσότερο τον τρόπο με τον οποίο οργανώνουμε τις δουλειές μας τόσο στο εργασιακό μας περιβάλλον όσο και στην καθημερινή μας ζωή. Έτσι πολλές εργασίες γίνονται πλέον με τη βοήθεια των ηλεκτρονικών υπολογιστών και των κατάλληλων εφαρμογών λογισμικού. Νέα προοπτική στην αυτοματοποίηση των διαφόρων εργασιών ήρθε να δώσει η ανάπτυξη των δικτύων και ιδιαίτερα του Διαδικτύου. Μέσα από το Διαδίκτυο είναι δυνατόν πληροφορίες αλλά και εφαρμογές να χρησιμοποιούνται από πολύ περισσότερο κόσμο. Σε αυτό το πλαίσιο αναπτύσσεται και η τεχνολογία των υπηρεσιών Διαδικτύου που θα λέγαμε ότι δεν είναι τίποτε άλλο παρά εφαρμογές τις οποίες μπορούμε να προσπελάσουμε μέσα από το Διαδίκτυο.

Η ανάπτυξή αυτής της νέας τεχνολογίας, των υπηρεσιών Διαδικτύου, έγινε το αντικείμενο μελέτης διαφόρων επιστημόνων. Ο τρόπος ανάπτυξης τους, η δυνατότητα προσπέλασης τους αλλά και η δυνατότητα συνδυασμού τους, απασχόλησε και απασχολεί όσους είτε χρησιμοποιούν αυτή την τεχνολογία είτε ασχολούνται με αυτήν ερευνητικά. Σε αυτό το πλαίσιο κινείται και η παρούσα εργασία, η οποία προτείνει ένα τρόπο επιλογής της «καλύτερης» προς εκτέλεση υπηρεσίας σε ένα περιβάλλον σύνθεσης τέτοιων υπηρεσιών με στόχο την εκτέλεση πολύπλοκων διαδικασιών.

Πιο συγκεκριμένα, στην παρούσα εργασία προτείνεται ένας «λαίμαργος» αλγόριθμος βάσει του οποίου αν έχουμε μια ροή εργασίας που ορίζει εναλλακτικούς τρόπους

σύνθεσης διαφόρων υπηρεσιών επιλέγουμε εκείνον τον τρόπο που θεωρείται καλός έχοντας κάποιες παραμέτρους όπως είναι για παράδειγμα ο χρόνος εκτέλεσης, Επιπλέον, προσομοιώνεται η χρήση του συγκεκριμένου αλγορίθμου και μελετάται η συμπεριφορά του.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή

Στην εποχή της πληροφορίας και του Διαδικτύου οι τεχνολογίες που αφορούν στην αξιοποίηση του παγκόσμιου ιστού κερδίζουν συνεχώς έδαφος. Μια τέτοια τεχνολογία είναι αυτή που αφορά στις υπηρεσίες Διαδικτύου δηλαδή τις εφαρμογές εκείνες που μπορούν να προσδιοριστούν από ένα URI και των οποίων οι διεπαφές προσδιορίζονται με τη χρήση XML εγγράφων. Το θετικό με αυτού του τύπου τις εφαρμογές είναι ότι είναι δημοσιευμένες οι περιγραφές τους στον παγκόσμιο ιστό και επομένως είναι εύκολη η εύρεση και η χρήση τους ενώ επιπλέον δεν εξαρτώνται ούτε από το λειτουργικό σύστημα ούτε από το ακριβές περιβάλλον από το οποίο καλούνται. Όσο για το λόγο που η τεχνολογία στρέφεται προς τις υπηρεσίες Διαδικτύου δεν είναι παρά το γεγονός ότι με τη χρήση τους οι διάφορες ιστοσελίδες γίνονται όλο και πιο φιλικές προς τους χρήστες παρέχοντας όλο και περισσότερες ευκολίες σε αυτόν. Τη λειτουργία υπηρεσιών Διαδικτύου πλαισιώνουν διαφορά πρωτόκολλα τα οποία καθορίζουν: τον τρόπο ανταλλαγής της πληροφορία στο περιβάλλον του παγκόσμιου ιστού, τη μορφή που θα πρέπει να έχει η πληροφορία που ανταλλάσσεται (μορφή του μηνύματος) και τέλος τον τρόπο με τον οποίο θα περιγράφεται η λειτουργία και ο τρόπος χρήσης κάθε υπηρεσίας έτσι ώστε να είναι δυνατός ο εντοπισμός της.

Η έρευνα σε αυτό το χώρο έχει ένα ευρύ πεδίο να καλύψει. Η ιδέα της ηλεκτρονικής συνεργασίας προϋπήρχε από τις B2B εφαρμογές και θεωρήθηκε σκόπιμο να υλοποιηθεί και στην περίπτωση της τεχνολογίας των υπηρεσιών Διαδικτύου. Για το σκοπό αυτό χρειάστηκε να καθοριστεί ο τρόπος εντοπισμού μια υπηρεσίας Διαδικτύου με συγκεκριμένα χαρακτηριστικά. Επίσης για τον ίδιο λόγο δημιουργήθηκαν γλώσσες σύνθεσης υπηρεσιών Διαδικτύου ή ακόμη και πλαίσια σύνθεσης και εκτέλεσης τέτοιων υπηρεσιών. Μια άλλη κατεύθυνση προς την οποία κινήθηκαν οι ερευνητές είναι ο καθορισμός της ποιότητας των υπηρεσιών Διαδικτύου απλών ή και σύνθετων καθώς και η εξοικονόμηση κατά το δυνατόν των πόρων του δικτύου με την ταυτόχρονη ποιοτική λειτουργία των υπηρεσιών

Στη παρούσα εργασία μελετάμε η δυνατότητα βελτιστοποίησης της επιλογής των απλών υπηρεσιών Διαδικτύου που απαρτίζουν μια συνθέτη έτσι ώστε να ελαχιστοποιηθεί το κόστος της σύνθετης υπηρεσίας Διαδικτύου. Με τον ορό σύνθετη υπηρεσία εννοούμε την υπηρεσία

που επιτυγχάνει ένα στόχο χρησιμοποιώντας άλλες ήδη έτοιμες υπηρεσίες. Η πληθώρα των υπηρεσιών Διαδικτύου που υπάρχουν στο Διαδίκτυο εξυπηρετώντας τον ίδιο ή παρόμοιο στόχο έχει ως αποτέλεσμα τη δυνατότητα ορισμού με πολλούς διαφορετικούς τρόπους της σύνθετης υπηρεσίας (κάθε τρόπος αφορά στην επιλογή άλλων υπηρεσιών). Κάθε υπηρεσία διαδικτύου μπορούμε να τη χαρακτηρίσουμε με βάση κριτήρια όπως ο χρόνος εκτελέσεις της ή η αξιοπιστία της και με βάση αυτά μπορούμε να πούμε ότι μια υπηρεσία είναι καλύτερη ή πιο συμφέρουσα από κάποια άλλη. Όμοια μπορούμε να πούμε ότι ένας τρόπος εκτέλεσης μιας σύνθετης υπηρεσία είναι καλύτερος από έναν άλλο. Είναι προφανές ότι η ποιότητα της σύνθετης υπηρεσίας εξαρτάται από την ποιότητα των υπηρεσιών που την απαρτίζουν. Αν θεωρήσουμε ως ένα γράφο τους διάφορους τρόπους δημιουργίας μιας σύνθετης υπηρεσίας (κόμβοι του γράφου προφανώς είναι οι απλές υπηρεσίες που χρησιμοποιούνται) τότε το πρόβλημα αυτής της εργασίας ανάγεται σε εκείνο της εύρεσης του ελάχιστου μονοπατιού σε ένα κατευθυνόμενο γράφο. Θεωρήσαμε, όμως, ότι τέτοιοι αλγόριθμοι δεν θα ήταν η καλύτερη λύση λόγω της δυναμικής και κατανεμημένης φύσις του Διαδικτύου και έτσι προτείνουμε τη χρήση ενός greedy αλγόριθμου ο οποίος προφανώς δεν δίνει πάντοτε τη βέλτιστη λύση την πλησιάζεις όμως αρκετά όπως έδειξε η προσομοίωση που υλοποιήσαμε. Στη συνέχεια επεκτείνουμε τον αλγόριθμο αυτό για περιπτώσεις οπού η δυνατότητα λειτουργίας όλων των υπηρεσιών δεν είναι γνωστή εκ των προτέρων.

Σα μέτρο σύγκρισης για τον αλγόριθμο που προτείνουμε χρησιμοποιούμε τον γνωστό αλγόριθμο υπολογισμού ελάχιστων διαδρομών Bellmann – Ford. Επίσης υλοποιήσαμε μια παραλλαγή για να καλύψουμε και τις περιπτώσεις αποτυχημένων υπηρεσιών.

Τα αποτελέσματα της εργασίας ήταν ενθαρρυντικά αφού η προσομοίωση έδειξε ότι ο greedy αλγόριθμος που προτάθηκε για την επιλογή των υπηρεσιών που θα εκτελεστούν με στόχο την εκτέλεση μιας σύνθετης υπηρεσίας έχει αρκετά καλές επιδόσεις παρόμοιες με αυτές του βέλτιστου ειδικά όταν το βήμα του greedy είναι τουλάχιστον 3.

1.2 Δομή της Εργασίας

Αρχικά λοιπόν, στο πρώτο κεφάλαιο παρουσιάζονται οι τεχνολογίες των υπηρεσιών Διαδικτύου και των ροών εργασίας. Και οι δυο αυτές τεχνολογίες αποτελούν το πλαίσιο μέσα στο οποίο κινείται κάθε εργασία που στοχεύει στη βελτίωση της ποιότητας των υπηρεσιών που προσφέρονται από κατάλληλα συνδυασμένες υπηρεσίες Διαδικτύου. Στη συνέχεια, στο δεύτερο κεφάλαιο παρουσιάζεται σχετική εργασία, ενώ στο τρίτο κεφάλαιο παρουσιάζεται ο αλγόριθμος που προτείνεται σε αυτή την εργασία καθώς και ένας βέλτιστος αλγόριθμος για αυτό, που χρησιμοποιείται στη συνέχεια ως μέτρο σύγκρισης για την απόδοση του πρώτου. Στο τέταρτο κεφαλαίο, φαίνονται τα αποτελέσματα της εκτέλεσης της προσομοίωσης.

ΚΕΦΑΛΑΙΟ 2. ΑΠΛΕΣ ΚΑΙ ΣΥΝΘΕΤΕΣ ΥΠΗΡΕΣΙΕΣ ΔΙΑΔΙΚΤΥΟΥ

Το κεφάλαιο αυτό έχει στόχο να εισάγει τον αναγνώστη στις έννοιες και στις τεχνολογίες που αναφέρονται στο υπόλοιπο της εργασίας. Στις πρώτες ενότητες δίνεται σύντομη περιγραφή των υπηρεσιών Διαδικτύου και των τεχνολογιών που διέπουν την χρήση τους, Στη συνέχεια περιγράφονται οι ροές εργασίας (workflows) και η χρήση τους σε γενικό επίπεδο καθώς επίσης και πιο συγκεκριμένα η χρήση τους στις ροές εργασίας Υπηρεσιών Διαδικτύου που χρησιμοποιούνται για τον ορισμό σύνθετων υπηρεσιών Διαδικτύου. Σε αυτό το σημείο παρουσιάζεται και η BPEL ως ένα παράδειγμα μιας γλώσσας για την έκφραση ροών Υπηρεσιών Διαδικτύου.

2.1. Υπηρεσίες Διαδικτύου

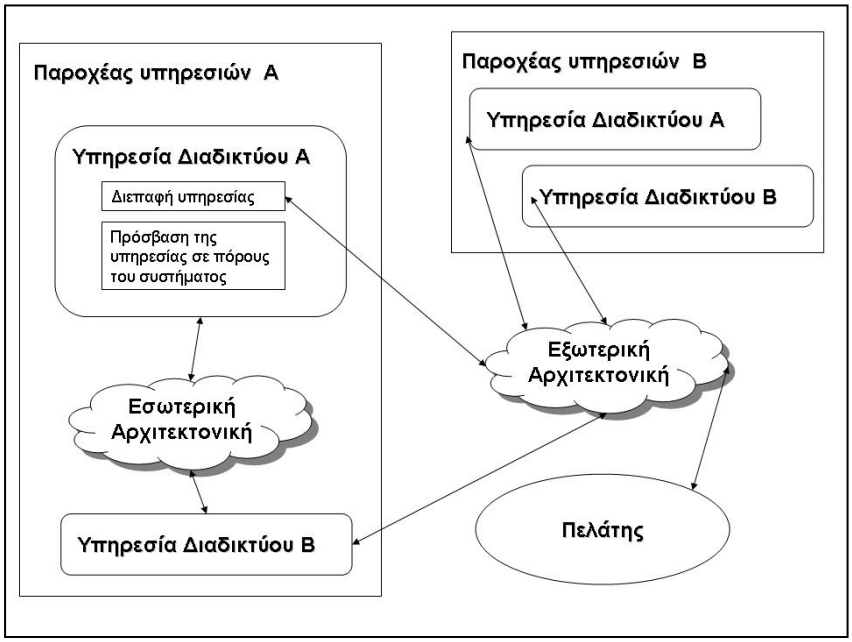
2.1.1. Βασικές Αρχές για τις Υπηρεσίες Διαδικτύου

Ορισμός

Με την ταχύτατη ανάπτυξη του παγκόσμιου ιστού, ο όρος Υπηρεσίες Διαδικτύου (Web services) συναντάται όλο και περισσότερο. Πολλοί θεωρούν ως υπηρεσία Διαδικτύου κάθε εφαρμογή στην οποία μπορεί κανείς να έχει πρόσβαση μέσω του παγκόσμιου ιστού. Με την έννοια αυτή, υπηρεσίες Διαδικτύου θεωρούνται ακόμη και τα CGI scripts τα οποία δεν είναι παρά διεπαφές που συνδέουν τις HTML φόρμες με εκτελέσιμα προγράμματα. Όμως, ο όρος υπηρεσίες Διαδικτύου δεν είναι τόσο γενικός και αναφέρεται σε εφαρμογές συγκεκριμένης τεχνολογίας. Έτσι όταν αναφερόμαστε σε μια υπηρεσία Διαδικτύου με βάση τον ορισμό που έδωσε το W3C (World Wide Web Consortium) αναφερόμαστε σε: «*μια εφαρμογή λογισμικού που προσδιορίζεται από ένα URI, και της οποίας οι διεπαφές είναι δυνατόν να προσδιοριστούν, να περιγραφούν και να βρεθούν ως XML έγγραφα. Μία υπηρεσία Διαδικτύου υποστηρίζει την άμεση επικοινωνία με άλλους πράκτορες λογισμικού οι οποίοι υποστηρίζουν την ανταλλαγή μηνυμάτων βασισμένων στην τεχνολογία XML μέσω ενός πρωτοκόλλου του Διαδικτύου»[13]*.

Αρχιτεκτονική

Όταν ασχολείται κανείς με υπηρεσίες Διαδικτύου ενδιαφέρεται για δυο βασικά πράγματα βάσει των οποίων χωρίζεται σε δύο μέρη και η αρχιτεκτονική ενός συστήματος που χρησιμοποιεί υπηρεσίες Διαδικτύου. Στο Σχήμα 2.1 φαίνεται πώς αλληλεπιδρούν οι υπηρεσίες με τα δύο αυτά μέρη αλλά και με υπηρεσίες ή πελάτες εκτός του τοπικού μηχανήματος,



Σχήμα 2.1: Σχηματική αναπαράσταση της αρχιτεκτονικής

Καταρχήν, οι υπηρεσίες Διαδικτύου είναι ένας τρόπος να δημοσιοποιηθούν οι εσωτερικές διαδικασίες έτσι ώστε να μπορούν να κληθούν και μέσω του παγκόσμιου ιστού. Έτσι, απαιτείται το σύστημα εκείνου που θέλει να παρέχει υπηρεσίες Διαδικτύου να είναι σε θέση να λάβει τα αιτήματα μέσω του παγκόσμιου ιστού και να τα κατανοήσει. Την δουλειά αυτή αναλαμβάνει το «internal middleware», την αρχιτεκτονική του οποίου αναφέρουμε ως «εσωτερική αρχιτεκτονική» [13] .

Εκτός όμως από αυτό, για να εργαστεί κάποιος με υπηρεσίες Διαδικτύου είναι απαραίτητο να έχει το υλικό και το λογισμικό που θα φροντίσει για την ενσωμάτωση (integration) διαφορετικών υπηρεσιών Διαδικτύου. Ένας πελάτης είναι πιθανόν να θέλει να χρησιμοποιήσει υπηρεσίες από διάφορους παροχείς και ίσως και να

συνδυάσει τα αποτελέσματα των υπηρεσιών αυτών. Για το σκοπό αυτό, φροντίζει η μονάδα της οποίας η αρχιτεκτονική θα αναφέρεται ως «εξωτερική αρχιτεκτονική».

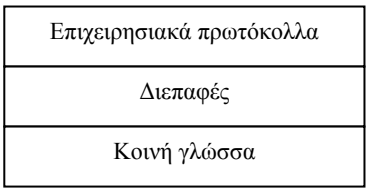
Η αρχιτεκτονική των υπηρεσιών Διαδικτύου παρέχει αρκετά πλεονεκτήματα τα οποία έχουν κάνει δημοφιλή τη χρήση αυτών των υπηρεσιών. Τέτοια πλεονεκτήματα είναι:

- ***η Διαλειτουργικότητα.*** Μία υπηρεσία Διαδικτύου παρέχει ανεξαρτησία τόσο από το λειτουργικό σύστημα όσο και από το υλικό. Οποιοδήποτε πρόγραμμα που συμβαδίζει με αυτή τη τεχνολογία μπορεί πολύ εύκολα να προσπελάσει μία τέτοια υπηρεσία.
- ***η Ενσωμάτωση***. Σε ένα υπάρχον λογισμικό σύστημα που λειτουργεί μέσα στο Internet η δημιουργία μιας υπηρεσίας Διαδικτύου δεν απαιτεί αλλαγές στον μηχανισμό του συστήματος.
- ***η Διαθεσιμότητα και δημοσίευση***. Οι πληροφορίες για τις υπηρεσίες Διαδικτύου δημοσιεύονται, οπότε η εύρεση και η χρήση τους μπορεί να είναι ταχύτατες.
- ***η Επέκταση***. Μία έτοιμη υπηρεσία Διαδικτύου είναι δυνατό να ανανεωθεί με εύκολο τρόπο παρέχοντας έτσι επιπρόσθετες υπηρεσίες στους χρήστες της.
- ***το Μικρό κόστος δημιουργίας και χρήσης***. Σε ένα λογισμικό σύστημα που υπάρχει ήδη κάποια διαδικασία που χρειάζεται να επεκταθεί σε on-line υπηρεσία, η δημιουργία της υπηρεσίας Διαδικτύου κοστίζει ελάχιστα. Επίσης, το κόστος ενσωμάτωσης μιας υπηρεσίας Διαδικτύου σε κάποιο **ΔΙΚΤΥΑΚΟ** τόπο ή σε δικτυακή εφαρμογή είναι πάρα πολύ μικρό. Ακόμα και στις περιπτώσεις που η χρήση κάποιας υπηρεσίας Διαδικτύου γίνεται με ενοικίαση σίγουρα το συνολικό κόστος της χρήσης είναι αρκετά πιο μικρό από το κόστος δημιουργίας της υπηρεσίας αυτής.
- ***η Χρήση λογισμικών συστημάτων***. Όλα τα λογισμικά συστήματα και ειδικότερα οι ιστοσελίδες που χρησιμοποιούν έτοιμες υπηρεσίες γίνονται πιο λειτουργικά και πιο φιλικά αφού παρέχουν περισσότερες υπηρεσίες στους χρήστες.

Τεχνολογίες Υπηρεσιών Διαδικτύου

Δουλεύοντας με υπηρεσίες Διαδικτύου είναι απαραίτητο να έχουμε ένα τρόπο να περιγράφουμε αυτές τις υπηρεσίες, όπως κάτι ανάλογο χρειαζόμαστε και όταν δουλεύουμε με συμβατικές υπηρεσίες για αυτό χρησιμοποιούμε τις IDL περιγραφές. Μια λογική σκέψη θα ήταν να χρησιμοποιούσαμε αυτού του τύπου τις περιγραφές για να περιγράψουμε και μια υπηρεσία Διαδικτύου. Όμως, οι περιγραφές αυτές δεν είναι αρκετές για να προσδιορίσουν μια υπηρεσία Διαδικτύου κι αυτό γιατί για τις συμβατικές υπηρεσίες η ίδια ομάδα προγραμματιστών προγραμμάτιζε τόσο τους εξυπηρετητές (υπηρεσίες) όσο και τους πελάτες. Επιπλέον, το περιβάλλον στο οποίο θα έτρεχαν οι εφαρμογές ήταν γνωστό κατά τη συγγραφή των κλασσικών υπηρεσιών και των πελατών τους και άρα οι προγραμματιστές το λάμβαναν υπόψη και αυτό [13]. Το Σχήμα 2.2 δείχνει μια στοίβα με βάση την οποία περιγράφεται μια υπηρεσία Διαδικτύου. Κάθε τι που βρίσκεται ψηλότερα στην στοίβα προσδιορίζει ή βελτιώνει την περιγραφή της υπηρεσίας που γίνεται στα χαμηλότερα επίπεδα. Η στοίβα αυτή περιγράφεται στη συνέχεια.

Κοινή γλώσσα. Σε πρώτη φάση το πρόβλημα που εξετάζεται είναι ο καθορισμός μιας κοινής μετα- γλώσσας που να μπορεί να χρησιμοποιηθεί ως βάση για όλες τις γλώσσες που περιγράφουν τις διαφορετικές πτυχές μιας υπηρεσίας. Η XML χρησιμοποιείται για αυτό το σκοπό, επειδή είναι ευρέως αποδεκτό πρότυπο και επειδή έχει μια σύνταξη αρκετά ευέλικτη να επιτρέψει τον καθορισμό τόσο των γλωσσών περιγραφής υπηρεσιών όσο και των πρωτοκόλλων.



Σχήμα 2.2: Ιεραρχία βάση της οποίας
περιγράφεται κάθε υπηρεσία Διαδικτύου

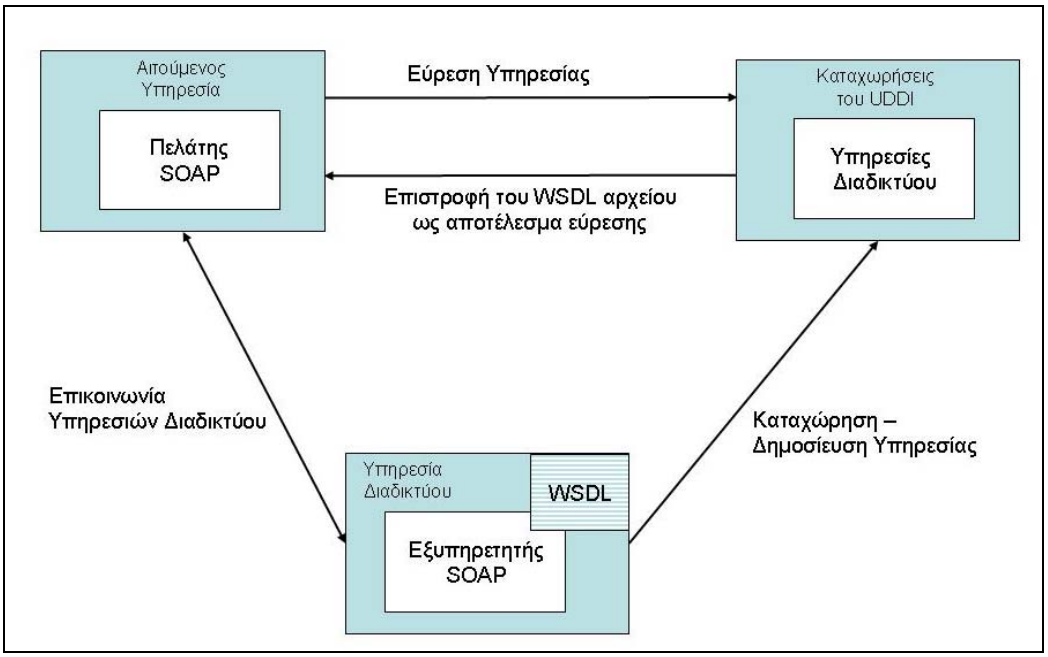
Ορισμός διεπαφών. Βάση για κάθε τι που χρησιμοποιεί την αντικειμενοστραφή λογική είναι μια γλώσσα ορισμού διεπαφών ανάμεσα στα διάφορα αντικείμενα. Οι διεπαφές των υπηρεσιών Διαδικτύου μοιάζουν με CORBA IDLs με τις διεπαφές δηλαδή που ορίζονται όταν χρησιμοποιώντας CORBA στήνουμε ένα κατανεμημένο σύστημα. Το γεγονός ότι μοιάζουν δεν σημαίνει ότι είναι και ίδιες, μερικές από τις διαφορές αυτές είναι η διαθεσιμότητα των διαφορετικών τρόπων επικοινωνίας καθώς επίσης και η χρήση του XML σχήματος. Ακόμη, όπως ήδη έχει αναφερθεί, οι περιγραφές των υπηρεσιών Διαδικτύου πρέπει να περιέχουν επιπλέον πληροφορίες. Είναι απαραίτητο, για παράδειγμα, για να υπάρχει δυνατότητα κλήσης μιας υπηρεσίας Διαδικτύου να διευκρινιστεί το URI της υπηρεσίας και το πρωτόκολλο μεταφοράς που θα χρησιμοποιεί. Η κυρίαρχη πρόταση για IDLs σε αυτή την περιοχή είναι η γλώσσα περιγραφής υπηρεσιών Διαδικτύου WSDL που περιγράφεται παρακάτω.

Επιχειρησιακά πρωτόκολλα. Μια υπηρεσία Διαδικτύου συχνά προσφέρει διάφορες διαδικασίες τις οποίες οι πελάτες πρέπει να καλέσουν με μια ορισμένη σειρά προκειμένου να επιτευχθούν οι στόχοι τους. Για παράδειγμα, για να γίνει μια προμήθεια (έστω ότι αυτή είναι η υπηρεσία Διαδικτύου), ο πελάτης θα πρέπει πρώτα να παραγγείλει τα αγαθά, και να κάνει τελικά μια πληρωμή. Τέτοιες ανταλλαγές μεταξύ των πελατών και των υπηρεσιών Διαδικτύου καλούνται συνομιλίες (conversations). Οι φορείς παροχής υπηρεσιών θέλουν να επιβάλουν τους κανόνες που διέπουν τη συνομιλία, δηλώνοντας ποιες συνομιλίες είναι έγκυρες και κατανοητές από την υπηρεσία. Αυτό το σύνολο κανόνων διευκρινίζεται στο επιχειρησιακό πρωτόκολλο. Προτείνονται διάφορες γλώσσες για την συγγραφή αυτών των επιχειρησιακών πρωτοκόλλων. Τέτοιες είναι οι WSCL και BPEL.

Βασικά πρωτόκολλα και γλώσσες των υπηρεσιών Διαδικτύου

SOAP (Simple Object Access Protocol): Το SOAP είναι ένα πρωτόκολλο που χρησιμοποιείται για την ανταλλαγή δομημένης πληροφορίας σε ένα αποκεντροποιημένο και κατανεμημένο περιβάλλον, όπως ο Παγκόσμιος Ιστός. Αυτό το πρωτόκολλο έρχεται να λύσει τα προβλήματα που δημιουργούν οι χαλαρά συνδεδεμένες επικοινωνίες, και η έλλειψη προτύπων στην ομαλή λειτουργία των υπηρεσιών Διαδικτύου. Πιο συγκεκριμένα, το SOAP χρησιμοποιώντας την XML

καθορίζει τον τρόπο οργάνωσης της πληροφορίας έτσι ώστε αυτή να μπορεί μεταφερθεί ανάμεσα στους κόμβους (peers) του Διαδικτύου. Αναλυτικότερα, το πρωτόκολλο SOAP καθορίζει τα ακόλουθα:



Σχήμα 2.3: Βασικές τεχνολογίες και η σχέση που τις συνδέει

- Μια μορφή μηνύματος (μορφή SOAP) για μια μονόπλευρη (one-way) επικοινωνία. Στη μορφή αυτή περιγράφεται πώς μπορεί η πληροφορία να συγκεντρωθεί σε ένα XML έγγραφο.
- Ένα σύνολο από συνθήκες για το πώς χρησιμοποιώντας SOAP μηνύματα μπορεί κανείς να υλοποιήσει απομακρυσμένη επικοινωνία τύπου RPC . Προσδιορίζει δηλαδή το πρωτόκολλο με βάση το οποίο οι πελάτες μπορούν να καλέσουν μια απομακρυσμένη διαδικασία στέλνοντας ένα μήνυμα της μορφής SOAP και πώς οι υπηρεσίες μπορούν να απαντήσουν στέλνοντας πίσω ένα άλλο μήνυμα της ίδιας μορφής.
- Ένα σύνολο από κανόνες τους οποίους πρέπει να ακολουθήσει κάθε οντότητα που επεξεργάζεται SOAP μηνύματα. Οι κανόνες αυτοί προσδιορίζουν τα XML στοιχεία που κάθε οντότητα πρέπει να διαβάζει

και να καταλαβαίνει καθώς και τις ενέργειες που πρέπει να κάνει μια οντότητα που δεν καταλαβαίνει το περιεχόμενο ενός μηνύματος.

- Μια περιγραφή του τρόπου με τον οποίο ένα μήνυμα τύπου SOAP πρέπει να μεταφερθεί πάνω από το HTTP ή το SMTP προς το παρόν, αλλά προβλέπεται και σύνδεση με αλλά πρωτόκολλα μεταφοράς στο μέλλον.

WSDL (Web Services Description Language) : Το WSDL είναι ένα XML schema, που αναπτύχθηκε από την Microsoft και την IBM με σκοπό να ορίσει το XML μήνυμα, τη λειτουργία και το πρωτόκολλο αντιστοίχησης μιας υπηρεσίας διαδικτύου που προσπελαύνεται χρησιμοποιώντας SOAP ή κάποιο άλλο XML πρωτόκολλο. Το συντακτικό του WSDL επιτρέπει τον αφαιρετικό ορισμό τόσο των μηνυμάτων όσο και των λειτουργιών των μηνυμάτων, έτσι ώστε να μπορούν να αντιστοιχηθούν σε πολλαπλές φυσικές υλοποιήσεις.

Ο ρόλος της WSDL είναι και πάλι παρόμοιος με τον ρόλο των IDLs στις συμβατικές υπηρεσίες. Μια βασική διαφορά είναι το γεγονός ότι εκτός από των προσδιορισμό των λειτουργιών που προσφέρει μια υπηρεσία, χρειάζεται η WSDL να είναι σε θέση να προσδιορίσει και τους μηχανισμούς που θα μας επιτρέψουν την πρόσβαση σε αυτήν. Στο περιβάλλον του παγκόσμιου ιστού σε κάθε υπηρεσία είναι δυνατόν να έχουμε πρόσβαση μέσω διαφορετικού πρωτοκόλλου, άλλωστε όπως αναφέρθηκε το SOAP υποστηρίζει πολλαπλά πρωτόκολλα μεταφοράς.

Επιπλέον, στην WSDL περιγραφή είναι απαραίτητο να προσδιοριστεί και η θέση της κάθε υπηρεσίας λόγω της κατανεμημένης φύσης του παγκόσμιου ιστού. Η γνώση της θέσης αυτής είναι απαραίτητη για να ξέρουμε πού θα στείλουμε το SOAP μήνυμα.

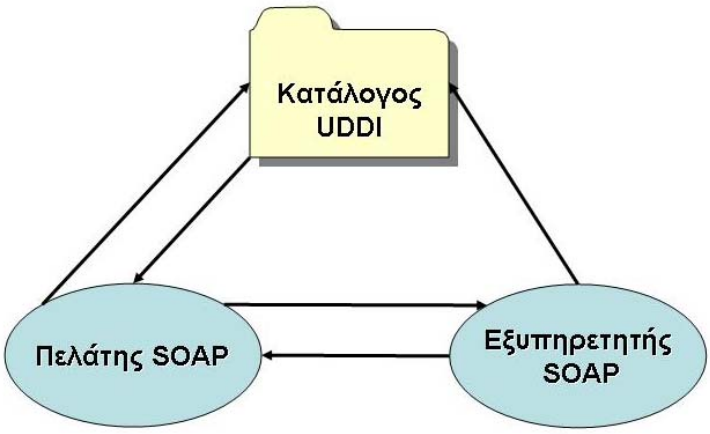
UDDI (Universal Description Discovery and Integration) : Το UDDI, ορίζει ένα μοντέλο δεδομένων (σε XML) και SOAP APIs για την καταχώρηση και αναζήτηση των υπηρεσιών Διαδικτύου και των φορέων που τις παρέχουν. Οι φορείς χρησιμοποιούν τα SOAP APIs για να καταχωρούν τις υπηρεσίες που παρέχουν στο UDDI. Ο πελάτες ψάχνουν στο UDDI όταν θέλουν να ανακαλύψουν μια υπηρεσία. Το UDDI μπορεί να παρέχει πληροφορία εύρεσης των υπηρεσιών, δίνοντας ουσιαστικά έναν «δείκτη» στο WSDL αρχείο που περιγράφει τις υπηρεσίες

Διαδικτύου που παρέχει ο συγκεκριμένος φορέας. Σε μια καταχώρηση του UDDI είναι δυνατόν να συναντήσουμε τρία είδη πληροφοριών:

1. Πληροφορίες που αφορούν στον οργανισμό που παρέχει την υπηρεσία
2. Πληροφορίες που αφορούν στον τρόπο κλήσης της πληροφορίας
3. Πληροφορίες για τις προδιαγραφές των υπηρεσιών, που περιγράφονται από διάφορες οντότητες

Στης συνέχεια, σκόπιμο θα ήταν να περιγραφεί λίγο ο τρόπος με τον οποίο οργανώνονται οι παραπάνω πληροφορίες με βάση το UDDI. Οι καταχωρήσεις του UDDI μπορούν να χωριστούν στις εξής τρεις κατηγορίες:

Λευκές σελίδες Πρόκειται για λίστες από οργανισμούς, στοιχεία οργανισμών όπως τηλέφωνα ή ηλεκτρονικές διευθύνσεις καθώς και υπηρεσίες που οι συγκεκριμένοι οργανισμοί προσφέρουν. Οι λευκές σελίδες μπορούν να χρησιμοποιηθούν σαν κατάλογοι εύρεσης υπηρεσιών.



Σχήμα 2.4: Σχηματικά ο ρόλος του UDDI

Κίτρινες σελίδες. Είναι ταξινομήσεις των οργανισμών και των υπηρεσιών Διαδικτύου με βάση προκαθορισμένες ή δηλωμένες από τον χρήστη ταξονομίες. Μέσω των κίτρινων σελίδων, είναι δυνατό να αναζητηθούν υπηρεσίες με βάση την κατηγορία στην οποία ανήκουν, σύμφωνα με ένα δεδομένο σχέδιο ταξινόμησης.

Πράσινες σελίδες. Αυτές οι σελίδες περιγράφουν πώς μια δεδομένη υπηρεσία Διαδικτύου μπορεί να κληθεί. Παρέχεται με τη βοήθεια δεικτών στα έγγραφα περιγραφής υπηρεσιών, που αποθηκεύονται έξω από την UDDI σελίδα.

Μέτρα για την ποιότητα απλών υπηρεσιών

Για την ίδια λειτουργία είναι δυνατόν να βρούμε στο Διαδίκτυο περισσότερες από μια υπηρεσίες. Υπάρχει τότε η ανάγκη επιλογής της υπηρεσίας εκείνης που θα χρησιμοποιήσουμε για την εκτέλεση της λειτουργίας. Θα επιθυμούσαμε να επιλέξουμε την καλύτερη υπηρεσία. Πρέπει όμως για να μπορεί να γίνει αυτό να οριστούν ποσοτικά μέτρα εκτίμησης της ποιότητας μιας υπηρεσίας. Παρακάτω αναφέρονται μέτρα που έχουν προταθεί για αυτό το σκοπό.

1) Κόστος εκτέλεσης. Δοθείσης μιας λειτουργίας op μίας υπηρεσίας s το κόστος εκτέλεσης $q_{price}(s,op)$ της είναι το χρηματικό πόσο που πρέπει να πλήρωση ο αιτών την εκτέλεση της λειτουργίας op. Οι παροχείς υπηρεσιών web είτε άμεσα ενημερώνουν για αυτό το κόστος είτε δίνουν τη δυνατότητα να ρωτήσει κανείς για αυτό.
2) Διάρκεια εκτέλεσης. Δοθείσης μιας λειτουργίας op μίας υπηρεσίας s η διάρκεία εκτέλεσης $q_{du}(s,op)$ μετράει, σε δευτερόλέπτα, την αναμενόμένη καθυστέρηση από τη στιγμή που στέλνεται η αίτηση μέχρι τη στιγμή που λαμβάνονται τα αποτελέσματα. Η διάρκεια εκτέλεσης υπολογίζεται ως εξής $q_{du}(s,op) = T_{process}(s,op) + T_{trans}(s,op)$, όπου $T_{process}$ ο χρόνος εκτέλεσης και T_{trans} ο χρόνος μεταφοράς. Για τον χρόνο εκτέλεσης οι υπηρεσίες web είτε άμεσα ενημερώνουν είτε δίνουν τη δυνατότητα να ρωτήσει κανείς για αυτόν. Ο χρόνος μεταφοράς από την άλλη εκτιμάται με βάση τις προηγούμενες εκτελέσεις των λειτουργιών της υπηρεσίας έτσι $T_{trans}(s,op) = \frac{\sum_{i=1}^{n} T_i(s,op)}{n}$ όπου $T_i(s,op)$ οι περασμένοι χρόνοι μεταφοράς και n ο αριθμός των παρατηρήσεων που έχουμε από άλλες φορές.
3) Αξιοπιστία. Η αξιοπιστία μιας υπηρεσίας s, $q_{rel}(s)$ είναι η πιθανότητα μια αίτηση να απαντηθεί σωστά μέσα στο αναμενόμενο χρονικό περιθώριο (το τελευταία υπάρχει στη τεχνική περιγραφή της υπηρεσίας). Η αξιοπιστία είναι μια παράμετρος που έχει να κάνει με την υλική και λογισμική τεκμηρίωση της υπηρεσίας καθώς και με την δικτυακή υποδομή ανάμεσα στον παροχέα της υπηρεσίας και αυτόν που την ζητάει. Η τιμή της αξιοπιστίας υπολογίζεται με

βάση παλιά δεδομένα με χρήση του τύπου $q_{rel}(s) = N_c(s)/K$ όπου $N_c(s)$ το πλήθος των φορών που η υπηρεσία s απαντήθηκε σωστά μέσα στο αναμενόμενο χρονικό περιθώριο και K το συνολικό πλήθος των φορών που κλήθηκε η S.

4) Διαθεσιμότητα. Η διαθεσιμότητα μιας υπηρεσίας s είναι η πιθανότητα η υπηρεσία να είναι διαθέσιμη. Η τιμή της υπολογίζεται με βάση τον ακόλουθο τύπο $q_{av}(s) = T_a(s)/\theta$ όπου T_a ο συνολικός χρόνος σε δευτερόλεπτα που η s ήταν διαθέσιμη τα τελευταία θ δευτερόλεπτα. Η τιμή του θ μπορεί να διαφέρει ανάλογα με την εφαρμογή.
5) Reputation μιας υπηρεσίας s είναι ένα μέτρο της «αξιοπιστίας» της υπηρεσίας. Βασικά εξαρτάται από την εμπειρία που έχει ο χρήστης με την υπηρεσία. Έτσι η τιμή της ορίζεται με βάση την βαθμολογία rank που της έδωσαν οι χρήστες. $Q_{rep}(s) = \frac{\sum_{i=1}^{n} R_i}{n}$ όπου R_i οι βαθμοί και n οι φορές που βαθμολογήθηκε η υπηρεσία.

2.2. Ροές Εργασίας

2.2.1. Ορισμός & Βασικές Έννοιες

Ορισμός

Υπάρχουν πολλοί και διαφορετικοί ορισμοί για τον όρο Ροή Εργασίας (workflow). Με τον ορό αυτό μπορεί κανείς να εννοεί μια επιχειρησιακή διαδικασία ή μια εφαρμογή λογισμικού που απλά υποστηρίζει τη συνεργασία πολλών ανθρώπων προκείμενου να επιτευχθεί ένας στόχος. Σύμφωνα πάντως με το πρότυπο WFMC (WorkFlow Management Coalition) Ροή Εργασίας είναι η απλοποίηση ή η αυτοματοποίηση μιας διαδικασίας ή μέρους αυτής, με χρήση υπολογιστών. Επομένως, μπορούμε να δούμε μια ροή εργασίας σα μια συλλογή από εργασίες οργανωμένες έτσι ώστε να ολοκληρώνεται μια διαδικασία. Επιπλέον, σε μια Ροή εργασίας ορίζεται η σειρά εκτέλεσης των εργασιών, οι συνθήκες κάτω από τις οποίες εκτελούνται, ποιες εργασίες είναι απαραίτητο να συγχρονιστούν μεταξύ τους και η ροή των δεδομένων ανάμεσα στις εργασίες. Παρακάτω περιγράφονται οι βασικές έννοιες που χρησιμοποιεί κανείς όταν δουλεύει με Ροές Δεδομένων.

Σχήμα 2.5: Διαχείριση Ροών Εργασίας

Βασικές Έννοιες

- Διαχείριση Ροής Εργασίας (Workflow management, WFM) είναι η τεχνολογία που υποστηρίζει τον επανασχεδιασμό των επιχειρησιακών και

πληροφοριακών διαδικασιών. Η Διαχείριση Ροής Εργασίας χωρίζεται σε δύο μέρη (Σχήμα 2.5):

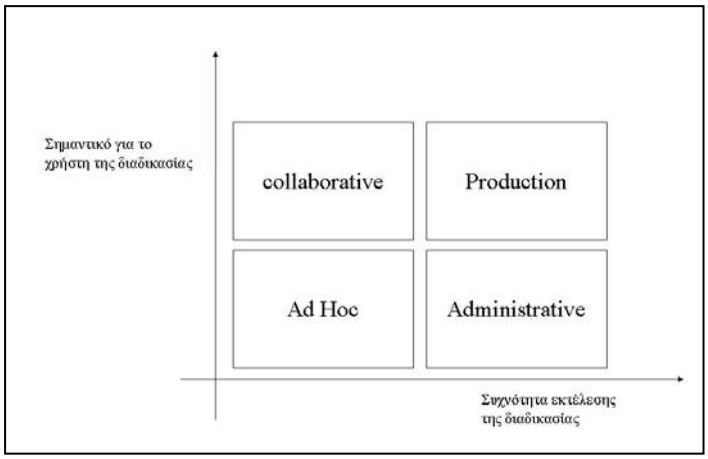
- ♦ Ορισμός των ροών εργασίας περιγράφοντας για παράδειγμα τις εργασίες που εμπλέκονται, τη σειρά εκτέλεσης τους και τους περιορισμούς κάτω από τους οποίους θα πρέπει να εκτελεστούν.
- ♦ Παροχή της δυνατότητας εκτέλεσης των ροών που σχεδιάστηκαν στο προηγούμενο μέρος.

- Σύστημα Διαχείρισης Ροών Εργασίας (WFMS) είναι το σύστημα που υποστηρίζει την τεχνολογία των Ροών Εργασίας.
- Μοντέλο Ροής Εργασίας είναι εκείνο το κομμάτι όπου ορίζεται η αρχή και το τέλος της διαδικασίας, οι συνθήκες κάτω από τις οποίες εκτελούνται οι εργασίες και η ροή των δεδομένων ανάμεσα σε αυτές.
- Δραστηριότητα ονομάζεται κάθε λογικό βήμα μέσα σε μια Ροή Εργασίας συμπεριλαμβανομένων και των συνθηκών κάτω από τις οποίες εκτελείται καθώς και εκείνων που εξασφαλίζουν την ορθή της εκτέλεση.
- Στιγμιότυπο διαδικασίας είναι μια συγκεκριμένη εκτέλεση μιας ροής εργασίας.
- Μηχανή ροών εργασίας είναι η διαδικασία που φροντίζει για την πραγματική εκτέλεση στη σωστή σειρά των διαφόρων εργασιών που απαρτίζουν μια ροή εργασίας.

2.2.2. Κατηγοριοποίηση των Ροών Εργασίας

Όλες οι ροές εργασίας δεν είναι όμοιες, είτε γιατί οι διαδικασίες που μοντελοποιούν δεν είναι όμοιες είτε γιατί οι τεχνολογίες με τις οποίες υλοποιούνται διαφέρουν. Στη παρούσα φάση, αφού αναφέρουμε γενικά τα κριτήρια με βάση τα οποία μπορούμε να κατατάξουμε σε κατηγορίες τις διάφορες ροές εργασίας, δίνουμε μια από τις προτεινόμενες κατηγοριοποιήσεις. Οι ροές εργασίας χρησιμοποιούνται για να περιγράψουν διαδικασίες πολύ διαφορετικές μεταξύ τους με πολύ διαφορετικές απαιτήσεις. Έτσι είναι δυνατόν να κατηγοριοποιήσουμε και τις ροές εργασίας [12] με βάση αυτές τις απαιτήσεις ή και με βάση τις απαιτήσεις του χρήστη από αυτές. Ένας άλλος τρόπος να γίνει η κατηγοριοποίηση των ροών εργασίας είναι ο τρόπος με τον

οποίο μοντελοποιούν σε ροές τις διάφορες εργασίες και ο τρόπος με τον οποίο ξεκινούν και ελέγχουν την εκτέλεση τους .



Σχήμα 2.6: Κατηγοριοποίηση GIGA

Μια καλή κατηγοριοποίηση των ροών εργασίας πρότεινε το GIGA Information Group. Η ομάδα αυτή, όπως φαίνεται και στο Σχήμα 2.6 , κατέταξε σε κατηγορίες τις ροές εργασίας με βάση το πόσο σημαντικές είναι αυτές για την επιχείρηση ή τον οργανισμό που τις χρησιμοποιεί αλλά και με βάση τη συχνότητα εκτέλεσης μιας διαδικασίας.

Αναλυτικότερα η κατηγοριοποίηση έχει ως εξής:

Κατηγορία Collaborative: περιλαμβάνει ροές εργασίας που η ύπαρξη τους είναι πολύ σημαντική για την εταιρία που χρησιμοποιεί τη διαδικασία που μοντελοποιούν, εκτελούνται λίγες μόνο φορές και η διαδικασία που υλοποιούν είναι αρκετά πολύπλοκη. Δημιουργούνται εξάλλου για την εκτέλεση της συγκεκριμένης και μόνο διαδικασίας.

Κατηγορία AD Hoc: περιλαμβάνει ροές εργασίας που η ύπαρξη τους δεν είναι και τόσο σημαντική για τη εταιρία και που επίσης δεν μοντελοποιούν διαδικασίες που επαναλαμβάνονται συχνά. Οι δραστηριότητες που περιλαμβάνουν τέτοιου τύπου ροές εργασίες συχνά χρειάζονται την ενεργή συμμετοχή του ανθρώπινου δυναμικού. Επιπλέον, τέτοιες ροές συχνά δημιουργούνται απλά και μόνο για να καλύψουν ανάγκες που έτυχε να προκύψουν και που δεν είναι μόνιμες ή προβλέψιμες.

Κατηγορία administrative: περιλαμβάνει ροές εργασίας που παρότι η ύπαρξη τους δεν είναι και τόσο σημαντική για τη εταιρία, η χρήση τους είναι παρά πολύ συχνή.

Κατηγορία Production: περιλαμβάνει ροές εργασίας που η ύπαρξη τους είναι πολύ σημαντική για την εταιρία και η χρήση τους είναι παρά πολύ συχνή.

2.3. Σύνθετες Υπηρεσίες Διαδικτύου

Η τεχνική της επαναχρησιμοποίησης κώδικα είναι ευρέως διαδεδομένη στο χώρο του προγραμματισμού. Η χρήση της παραπάνω τακτικής γλιτώνει τους προγραμματιστές από τον κόπο να προγραμματίζουν και να ξανά προγραμματίζουν τις ίδιες ρουτίνες και τους δίνει τη δυνατότητα να παράγουν εύκολα και γρήγορα κώδικα για περίπλοκες διαδικασίες χρησιμοποιώντας και συνθέτοντας κομμάτια ήδη έτοιμου κώδικα. Επιπλέον, ο κώδικας έτσι γίνεται πιο ευανάγνωστος και κατανοητός αφού είναι σε θέση κανείς να καταλάβει τι γίνεται χωρίς να πρέπει να εξετάσει με λεπτομέρεια τον κάθε αλγόριθμο που υλοποιεί μια απλή διαδικασία. Θα ήταν, λοιπόν, χρήσιμο να μπορούμε να εφαρμόσουμε την ίδια τακτική και με τις υπηρεσίες Διαδικτύου. Θα ήταν με άλλα λόγια χρήσιμο να μπορούσαμε να συνθέτουμε υπάρχουσες υπηρεσίες δημιουργώντας έτσι άλλες πιο πολύπλοκες. Για παράδειγμα, αν υποθέσουμε ότι θέλουμε να δημιουργήσουμε μια υπηρεσία μέσω της οποίας κάποιος θα είναι σε θέση να κανονίσει τις διακοπές του τότε θα ήταν καλό να μπορούσαμε να χρησιμοποιήσουμε ήδη έτοιμες υπηρεσίες για να κάνουμε κράτηση σε ξενοδοχεία, να κλείσουμε θέσεις σε αεροπλάνα ή λεωφορεία. Είναι λοιπόν χρήσιμο να μπορούμε να μοντελοποιούμε νέες υπηρεσίες διαδικτύου ως ροές εργασίας με δραστηριότητες τις ήδη υπάρχουσες υπηρεσίες.

2.3.1. Κύρια στοιχεία του υλικολογισμικού σύνθεσης υπηρεσιών Διαδικτύου

Τη σύνθεση υπηρεσιών πρέπει να υποστηρίξει ένα σύστημα middleware. Το σύστημα αυτό θα πρέπει να καθορίζει πώς θα γίνει η σύνθεση μιας υπηρεσίας Διαδικτύου με τις άλλες. Αντίστοιχα, το υλικολογισμικό σύνθεσης υπηρεσιών Ιστού περιλαμβάνει τις αφαιρέσεις και τα εργαλεία που διευκολύνουν τον καθορισμό και την εκτέλεση μιας σύνθετης υπηρεσίας επιτρέποντας στους υπεύθυνους για την ανάπτυξη, να στραφούν στην επιχειρησιακή λογική παρά στις χαμηλού επιπέδου λεπτομέρειες. Το παραπάνω middleware αποτελείται από:

- **Ένα πρότυπο και μια γλώσσα σύνθεσης**, που επιτρέπουν την σύνθεση των περιγραφών των υπηρεσιών, καθορίζουν τη σειρά εκτέλεσης των διαφόρων υπηρεσιών (με βάση ενδεχομένως τις συνθήκες που υπάρχουν κατά τον χρόνο εκτέλεσης), και ο τρόπος καθορισμού των παραμέτρων κλήσης των υπηρεσιών (δηλ., ο τρόπος με τον οποίο τα μηνύματα χτίζονται). Η περιγραφή μιας σύνθετης υπηρεσίας που εκφράζεται μέσω μιας γλώσσας σύνθεσης καλείται σχήμα σύνθεσης. Το σχήμα, επομένως, καθορίζει την επιχειρησιακή λογική μιας σύνθετης υπηρεσίας Διαδικτύου, και μπορεί να θεωρηθεί ως πρόγραμμα που γράφεται σε μια γλώσσα που σχεδιάζεται συγκεκριμένα για τη σύνθεση των υπηρεσιών middleware παρά σε μια συμβατική γλώσσα προγραμματισμού.
- **Ένα περιβάλλον ανάπτυξης,** με γραφική συνήθως διεπαφή χρήστη, μέσω της οποίας οι σχεδιαστές μπορούν να καθορίσουν ένα σχήμα σύνθεσης σέρνοντας τις διάφορες υπηρεσίες Διαδικτύου σε έναν καμβά και καθορίζοντας με διαγράμματα, κυρίως, τη σειρά κλήσης τους. Έπειτα, οι γραφικές παραστάσεις και άλλες περιγραφικές πληροφορίες μετατρέπονται σε κείμενο που περιγράφει τις συνθέτες υπηρεσίες Διαδικτύου.
- **Ένα περιβάλλον χρόνου εκτέλεσης**, συχνά αναφέρεται ως μηχανή σύνθεσης, η οποία εκτελεί την επιχειρησιακή λογική της σύνθετης υπηρεσίας Διαδικτύου με την κλήση των συστατικών υπηρεσιών όπως καθορίζονται στο σχήμα. Κάθε εκτέλεση μιας σύνθετης υπηρεσίας καλείται στιγμιότυπο (instance) της σύνθεσης.

Στη συνέχεια, αφού αναφερθούμε για λίγο γενικά στις δυνατότητες σύνθεσης των υπηρεσιών Διαδικτύου θα δούμε μερικά βασικά στοιχεία των πρότυπων, γλωσσών που έχουν αναπτυχθεί για τη σύνθεση αυτών των υπηρεσιών, και θα μελετήσουμε κάπως εκτενέστερα μια από αυτές.

2.3.2. Δυνατότητες σύνθεσης Υπηρεσιών Διαδικτύου

Στο σημείο αυτό, ας δούμε τι είναι αυτό που καθιστά τις υπηρεσίες Διαδικτύου διαφορετικές από την άποψη της σύνθεσης. Η κύρια διαφορά είναι ότι οι υπηρεσίες Διαδικτύου εμφανίζονται να είναι αυτάρκη συστατικά. Έχουν καθορισμένες με σαφήνεια διεπαφές, και η συμπεριφορά τους διευκρινίζεται ως τμήμα των

πληροφοριών που παρέχονται στα registries των υπηρεσιών Διαδικτύου. Επιπλέον, περιγράφονται χρησιμοποιώντας την ίδια γλώσσα (WSDL) και καλούνται με την αποστολή εγγράφων XML που ομαδοποιούνται σε SOAP μηνύματα. Με βάση τα προηγούμενα, θα λέγαμε ότι οι υπηρεσίες Διαδικτύου μπορούν να ταιριάξουν καλά στη σύνθεση. Αυτό είναι περισσότερο εμφανές αν εξετάσει κανείς ένα συμβατικό σύστημα για workflows και ένα σύστημα workflows για υπηρεσίες Διαδικτύου. Το πρώτο θα είναι αναγκασμένο να παρακολουθεί συνεχώς την ετερογένεια των συστατικών. Το τελευταίο, δεδομένου ότι το σύστημα σύνθεσης μπορεί να υποστηρίξει τα πρωτόκολλα αλληλεπίδρασης, μπορεί να δουλέψει με βάση τις τυποποιημένες διεπαφές και τα πρωτόκολλα και να παρέχει μια πολύ γρηγορότερη και αποδοτικότερη σύνθεση.

Ας τονίσουμε ότι η τυποποίηση δεν επιλύει αμέσως όλα τα προβλήματα διαλειτουργικότητας. Μια υπηρεσία βασισμένη σε RPC είναι ακόμα ασυμβίβαστη με μια υπηρεσία βασισμένη στις ουρές αναμονής, παρότι και οι δύο είναι υπηρεσίες Διαδικτύου. Εάν αυτές οι δύο υπηρεσίες πρέπει να συνδεθούν, οι προσπάθειες που απαιτούνται είναι οι ίδιες ανεξάρτητα από το αν τα συστατικά χρησιμοποιούν το υλικολογισμικό συμβατικών ή υπηρεσιών Διαδικτύου. Ένα πλεονέκτημα των προτύπων υπηρεσιών Διαδικτύου είναι ότι υποστηρίζουν μια κοινή γλώσσα για τα συστατικά που μπορούν να χρησιμοποιηθούν σε όλες τις πλατφόρμες. Αν και τα συστατικά μπορούν ακόμα να είναι μη συμβατά, τουλάχιστον δεν πρέπει να ανησυχούμε για τη διάλεκτο IDL που χρησιμοποιείται σε κάθε συστατικό ή για το πώς να φέρουμε μια κλήση σε ένα μακρινό σύστημα.

Οι υπηρεσίες Διαδικτύου υπόσχονται επίσης να μειώσουν σημαντικά το κόστος και την πολυπλοκότητα της ανάπτυξης τέτοιων συστημάτων, δεδομένου ότι πολλές ενότητες συστημάτων θα είναι παρόμοιες για όλες τις πλατφόρμες. Παραδείγματος χάριν, η προσφορά υπηρεσιών Διαδικτύου τυποποίησε τους μηχανισμούς επιλογής καταλόγου και υπηρεσιών (π.χ., UDDI) που μπορούν να υιοθετηθούν εύκολα (και να προσαρμοστεί εάν είναι απαραίτητο) από οποιοδήποτε σύστημα σύνθεσης. Ως εκ τούτου, η σύνθεση υπηρεσιών μπορεί να παρασχεθεί από ένα σχετικά ελαφρύ συστατικό όσον αφορά στις ροές εργασίας.

2.3.3. Πρότυπα και γλώσσες σύνθεσης

Πρόσφατα, διάφορες γλώσσες σύνθεσης έχουν αναπτυχθεί για τον καθορισμό των διαδικασιών ως σύνολα υπηρεσιών που εκτελούνται με έναν δομημένο τρόπο. Ο καθορισμός μιας διαδικασίας είναι η αναπαράσταση της σε μια μορφή που να υποστηρίζει τον αυτοματοποιημένο χειρισμό της, όπως η μοντελοποίηση με ένα σύστημα διαχείρισης ροών εργασίας. Για να καθορίσουμε μια διαδικασία καθορίζουμε ένα δίκτυο των δραστηριοτήτων που την αποτελούν και των σχέσεών τους καθώς επίσης και κριτήρια για την έναρξη και τη λήξη της διαδικασίας, και άλλες πληροφορίες, όπως ακριβώς και κατά την μοντελοποίηση μια ροής εργασίας.

Μια γλώσσα περιγραφής διαδικασιών παρέχει ένα περιβάλλον για μια πλούσια περιγραφή μιας διαδικασίας που μπορεί να χρησιμοποιηθεί:

- Σαν πρότυπο για τη δημιουργία και τον έλεγχο των περιπτώσεων εκείνης της διαδικασίας κατά τη διάρκεια της θέσπισης διαδικασίας.
- Για την προσομοίωση και την πρόβλεψη της συμπεριφοράς της διαδικασίας.
- Σαν βάση για τον έλεγχο και την ανάλυση των υπαρχόντων διαδικασιών.
- Για την τεκμηρίωση, την απεικόνιση, και τη διαχείριση γνώσης

Σε κάθε γλώσσα σύνθεσης μπορούμε και πρέπει να ορίσουμε τα παρακάτω. Ο τρόπος με τον οποίο ορίζονται οι έξι αυτοί παράγοντες μπορεί να διαφέρει από ροή εργασίας σε ροή εργασίας ανάλογα με τις ανάγκες.

1. Το πρότυπο Συστατικό το οποίο καθορίζει τη φύση των στοιχείων που συντίθενται.
2. Το πρότυπο οργάνωσης το οποίο καθορίζει τις γλώσσες που χρησιμοποιούνται για να καθορίσουν τη σειρά με την οποία οι υπηρεσίες πρόκειται να κληθούν. Τέτοια πρότυπα μπορεί να είναι: τα διαγράμματα δραστηριότητας, τα petri-nets, ο π - calculus, τα statecharts, οι ιεραρχίες δραστηριότητας, και η βασισμένη σε κανόνες οργάνωση.

3. Το πρότυπο δεδομένων και πρόσβασης πάνω στο οποίο καθορίζει πώς το στοιχείο διευκρινίζεται και πώς ανταλλάσσεται μεταξύ των συστατικών.
4. Το πρότυπο επιλογής υπηρεσιών το οποίο καθορίζει πώς η στατική ή η δυναμική σύνδεση πραγματοποιείται, δηλ., πώς μια συγκεκριμένη υπηρεσία επιλέγεται ως συστατικό μιας σύνθεσης.
5. Το πρότυπο Transactions το οποίο καθορίζει ποια είναι η σημασιολογία των συναλλαγών επιτρέπεται.
6. Το πρότυπο Χειρισμού εξαιρέσεων το οποίο καθορίζει πώς οι εξαιρετικές καταστάσεις που εμφανίζονται κατά τη διάρκεια της εκτέλεσης της σύνθετης υπηρεσίας μπορούν να αντιμετωπιστούν.

2.3.4. BPEL4WS

Παρακάτω περιγράφεται περιληπτικά μια από τις πολλές τέτοιες γλώσσες που έχουν αναπτυχθεί τελευταία, η BPEL4WS. Η Business Process Execution Language for Web Services (BPEL4WS) είναι μια γλώσσα περιγραφής διαδικασιών βασισμένη στα workflows. Η BPEL4WS εισάγει καταρχήν ένα σύνολο από βασικές δραστηριότητες (activities) για να ορίσει:

- την αλληλεπίδραση μεταξύ των εφαρμογών που συνθέτονται (invoke, reply, receive activities)
- την αναμονή για κάποιο χρονικό διάστημα (wait activity)
- την αντιγραφή των δεδομένων (assign activity)
- την εμφάνιση λάθους (throw activity)
- τον τερματισμό ολόκληρης της σύνθετής λειτουργίας (terminate activity)
- την αδρανή κατάσταση (wait activity)

Επιπλέον η BPEL4WS μας παρέχει ένα σύνολο από δομημένες δραστηριότητες (structured activities) κι έτσι μας δίνεται η δυνατότητα να ορίσουμε:

- διατεταγμένα σύνολα από δραστηριότητες (sequence activity)

- διαφορετική συμπεριφορά με βάση κάποιες συνθήκες (switch activity)
- βρόχους (while activity)
- εκτέλεσης ενός από περισσότερα εναλλακτικά μονοπάτια (pick activity)
- παράλληλη εκτέλεση κάποιων βημάτων (flow activity)

Επίσης, χρησιμοποιώντας την BPEL4WS έχουμε τη δυνατότητα να ορίσουμε διαχειριστές λαθών καθώς και compensation handlers. Στο παράρτημα, δίνεται ενδεικτικά και η γραμματική με βάση την οποία ορίζουμε μερικές από τις παραπάνω δραστηριότητες όπως αυτή περιγράφεται από την BPEL4WS.

Με τη χρήση των παραπάνω δραστηριοτήτων, και κυρίως με τη βοήθεια των τελευταίων (δομημένων δραστηριοτήτων) είναι φανερό ότι μπορούμε να συνθέσουμε μια διαδικασία με βάση κάποιες ήδη υπάρχουσες. Επιπλέον η BPEL4WS μας δίνει και την δομή που πρέπει να έχει μια νέα διαδικασία ορισμένη με τη βοήθεια της. Η δομή αυτή δίνεται αμέσως παρακάτω (Σχήμα 2.7).

Έτσι θα ορίζαμε μια νέα διαδικασία με όνομα MyProcess. Στη συνέχεια τη διαδικασία αυτή μπορούμε να την χρησιμοποιούμε ως ανεξάρτητη υπηρεσία αν την δηλώσουμε πρώτα ως τέτοια με χρήση του πρωτοκόλλου WSDL

Στη συνέχεια της εργασίας θα αναφερθούμε σε ροές εργασίας που αναπαρίστανται με γράφους και υποστηρίζουν μόνο διατεταγμένα σύνολα από δραστηριότητες, εκτέλεσης ενός από περισσότερα εναλλακτικά μονοπάτια (OR), παράλληλη εκτέλεση κάποιων βημάτων (AND). Αυτές οι ροές όπως ήδη φάνηκε, δεν αποτελούν παρά ένα υποσύνολο των ροών που μπορούν να οριστούν με μια γλώσσα σαν την BPEL4WS.

```
<process name=”MyProcess”>
        <partnerLinks>
        <! -- εδώ παραθέτουμε όλες τις υπηρεσίες που εμπλέκονται στη σύνθετη
        υπηρεσία που πάμε να ορίσουμε -->
        </partnerLinks>
        <variables>
        <! – εδώ προσδιορίζονται τα δεδομένα που μεταφέρονται κατά τη διάρκεια της
```

```
        διαδικασίας -->
        </variables>
        <correlationSets>
        <! – συχετιζει συνολα λειτουργιών με συγκεκριμένα στιγμιότυπα της υπηρεσίας -->
        </correlationSets>
        <fautHandlers>
        <! – λίστα στοιχείων που χρησιμευουν στον εντοπισμό των λαθών  -->
        </faultHandlers>
        <compensationHandler>
        <! – Προσδιορίζονται οι διαδικασίες ανατροπής μιας σειράς δραστηριοτήτων σε
        περίπτωση ανίχνευσης σφάλματος-->
        </compensationHandler>
        <eventHandlers>
        <! – Λαμβάνει γεγονότα που προέρχονται από το περιβάλλον έξω από το
        workflow  -->
        </eventHandlers>
        <sequence>
        <! – Η λογική εκτέλεση του workfow  -->
        </sequence>
</process>
```

Σχήμα 2.7: Δομή διαδικασίας ορισμένης με BPEL4WS

ΚΕΦΑΛΑΙΟ 3. ΣΧΕΤΙΚΗ ΕΡΕΥΝΑ

Στο κεφάλαιο αυτό θα παρουσιάσουμε με συντομία την υπάρχουσα δουλειά που έχει γίνει στα πλαίσια των ροών εργασίας και υπηρεσιών Διαδικτύου. Αναφέρουμε ότι οι εργασίες που παρουσιάζονται σε αυτό το κεφάλαιο δεν αφορούν όλες στην επιλογή ποιοτικών σύνθετων υπηρεσιών. Αφορούν όμως προβλήματα βελτιστοποίησης στο χώρο των συνθετών υπηρεσιών Διαδικτύου. Στη συνέχεια αφού αναφερθούμε γενικά στη σχετική εργασία παρουσιάζουμε εκτενέστερα το Self-serv ένα πλαίσιο για την αποτελεσματική σύνθεση υπηρεσιών Διαδικτύου κι αυτό γιατί η εργασία αυτή πλησιάζει περισσότερο τη δική μας εργασία. Άλλωστε η μελέτη αυτής της εργασίας οδήγησε και στην ιδέα για τη δική μας μελέτη.

3.1. Ερευνητικά Θέματα στις Υπηρεσίες Διαδικτύου

Σε προηγούμενο κεφάλαιο περιγράφηκαν οι βασικές αρχές που διέπουν τη τεχνολογία των υπηρεσιών Διαδικτύου. Στο παρόν κεφάλαιο θα δοθούν οι βασικές κατευθύνσεις προς τις οποίες κινείται η έρευνα γύρω από αυτήν την τεχνολογία.

Ξεκινώντας, κρίνεται σκόπιμο να δοθεί μια εκτίμηση για τον λόγο που οδήγησε στην ενασχόληση με αυτή την τεχνολογία τόσων ερευνητών. Η ανάπτυξη της τεχνολογίας των υπηρεσιών Διαδικτύου έδωσε νέα ώθηση στον τομέα των ηλεκτρονικών επιχειρήσεων. Άλλωστε η ιδέα της ηλεκτρονικής συνεργασίας προϋπήρχε και με τις B2B εφαρμογές. Σε αυτού του τύπου τη συνεργασία η ανάπτυξη των υπηρεσιών Διαδικτύου έδωσε τη δυνατότητα επέκτασης αφού με τη νέα αυτή τεχνολογία όλο και περισσότεροι μπορούν να επωφεληθούν των ευεργεσιών της συνεργασίας αφού το μόνο που προϋποθέτει είναι η δημοσίευση των υπηρεσιών που προσφέρει μια επιχείρηση μέσω των καθορισμένων από την τεχνολογία τρόπων (περιγραφή της υπηρεσίας σε ένα WSDL αρχείο και δημοσίευση της στο UDDI). Η χρησιμότητα αυτής της νέας τεχνολογίας οδήγησε την έρευνα προς:

1. Τη δημιουργία γλωσσών είτε για την εύρεση υπηρεσιών με δεδομένα χαρακτηριστικά μέσα από των όγκο των υπηρεσιών που δηλώνονται στο UDDI
2. Τη δημιουργία γλωσσών σύνθεσης υπηρεσιών Διαδικτύου ή ακόμη και πλαισίων σύνθεσης και εκτέλεσης υπηρεσιών Διαδικτύου. Τον καθορισμό της ποιότητας των υπηρεσιών Διαδικτύου απλών ή και σύνθετων
3. Την εξοικονόμηση κατά το δυνατόν των πόρων του δικτύου με την ταυτόχρονη ποιοτική λειτουργία των υπηρεσιών.

3.1.1. Εύρεση Υπηρεσιών Διαδικτύου

Στα πλαίσια του πρώτου στόχου αναπτύχθηκε η XSRL [2] μια γλώσσα που οι κατασκευαστές της θέλησαν να μπορεί να χρησιμοποιείται για την εύρεση υπηρεσιών ανεξάρτητα από τον τύπο ή τις υπηρεσίες που θα προσέφεραν. Επιπλέον, με τη χρήση αυτής της γλώσσας το ερώτημα που θα τεθεί μπορεί να αφορά τόσο τις λειτουργίες που θέλουμε να επιτελεί η υπηρεσία που αναζητούμε όσο και την ποιότητα της ζητούμενης υπηρεσίας. Είναι επίσης δυνατόν να οριστούν εναλλακτικοί στόχοι για την υπηρεσία ανάλογα με το άγνωστο εκ των προτέρων αποτέλεσμα ενδιάμεσων σταδίων εκτέλεσης της υπηρεσίας. Επιπλέον, η XSRL φιλοδοξεί να δώσει στους χρήστες μια γλώσσα αρκετά «ευέλικτη» που να τους επιτρέπει να εκφράζουν ερωτήματα με κάποια αοριστία. Δίνει λοιπόν καταρχήν, τη δυνατότητα ορισμού μιας σειράς από εναλλακτικές επιθυμίες του χρήστη. Για παράδειγμα, είναι δυνατόν κανείς να ορίσει ότι προτιμά μία λειτουργία περισσότερο από μια άλλη που όμως θα την προτιμούσε από το να μην ικανοποιηθεί καθόλου το αίτημα του. Ενώ ταυτόχρονα δίνει τη δυνατότητα χρήσης τελεστών ομοιότητας έτσι ώστε ο χρήστης να μπορεί να ζητά μια υπηρεσία που θα του επιστρέψει κάτι παρόμοιο με αυτό που δίνει σαν είσοδο ή μια υπηρεσία που θα εκτελεί μια λειτουργία παρόμοια με αυτήν που ο χρήστης περιγράφει ή ακόμα και μια υπηρεσία παρόμοια με μια άλλη που έχει ήδη εκτελεστεί επιτυχώς.

Επίσης κινούμενοι προς τον πρώτο στόχο άλλοι ερευνητές [3] πρότειναν την χρήση οντολογιών ή και ταξονομιών για τη σημασιολογική περιγραφή των υπηρεσιών Διαδικτύου. Έτσι αναπτύχθηκαν διάφορες ταξονομίες όπως η NAICS και η ISO3166. Μια ταξονομία δεν ορίζει παρά μια ιεραρχία κλάσεων. Στην περιγραφή λοιπόν μιας υπηρεσίας είναι δυνατόν αν έχουμε ορίσει μια ταξονομία να

συμπεριλάβουμε και την υποκλάση στην οποία ανήκει η συγκεκριμένη υπηρεσία. Το πρόβλημα με τις ταξονομίες είναι ότι επειδή δεν είναι παρά ιεραρχίες δεν είναι δυνατόν να περιγράψουν παρά μόνο πολύ γενικά μια υπηρεσία. Ψάχνοντας λοιπόν με βάση την κλάση που ανήκει μια υπηρεσία είναι πιθανόν να μας επιστραφούν υπηρεσίες άσχετες ουσιαστικά μεταξύ τους ή ακόμη και υπηρεσίες που δεν μπορούν να χρησιμοποιηθούν παρά μόνο κάτω από κάποιους περιορισμούς. Δεν μας επιτρέπει λοιπόν η χρήση ταξονομιών την αυτόματη εύρεση της κατάλληλης για μια εργασία υπηρεσίας Διαδικτύου. Σε αντίθεση με τις ταξονομίες μια οντολογία είναι σε θέση να προσδιορίσει πολύ καλύτερα μια υπηρεσία Διαδικτύου. Για να ορίσουμε μια οντολογία για την περιγραφή υπηρεσιών Διαδικτύου μπορούμε να χρησιμοποιήσουμε την DAML+OIL μια γλώσσα στην οποία έχει ήδη οριστεί μια οντολογία για την βασική περιγραφή μιας υπηρεσίας Διαδικτύου και επιπλέον μας δίνει τη δυνατότητα να προσθέσουμε και νέα στοιχεία. Ένα επίσης θετικό χαρακτηριστικό των οντολογιών που αναπτύσσονται σε DAML+OIL είναι ότι οι περιγραφές υπηρεσιών που προκύπτουν βάση των οντολογιών αυτών ενσωματώνονται εύκολα στις καταχωρήσεις των υπηρεσιών στο UDDI οπού δηλαδή θα ψάξει κανείς για να εντοπίσει την υπηρεσία. Η σημασιολογική περιγραφή των υπηρεσιών του Διαδικτύου αλλά και των δεδομένων που υπάρχουν στον παγκόσμιο ιστό, με γλώσσες όπως η DAML+OIL και η δυνατότητα δημιουργίας και εκτέλεσης υπηρεσιών Διαδικτύου που θα ανασύρουν δεδομένα με βάση τη σημασιολογία τους μπορεί να βοηθήσει σημαντικά και στην αναζήτηση δεδομένων αν βασιστεί πάνω σε αυτές τις τεχνολογίες η κατασκευή νέων μηχανών αναζήτησης.

3.1.2. Σύνθεση υπηρεσιών

Με βάση το δεύτερο στόχο έρευνας στην τεχνολογία των υπηρεσιών Διαδικτύου αναπτύχθηκαν πλαίσια σύνθεσης και εκτέλεσης υπηρεσιών όπως το VISPO, το Commerce One Conductor και το Self-Serv. επίσης, για να μπορούν να οριστούν ροές που να περιλαμβάνουν και υπηρεσίες Διαδικτύου εκτός από απλές εφαρμογές επεκτάθηκαν ήδη υπάρχουσες γλώσσες όπως η WEBML αλλά και πλατφόρμες όπως η WLI. Στις αμέσως επόμενες παραγράφους θα αναφερθούμε περιληπτικά στην WLI και στη WEBML, ενώ για το Self-Serv υπάρχει, όπως ήδη αναφέρθηκε, εκτενέστερη αναφορά στο τέλος του κεφαλαίου.

Το VISPO [5] έχει ως στόχο τη δημιουργία ενός περιβάλλοντος όπου θα είναι δυνατή η αυτόματη σύνθεση υπηρεσιών Διαδικτύου. Στα πλαίσια όμως αυτά, έχει υλοποιήσει και ένα τρόπο για την εύρεση των κατάλληλων υπηρεσιών για μια λειτουργία (βλ. τρίτη μονάδα). Το πλαίσιο σύνθεσης και εκτέλεσης υπηρεσιών VISPO αποτελείται από τέσσερις μονάδες. Στη πρώτη μονάδα, κάθε υπηρεσία Διαδικτύου περιγράφεται με βάση το UDDI ενώ επίσης περιγράφεται ο τρόπος κλήσης των υπηρεσιών και τα δυνατά σενάρια σύνθεσης υπηρεσιών. Η δεύτερη μονάδα δεν είναι παρά η μηχανή εκτέλεσης των ροών εργασίας που περιγράφουν τις σύνθετες υπηρεσίες και βρίσκονται στο πρώτο μέρος. Είναι σημαντικό το γεγονός ότι στην ουσία μια τέτοια μηχανή υπάρχει για κάθε παροχεα υπηρεσιών. Η τρίτη μονάδα αναλαμβάνει την εύρεση, ανάμεσα στις υπηρεσίες πού περιγράφονται στη πρώτη μονάδα, εκείνων των υπηρεσιών που επιτελούν μια συγκεκριμένη λειτουργία ή τουλάχιστον παρόμοιες λειτουργίες. Τέλος, η τέταρτη μονάδα αναλαμβάνει την επιλογή και κλήση μιας υπηρεσίας από τις διαθέσιμες για την εκτέλεση τελικά της λειτουργίας.

Το Commerce One Conductor [4] είναι ένα πλαίσιο σύνθεσης υπηρεσιών που αναπτύχθηκε για ένα πιο ειδικό σκοπό: για να μειώσει τα προβλήματα των συναλλαγών των εταιρειών που χρησιμοποιούσαν για την παροχή υπηρεσιών ηλεκτρονικού εμπορίου την πλατφόρμα με το όνομα MarketSite. Παρόλα αυτά, το σύστημα αυτό μοιάζει αρκετά με τις υπηρεσίες που παρέχει με το VISPO. Αποτελείται κι αυτό από τέσσερις βασικές μονάδες. Στην πρώτη, δημοσιεύονται όλες οι υπηρεσίες. Η δεύτερη, διαχειρίζεται τις καταχωρημένες υπηρεσίες έτσι ώστε να μην έχουν πρόσβαση σε αυτές μη εξουσιοδοτημένοι χρήστες (υπηρεσίες). Η τρίτη επιτρέπει τη δημιουργία νέων υπηρεσιών συνθέτοντας τις ήδη υπάρχουσες. Στη συγκεκριμένη εργασία, η έμφαση δόθηκε στην ασφάλεια και την αξιοπιστία οι οποίες για να εξασφαλίσουν δεν αποτελούν πρώτη προτεραιότητα για τα πρωτόκολλα που διέπουν την τεχνολογία των υπηρεσιών Διαδικτύου όπως το SOAP ή το WSDL κι αυτό για λόγους απλότητας. Για αυτό και οι καταχωρήσεις για κάθε υπηρεσία στο σύστημα αυτό είναι αυξημένες σε σχέση με μία απλή UDDI καταχώρηση. Τις πληροφορίες αυτές σχετικά με την ασφάλεια συγκεντρώνει η τελευταία μονάδα του συστήματος.

Επίσης, για την επίτευξη του δευτέρου στόχου επεκτάθηκε το WLI. Πιο συγκεκριμένα, στην έκδοση WLI 8.0 που δεν είναι παρά μια πλατφόρμα για την δημιουργία αλλά και εκτέλεση επιχειρησιακών εφαρμογών σαν βάση για την

επικοινωνία των εφαρμογών, τίθενται το SOAP και η XML ενώ η ίδια η πλατφόρμα διευκολύνει τους χρήστες να φτιάξουν τις δικές τους υπηρεσίες Διαδικτύου παράγοντας J2EE κώδικα με γραφικό τρόπο. Επιπλέον, δίνει τη δυνατότητα ορισμού ροών εργασίας με χρήση των δομών που προσφέρει η JWF.

3.1.3. Εξοικονόμηση πόρων

Σε άλλη εργασία, οι Jingwen Jin και Klara Nahrstedt βλέπουν το θέμα της σύνθεσης υπηρεσιών από μια άλλη σκοπιά με στόχο την ποιοτική εκτέλεση των υπηρεσιών εξοικονομώντας όμως πόρους για το δίκτυο με την μείωση κατά το δυνατών του αριθμού των εκτελέσεων μιας υπηρεσίας.. Η ιδέα πίσω από τη δική τους εργασία είναι ότι κατά την εκτέλεση σύνθετων υπηρεσιών εκτελούνται διάφορες ακολουθίες από υπηρεσίες μέρος των οποίων τα αποτελέσματα χρειάζονται και άλλοι χρήστες εκτός από εκείνον που ζήτησε «πρώτος» τη σύνθετη υπηρεσία. Έτσι εξοικονομούνται πόροι του συστήματος αφού μειώνεται το πλήθος των εκτελέσεων των διαφόρων υπηρεσιών. Το παράδειγμα που αναφέρεται είναι η μορφοποίηση ενός βίντεο για τη λήψη του από διαφορετικούς δέκτες. Η μορφοποίηση αυτή απαιτεί μια σειρά από ενέργειες οι οποίες αν και εξαρτώνται από τον τύπο του δέκτη δεν είναι όλες διαφορετικές ανάλογα με το δέκτη. Η ιδέα λοιπόν είναι καταρχήν να κάνουμε όσο το δυνατόν περισσότερες ενέργειες μία φορά και στη συνέχεια να διαχωρίσουμε τα μονοπάτια της εκτέλεσης των ενεργειών στο σημείο που αυτό είναι απαραίτητο.

Ορίζεται λοιπόν, στη συγκεκριμένη εργασία καταρχήν ένας τρόπος για τη δρομολόγηση εκτέλεσης μιας σύνθετης υπηρεσίας όταν αυτή έχει μοναδικό αποδεκτή. Επειδή στο διαδίκτυο είναι δυνατόν πολλοί να παρέχουν την ίδια απλή υπηρεσία (μέρος της σύνθετης) δεν είναι προφανές ποιες υπηρεσίες θα εκτελεστούν όταν ζητήσουμε την εκτέλεση μιας σύνθετης. Προϋποτίθεται ότι ο ορισμός της σύνθετης έγινε με βάση τις ενέργειες που πρέπει να εκτελεστούν και όχι με βάση συγκεκριμένες υπηρεσίες Διαδικτύου. Ο πιο απλοϊκός τρόπος είναι να οριστεί κάποιο κριτήριο κόστους και με βάση αυτό να επιλέγεται κάθε φόρα η καλύτερη υπηρεσία. Το αρνητικό αυτής της προσέγγισης είναι ότι δεν εγγυάται την ποιότητα της σύνθετης υπηρεσίας αλλά τοπικά κάθε φορά την ποιότητα της απλής υπηρεσίας. Το πρόβλημα αυτό μπορεί να ξεπεραστεί αν κάθε φορά επιλέγουμε την επόμενη προς εκτέλεση απλή υπηρεσία με βάση ολόκληρη την ακολουθία από εκεί και πέρα. Ορίζεται δηλαδή, ένας γράφος με εναλλακτικά μονοπάτια και εφαρμόζοντας σε

αυτόν ένα αλγόριθμο εύρεσης ελάχιστου μονοπατιού βρίσκουμε την επόμενη υπηρεσία. Ο λόγος που δεν επιλέγεται με αυτό τον τρόπο το σύνολο των προς εκτέλεση υπηρεσιών αλλά μόνο η επόμενη είναι ότι δεν είμαστε σίγουροι ότι όλες οι υπηρεσίες θα παραμείνουν ενεργές μέχρι να τις χρειαστούμε. Στη συνέχεια ορίζεται το πώς θα μπορέσουν πολλαπλοί χρήστες να εκμεταλλευτούν τις εκτελούμενες υπηρεσίες ουσιαστικά ορίζονται νέοι γράφοι με βάση τα κομμάτια των υπηρεσιών που ζητάει ο κάθε χρήστης και επίσης με βάση τη θέση του κάθε χρήστη. Για την εξοικονόμηση μάλιστα ακόμη περισσότερων δικτυακών πόρων οι συγγραφείς προτείνουν οι χρήστες που τελικά λαμβάνουν τα ίδια δεδομένα από την εκτέλεση μιας υπηρεσίας να μην τα λαμβάνουν όλοι κατευθείαν από τον παροχέα της υπηρεσίας αλλά να τα λαμβάνει ένας και να τα μοιράζονται μετά οι χρήστες μεταξύ τους.

3.2. Self-serv: ένα σύστημα σύνθεσης υπηρεσιών

Με τη σύνθεση υπηρεσιών Διαδικτύου ασχολήθηκε εκτεταμένα ο Boualem Benatallah με τους συνεργάτες του. Η ομάδα αυτή υλοποίησε ένα πλαίσιο για την αποτελεσματική σύνθεση υπηρεσιών Διαδικτύου το οποίο ονομάζεται Self-Serv [7],[8],[14], και χρησιμοποιεί τα διαγράμματα καταστάσεων ως γλώσσα για τη σύνθεση των διάφορων υπηρεσιών Διαδικτύου. Κατάσταση ενός διαγράμματος ορίζεται να είναι μια απλή ή και μια συνθέτη υπηρεσία.

Στο σημείο αυτό κάνοντας μια παρένθεση αναφέρουμε ότι τα διαγράμματα καταστάσεων προτάθηκαν και χρησιμοποιήθηκαν για την μοντελοποίηση μεγάλων, σύνθετων ή και αλληλεπιδραστικών συστημάτων πριν την ανάπτυξη του Διαδικτύου. Αυτό το είδος διαγραμμάτων δεν είναι παρά κατευθυνόμενοι γράφοι των οποίων οι κόμβοι συμβολίζουν καταστάσεις και οι ακμές τις μεταβάσεις από την μια κατάσταση στην άλλη. Επιπλέον, μπορούμε χρησιμοποιώντας αυτά τα διαγράμματα να εμφωλεύσουμε καταστάσεις την μια μέσα στην άλλη. Έτσι είναι δυνατόν να μελετήσουμε ένα σύστημα σε βάθος ή και πιο επιφανειακά αναλύοντας ή όχι τα υποσυστήματα του. Μας δίνεται επίσης η δυνατότητα να ορίσουμε μια κατάσταση ως το αποτέλεσμα των πράξεων AND και XOR ανάμεσα σε περισσότερες καταστάσεις,

Το Self-Serv, λοιπόν, αποτελεί ένα περιβάλλον που υποστηρίζει τη σύνθεση web υπηρεσιών. Το περιβάλλον αυτό στοχεύει στη γρήγορη δημιουργία υπηρεσιών των οποίων η λειτουργία βασίζεται σε άλλες προϋπάρχουσες υπηρεσίες με έναν εύκολο τρόπο ο οποίος να επιδέχεται και κλιμάκωση. Στόχος, δηλαδή είναι η εύκολη σύνθεση τόσο ενός μικρού (σε αριθμό) συνόλου υπηρεσιών όσο κι ενός μεγαλύτερου. Για τη λειτουργία του το Self-Serv απαιτεί την ύπαρξη μιας προγραμματιστικής διεπαφής βασισμένης στο SOAP και στο WSDL για κάθε μια από τις διαθέσιμες υπηρεσίες. Δύο είναι οι βασικές έννοιες πάνω στις οποίες βασίζεται ο μηχανισμός σύνθεσης υπηρεσιών του Self-Serv, η έννοια της σύνθετης υπηρεσίας (composite service) και η έννοια του συλλογέα υπηρεσιών (container). Παρακάτω περιγράφουμε αναλυτικότερα αυτές τις δύο έννοιες.

Σύνθετες υπηρεσίες, ονομάζονται οι υπηρεσίες εκείνες που λειτουργούν μαζεύοντας διάφορες άλλες υπηρεσίες (απλές ή σύνθετες) και επιτυγχάνουν κατά κάποιο τρόπο τη συνεργασία ανάμεσα τους, για την επίτευξη ενός συγκεκριμένου στόχου. Στο Self-Serv η σειρά εκτέλεσης των διαφόρων υπηρεσιών που αποτελούν μια σύνθετη υπηρεσία, καθώς και η ύπαρξη διαφορετικών ροών εκτέλεσης ανάλογα με το αποτέλεσμα μιας από τις υπηρεσίες που συνεισφέρουν στη σύνθετη υπηρεσία εκφράζεται με χρήση διαγραμμάτων καταστάσεων (State charts). Πιο συγκεκριμένα, στο Self-Serv οι καταστάσεις αντιστοιχούν στις εκτελέσεις συγκεκριμένων υπηρεσιών web και οι ακμές ορίζουν την αλληλουχία αυτών των υπηρεσιών. Σημειώνουμε επίσης ότι χρησιμοποιώντας Data mapping καθορίζεται και η ροή των δεδομένων ανάμεσα στις υπηρεσίες.

Όπως προαναφέρθηκε, στο Self-Serv έκτος από τις σύνθετες υπηρεσίες ορίζονται επίσης και οι συλλογείς υπηρεσιών (containers ή communities). Οι συλλογείς υπηρεσιών δεν είναι παρά ένας τρόπος ομαδοποίησης των υπηρεσιών που υπάρχουν στο σύστημα (πιθανώς παρέχονται από διαφορετικούς κόμβους του συστήματος) για την εκτέλεση της ίδιας ή παρόμοιας διαδικασίας κάτω από μια κοινή διεπαφή (interface). Έτσι, καλώντας τον συλλογέα ενεργοποιείται ανάλογα με την περίσταση η καταλληλότερη υπηρεσία. Γίνεται επομένως φανερό ότι για να ορίσουμε πλήρως ένα τέτοιο συλλογέα πρέπει να ορίσουμε τις υπηρεσίες που συμπεριλαμβάνει καθώς επίσης και τη διαδικασία με βάση την οποία επιλέγεται μια από αυτές για να εκτελεστεί όταν κληθεί ο συλλογέας.

Στο Self-Serv ορίζεται κι ένας τρόπος δρομολόγησης της εκτέλεσης των σύνθετων υπηρεσιών χωρίς την ύπαρξη κεντρικοποιημένου ελέγχου. Αναφέρουμε απλά εδώ ότι αυτή η δρομολόγηση βασίζεται στους συντονιστές καταστάσεων, το Self-Serv δημιουργεί έναν τέτοιο για κάθε κατάσταση και σε δυο πίνακες δρομολόγησης έναν με τις συνθήκες που οδηγούν σε μια κατάσταση και ένα με εκείνες που οδηγούν στην επόμενη κατάσταση. Οι πίνακες αυτοί, δημιουργούνται με δεδομένο το διάγραμμα καταστάσεων που περιγράφει την υπηρεσία.

Το πιο ενδιαφέρον όμως στα πλαίσια αυτής της εργασίας είναι ότι για το self-serv αναπτύχθηκε και ένας αλγόριθμος για την επιλογή εκτέλεσης της καταλληλότερης, όπως χαρακτηρίζεται, με βάση την ποιότητα υπηρεσίας για κάθε βήμα μιας σύνθετης υπηρεσίας. Για το σκοπό αυτό ορίζονται καταρχήν τα μέτρα με βάση τα οποία κρίνεται η ποιότητα των σύνθετων υπηρεσιών που γίνεται με βάση τους συγγραφείς το κόστος εκτέλεσης, τη διάρκεια εκτέλεσης, την αξιοπιστία, τη διαθεσιμότητα και το reputation. Τα μέτρα αυτά παρουσιάστηκαν αναλυτικά στο Κεφάλαιο 1. Τέλος, κάνοντας χρήση γραμμικών μεθόδων βελτιστοποίησης επιλέγεται το καλύτερο ως προς το κόστος πλάνο εκτέλεσης της σύνθετης υπηρεσίας. Αξιοσημείωτο είναι ότι υπάρχουν πολλά πλάνα εκτέλεσης για διαφορετικές ροές του ίδιου workflow.

Πιο συγκεκριμένα, στη συγκεκριμένη δημοσίευση αναφέρεται ότι δημιουργείται, ένας ακυκλικός γράφος για κάθε εναλλακτική ροή στο διάγραμμα καταστάσεων που αντιπροσωπεύει τη σύνθετη υπηρεσία ο οποίος αναφέρεται ως μονοπάτι εκτέλεσης. Ένα πλάνο εκτέλεσης ορίζεται στη συνέχεια για κάθε τέτοιο μονοπάτι και ορίζει ποια υπηρεσία θα ενεργοποιηθεί για την εκτέλεση καθεμιάς από τις εργασίες του γράφου. Έτσι αντιστοιχίζονται πολλά πλάνα σε κάθε μονοπάτι εκτέλεσης. Στη δημοσίευση δίνονται αυστηροί ορισμοί των μονοπατιών εκτέλεσης καθώς και των πλάνων εκτέλεσης

Προκειμένου να ελαχιστοποιηθεί το κόστος μιας σύνθετης υπηρεσίας υπάρχουν με βάση πάντα τη δημοσίευση, δύο τρόποι:

1. Η εξαντλητική προσέγγιση που υπολογίζει το κόστος κάθε δυνατού πλάνου εκτέλεσης και επιλέγει τελικά το καλύτερο πλάνο εκτέλεσης
2. Η προσέγγιση με τη βοήθεια του γραμμικού προγραμματισμού

Η πρώτη λύση έχει πολύ μεγάλη χρονική πολυπλοκότητα για αυτό επιλέγεται από τους συγγραφείς η δεύτερη, η οποία υλοποιείται με τη βοήθεια LP solver που είναι διαθέσιμοι στο Internet και υλοποιούν αλγορίθμους όπως ο Simplex. Βεβαία για να οριστεί ένα πρόβλημα γραμμικού προγραμματισμού είναι απαραίτητο να οριστούν η συνάρτηση για ελαχιστοποίηση κα οι περιορισμοί του προβλήματος. Αυτά ορίζονται από τους συγγραφείς όπως φαίνεται αμέσως παρακάτω στους Πίνακες 3.1 και 3.2.

Συνάρτηση για μεγιστοποίηση
$\sum_{l=0}^{2}\left(\frac{Q_l^{\max}-Q_{i,l}}{Q_l^{\max}-Q_l^{\min}}*W_l\right)+\sum_{l=3}^{5}\left(\frac{Q_{i,l}-Q_l^{\min}}{Q_l^{\max}-Q_l^{\min}}*W_l\right)$

Πίνακας 3.1

Περιορισμοί			
1.	$\sum_{i\in S_j} y_{ij}=1,\forall j\in A$		Μόνο μια από τις υποψήφιες υπηρεσίες θα εκτελέσει την j task
2.	$\sum_{i\in S_j} p_{ij}y_{ij}=p_j,\forall j\in A$		Η διάρκεια εκτέλεσης της task t_j είναι ίση με τον χρόνο εκτέλεσης της από μια υπηρεσία
3.	$x_k-(p_j+x_j)\geq 0,\forall t_j\rightarrow t_k,j,k\in A$		Αν η t_k ακολουθεί άμεσα την t_j, τότε η t_k δεν μπορει να ξεκινήσει αν δεν ολοκληρωθεί η t_j
4.	$Q_{du}-(x_j+p_j)\geq 0,\forall j\in A$		Η εκτέλεση μιας σύνθετης υπηρεσίας ολοκληρώνεται μόνο αν ολοκληρωθούν αυτές από τις οποίες συντίθεται
5.	$Q_{du}=\sum_{j\in A}\sum_{i\in S_j} p_{ij}z_{ij}$		Η διάρκεια εκτέλεσης της υπηρεσίας θα ισούται με την διάρκεια εκτέλεσης των κρίσιμων υπηρεσιών
6.	α	$Q_{price}=\sum_{j\in A}\sum_{j\in S_j} c_{ij}y_{ij}$	Το συνολικό κόστος ισούται με το άθροισμα των τιμών του κόστους των υπηρεσιών που συμμετέχουν
	β	$\sum_{j\in A}\sum_{j\in S_j} c_{ij}y_{ij}\leq B, B>0$	
7.	$Q_{rep}=\sum_{j\in A}\sum_{j\in S_j} r_{ij}y_{ij}$		
8.	$Q'_{rel}=\sum_{j\in A}\sum_{i\in S_j}\alpha_{ij}z_{ij}\quad Q'_{rel}=\ln(Q_{rel})$		
9.	$Q'_{av}=\sum_{j\in A}\sum_{i\in S_j} b_{ij}z_{ij}\quad Q'_{av}=\ln(Q_{av})$		

Πίνακας 3.2

Q	Το διάνυσμα κόστους
W	Ορός για την κανονικοποιήση
A	Όλες οι tasks
y_{ij}	Συμμετοχή της υπηρεσίας WEB S_{ij} στο επιλεγμένο πλάνο
x_j	Ο αρχικός χρόνος εκκίνησης της task t_j
p_j	Η διάρκεια εκτέλεσης της task t_j
p_{ij}	Η διάρκεια εκτέλεσης της task t_j από την υπηρεσία S_{ij}
z_{ij}	Δείχνει αν η S_{ij} είναι κρίσιμη υπηρεσία
c_{ij}	Το κόστος της υπηρεσίας S_{ij}
α_{ij}	Η αξιοπιστία της υπηρεσίας S_{ij}
b_{ij}	Η διαθεσιμότητα της υπηρεσίας S_{ij}

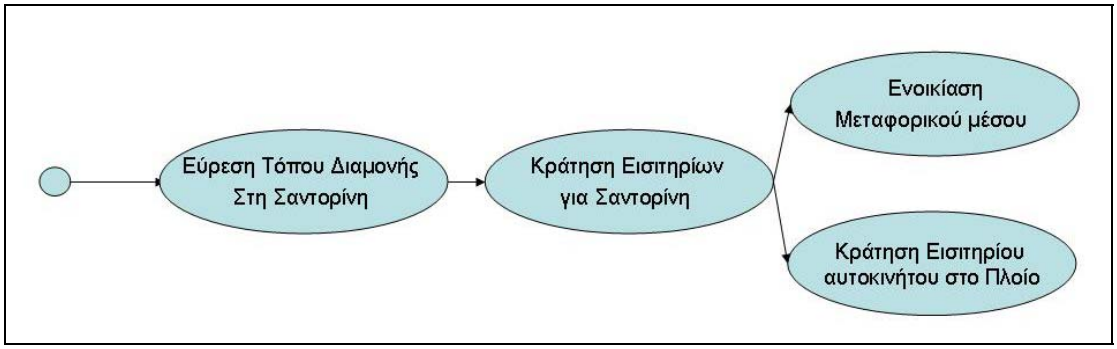
Πίνακας 3.3: Εξήγηση συμβόλων των προηγούμενων πινάκων

ΚΕΦΑΛΑΙΟ 4. ΠΡΟΤΕΙΝΟΜΕΝΟΙ ΑΛΓΟΡΙΘΜΟΙ

Στο κεφάλαιο αυτό, αφού πρώτα παρουσιαστεί το πρόβλημα το οποίο μελετά η παρούσα εργασία, παρουσιάζονται τρεις αλγόριθμοι. Ο πρώτος είναι ένας κλασσικός αλγόριθμος γραφημάτων, ενώ οι άλλοι δύο αναπτυχθήκαν στα πλαίσια της εργασίας για να υπολογιστεί το καλύτερο κόστος μιας συνθέτης υπηρεσίας Διαδικτύου που ορίζεται από μια ροή εργασίας. Οι ροές αυτές έχουν αναπαρασταθεί ως ειδικού τύπου κατευθυνόμενα γραφήματα τα οποία ορίζονται επίσης εδώ.

4.1. Ελαχιστοποίηση κόστους σύνθετων υπηρεσιών

Σε προηγούμενο κεφάλαιο αναφέραμε ότι κάθε Υπηρεσία Διαδικτύου έχει ένα κόστος. Επίσης επισημάνθηκε ότι το κόστος μιας τέτοιας υπηρεσίας δεν είναι κατ' ανάγκη χρηματικό και δόθηκαν οι ορισμοί με βάση τους οποίους το κόστος μια υπηρεσίας Διαδικτύου μπορεί να μετρηθεί λαμβάνοντας υπόψη το κόστος εκτέλεσης, τη διάρκεια εκτέλεσης, την αξιοπιστία, τη διαθεσιμότητα, το κόστος που επιφέρει η επικοινωνία και το reputation. Επίσης, στο ίδιο κεφάλαιο αναφέρθηκε η δυνατότητα ορισμού σύνθετων υπηρεσιών διαδικτύου ως ροών εργασίας όπου κάθε εργασία δεν θα είναι παρά μια από τις ήδη υπάρχουσες υπηρεσίες. Το πρόβλημα που έρχεται να αντιμετωπίσει η παρούσα εργασία, είναι αυτό της επιλογής και εκτέλεσης των καλύτερων με βάση το κόστος υπηρεσιών της ροής εργασίας έτσι ώστε η συνολική υπηρεσία να είναι συμφέρουσα. Υπενθυμίζεται ότι σε μια ροή εργασίας μπορούμε να έχουμε επιλογές στον τρόπο με τον οποίο θα πετύχουμε τον ίδιο στόχο. Έτσι πιθανώς το επιθυμητό αποτέλεσμα να μπορούμε να το πετύχουμε επιλέγοντας διαφορετικά μονοπάτια στη ροή εργασίας και επομένως εκτελώντας περισσότερες ή λιγότερες υπηρεσίες Διαδικτύου. Οι επιλογές αυτές κάνουν πολύπλοκη την επιλογή των εργασιών με το μικρότερο κόστος.



Σχήμα 4.1: Οργάνωση ταξιδιού στη Σαντορίνη

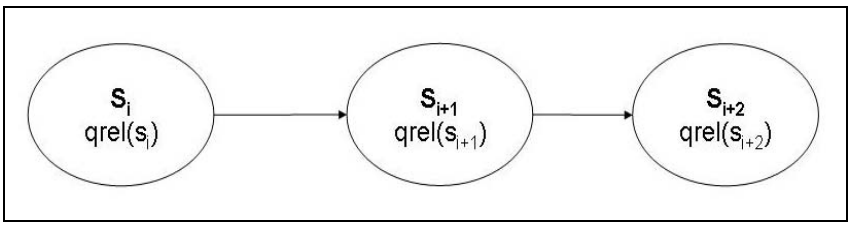
Αρχικά όμως, ας δούμε πως μπορούμε να ορίσουμε το κόστος μιας σύνθετης υπηρεσίας. Αν θεωρήσουμε μια απλή ακολουθία υπηρεσιών διαδικτύου και θέλουμε να υπολογίσουμε το κόστος της με βάση κάποιο από τα κριτήρια κόστους που αναφέραμε προηγουμένως, τότε το κόστος αυτό θα είναι μια συνάθροιση των τιμών του κόστους των υπηρεσιών που αποτελούν την ακολουθία. Για παράδειγμα στο Σχήμα 4.1 αν θα θέλαμε να υπολογίσουμε το κόστος, (με την γνωστότερη σε μας μορφή του, αυτή του χρηματικού κόστους) για την οργάνωση ενός ταξιδιού στη Σαντορίνη θα προσθέταμε το κόστος για την εύρεση τόπου διαμονής με το κόστος για την κράτηση εισιτηρίων και τέλος αφού αποφασίζαμε αν θα πάρουμε μαζί μας το δικό μας αυτοκίνητο ή αν θα νοικιάζαμε κάποιο μέσο μεταφοράς στη Σαντορίνη θα προσθέταμε στο παραπάνω ποσό και το αντίστοιχο κόστος. Όμοια θα πρέπει να κινηθούμε και για οποιαδήποτε άλλη μορφή κόστους υπηρεσίας Διαδικτύου πάνω σε μια ροή εργασίας που θα απεικονίζει μια σύνθετη υπηρεσία Διαδικτύου. Στον Πίνακα 4.1 φαίνονται οι συναθροιστικές συναρτήσεις για αυτόν υπολογισμό όπως έχουν ήδη προταθεί για το περιβάλλον του Self Serv [14].

Μέτρο	**Συναθροιστική συνάρτηση**
Κόστος εκτέλεσης	$Q_{price}(p) = \sum_{i=1}^{N} q_{price}(s_i, op_i)$
Διάρκεια εκτέλεσης	$Q_{du}(p) = CPA(q_{du}(s_1, op_1), ..., q_{du}(s_N, op_N))$
Reputation	$Q_{rep}(p) = \frac{1}{N} \sum_{i=1}^{N} q_{rep}(s_i)$

Αξιοπιστία	$Q_{rel}(p) = \prod_{i=1}^{N} (e^{q_{rel}(s_i)*z_i})$
Διαθεσιμότητα	$Q_{av}(p) = \prod_{i=1}^{N} (e^{q_{av}(s_i)*z_i})$

Πίνακας 4.1: Μέτρα ποιότητας σύνθετων υπηρεσιών

Παρατηρούμε ότι όσον αφορά στο κόστος εκτέλεσης καθώς και στο Reputation το κόστος της σύνθετης υπηρεσίας υπολογίζεται αθροιστικά με βάση το κόστος των απλών υπηρεσιών που την συνθέτουν. Για τον υπολογισμό της διαρκείας εκτέλεσης της σύνθετης υπηρεσίας χρησιμοποιείται ο αλγόριθμος CPA (Critical Path Algorithm) με βάση τον οποίο η διάρκεια εκτέλεσης της σύνθετης υπηρεσίας καθορίζεται από το μακρύτερο ως προς την διάρκεια μονοπάτι που εκτελείται κατά την εκτέλεση της σύνθετης υπηρεσίας. Η διάρκεια υπολογίζεται αθροιστικά αλλά μόνο σε αυτό το μονοπάτι. Η εκτέλεση μιας σύνθετης υπηρεσίας μπορεί να περιλαμβάνει την ταυτόχρονη εκτέλεση διαφορετικών μονοπατιών και όπως είναι φυσικό μόνο το μακρύτερο μονοπάτι τελικά καθορίζει την διάρκεια εκτέλεσης της σύνθετης υπηρεσίας. Τέλος το μέτρο της διαθεσιμότητας και της αξιοπιστίας δεν είναι αθροιστικό, για το λόγο αυτό αν θέλουμε να τα χειριστούμε αθροιστικά δεν μετράμε στην τιμή του κόστους την πραγματική τιμή των μέτρων του πίνακα 4.1, αλλά το λογάριθμο ln της τιμής αυτής. Πιο συγκεκριμένα $q_{rel}(s_i)$ είναι η αξιοπιστία στην εκτέλεση της υπηρεσίας s_i. Το $q_{rel}(s_i)$ αντιστοιχεί με βάση τον ορισμό της αξιοπιστίας σε μια πιθανότητα δηλαδή σε έναν αριθμό ανάμεσα στο 0 και το 1.



Σχήμα 4.2: Ένα μονοπάτι από τρεις υπηρεσίες

Έτσι με βάση τη θεωρία των πιθανοτήτων η συνολική πιθανότητα για μια ακολουθία υπηρεσιών θα είναι το γινόμενο των πιθανοτήτων για κάθε υπηρεσία χωριστά

$Q_{rel} = \prod_{i=1}^{N} q_{rel}(s_i)$. Έτσι για παράδειγμα η αξιοπιστία του μονοπατιού που φαίνεται στο Σχήμα 4.2 θα περιμέναμε να είναι: qrel(si)* qrel(si_{+1})* qrel(si_{+2}). Όμως στη δημοσίευση όπου ορίζονται τα μέτρα των σύνθετων υπηρεσιών έχουμε $\prod_{i=1}^{N} (e^{q_{rel}(s_i)*z_i})$ αντί $\prod_{i=1}^{N} q_{rel}(s_i)$. Αντί δηλαδή να πάρουμε το γινόμενο των πιθανοτήτων παίρνουμε το γινόμενο του e υψωμένου στην πιθανότητα (το z_i ξάπλα εκφράζει την συμμετοχή ή μη της υπηρεσίας s_i στις υπηρεσίες που επιλέγονται να εκτελεστούν όταν εκτελείται η σύνθετη υπηρεσία). Η αξιοπιστία ορίζεται με αυτόν τον τρόπο για να είναι δυνατός στη συνέχεια ο χειρισμός της σαν να ήταν ένα αθροιστικό μέτρο. Στη συνέχεια λοιπόν μπορούμε να πάρουμε το λογάριθμο της παραπάνω σχέσης και να έχουμε:

$$\ln(Q_{rel}) = \ln(\prod_{i=1}^{N} (e^{q_{rel}(s_i)*z_i})) = \sum_{i=1}^{n} \ln(e^{q_{rel}(s_i)*z_i}) = \sum_{i=1}^{n} e^{q_{rel}(s_i)*z_i}$$

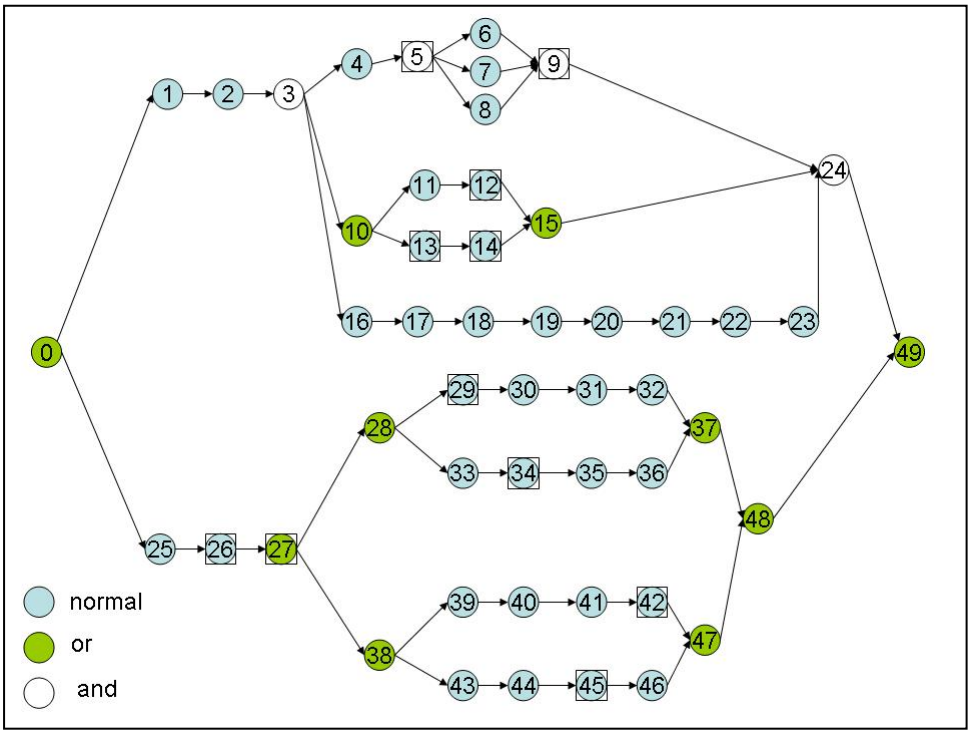
Αν τώρα αντί τη πραγματική τιμή της αξιοπιστία υπολογίζουμε την τιμή της $Q'_{rel} = \ln(Q_{rel})$ η τιμή της οποίας αυξάνει με την αύξηση της τιμής της πραγματικής αξιοπιστίας έχω μετατρέψει και την αξιοπιστία σε ένα αθροιστικό μέτρο. Με όμοιο ακριβώς τρόπο μπορεί να μετατραπεί σε αθροιστικό μέτρο και η διαθεσιμότητα. Επομένως μπορούμε να θεωρούμε ότι το κόστος μιας σύνθετης υπηρεσίας είναι τελικά αθροιστικό. Σημειώνουμε ότι για αυτό το λόγο στην προσομοίωση που υλοποιήσαμε θεωρήσαμε το κόστος ως ένα απλό νούμερο και αγνοήσαμε για λόγους απλότητας το διάνυσμα του κόστους που περιγράφεται εδώ. Στην περίπτωση πραγματικής εφαρμογής των αλγορίθμων, τόσο αυτού που εδώ προτείνεται όσο και του Bellman – Ford, η εκτέλεση του θα πρέπει να γίνει για κάθε μια από τις συνιστώσες του διανύσματος και εφαρμόζεται για όλες άνετα εκτός από τη διάρκεια εκτέλεσης όπου απαιτείται επιπλέον εργασία για τον εντοπισμό εργασιών που εκτελούνται παράλληλα.

4.2. Ορισμός Γράφου

Για την εκτέλεση των αλγορίθμων δρομολόγησης μοντελοποιήσαμε τις ροές εργασίας με κατευθυνόμενους γράφους. Θεωρήσαμε ότι οι γράφοι που ορίζουν τις σύνθετες

υπηρεσίες είναι κατευθυνόμενοι γράφοι με κόμβους που ανήκουν σε 5 κατηγορίες οι οποίες περιγράφονται παρακάτω.

Κόμβοι τύπου And: ένας τέτοιος κόμβος ακολουθείται από περισσότερους από ένα κόμβους και δηλώνει την αρχή κάποιων ανεξάρτητων μονοπατιών. Με δεδομένο ότι οι κόμβοι στη δική μας περίπτωση αναπαριστούν υπηρεσίες Διαδικτύου ένας κόμβος



Σχήμα 4.3: Παράδειγμα γράφου σύνθετης υπηρεσίας

αυτής της κατηγορίας ορίζει ότι αμέσως μετά πρέπει να εκτελεστούν όλες οι υπηρεσίες που αντιπροσωπεύονται από τους επόμενους κόμβους.

Κόμβοι τύπου end_of_And: ένας τέτοιος κόμβος ορίζει το τέλος, την συγχώνευση των ανεξάρτητων μονοπατιών που ορίστηκαν από ένα κόμβο τύπου And. Στην πράξη όταν συναντήσουμε ένα τέτοιο κόμβο, πρέπει να φροντίσουμε για τον συγχρονισμό όλων των μονοπατιών που καταλήγουν σε αυτόν. Δεν πρέπει δηλαδή να αφήσουμε να ξεκινήσει η επόμενη υπηρεσία αν όλες όσες φτάνουν στο συγκεκριμένο κόμβο δεν ολοκληρώσουν τη λειτουργία τους.

Κόμβοι τύπου Or: ένας τέτοιος κόμβος ακολουθείται από περισσότερους από ένα κόμβους και δηλώνει και αυτός, την αρχή κάποιων ανεξάρτητων μονοπατιών. Με δεδομένο ότι οι κόμβοι στη δική μας περίπτωση αναπαριστούν web services ένας κόμβος αυτής της κατηγορίας ορίζει ότι μετά πρέπει να εκτελεστεί μόνο ένα από τα μονοπάτια που ξεκινούν από τον κόμβο αυτού του τύπου.

Κόμβοι τύπου end_of_Or: ένας τέτοιος κόμβος ορίζει το τέλος, την συγχώνευση των ανεξάρτητων μονοπατιών που ορίστηκαν από ένα κόμβο τύπου Or. Οι κόμβοι αυτού του τύπου δεν έχουν ουσιαστική λειτουργία αλλά χρησιμοποιούνται μόνο για την καλύτερη απεικόνιση των ροών εργασίας.

Κάθε κόμβος έχει ένα κόστος το οποίο αντιπροσωπεύει το κόστος της υπηρεσίας που αντιπροσωπεύει ο κόμβος. Επιπλέον υπάρχει ένας μόνο αρχικός καθώς και ένας μόνο τελικός κόμβος σε κάθε γράφο. Η κατασκευή των γράφων με αυτό τον τρόπο κάνει ευκολότερους τους υπολογισμούς χωρίς να περιορίζει καθόλου τους γράφους στους οποίους δουλεύουμε αφού ένας αρχικός και ένας τελικός κόμβος μπορεί να προστεθεί εύκολα σε κάθε γράφο με διαφορετική μορφή.

Όπως έγινε πλέον φανερό οι γράφοι που χρησιμοποιούμε υποστηρίζουν μόνο εκτέλεση διατεταγμένων συνόλων από δραστηριότητες, εκτέλεση ενός από περισσότερα εναλλακτικά μονοπάτια (OR), παράλληλη εκτέλεση κάποιων βημάτων (AND). Αυτές αποτελούν υποσύνολο των ροών που μπορούν να οριστούν με μια γλώσσα σαν την BPEL4WS και επομένως κάθε σύνθετη υπηρεσία που ορίζεται με ένα γράφο μπορεί να οριστεί και με χρήση της BPEL4WS. Στην επόμενη παράγραφο (4.3) θα φανεί ποια είναι ακριβώς η σχέση των κόμβων συζητήθηκαν παραπάνω με τις δομημένες δραστηριότητες της BPEL4WS.

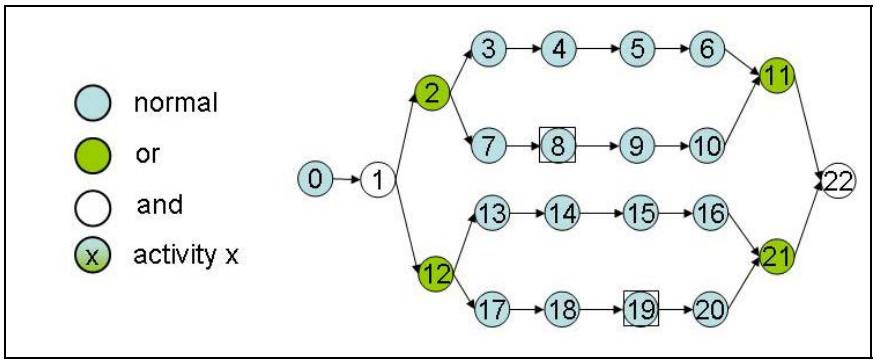
Πρώτα όμως ας αναφερθούμε λίγο στα πιθανά σενάρια σύνθεσης υπηρεσιών Διαδικτύου που μοιάζει να μην μπορούν να υλοποιηθούν χρησιμοποιώντας τους γράφους που ορίστηκαν εδώ.

Δεν προβλέφθηκαν στη δημιουργία των γράφων αυτών περιπτώσεις υπό συνθήκη μετάβασης σε κάποια υπηρεσία Η περίπτωση αυτή είναι παρά πολύ συχνή στις ροές εργασίας και στη δημιουργία ροών εργασίας για υπηρεσίες διαδικτύου. Η λύση που δίνεται για τη γραφική απεικόνιση των συγκεκριμένων περιπτώσεων να είναι η σύνδεση των ακμών των γράφων με κανόνες που έχει επικρατήσει να ονομάζονται

«ECA rules» λόγω της μορφή τους. Πάνω λοιπόν σε κάθε ακμή του γράφου θα μπορούσε να υπάρχει μια ετικέτα με ένα κανόνα που θα αποτελείται το πολύ από τρία μέρη: Γεγονός το οποίο προκαλεί την ενεργοποίηση του κανόνα (Event), Συνθήκη που ελέγχεται όταν ενεργοποιηθεί ο κανόνας (Condition), Πράξη που εκτελείται όταν ικανοποιείται η συνθήκη (Action). Στη περίπτωση μας στο τρίτο μέρος συνήθως απλά έχουμε την μετάβαση στον κόμβο όπου καταλήγει η ακμή. Αγνοήσαμε αυτές τις περιπτώσεις διότι θεωρήσαμε ότι δεν επηρεάζουν το κόστος της σύνθετης υπηρεσίας για το οποίο εμείς ενδιαφερόμαστε. Αν κανείς είναι υποχρεωμένος λόγω κάποιας συνθήκης να ακολουθήσει ένα συγκεκριμένο μονοπάτι στο γράφο τότε δεν τίθεται θέμα να επιλέξει το καλύτερο με βάση το κόστος μονοπάτι και από την αρχή ο δικός μας στόχος ήταν να μπορούμε να αποφασίζουμε το καλύτερο μονοπάτι κατά την εκτέλεση οπότε οι κανόνες θα είναι υπολογισμένοι στην πλειοψηφία τους. Αν πάλι θέλουμε να υπολογίσουμε το κόστος και ο κανόνας δεν είναι υπολογισμένος τότε δεν μπορούμε παρά με κάποιο τρόπο να συνυπολογίσουμε το κόστος όλων των ενδεχόμενων μονοπατιών που ορίζουν οι κανόνες παίρνοντας για παράδειγμα τον μέσο όρο του κόστους τους.

Αν υποθέσουμε τώρα ότι για την ολοκλήρωση μιας σύνθετης υπηρεσίας μέρος των υπηρεσιών πρέπει να εκτελεστούν πολλές φορές μέχρι να ικανοποιηθεί κάποια συνθήκη ή και συγκεκριμένο αριθμό φορών τότε στο γράφημα θα έπρεπε να μπορούμε να έχουμε ακμές προς τα πίσω και κανόνες σαν αυτούς που περιγράφηκαν παραπάνω για τον τερματισμό του βρόγχου. Οι ακμές προς τα πίσω δεν απαγορεύονται από τον ορισμό του γράφου άρα ουσιαστικά μιλάμε για την ίδια με πριν περίπτωση. Επιπλέον στην περίπτωση συγκεκριμένου αριθμού επαναλήψεων δεν έχουμε παρά να «ξεδιπλώσουμε» τον βρόγχο παραθέτοντας την ομάδα των υπηρεσιών που πρέπει να επαναληφθούν τόσες φορές όσες είναι απαραίτητο.

4.3. Ο γράφος ως υποσύνολο της BPEL4WS



Σχήμα 4.4: Γράφος μιας σύνθετης υπηρεσίας

Στην BPEL4WS ορίζονται, όπως αναφέρθηκε ήδη στη σχετική παράγραφο, κάποιες δραστηριότητές οι οποίες ονομάζονται δομημένες δραστηριότητες και στόχο έχουν τον καθορισμό της σειράς εκτέλεσης των διαφόρων εργασιών που εμπλέκονται σε μια ροή εργασίας τον ορισμό της οποίας υλοποιούμε χρησιμοποιώντας τη συγκεκριμένη γλώσσα. Οι δομημένες αυτές δραστηριότητες είναι οι sequence, switch, while, flow και pick και η λειτουργία τους περιγράφεται στην παράγραφο 2.3.4.

Στην εργασία μας ορίσαμε, όπως φαίνεται παραπάνω γράφους που αποτελούνται από:

1. ακμές που δηλώνουν την διαδοχική εκτέλεση των υπηρεσιών που ορίζουν και αντιστοιχούν κατά κάποιο τρόπο στη δομημένη δραστηριότητα Sequence
2. κόμβους τύπου Or που αντιστοιχούν στη δομημένη δραστηριότητα pick της BPEL4WS.
3. κόμβους τύπου And που στην BPEL4WS θα μπορούσαν να υλοποιηθούν με τη δομημένη δραστηριότητα flow.
4. κόμβους τύπους end_of_and και end_of_or που ουσιαστικά δηλώνουν το τέλος μιας ακολουθίας εργασιών που ξεκίνησαν με έναν από τους προηγούμενους τύπους κόμβων και δεν χρειάζεται να αντιστοιχιστούν με κάποια δομημένη δραστηριότητα.

```
[1]   <sequence>
[2]     <activity 0>
[3]     <activity 1>
[4]     <flow>
[5]       <sequence>
[6]         <activity 2>
[7]         <pick>
[8]           <sequence>
[9]             <activity 3>
[10]            <activity 4>
[11]            <activity 5>
[12]            <activity 6>
[13]          </sequence>
[14]          <sequence>
[15]            <activity 7>
[16]            <activity 8>
[17]            <activity 9>
```

Σχήμα 4.5: BPEL4WS για τον γράφο του Σχήματος 4.4

Στο Σχήμα 4.4 απεικονίζεται ένας γράφος της μορφής που χρησιμοποιήσαμε στην παρούσα εργασία για την απεικόνιση μιας σύνθετης υπηρεσίας ενώ στο Σχήμα 4.5 φαίνεται η δομή του XML εγγράφου τύπου BPEL4WS που αντιστοιχεί στη ίδια σύνθετη υπηρεσία. Στο τελευταίο σχήμα φαίνονται οι ετικέτες που χρησιμοποιεί η BPEL4WS για τις δομημένες δραστηριότητες ενώ σαν μια ετικέτα με το κείμενο activity x έχουμε συμπεριλάβει και τις υπηρεσίες που αποτελούν τη σύνθετη υπηρεσία. Σημειώνεται ότι στο πραγματικό XML έγγραφο στη θέση των ετικετών «activity x» θα υπάρχουν ετικέτες τύπου invoke που θα φροντίζουν για την κλήση της αντίστοιχης υπηρεσίας. Μια επιπλέον παρατήρηση όσον αφορά την αντιστοιχία των σχημάτων 4.4, 4.5 είναι ότι από το Σχήμα 4.5 φαίνεται να απουσιάζουν οι υπηρεσίες 11 και 21. Αυτό συμβαίνει γιατί ουσιαστικά η κόμβοι αυτοί είναι τύπου end_of_or και προστέθηκαν για πρακτικούς λόγους στο γράφο ενώ δεν έχουν καμιά πρακτική σημασία.

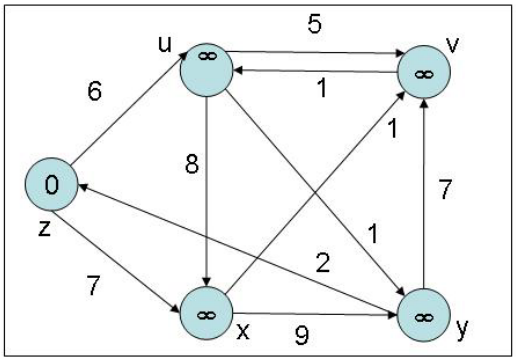
4.4. Αλγόριθμος Bellman Ford

Στη ενότητα που ακολουθεί παρουσιάζεται ένας κλασσικός αλγόριθμος υπολογισμού ελάχιστων διαδρομών απλού ζεύγους σε κατευθυνόμενα γραφήματα. Πιο συγκεκριμένα, ο αλγόριθμος αυτός εφαρμόζεται σε κατευθυνόμενα γραφήματα G όπου κάθε ακμή <u,v> έχει ένα βάρος w(u,v) και βρίσκει τα βάρη των ελάχιστων μονοπατιών από μια δεδομένη πηγή s προς κάθε κορυφή v του γράφου G. Στην πρώτη φάση του αλγόριθμου αποδίδονται οι αρχικές τιμές στα d. Στη συνέχεια για |V| - 1 φορές γίνεται 'χαλάρωση' της κάθε ακμής, δηλαδή μικραίνει η απόσταση της κάθε ακμής από την πηγή s. Στο Σχήμα 4.2 φαίνεται ένα τέτοιο γράφημα G.

Αλγόριθμος 4.1

Είσοδος: Ένας γράφος G με n κόμβους με βάρη w και ο αρχικός κόμβος του γράφου s

Έξοδος: Η απόσταση του s από τον τελικό κόμβο



Σχήμα 4.6: Παράδειγμα γράφου

Αρχικοποίηση αποστάσεων d στο άπειρο

Για i=1 έως n-1 (όπου n το πλήθος των κόμβων του γράφου)

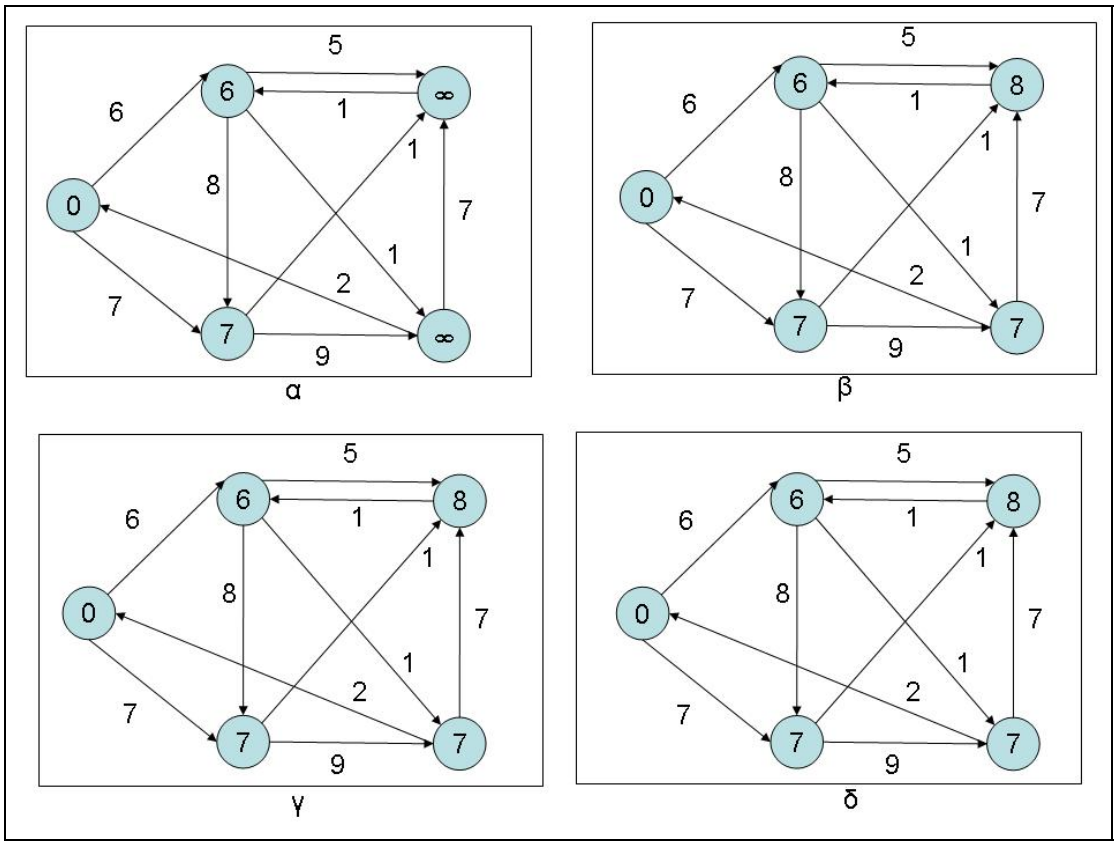
Για κάθε ακμή <u,v> (όποου u ο κόμβος i του γράφου)

(*) Αν d[v]>d[u]+w(u,v) (όπου w(u,v) το βάρος του κόμβου d)

d[v]= d[u]+w(u,v)

Πολυπλοκότητα Αλγορίθμου: Η πολυπλοκότητα του αλγορίθμου είναι γνωστή και ίση με O(n*m) όπου n το πλήθος των κόμβων του γραφήματος και m το πλήθος των ακμών του.

Πρέπει να σημειωθεί σε αυτό το σημείο ότι στην κανονική του μορφή ο αλγόριθμος έχει και τη δυνατότητα εντοπισμού και κύκλων με αρνητικό βάρος πάνω στο γράφημα. Εδώ το σκέλος αυτό παραλείπεται αφού θεωρούμε ότι το κόστος καμιάς υπηρεσίας δεν είναι αρνητικό. Η θεώρηση αυτή δεν είναι τυχαία αλλά ανταποκρίνεται στην πραγματικότητα αφού μια υπηρεσία στην καλύτερη περίπτωση μπορώ να θεωρήσω ότι δεν έχει καθόλου κόστος (κι αυτό μόνο για κάποιες μορφές κόστους όπως π.χ. το χρηματικό κόστος) δεν είναι όμως δυνατόν να μου αποφέρει κέρδος στο οποίο μεταφράζεται η ύπαρξη αρνητικού κόστους.



Σχήμα 4.7: Εφαρμογή του αλγορίθμου Bellman-Ford

Ένα παράδειγμα εφαρμογής του αλγορίθμου στο γράφημα G του Σχήματος 4.6 φαίνεται στο Σχήμα. 4.7. Και στα δύο σχήματα πάνω από κάθε ακμή αναγράφεται το κόστος διασχίζοντας τη συγκεκριμένη ακμή και μέσα σε κάθε κόμβο υπολογίζεται σιγά - σιγά το κόστος πρόσβασης σε αυτόν από τον αρχικό κόμβο. Αρχικός κόμβος στα σχήματα είναι προφανώς ο κόμβος που το κόστος του έχει αρχικοποιηθεί σε 0 ήδη από το Σχήμα 4.6 δηλαδή ο κόμβος z. Τα ονόματα των κόμβων φαίνονται επίσης στο σχήμα 4.6. Ο έλεγχος (*) του αλγορίθμου 4.1 και ότι αυτός συνεπάγεται γίνεται με σειρά για τις ακμές Σχήμα 4.7 (α): (z, u), (z,x), Σχήμα 4.7 (β): (u, v), (u, x), (u, y),

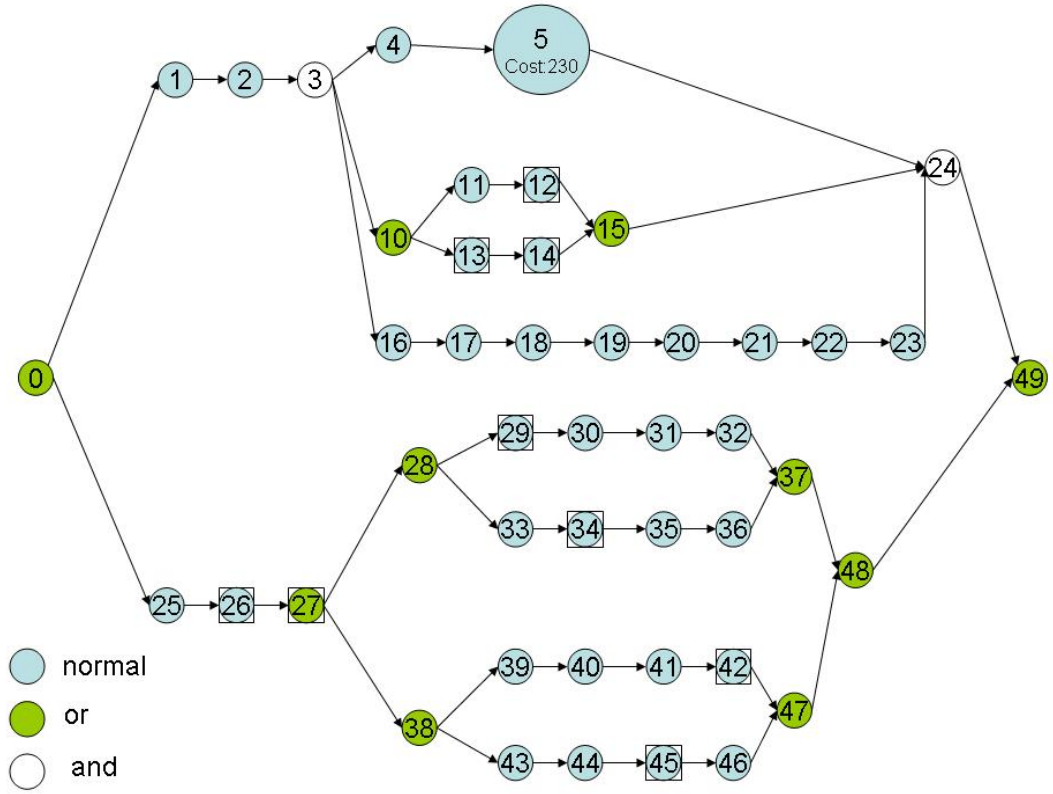
Σχήμα 4.7 (γ): (x, v), (x, y) και Σχήμα 4.7 (δ): (y, v), (y, z). Επομένως στο Σχήμα 4.5 (δ) μέσα σε κάθε κόμβο αναγράφεται η ελάχιστη απόσταση του από τον κόμβο z.

Ο αλγόριθμός αυτός υπολογίζει ελάχιστα μονοπάτια σε κατευθυνόμενους γράφους. Ο γράφος που χρησιμοποιήσαμε για τη μοντελοποίηση του προβλήματος εύρεσης των προς εκτέλεση υπηρεσιών διαδικτύου από μια ροή εργασίας είναι ένας τέτοιος γράφος. Η σκέψη αυτή μας οδήγησε στη δημιουργία του παρακάτω αλγορίθμου που χρησιμοποιεί τον Bellman-Ford αλγόριθμο.

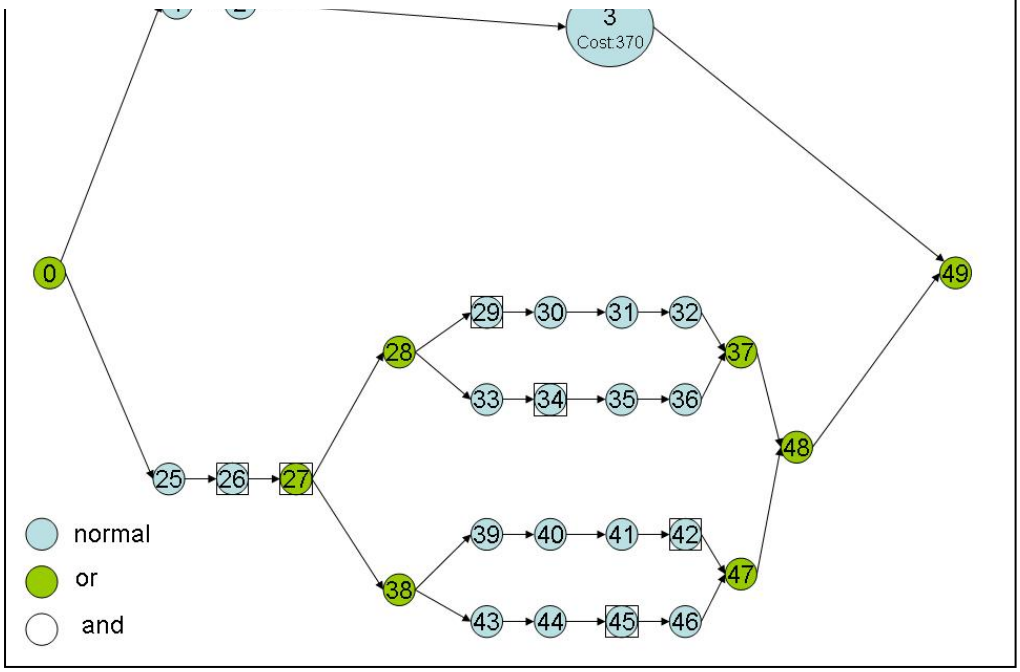
4.5. Αλγόριθμος Υπολογισμού του κόστους του βέλτιστου μονοπατιού

Ο αλγόριθμος εφαρμόζει τον αλγόριθμο των Bellman Ford στους γράφους που αναπαριστούν τις ακολουθίες των υπηρεσιών web για να βρούμε την καλύτερη τέτοια ακολουθία. Θεωρούμε ως βάρος μιας ακμής το κόστος της υπηρεσίας στην οποία καταλήγει . Επιπλέον, συρρικνώνεται κάθε υπογράφος που ξεκινάει από κόμβο and και τελειώνει σε end_of_and σε ένα μόνο κόμβο με κόστος το μικρότερό κόστος της διαδρομής από τον κόμβο And στον αντίστοιχο end_of_and που υπολογίζεται και πάλι με Bellman Ford στον υπογράφο. Το τελευταίο γίνεται γιατί ο αλγόριθμος των Bellman Ford δεν προβλέπει κόμβους τύπου And.

Ο συρρικνωμένος υπογράφος του Σχήματος 4.3 φαίνεται στο Σχήμα 4.7. Σημειώνεται ότι όλοι οι κόμβοι στο Σχήμα 4.3 έχουν κόστος 10 εκτός από τους τετράγωνους που έχουν κόστος 100. Πιο συγκεκριμένα η συρρίκνωση του γράφου ξεκινάει από τους εσωτερικότερους κόμβους τύπου And και σταδιακά προχωράει στους πιο εξωτερικούς. Έτσι ξεκινώντας από το γραφό του Σχήματος 4.3 πρώτα συρρικνώνουμε τον υπογράφο που ξεκινάει από τον κόμβο 5 κι αυτό γίνεται εφαρμόζοντας Bellman Ford σε κάθε μονοπάτι που ξεκινάει από κόμβο που έπεται του κόμβου 3 και καταλήγει σε κόμβο που προηγείται του κόμβου 9 (κόμβος End_of_And). Στη προκειμένη περίπτωση δεν έχω μονοπάτια άλλα μεμονωμένους κόμβους τους 6,7,8. Η συρρίκνωση του κόμβου 5 οδηγεί στο γράφο του Σχήματος 4.8. Ακολουθώντας ύστερα την ίδια διαδικασία για τον κόμβο 5 οδηγούμαστε στο γράφο του Σχήματος 4.9 που είναι απαλλαγμένος από κόμβους τύπου And.



Σχήμα 4.8: Παράδειγμα γράφου μετά από συρρίκνωση του κόμβου 5



Σχήμα 4.9: Παράδειγμα γράφου μετά από συρρίκνωση μετά από συρρίκνωση του κόμβου 3

Αλγόριθμος 4.2

Είσοδος: Ένας γράφος με n κόμβους

Έξοδος: Ένας συρρικνωμένος γράφος με έναν κόμβο για κάθε υπογράφο που ξεκινά με ένα κόμβο διακλάδωσης AND

Δημιουργία συρρικνωμένου γράφου

Για κάθε ένα από του κόμβους του αρχικού γραφήματος με αντιστροφή σειρά (έτσι ώστε να συρρικνωθούν πρώτα τα πιο εσωτερικά AND του αρχικού γράφου)

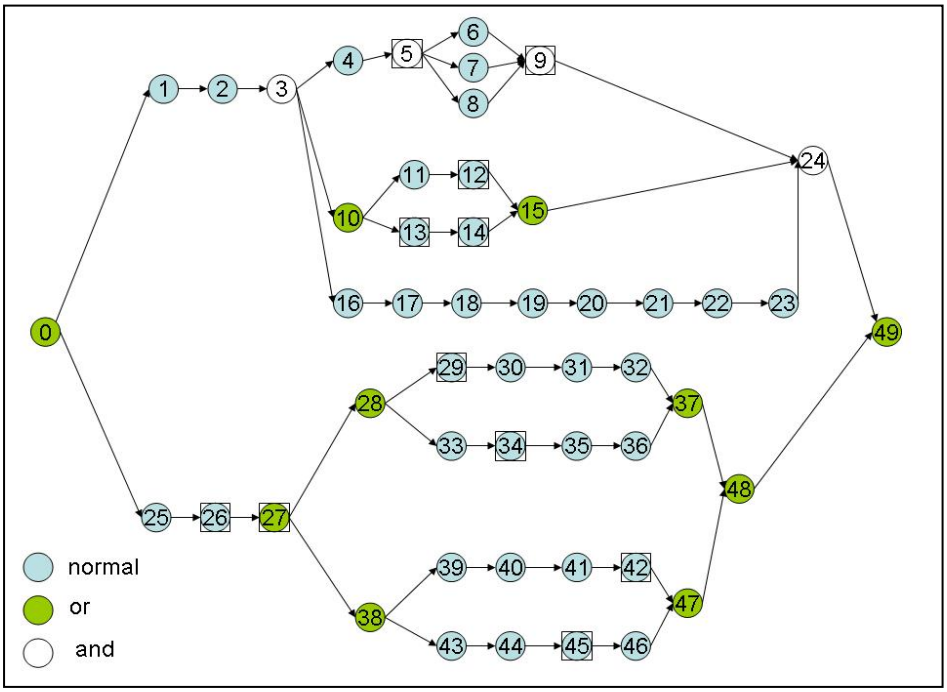
Αν ο κόμβος είναι τύπου AND

Εφαρμόζουμε Bellman-ford στον υπογράφο που ορίζει ο κόμβος και κρατάμε στο συρρικνωμένο γράφο μόνο τον αρχικό κόμβο με κόστος το κόστος που επιστρέφει ο Bellman-ford.

Εφαρμόζουμε Bellman-ford στον συρρικνωμένο γράφο

Πολυπλοκότητα Αλγορίθμου: Η πολυπλοκότητα αυτού του αλγορίθμου είναι $O(n*n*m) = O(n^2*m)$όπου n το πλήθος των κόμβων του γραφήματος και m το πλήθος των ακμών του. Ουσιαστικά εκτελείται ο Bellman-ford τόσες φορές όσοι είναι και οι κόμβοι and συν μία. Επομένως εκτελείται ο Bellman-ford το πολύ κάτι λιγότερο από n/2 φορές.

4.6. Greedy Αλγόριθμος Υπολογισμού του κόστους του καλύτερου μονοπατιού



Σχήμα 4.10: Παράδειγμα γράφου

Ο αλγόριθμός αυτός είναι ένας greedy αλγόριθμος και δεν δίνει τη βέλτιστη λύση. Ο λόγος για τον οποίο μπορεί να τον προτιμήσει κανείς από τον βέλτιστό είναι ότι επιτρέπει την εκκίνηση εκτέλεσης των υπηρεσιών πριν την ολοκλήρωση του αφού αποφασίζει βήμα βήμα για την επόμενη προς εκτέλεση υπηρεσία . Επιπλέον δεν είναι απαραίτητο κάθε κόμβος της ροής εργασίας το κόστος όλων των υπηρεσιών αλλά μόνο αυτών που μπορεί να εκτελεστούν κ βήματα μετά την τρέχουσα υπηρεσία. Επιπλέον είναι δυνατό κάθε κόμβος να μη γνωρίζει ούτε και τη ολοκληρωμένη μορφή της ροής εργασίας αλλά μέρος αυτής.

Αλγόριθμος 4.3

Είσοδος: Ένας γράφος με n κόμβους του οποίου γνωρίζουμε τον αρχικό s και τελικό κόμβο e

Έξοδος: Το κόστος που συμφώνα με τον αλγόριθμο έχει το καλύτερο μονοπάτι. (Προφανώς αφού ο αλγόριθμός είναι greedy και όχι εξαντλητικός, το κόστος δεν θα είναι πάντα το βέλτιστο)

Εκτέλεση του k-greedy αλγορίθμου με κόμβο εκτέλεσης i τον αρχικό κόμβο s του γράφου

Μέχρι ο κόμβος i να ταυτιστεί με τον τελευταίο του γράφου

Αν ο κόμβος i δεν είναι or

Το κόστους αρχικοποιείται στη τιμή κόστους του κόμβου i

Εκτελούμε τον k-1greedy αλγόριθμο με αρχικό κόμβο καθένα από τους γείτονες του i και προσθέτουμε το κόστος που επιστρέφει ο αλγόριθμος στο προηγούμενο κόστος.

Κόμβος εκτέλεσης γίνεται διαδοχικά καθένας από τους γείτονες του i και η διαδικασία επαναλαμβάνεται

Αλλιώς

εκτελούμε τον k-greedy αλγόριθμο με αρχικό κόμβο καθένα από τους γείτονες του i και κρατάμε τον γείτονα j για τον οποίο προκύπτει μικρότερο κόστος

κόμβος εκτέλεσης γίνεται ο γείτονας j και η διαδικασία επαναλαμβάνεται

προσθέτουμε στο κόστος το κόστος του i

Πολυπλοκότητα Αλγορίθμου: Η πολυπλοκότητα αυτού του αλγορίθμου είναι O(n*k)όπου n το πλήθος των κόμβων του γραφήματος και k το βήμα του greedy αλγορίθμου. Με βάση τον αλγόριθμο στη χειρότερη περίπτωση (όλοι σχεδόν οι κόμβοι είναι τύπου and, όπως και πριν) διασχίζουμε μια φορά όλους τους κόμβους του γραφήματος και για κάθε ένα από αυτούς διασχίζουμε τους d*k επομένους κόμβους.

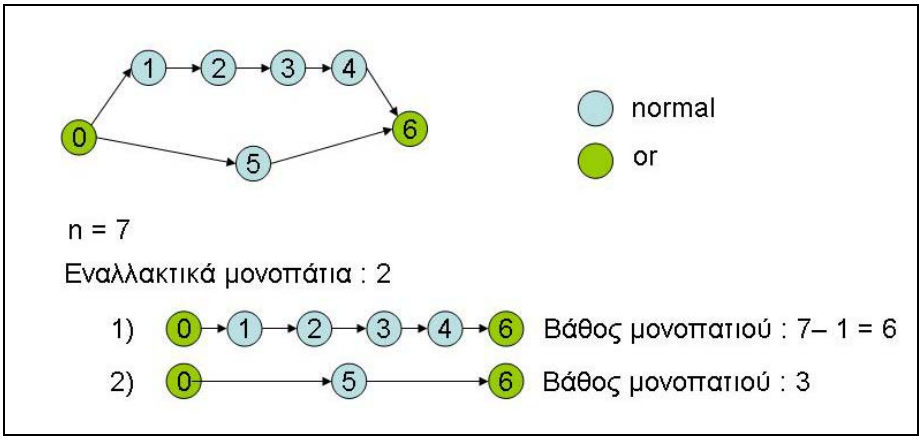
Ακολουθεί παράδειγμα όπου φαίνεται η διαφορετική λύση που δίνει ο αλγόριθμος για διαφορετικό βαθμό k. Στο Σχήμα 4.10 όλοι οι κόμβοι έχουν κόστος 10 εκτός από τους τετράγωνους που έχουν κόστος 100 Η εφαρμογή του 1-greedy στον κόμβο 28 θα μας υποδείξει να ακολουθήσουμε ως καλύτερο γείτονα ανάμεσα στους 29 και 33 τον 33 γιατί cost(33)< cost(29) Αντίθετα, αν εφαρμόσουμε 2-greedy για να κάνουμε την ίδια επιλογή οι κόμβοι φαίνεται να είναι ίδιοι cost(33)+ cost(34) =cost(29) + cost(30) και επομένως ανάλογα με την υλοποίηση μπορεί να επιλέξουμε τον 29 ή τον 33

4.7. k-greedy: η τιμή του k

Όπως έγινε φανερό από την προηγούμενη παράγραφο το βήμα k του αλγορίθμου που προτείνεται αντιστοιχεί στο βάθος μέχρι το οποίο θα αθροίζεται το κόστος των απλών υπηρεσιών της ροής εργασίας που αναπαριστά τη σύνθετη. Το άθροισμα αυτό υπολογίζεται κάθε φορά που εκτελείται μια υπηρεσία η οποία αντιστοιχεί σε κόμβο τύπου Or για να αποφασιστεί ποια θα είναι η επόμενη προς εκτέλεση υπηρεσία.

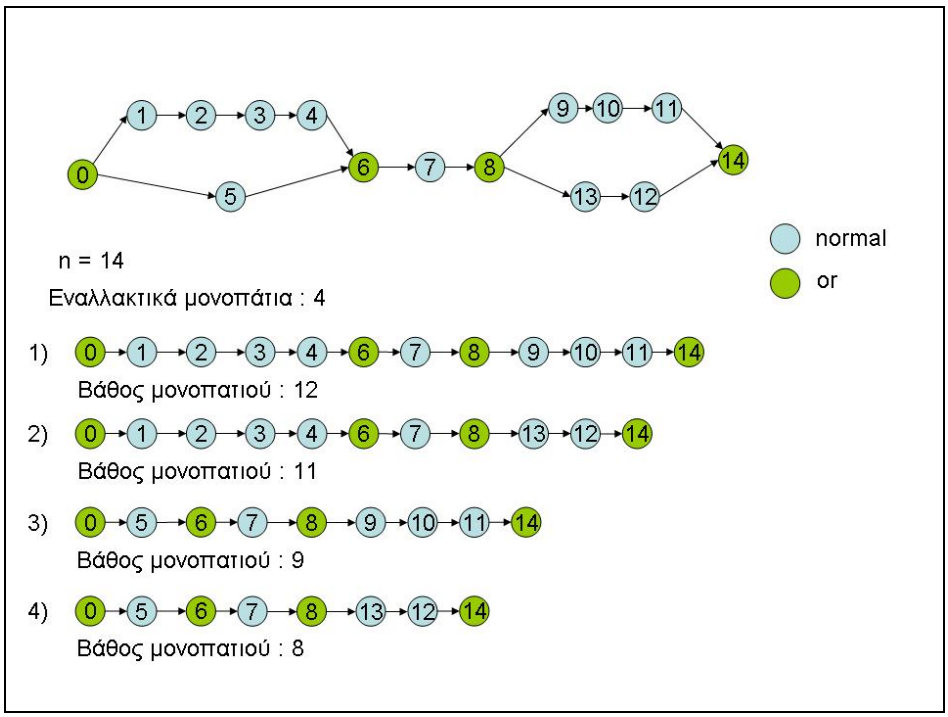
Το σίγουρο είναι ότι όσο μεγαλύτερη είναι η τιμή του k τόσο πιο βέβαιοι μπορούμε να είμαστε ότι η λύση που θα υπολογιστεί από την εκτέλεση του αλγορίθμου 4.3 θα είναι η βέλτιστη λύση. Προφανώς αν k = n, όπου n το συνολικό πλήθος των κόμβων του γράφου είναι σίγουρο ότι θα πάρουμε τη βέλτιστη λύση γιατί σίγουρα θα υπολογίζεται το κόστος μέχρι τον τελευταίο κόμβο του γράφου και άρα δεν υπάρχει περίπτωση λάθους.

Όμως θα θέλαμε να πετυχαίνουμε τη βέλτιστη λύση με k όσο το δυνατόν μικρότερο. Μια πρώτη παρατήρηση είναι ότι η ύπαρξη κόμβων Or μειώνει το συνολικό βάθος (διάμετρο) του γράφου άρα σίγουρα με k κάπως μικρότερο από n αλλά μεγαλύτερο ή ίσο με τη διάμετρο του γράφου, και πάλι ο αλγόριθμος θα δίνει τη βέλτιστη λύση. Αυτό συμβαίνει γιατί στην ουσία και πάλι ελέγχονται όλοι οι κόμβοι μέχρι το τέλος του γράφου. Ο λόγος που οι κόμβοι τύπου Or μειώνουν τη διάμετρο του γράφου είναι προφανώς το γεγονός ότι τέτοιοι κόμβοι έχουν βαθμό εξόδου μεγαλύτερο ή ίσο του 2. Το ίδιο ισχύει και για τους κόμβους τύπου and. Επομένως μετά από τέτοιου τύπου κόμβους, διακρίνουμε τουλάχιστον δυο διαφορετικά μονοπάτια στα οποία θα μοιράζονται οι κόμβοι του γράφου. Αποτέλεσμα αυτής της διαίρεσης των κόμβων είναι η διάμετρος του γράφου να είναι μικρότερη από n, το πολύ n-1 αλλά το αναμενόμενο είναι να είναι ακόμη μικρότερη.



Σχήμα 4.11: Παράδειγμα εκτέλεσης Αλγορίθμου 4.4

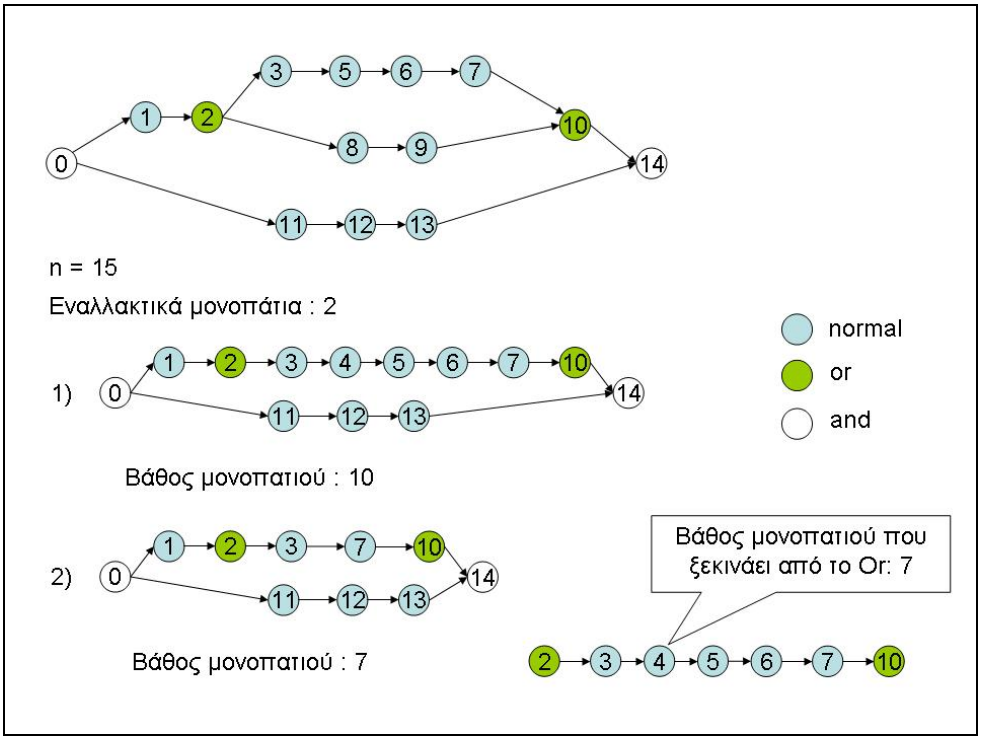
Η διάμετρος ενός γράφου ισούται με το μέγιστο βάθος μονοπατιού στο γράφο. Στο Σχήμα 4.11, για παράδειγμα, k = 6 είναι αρκετό γιατί 6 είναι η διάμετρος του γράφου.



Σχήμα 4.12: Παράδειγμα εκτέλεσης Αλγορίθμου 4.4

Όμοια και στην περίπτωση του Σχήματος 4.12 θα μπορούσα να βρω τη βέλτιστη λύση με k = 14 = n αλλά και με k = 12 δηλαδή ίσο με τη διάμετρο του γράφου.

Αντίθετα στο Σχήμα 4.13 παρότι η διάμετρος του γράφου είναι ίση με 10 επειδή ο πρώτος κόμβος Or συναντάτε σε βάθος 3 ικανοποιητικό k θα ήταν και το 10-3 = 7.



Σχήμα 4.13: Παράδειγμα εκτέλεσης Αλγορίθμου 4.4

Μια καλή λοιπόν τιμή για το k, μοιάζει να είναι η απόσταση του πρώτου κόμβου τύπου Or που συναντάμε από τον τελευταίο κόμβο του γράφου.

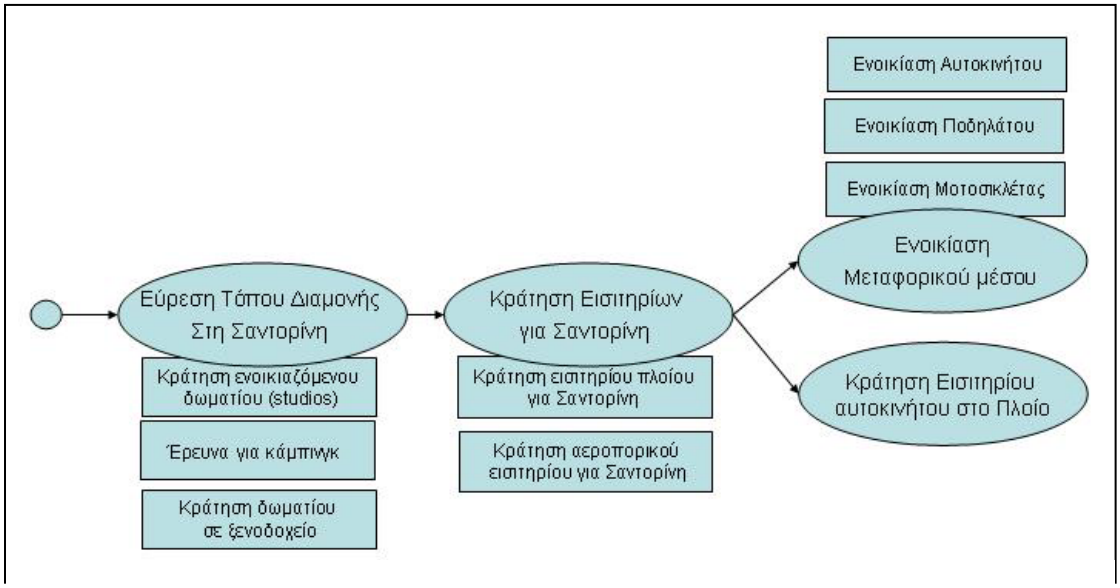
Για να δούμε με τι k θα τρέξουμε τον αλγόριθμο μας ενδιαφέρει να γνωρίζουμε, αν είναι δυνατόν, τη διάμετρο του γράφου αλλά και την απόσταση του πρώτου κόμβου τύπου Or που συναντάμε από τον τελευταίο κόμβο του γράφου. Και τα δύο στην περίπτωση μας δεν είναι δυνατόν να είναι γνωστά αντίθετα κρατώντας στατιστικά από διάφορες ροές εργασίας είναι δυνατόν να ξέρουμε περίπου το πλήθος των κόμβων από τους οποίους αποτελούνται καθώς και το ποσοστό των κόμβων and και or μέσα σε αυτούς. Είναι σίγουρο, όπως αναφέρθηκε , ότι όσο μεγαλύτερο είναι το ποσοστό των κόμβων and και or τόσο μικρότερη θα είναι η διάμετρος του γράφου. Επίσης όσο περισσότερους κόμβους τύπου and έχω στο γράφημα τόσο πιθανότερο είναι ο πρώτος κόμβος τύπου or να συναντηθεί σε μεγαλύτερο βάθος.

Στο επόμενο κεφάλαιο κάναμε πειράματα για να ελέγξουμε κατά πόσο πράγματι το k εξαρτάται από το ποσοστό των or κόμβων και το βαθμό εξόδου των κόμβων or και and.

4.8. Όμοιες υπηρεσίες

Στην παρούσα εργασία, όπως ήδη αναφέρθηκε μοντελοποιούμε με γραφήματα τις ροές εργασίας που απαρτίζουν μια σύνθετη υπηρεσία διαδικτύου. Κάθε σύνθετη υπηρεσία διαδικτύου απαρτίζεται προφανώς από άλλες απλές υπηρεσίες. Μέχρι τώρα όταν αναφερόμασταν σε ένα κόμβο του γραφήματος αναφερόμασταν απλά σε μια απλή υπηρεσία διαδικτύου. Είναι όμως δυνατόν να μοντελοποιήσουμε με έναν κόμβο ένα σύνολο από απλές υπηρεσίες διαδικτύου που επιτελούν την ίδια ή παρόμοια εργασία. Ας αναφέρουμε αυτές τις υπηρεσίες ως όμοιες. Η ιδέα αυτή δεν είναι καινούρια η ομαδοποίηση τω υπηρεσιών με βάση τη λειτουργία τους προτάθηκε και από τους κατασκευαστές του Self –Serv όπου η ομάδα των όμοιων υπηρεσιών αναφέρεται ως συλλογέας υπηρεσιών. Στόχος σε αυτή την περίπτωση είναι η εκμετάλλευση της ύπαρξης όμοιων υπηρεσιών για το συμφέρον του χρήστη. Βέβαια όταν μιλάμε για όμοιες υπηρεσίες πρέπει να ξεκαθαρίσουμε ότι ναι μεν επιτελούν την ίδια λειτουργία δεν έχουν όμως και το ίδιο κόστος [8]. Επομένως, γίνεται φανερό ότι κατά τον υπολογισμό του κόστους μιας σύνθετης υπηρεσίας θα πρέπει να υπολογίσουμε το κόστος μιας από τις όμοιες υπηρεσίες και προφανώς αφού θέλουμε να ελαχιστοποιήσουμε το κόστος θα πρέπει να υπολογίσουμε το κόστος της φθηνότερης από αυτές.

Στο Σχήμα 4.11 μπορεί κανείς να δει ξανά τον τρόπο οργάνωσης ενός ταξιδιού στη Σαντορίνη όμοια με το Σχήμα 4.1 μονό που τώρα είναι εμφανές ότι για την επίτευξη



Σχήμα 4.14: Παράδειγμα όμοιων υπηρεσιών

του ίδιου στόχου μπορεί κανείς να κάνει διαφορετικά πράγματα. Όμοια με το παράδειγμα λειτουργούν και οι όμοιες υπηρεσίες στο Διαδικτύο.

Επομένως, όσον αφορά τον αλγόριθμο υπολογισμού του κόστους της σύνθετης υπηρεσίας δεν αλλάζει τίποτα, απλά θεωρούμε κόστος ενός κόμβου το κόστος της φθηνότερης από τις όμοιες υπηρεσίες. Αυτό μπορεί να επιτευχθεί με δύο τρόπους: είτε να ψάχνουμε κάθε φορά την υπηρεσία με το χαμηλότερο κόστος είτε να έχουμε προεπεξεργαστεί τις όμοιες υπηρεσίες και να έχουμε κρατήσει το κόστος της μικρότερης. Στην προσομοίωση που υλοποιήσαμε θεωρήσαμε το δεύτερο κι αυτό γιατί είναι και η λογικότερη λύση αφού αλλιώς απλά σπαταλείται χρόνος κάθε φορά για την εύρεση της καταλληλότερης υπηρεσίας.

4.9. Αδυναμία εκτέλεσης υπηρεσιών Διαδικτύου

Ένα επιπλέον ζήτημα είναι ότι δεν είναι πάντα σίγουρο ότι η υπηρεσία που θα επιλεγεί προς εκτέλεση θα είναι διαθέσιμη εκείνη τη χρονική στιγμή. Αυτό μπορεί να συμβαίνει είτε λόγω προσωρινής βλάβης του παροχέα της υπηρεσίας, είτε λόγω ύπαρξης μη έγκυρων δεδομένων στον κόμβο που αποφασίζει την εκτέλεσή της, Άλλωστε είναι διαπιστωμένο ότι ο χρόνος ζωής μιας πληροφορίας στο Διαδικτύο είναι πολύ μικρός. Έτσι, θα πρέπει κατά την εκτέλεση της ροής εργασίας αν διαπιστωθεί ότι δεν μπορεί να προχωρήσει να μπορεί να γυρίσει πίσω και να αποφασίσει την εκτέλεση κάποιας άλλης. Αυτό σημαίνει ότι είτε θα πάρει την επόμενη όμοια υπηρεσία (αν υπάρχουν όμοιες υπηρεσίες όπως ορίστηκαν στην προηγούμενη παράγραφο) και αυτές είναι διαθέσιμες, είτε θα επιστρέψει στον προηγούμενο κόμβο or και θα επιλέξει από εκεί ένα νέο μονοπάτι.

4.10. Επέκταση αλγορίθμου Greedy σε περιπτώσεις ύπαρξης κόμβων εκτός λειτουργίας

Τον αλγόριθμο αυτό αναπτύξαμε για να μπορέσουμε να προσεγγισθούμε το βέλτιστο κόστος σε περίπτωση που υπάρχουν αποτυχίες στο σύστημα και δεν αποτελεί παρά παραλλαγή του Αλγορίθμου 4.3.

Αλγόριθμος 4.4

Είσοδος: Ένας γράφος με n κόμβους του οποίου γνωρίζουμε τον αρχικό s και τελικό κόμβο e

Σε κάθε κόμβο μια ταξινομημένη ακολουθία με βάση το κόστος των όμοιων υπηρεσιών

Έξοδος: Το κόστος που συμφώνα με τον αλγόριθμο έχει το καλύτερο μονοπάτι. (Προφανώς αφού ο αλγόριθμός είναι greedy και όχι εξαντλητικός το κόστος δεν θα είναι πάντα το βέλτιστο)

Εκτέλεση του k-greedy αλγορίθμου με κόμβο εκτέλεσης i τον αρχικό κόμβο s του γράφου

Μέχρι ο κόμβος i να ταυτιστεί με τον τελευταίο του γράφου

Αν ο κόμβος i δεν είναι or

Αν ο κόμβος i μπορεί να εκτελεστεί

Υπολογίζουμε το κόστος (Κόστος Αντικατάστασης) κ-βημάτων αντικαθιστώντας τη συγκεκριμένη υπηρεσία με μια όμοια της

Υπολογίζουμε το κόστος (Κόστος Επιστροφής) για να επιστρέψει η υπηρεσία στον τελευταίο κόμβο τύπου or και να προχωρήσει από εκεί κ-βήματα

(οι υπολογισμοί γίνονται με τον k-greedy αλγόριθμο θεωρώντας ότι δεν υπάρχει πιθανότητα να μη λειτουργεί κανένας κόμβος)

Αν μικρότερο είναι το Κόστος Αντικατάστασης κόμβος εκτέλεσης παραμένει ο i

Αλλιώς κόμβος εκτέλεσης i γίνεται το ο τελευταίος or κόμβος

Και εκτελούμε τον αλγόριθμο με αρχικό κόμβο καθένα από τους γείτονες του i

Αλλιώς

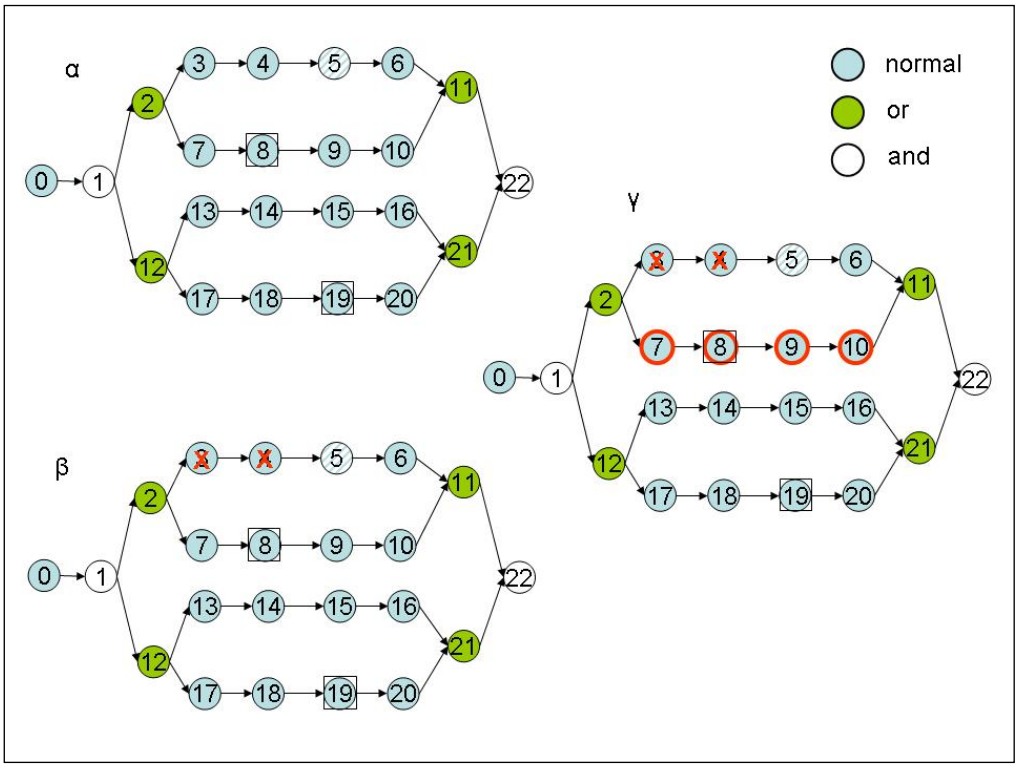
Εκτελούμε τον αλγόριθμο με αρχικό κόμβο καθένα από τους γείτονες του i και κρατάμε τον γείτονα j για τον οποίο προκύπτει μικρότερο κόστος

κόμβος εκτέλεσης γίνεται ο γείτονας j και η διαδικασία επαναλαμβάνεται

Προσθέτουμε στο κόστος το κόστος του i

Πολυπλοκότητα Αλγορίθμου: : Η πολυπλοκότητα αυτού του αλγορίθμου είναι η ίδια με αυτήν του Αλγορίθμου 4.3 δηλαδή O(n*k)όπου n το πλήθος των κόμβων του γραφήματος και k το βήμα του greedy αλγορίθμου. Η επέκταση του αλγορίθμου για να καλύπτει και περιπτώσεις όπου υπάρχουν σφάλματα δεν επιφέρει επιπλέον κόστος στον υπολογισμό ή καλύτερα δεν επηρεάζει την χειρότερη περίπτωση και άρα δεν αλλάζει και την πολυπλοκότητα του εκτέλεσης του αλγορίθμου.

Στη συνέχεια δίνεται ένα παράδειγμα για τον τρόπο λειτουργίας του συγκεκριμένου αλγορίθμου. Θεωρούμε τη ροή εργασίας που απεικονίζεται στο Σχήμα 4.12 (α). Επίσης θεωρούμε ότι θα εκτελεστεί η καλύτερη με βάση το κόστος ακολουθία υπηρεσιών που στην προκειμένη περίπτωση είναι η: 0,1,2,3,4,5,6,11,12,13,14,15,16,21,22.Σημειώνεται ότι στην πράξη η ακολουθία αυτή δε είναι γνωστή εκ των προτέρων αλλά αποφασίζετε όπως δείξαμε και στον προηγούμενο αλγόριθμο βήμα - βήμα η επόμενη προς εκτέλεση υπηρεσία. Κατά την εκτέλεση όμως της υπηρεσίας 5 ανακαλύπτουμε ότι αυτή δεν είναι δυνατόν να εκτελεστεί όποτε αναιρούμε τις λειτουργίες των υπηρεσιών 3,4 (Σχήμα 4.12 (β)) και συνεχίζουμε την



Σχήμα 4.15: Παράδειγμα εκτέλεσης Αλγορίθμου 4.4

εκτέλεση των υπηρεσιών 7,8,… (Σχήμα 4.12 (γ)). Η αναίρεση κάποιων λειτουργιών όπως αυτών που είχαν εκτελέσει στο παράδειγμα μας οι υπηρεσίες 3 και 4 επιφέρουν πρόσθετο κόστος ενώ ταυτόχρονα προϋποθέτουν την πρόβλεψη δυνατότητας αναίρεσης λειτουργιών κατά τον προγραμματισμό των υπηρεσιών ή τουλάχιστον κατά τη σύνθεση τους. Η δυνατότητα αυτή προσφέρεται ήδη από γλώσσες όπως την BPEL4WS στην οποία είναι δυνατόν να οριστούν χειριστές για την αναίρεση ολόκληρης ή μέρους της σύνθετης υπηρεσίας.

4.11. Επέκταση αλγορίθμου με χρήση Bellman- Ford σε περιπτώσεις ύπαρξης κόμβων εκτός λειτουργίας

Αλγόριθμος 4.5

Είσοδος: Ένας γράφος με n κόμβους του οποίου γνωρίζουμε τον αρχικό s και τελικό κόμβο e

Σε κάθε κόμβο μια ταξινομημένη ακολουθία με βάση το κόστος των όμοιων υπηρεσιών

Έξοδος: Το κόστος που συμφώνα με τον αλγόριθμο έχει το καλύτερο μονοπάτι. (Προφανώς αφού ο αλγόριθμός)

Στη προκειμένη περίπτωση αφού εκτελεστεί κανονικά ο αλγόριθμος 4.2 και φροντίσουμε έτσι ώστε να αποθηκευτεί το μονοπάτι που εκείνος ο αλγόριθμος επιλέγει . Παίρνουμε έναν ένα τους κόμβους του μονοπατιού κι αν είναι δυνατόν τους εκτελούμε αλλιώς γυρνάμε πίσω στον προηγούμενο κόμβο or αναιρώντας τις αντίστοιχες υπηρεσίες βρίσκουμε το καινούριο από εκεί μονοπάτι και συνεχίζουμε με τον ίδιο τρόπο.

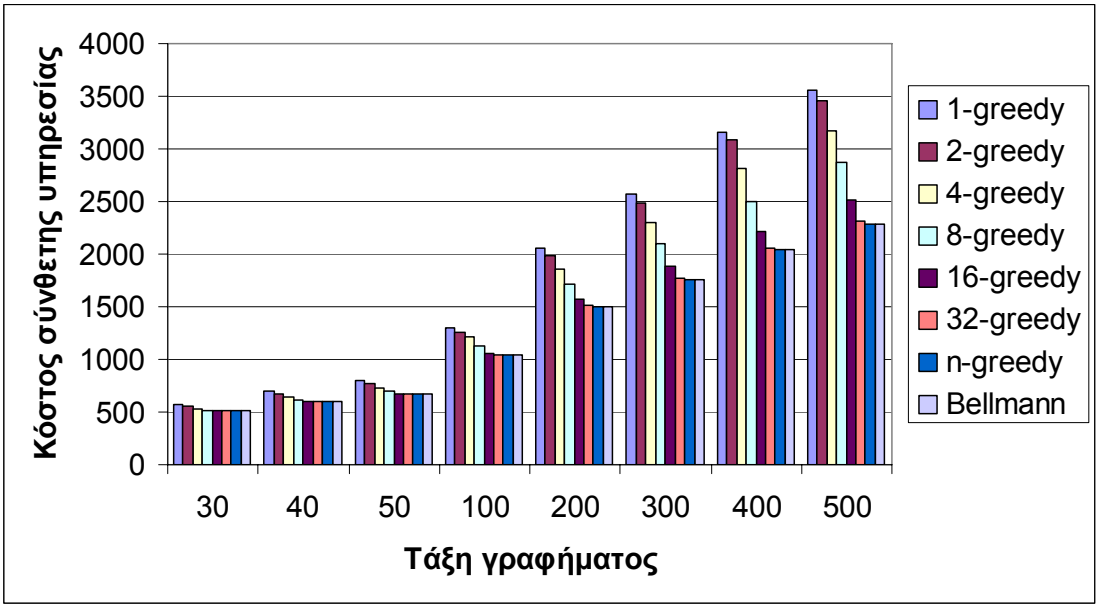
Πολυπλοκότητα Αλγορίθμου: Η πολυπλοκότητα του Αλγορίθμου 4.2 είναι $O(n^2*m)$ επομένως η πολυπλοκότητα του συγκεκριμένου αλγορίθμου ισούται με $O(n^3*m)$ αφού ουσιαστικά απλά εκτελείται το πολύ O(n) φορές ο αλγόριθμος 4.2 στη χειρότερη περίπτωση όπου O(n) κόμβοι δεν λειτουργούν.

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΑ

Στο κεφάλαιο αυτό, παρουσιάζονται τα αποτελέσματα και τα συμπεράσματα που βγάλαμε από την εκτέλεση των αλγορίθμων που περιγράφηκαν παραπάνω. Για την εκτέλεση των πειραμάτων αυτών υλοποιήθηκε πρόγραμμα προσομοίωσης της εκτέλεσης των σύνθετων υπηρεσιών καθώς και πρόγραμμα για την δημιουργία των γραφών που ορίζουν τις σύνθετες υπηρεσίες. Τα προγράμματα αυτά υλοποιήθηκαν σε γλώσσα C. Κατά την προσομοίωση της εκτέλεσης της σύνθετης υπηρεσίας εκτέλεση θεωρούμε την επίσκεψη στον κόμβο του γράφου που αντιστοιχεί στην υπηρεσία. Στα πειράματα που παρουσιάζονται παρακάτω μελετάμε την συμπεριφορά των αλγορίθμων όταν αλλάζει το πλήθος των κόμβων του γράφου, ο βαθμός διακλάδωσης, το ποσοστό των κόμβων And και Or και τέλος, η πιθανότητα αποτυχίας της απλής υπηρεσίας.

5.1. Πείραμα 1^{ο}: Κόστος Σύνθετης-Πλήθος Κόμβων

Για το πείραμα αυτό τρέξαμε τον απλό Greedy αλγόριθμο με βάθος 1, 2, 4, 8, 16, 32, n καθώς και τον βέλτιστο αλγόριθμο σε γράφους με 30, 40, 50, 100, 200,300, 400 και 500 κόμβους και πιθανότητα δημιουργίας or κόμβων και and κόμβων 10%, ενώ ο μέγιστος βαθμός εξόδου των and και or κόμβων ορίστηκε να είναι 2. Στο Σχήμα 5.1 φαίνεται ο μέσος όρος του κόστους του «καλύτερου» με βάση κάθε αλγόριθμο μονοπατιού στις 1000 εκτελέσεις του πειράματος με τα ίδια στοιχεία εισόδου.



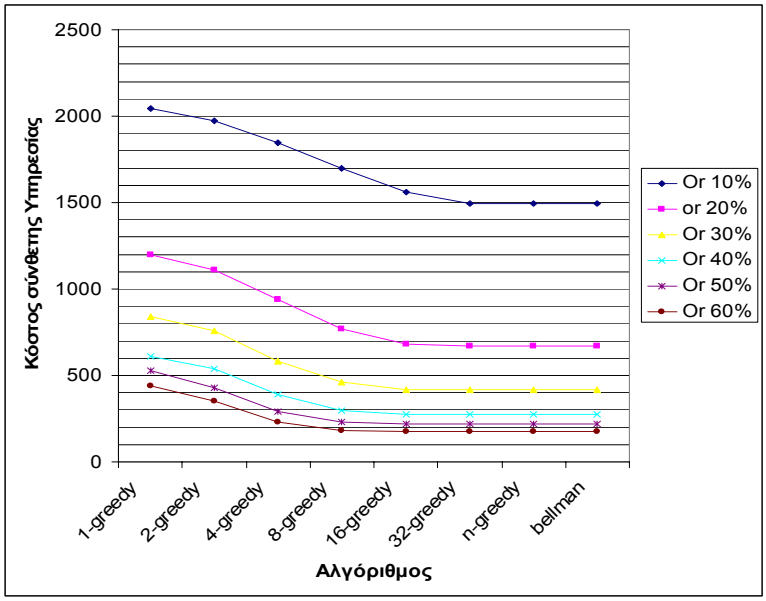
Σχήμα 5.1: Απεικόνιση αποτελεσμάτων 1ου πειράματος

Παρατηρούμε ότι, όπως είναι λογικό, το κόστος αυξάνεται όσο αυξάνει το πλήθος των κόμβων του γράφου. Επίσης παρατηρούμε ότι όσο το βάθος k αυξάνει τα αποτελέσματα εκτέλεσης του greedy αλγόριθμου σχεδόν ταυτίζονται με του βέλτιστου γεγονός που υποδεικνύει ότι ο greedy αλγόριθμος είναι μάλλον ικανοποιητικός. Αν σκεφτούμε μάλιστα και τα οφέλη που προκύπτουν από το γεγονός ότι μπορεί να εκτελείται παράλληλα με την εκτέλεση των υπηρεσιών σε ένα περιβάλλον όπου η ροή εργασίας δεν είναι γνωστή εξολοκλήρου σε κάθε κόμβο τότε θα λέγαμε ότι ο greedy πάρα το μικρό κόστος του είναι μάλλον προτιμότερος. Βέβαια τα αποτελέσματα του n-greedy ταυτίζονται ακριβώς με τα αποτελέσματα της εκτέλεσης του βέλτιστου, η εκτέλεση όμως αυτού του αλγορίθμου απαιτεί την επαναλαμβανομένη διάσχιση σχεδόν όλου του γράφου που απεικονίζει τη ροή εργασίας.

5.2. Πείραμα 2^{o}: Κόστος υπηρεσίας-Είδος κόμβων

Για το πείραμα αυτό τρέξαμε τον απλό Greedy αλγόριθμο με βήμα k ίσο με 2 καθώς και τον βέλτιστο αλγόριθμο σε γράφους με 200 κόμβους και πιθανότητα δημιουργίας κόμβου and 10% ενώ αλλάξαμε την πιθανότητα δημιουργίας κόμβου or σε 10%, 20%, 30%, 40% 50% και 60 %, το μέγιστο βαθμό εξόδου των κόμβων τον αφήσαμε σταθερό και ίσο με 2. Στο Σχήμα 5.2 φαίνεται ο μέσος όρος του κόστους του

«καλύτερου» με βάση κάθε αλγόριθμο μονοπατιού στις 1000 εκτελέσεις του πειράματος με τα ίδια στοιχεία εισόδου.



Σχήμα 5.2: Απεικόνιση αποτελεσμάτων 2ου πειράματος

Σημειώνουμε ότι ,όπως είναι αναμενόμενο, αυξάνοντας την πιθανότητα δημιουργία κόμβου τύπου Or αυξάνει και το πλήθος των κόμβων τύπου Or που υπάρχουν στο γράφημα. Ενδεικτικά αναφέρουμε ότι στο συγκεκριμένο πείραμα πιθανότητα δημιουργίας κόμβων τύπου Or 10% αντιστοιχεί σε ποσοστό κόμβων τύπου Or στο γράφημα 6% ενώ για πιθανότητα 30% έχουμε ποσοστό 13% και για πιθανότητα 60% έχουμε ποσοστό 21%. Επιπλέον σημειώνεται ότι επειδή στο γράφο υπάρχουν κόμβοι τύπου end για κάθε κόμβο τύπου or ή and το ποσοστό των κόμβων τύπου or δεν είναι δυνατόν να αυξάνει ανάλογα με την αύξηση της πιθανότητας δημιουργία κόμβων τύπου Or.

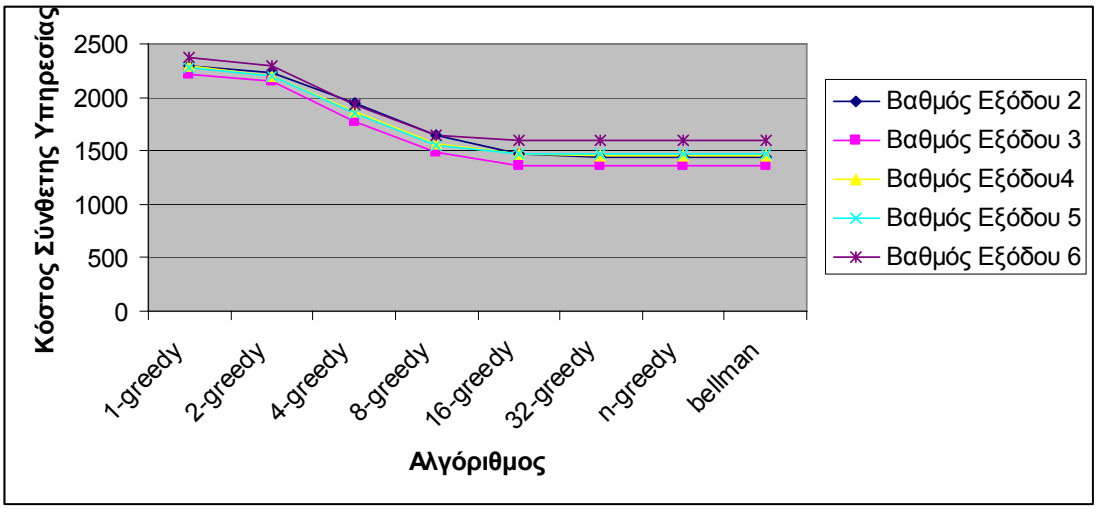
Παρατηρούμε ότι καθώς η πιθανότητα δημιουργίας κόμβου Or αυξάνεται το κόστος της υπηρεσίας μειώνεται. Το γεγονός αυτό οφείλεται στη δυνατότητα δημιουργίας περισσοτέρων εναλλακτικών μονοπατιών όσο αυξάνει το ποσοστό των κόμβων τύπου Or που υπάρχει στο γράφημα.

Επιπλέον όσο αυξάνει το βήμα του greedy αλγόριθμου τόσο το κόστος πλησιάζει το κόστος του βέλτιστου, την παρατήρηση αυτή κάναμε άλλωστε ήδη από το προηγούμενο πείραμα.

Αν κρατήσουμε τώρα σταθερό το ποσοστό των κόμβων Or και αλλάξουμε αυτό των κόμβων AND το αποτέλεσμα που παίρνουμε, όπως είναι αναμενόμενο, είναι το αντίστροφο καθώς το ποσοστό των κόμβων And μειώνεται το κόστος της υπηρεσίας αυξάνεται.

5.3. Πείραμα 3ο: Κόστος υπηρεσίας – Βαθμός διακλάδωσης

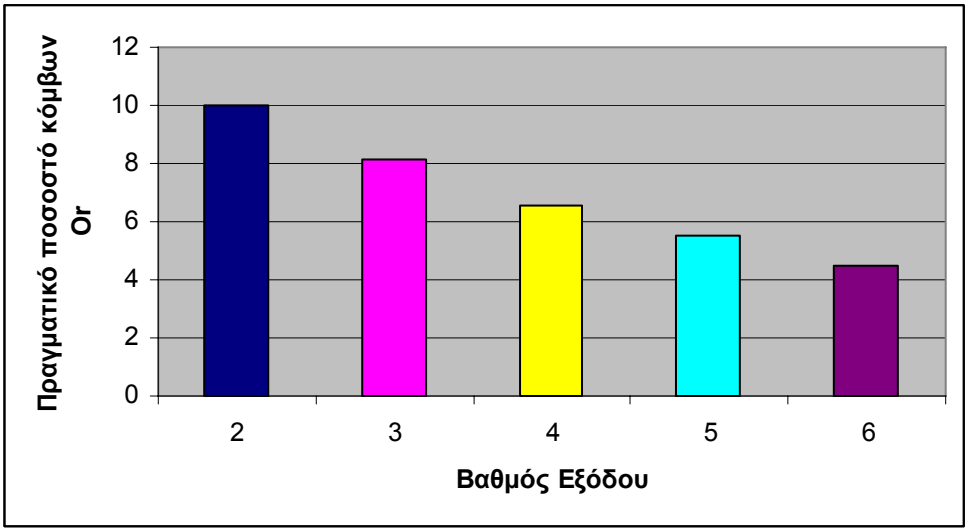
Στο πείραμα αυτό τρέξαμε τον απλό Greedy αλγόριθμο με βήμα k ίσο με 2 καθώς και τον βέλτιστο αλγόριθμο σε γράφους με 200 κόμβους και πιθανότητα δημιουργίας κόμβων And και Or 20% ενώ το μέγιστο βαθμό εξόδου των κόμβων τον σε 2, 3, 4, 5, 6. Στο Σχήμα 5.2 φαίνεται ο μέσος όρος του κόστους του «καλύτερου» με βάση κάθε αλγόριθμο μονοπατιού στις 1000 εκτελέσεις του πειράματος με τα ίδια στοιχεία εισόδου.



Σχήμα 5.3: Απεικόνιση αποτελεσμάτων 3ου πειράματος (πλήθος κόμβων 200)

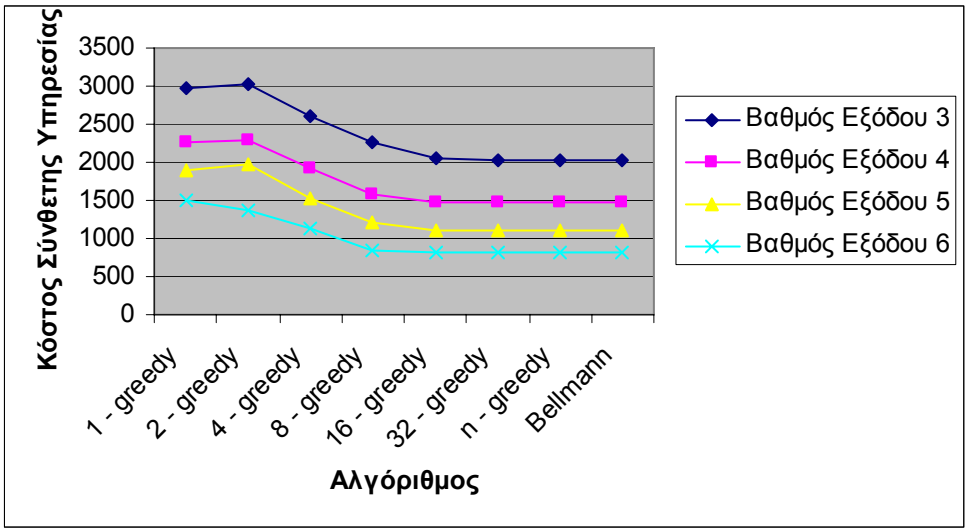
Παρατηρώντας το Σχήμα 5.3 θα μπορούσε να πει κανείς ότι το κόστος του γράφου είναι ανεξάρτητο από το βαθμό εξόδου του γράφου ή τουλάχιστον ότι δεν φαίνεται να υπάρχει κάποια αναλογική σχέση ανάμεσα σε αυτά τα δύο. Το συμπέρασμα αυτό είναι μάλλον παράλογο αν σκεφτεί κανείς ότι ,κρατώντας σταθερό το πλήθος των κόμβων του γραφήματος και την πιθανότητα δημιουργίας κόμβων Or, το αναμενόμενο είναι να δημιουργούνται περισσότερα εναλλακτικά μονοπάτια για κάθε

Or και επομένως το κόστος θα περιμέναμε να μειώνεται. Τα παράξενα αυτά αποτελέσματα έγιναν η αφορμή να μελετήσουμε περισσότερο τους γράφους που δημιουργήθηκαν. Στο Σχήμα 5.4 φαίνεται το πραγματικό ποσοστό των κόμβων Or που υπήρχαν στα γραφήματα για τα οποία εκτελέστηκε το πείραμα. Είχε αναφερθεί σε προηγούμενο πείραμα ότι αυξάνοντας την πιθανότητα δημιουργίας κόμβου τύπου Or αυξάνει και το πλήθος των κόμβων τύπου Or που υπάρχουν στο γράφημα. Επίσης όπως θα φανεί στα πειράματα που ακολουθούν το πραγματικό ποσοστό των κόμβων Or σε γράφους με ίδιο αριθμό κόμβων και ίδιο βαθμό εξόδου είναι σταθερό και αυξάνει όσο αυξάνει η πιθανότητα δημιουργίας κόμβων αυτού του τύπου. Οι παρατηρήσεις όμως αυτές δεν ισχύουν όπως φαίνεται στο Σχήμα 5.4 για την περίπτωση όπου η διακλάδωση αυξάνεται. Στο Σχήμα 5.4 βλέπουμε το πραγματικό ποσοστό των κόμβων Or να μειώνεται όσο αυξάνει ο βαθμός εξόδου παρότι η πιθανότητα δημιουργίας κόμβων τύπου Or παραμένει σταθερή. Όταν η διακλάδωση αυξάνεται οι κόμβοι του γράφου αναγκαστικά μοιράζονται σε περισσότερες ομάδες που αποτελούν τα εναλλακτικά μονοπάτια και οι κόμβοι της κάθε ομάδας είναι λιγότεροι, επομένως δεν υπάρχει λόγω του αριθμού των κόμβων περιθώριο να δημιουργηθούν πολλοί κόμβοι τύπου Or.



Σχήμα 5.4: Απεικόνιση αποτελεσμάτων 3ου πειράματος (πλήθος κόμβων 200)

Για να δούμε λοιπόν πως επηρεάζεται ο αλγόριθμος από τον βαθμό εξόδου επιλέξαμε από το δείγμα των 4000 χιλιάδων γράφων με βαθμό εξόδου 3, 4, 5 και 6 γράφους (70 από κάθε είδος) με πραγματικό αριθμό κόμβων Or 12 (Ο κάθε γράφος είχε 200 κόμβους) και μελετήσαμε το κόστος σε αυτούς τους γράφους. Η γραφική παράσταση στο Σχήμα 5.5 δείχνει αυτήν ακριβώς τη μελέτη του κόστους σε σχέση με το βαθμό εξόδου.



Σχήμα 5.5: Απεικόνιση αποτελεσμάτων 3ου πειράματος (πλήθος κόμβων 200)

Στο πείραμα η πιθανότητα δημιουργίας κόμβων τύπου Or ήταν 0,2 και αυτό σημαίνει ότι στην ιδανική περίπτωση, αν οι γράφοι δημιουργούνταν εντελώς τυχαία, θα είχαμε 40 κόμβους Or σε κάθε γράφο. Αυτό όμως είναι αδύνατο λόγω των περιορισμών που θέσαμε στη κατασκευή του γράφου. Οι περιορισμοί αυτοί φροντίζουν έτσι ώστε να συνδυάζονται σωστά οι κόμβοι Or (And) με τους κόμβους end_of_or (end_of_and). Με πιθανότητα όμως 0,2 οι περιορισμοί αυτοί μειώνονται και για αυτό δεν υπήρχε στο δείγμα μας γράφος με βαθμό εξόδου 2 και 12 κόμβους τύπου Or.

Στο Σχήμα 5.5 φαίνεται τελικά να επιβεβαιώνεται η υπόθεση μας ότι δηλαδή όσο ο βαθμός εξόδου του γράφου αυξάνεται τόσο το κόστος του μειώνεται.

5.4. Πείραμα 4^{ο}: Σχέση βάθους k του greedy με πιθανότητα Or

Ήδη, από τα πρώτα πειράματα, φάνηκε ότι το βάθος k για το οποίο πρέπει να εκτελεστεί ο αλγόριθμος έτσι ώστε να επιτύχει μια καλή λύση δεν είναι πάντα το ίδιο. Αναφέρθηκε παραπάνω ότι όσο αυξάνει το βήμα του greedy αλγόριθμου τόσο το κόστος πλησιάζει το κόστος του βέλτιστου. Επίσης είναι αναμενόμενο ότι η λύση που προκύπτει από την εκτέλεση του n-greedy αλγορίθμου είναι πάντα η βέλτιστη λύση μια και ο n-greedy αλγόριθμος κάνει εξαντλητικό έλεγχο σε όλους τους κόμβους του γράφου. Με βάση αυτή την παρατήρηση επιχειρήσαμε το παρακάτω πείραμα για να δούμε με τι βάθος k θα πρέπει να εκτελέσουμε τον αλγόριθμο για να πάρουμε τη βέλτιστη λύση.

Στο πείραμα αυτό χρησιμοποιήσαμε γράφους με 30, 40, 50, 100, 200, 300, 400 και 500 κόμβους και πιθανότητα δημιουργίας κόμβου and 10% ενώ αλλάξαμε την πιθανότητα δημιουργίας κόμβου or σε 10%, 20%, 40%, το μέγιστο βαθμό εξόδου των κόμβων τον αφήσαμε σταθερό και ίσο με 2. Κάθε φορά εκτελούσαμε τον βέλτιστο αλγόριθμο για να βρούμε τη βέλτιστη λύση και στη συνέχεια εκτελούσαμε τον απλό Greedy αλγόριθμο με βήμα 1,2 κ.λ.π. μέχρι να βρεθεί η βέλτιστη λύση. Εκτελέσαμε το παραπάνω πείραμα για 1000 γράφους με κάθε συνδυασμό των στοιχείων εισόδου που μόλις περιγράφηκαν και κρατήσαμε το βάθος k στο οποίο σταματούσαμε κάθε φορά.

Στο πείραμα αυτό ελέγξαμε δηλαδή πως μεταβάλλεται το k όταν μεταβάλλεται η πιθανότητα δημιουργίας κόμβων τύπου or κι αυτό γιατί όπως ήδη αναφέρθηκε η πιθανότητα αυτή αναμένουμε να επηρεάζει τη διάμετρο του γράφου αφού από αυτήν εξαρτάται το πλήθος των or κόμβων.

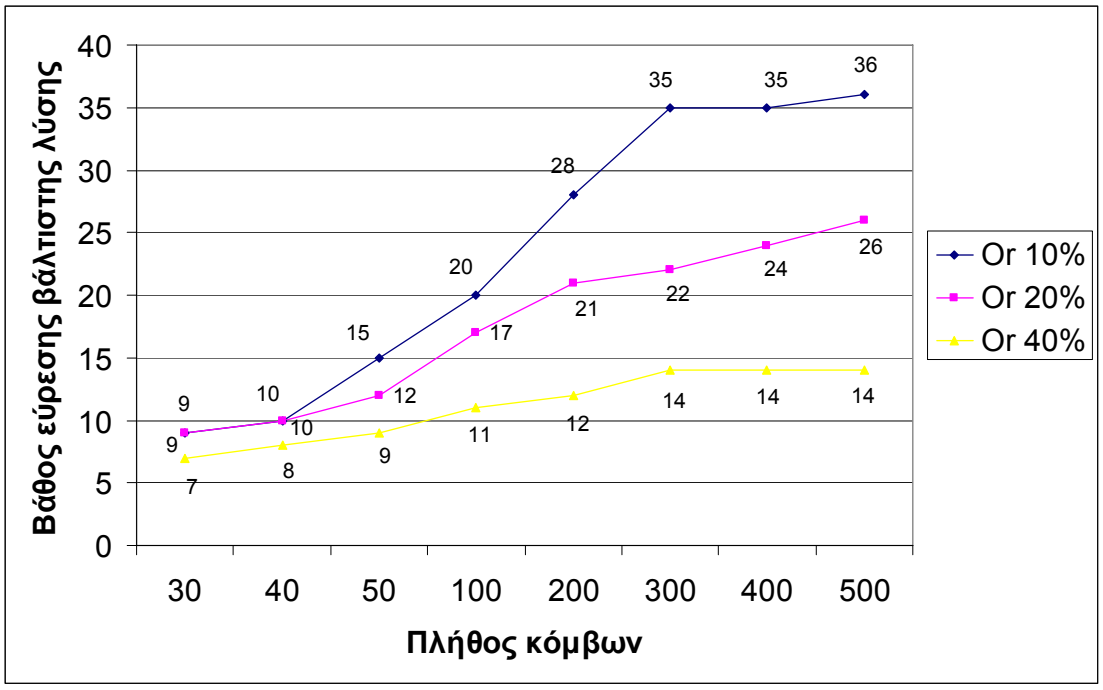
Στον Πίνακα 5.1 φαίνονται τα αποτελέσματα του πειράματος αυτού. Με τον όρο αξιοπιστία στον πίνακα αναφερόμαστε στην πιθανότητα εύρεσης της βέλτιστης λύσης αν εκτελέσουμε τον αλγόριθμο με βήμα k. Είναι λοιπόν πλέον φανερό ότι το k με το οποίο βρίσκεται η βέλτιστη λύση είναι πολύ μικρότερο από το πλήθος των κόμβων n.

Επίσης παρατηρούμε ότι όσο αυξάνει το πλήθος των κόμβων του γραφήματος τόσο αυξάνει και το k το οποίο πρέπει να χρησιμοποιήσουμε για να έχουμε αποτελέσματα

Πιθανότητα Or	**Αριθμός Κόμβων**	**Αξιοπιστία 95% → k**	**Αξιοπιστία 99% → k**
10%	30	6	9
	40	7	10
	50	9	15
	100	18	20
	200	21	28
	300	26	35
	400	29	35
	500	28	36
20%	30	6	9
	40	8	10
	50	9	12
	100	12	17
	200	15	21
	300	18	22
	400	20	24
	500	20	26
40%	30	6	7
	40	6	8
	50	8	9
	100	9	11
	200	11	12
	300	11	14
	400	12	14
	500	12	14

Πίνακας 5.1: **Βάθος k σε σχέση με το πλήθος των κόμβων Or**

που να πλησιάζουν τα βέλτιστα. Επίσης παρατηρούμε ότι όσο το πλήθος των κόμβων Or αυξάνει μέσα στο γράφημα τόσο λιγότερα είναι και τα βήματα που χρειάζεται να εκτελέσει ο αλγόριθμος δηλαδή τόσο μικρότερο το βέλτιστο k γεγονός που οφείλεται στην ύπαρξη περισσοτέρων αλλά μικρότερων μονοπατιών όταν τα Or είναι περισσότερα. Η αλήθεια είναι ότι μικρότερο μονοπάτι δεν σημαίνει σίγουρα μικρότερο κόστος αλλά όσο μικρότερά είναι τα μονοπάτια τόσο αυξάνεται και η πιθανότητα το κόστος τους να είναι μικρό.



Σχήμα 5.6: k για βέλτιστα αποτελέσματα σε ποσοστό 95%

Στο Σχήμα 5.6 φαίνεται γραφικά το βάθος k για την εύρεση βέλτιστων λύσεων σε ποσοστό 95%. Από το διάγραμμα αυτό παρατηρούμε ότι ενώ για μικρό αριθμό κόμβων το βάθος k δεν επηρεάζεται πολύ από την αύξηση των κόμβων τύπου Or, συμβαίνει ακριβώς το αντίθετο για μεγάλο πλήθος κόμβων.

5.5. Πείραμα 5ο: Σχέση βάθους k του greedy με βαθμό εξόδου

Θεωρητικά περιμέναμε ότι το βάθος k για το οποίο πρέπει να εκτελεστεί ο αλγόριθμος έτσι ώστε να επιτύχει μια καλή λύση πρέπει να επηρεάζεται και από το μέγιστο βαθμό εξόδου των κόμβων, μια και αυτός επηρεάζει τη διάμετρο του γράφου,

Βαθμός εξόδου	Αριθμός Κόμβων	Αξιοπιστία 95% → k	Αξιοπιστία 99% → k
2	30	6	9
	40	8	10
	50	9	12
	100	12	17
	200	15	21
	300	18	22
	400	20	24
	500	20	26
3	30	6	8
	40	7	9
	50	8	9
	100	11	14
	200	13	16
	300	14	18
	400	15	21
	500	16	20
6	30	6	8
	40	7	8
	50	7	9
	100	9	12
	200	10	14
	300	11	14
	400	11	14
	500	12	15

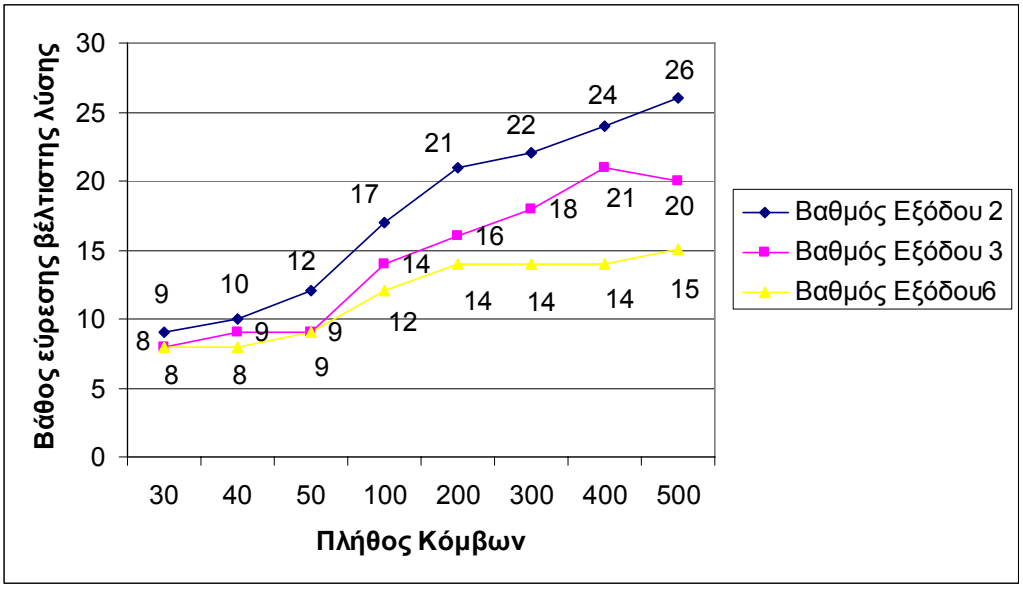
Πίνακας 5.2: **Βάθος k σε σχέση με το μέγιστο βαθμό εξόδου του γράφου**

για αυτό επιχειρήσαμε το παρακάτω πείραμα.

Στο πείραμα αυτό χρησιμοποιήσαμε γράφους με 30, 40, 50, 100, 200, 300, 400 και 500 κόμβους και πιθανότητα δημιουργίας κόμβων Or 20% και κόμβων And 10% ενώ αλλάξαμε το μέγιστο βαθμό εξόδου των κόμβων σε 2, 3, 6. Κάθε φορά εκτελούσαμε τον βέλτιστο αλγόριθμο για να βρούμε τη βέλτιστη λύση και στη συνέχεια εκτελούσαμε τον απλό Greedy αλγόριθμο με βήμα 1,2 κ.λ.π. μέχρι να βρεθεί η βέλτιστη λύση, όπως ακριβώς και στο προηγούμενο πείραμα. Εκτελέσαμε και αυτό το πείραμα για 1000 γράφους με κάθε συνδυασμό των στοιχείων εισόδου που μόλις περιγράφηκαν και κρατήσαμε το βάθος k στο οποίο σταματούσαμε κάθε φορά.

Στον Πίνακα 5.2 φαίνονται τα αποτελέσματα του πειράματος αυτού. Με τον όρο αξιοπιστία στον πίνακα αναφερόμαστε, ξανά, στην πιθανότητα εύρεσης της βέλτιστης λύσης αν εκτελέσουμε τον αλγόριθμο με βήμα k. Παρατηρούμε ότι όσο αυξάνει ο βαθμός εξόδου των κόμβων του γραφήματος τόσο μειώνεται το k το οποίο πρέπει να χρησιμοποιήσουμε για να έχουμε αποτελέσματα που να πλησιάζουν τα βέλτιστα.

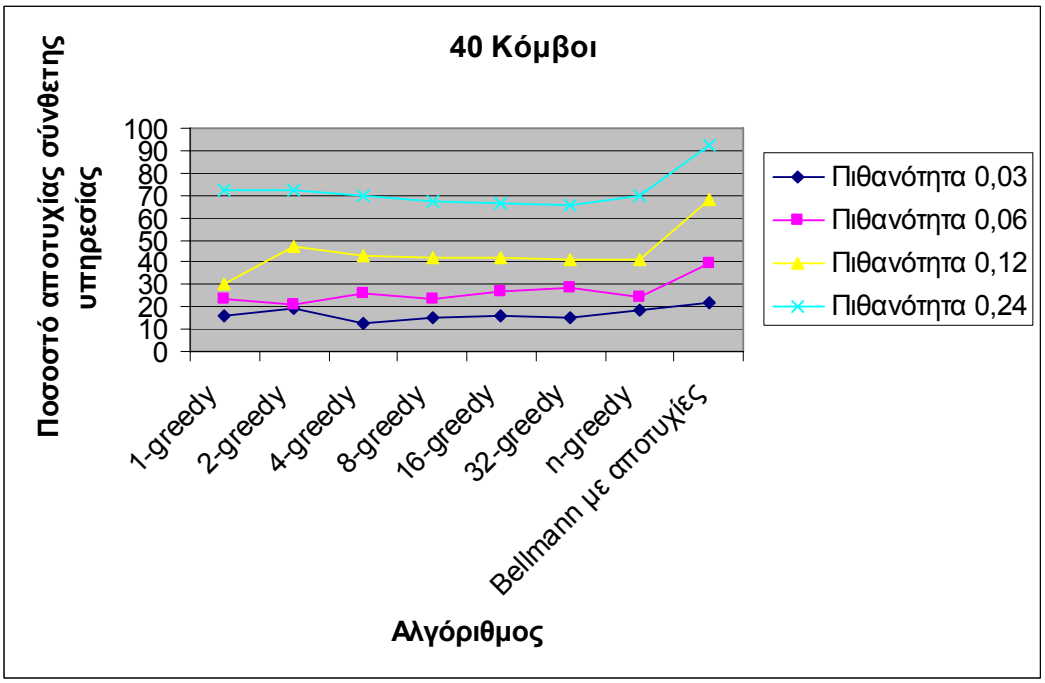
Στο Σχήμα 5.7 φαίνεται γραφικά το βάθος k για την εύρεση βέλτιστων λύσεων σε ποσοστό 99%. Από το διάγραμμα αυτό παρατηρούμε ότι ενώ για μικρό αριθμό κόμβων το βάθος k δεν επηρεάζεται πολύ από την αύξηση ου μέγιστου βαθμού εξόδου, συμβαίνει ακριβώς το αντίθετο για μεγάλο πλήθος κόμβων.



Σχήμα 5.7: k για βέλτιστα αποτελέσματα σε ποσοστό 99%

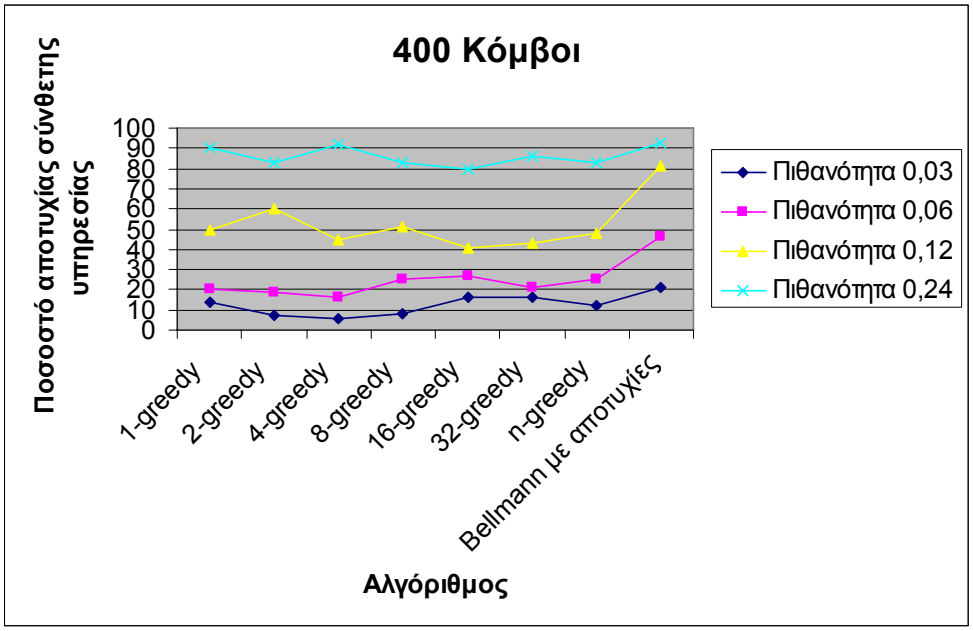
5.6. Πείραμα 6ο: Πιθανότητα αποτυχίας σύνθετης υπηρεσίας

Στο πείραμα αυτό στόχος μας ήταν να μελετήσουμε τη συμπεριφορά του greedy αλγορίθμου σε σχέση με την πιθανότητα αποτυχίας μιας από τις απλές υπηρεσίες που συνθέτουν τη ροή εργασίας.



Σχήμα 5.8: Απεικόνιση αποτελεσμάτων 6ου πειράματος (40 κόμβοι)

Πιο συγκεκριμένα θελήσαμε να δούμε πόσο πιθανό είναι να αποτύχει η σύνθετη υπηρεσία και αν αυτή η πιθανότητα σχετίζεται με την αντίστοιχη των απλών υπηρεσιών που την συνθέτουν. Εκτελέσαμε το πρόγραμμα που υλοποιεί greedy αλγόριθμο με βάθος k 1, 2, 4, 8, 16, 32, n και την παραλλαγή του Bellman με αποτυχίες, για γράφους με 40 και 400 κόμβους και πιθανότητα δημιουργίας κόμβων Or 20% και κόμβων And 10% ενώ το βαθμό εξόδου αφήσαμε σταθερό και ίσο με 2. Για κάθε γράφο εκτελέσαμε το πρόγραμμα με πιθανότητες αποτυχίας απλής υπηρεσία 0,03, 0,06, 0,12 και 0,24.



Σχήμα 5.9: Απεικόνιση αποτελεσμάτων 4ου πειράματος (400 κόμβοι)

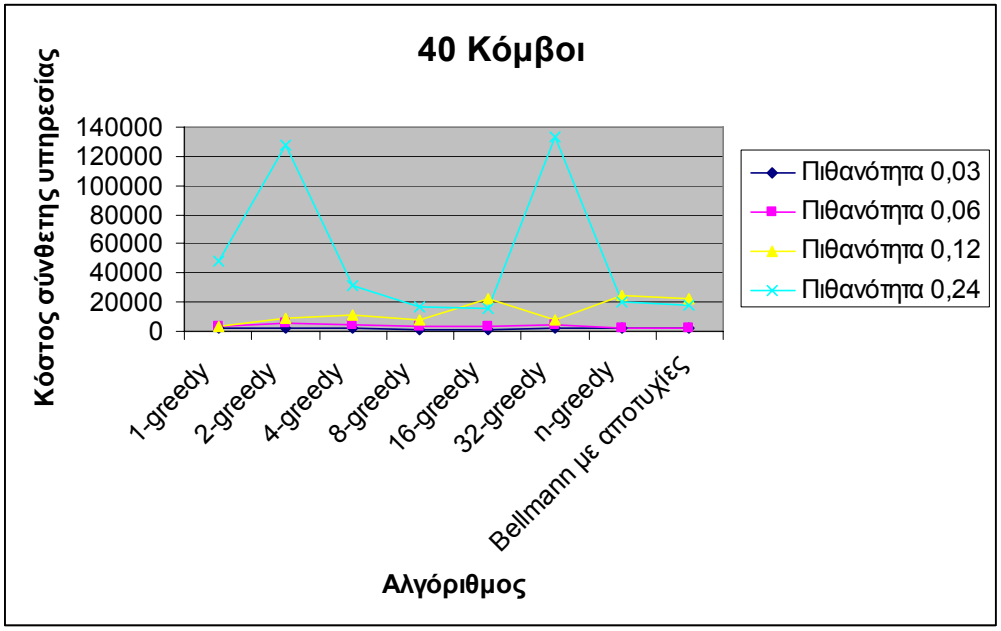
Από τα Σχήματα 5.8 και 5.9 συμπεραίνουμε ότι, όπως είναι λογικό, αύξηση της πιθανότητας αποτυχίας των απλών υπηρεσιών προκαλεί αύξηση της πιθανότητας αποτυχίας της σύνθετης υπηρεσίας. Επίσης παρατηρούμε ότι διπλασιάζοντας τη πιθανότητα αποτυχίας των απλών υπηρεσιών αυξάνεται στο διπλάσιο περίπου και η πιθανότητα αποτυχίας της σύνθετης υπηρεσίας.

Για να μειώσουμε κάπως την πιθανότητα αποτυχίας μπορούμε να χρησιμοποιήσουμε τις όμοιες υπηρεσίες που αναφέρθηκαν σε προηγούμενο κεφάλαιο. Σχετικό πείραμα υπάρχει στην παράγραφο 5.8.

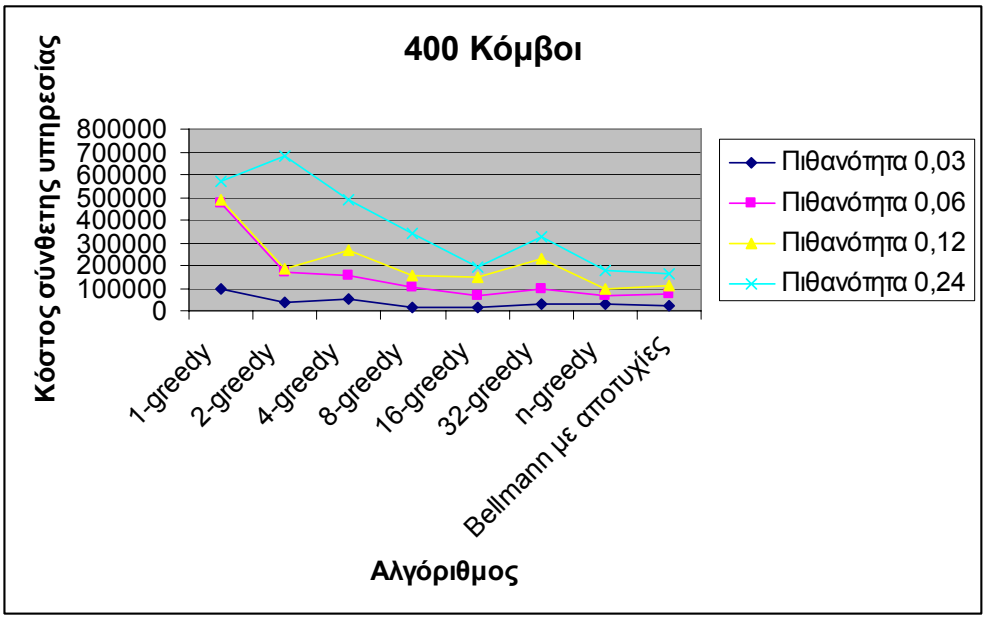
5.7. Πείραμα 7ο: Κόστος σύνθετης υπηρεσίας με αποτυχίες

Στο πείραμα αυτό στόχος μας ήταν να μελετήσουμε το κόστος της σύνθετης υπηρεσίας σε σχέση με την πιθανότητα αποτυχίας μιας από τις απλές υπηρεσίες που συνθέτουν τη ροή εργασίας. Εκτελέσαμε το πρόγραμμα που υλοποιεί τον greedy αλγόριθμο με βάθος k 1, 2, 4, 8, 16, 32, n και την παραλλαγή του Bellman με αποτυχίες, για γράφους με 40 και 400 κόμβους και πιθανότητα δημιουργίας κόμβων

Or 20% και κόμβων And 10% ενώ το βαθμό εξόδου αφήσαμε σταθερό και ίσο με 2. Για κάθε ροή εργασίας τρέξαμε τους αλγορίθμους με πιθανότητες αποτυχίας απλής υπηρεσία 0,03, 0,06, 0,12 και 0,24.



Σχήμα 5.10: Απεικόνιση αποτελεσμάτων 7ου πειράματος (40 κόμβοι)

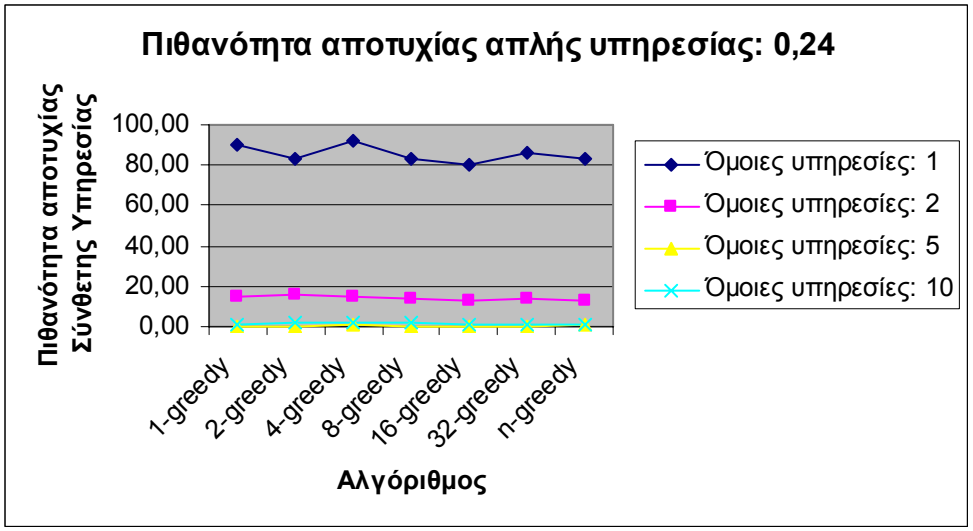


Σχήμα 5.11: Απεικόνιση αποτελεσμάτων 7ου πειράματος (400 κόμβοι)

Όπως ήταν αναμενόμενο το κόστος της υπηρεσίας όταν χρειαστεί να γυρίσουμε πίσω αυξάνει συμπέρασμα που βγαίνει αν συγκρίνουμε τις τιμές του κόστους που φαίνονται στα Σχήματα 5.10 και 5.11 με αυτές του Σχήματος 5.1. Επιπλέον παρατηρώντας τα Σχήματα 5.10, 5.11 γίνεται φανερό ότι αύξηση της πιθανότητας αποτυχίας των απλών υπηρεσιών από τις οποίες αποτελείται η ροή εργασίας της σύνθετης συνεπάγεται συνήθως και αύξηση του κόστους της σύνθετης υπηρεσίας γεγονός που οφείλεται κυρίως στην εκτέλεση και αναίρεση περισσότερων υπηρεσιών όταν υπάρχει μεγαλύτερη πιθανότητα αποτυχίας.

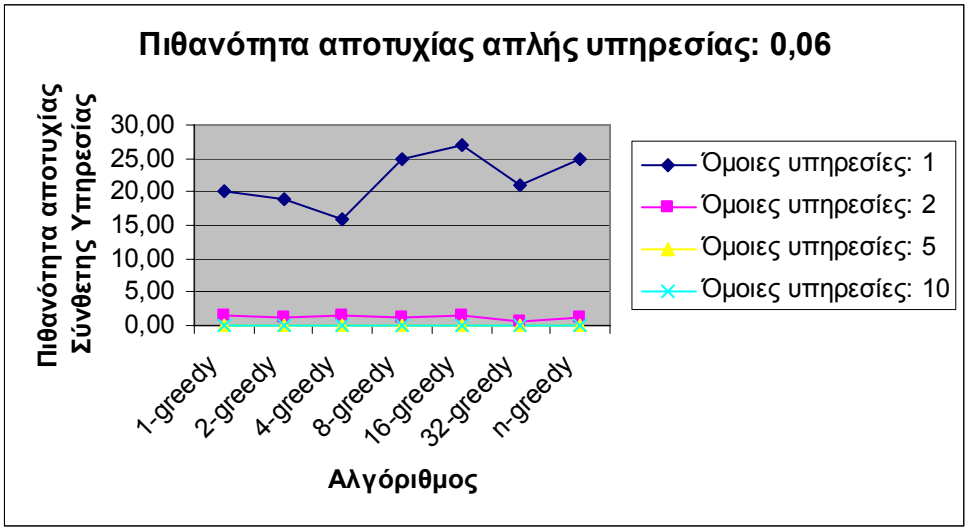
5.8. Πείραμα 8ο: Πιθανότητα αποτυχίας σύνθετης υπηρεσία με χρήση όμοιων υπηρεσιών

Στο κεφάλαιο 4 ορίστηκαν ως όμοιες υπηρεσίες, απλές υπηρεσίες διαδικτύου που επιτελούν την ίδια ή παρόμοια εργασία και στις οποίες αναφερόμαστε σαν να ήταν μια απλή υπηρεσία. Στο ίδιο κεφάλαιο αναφέραμε ότι η χρήση των όμοιων υπηρεσιών μειώνει το ποσοστό αποτυχίας των σύνθετων υπηρεσιών. Το πείραμα αυτό στόχο είχε να μελετήσει κατά ποσό αυτή η αναμενόμενη μείωση του κόστους υπάρχει και αν υπάρχει πόσο αξιόλογη είναι. Για το σκοπό αυτό χρησιμοποιήσαμε γράφους με 40 και 400 κόμβους, πιθανότητα δημιουργίας κόμβου and 10% και κόμβου or σε 20%, το μέγιστο βαθμό εξόδου των κόμβων τον αφήσαμε σταθερό και ίσο με 2. Κάθε φορά εκτελούσαμε τον απλό Greedy αλγόριθμο με βήμα 1, 2, 4, 8, 16, 32, n. Εκτελέσαμε το παραπάνω πείραμα για 1000 γράφους με κάθε συνδυασμό των στοιχείων εισόδου που μόλις περιγράφηκαν.



Σχήμα 5.12: Απεικόνιση αποτελεσμάτων 8ου πειράματος (πιθανότητα 0,24, κόμβοι 400)

Στο σχήμα 5.12 φαίνεται ότι η ύπαρξη ακόμη και μιας δεύτερης εναλλακτικής υπηρεσίας μειώνει αισθητά την πιθανότητα αποτυχίας της σύνθετης υπηρεσίας από 85% περίπου σε μικρότερο από 20% ενώ η ύπαρξη τριών ή περισσότερων όμοιων υπηρεσιών για κάθε λειτουργία που υπάρχει στη ροή εργασία απαλείφει θα λέγαμε τελείως την πιθανότητα αποτυχίας. Σημειώνεται ότι η πιθανότητα αποτυχίας των απλών υπηρεσιών στο πείραμα του οποίου τα αποτέλεσμα βλέπουμε στο Σχήμα 5.12 είναι αρκετά μεγάλη (0,24) και η εκτέλεση χωρίς την ύπαρξη όμοιων υπηρεσιών ήταν σχεδόν αδύνατη. Στο Σχήμα 5.13 φαίνεται ότι όταν οι απλές υπηρεσίες έχουν μικρότερη πιθανότητα αποτυχίας η πιθανότητα αποτυχίας της σύνθετης εξαλείφεται ακόμη και με την ύπαρξη δύο μόνο εναλλακτικών υπηρεσιών.



Σχήμα 5.13: Απεικόνιση αποτελεσμάτων 8ου πειράματος (πιθανότητα 0,06, κόμβοι 400)

Το ίδιο χρήσιμή φάνηκε η ύπαρξη όμοιων υπηρεσιών και σε ροές εργασίας με λιγότερους κόμβους για τις οποίες εκτελέσαμε το ίδιο ακριβώς πείραμα.

5.9. Σύγκριση μεθόδων

Στη παρούσα εργασία μελετήσαμε τη συμπεριφορά του greedy αλγόριθμου για την επιλογή της καλύτερης σύνθετης υπηρεσίας που προτάθηκε σε σχέση με τη συμπεριφορά του κλασσικού αλγορίθμου Bellman – Ford. Η προσομοίωση έδειξε ότι οι επιδόσεις του αλγορίθμου που προτάθηκε είναι πολύ καλές ιδίως όταν το βήμα του greedy αλγορίθμου είναι αρκετά μικρότερο από το πλήθος των κόμβων του γράφου. Συγκρίνοντας λοιπόν τις μεθόδους θα λέγαμε ότι αν βεβαία επιδιώκουμε τη βέλτιστη λύση τότε ο αλγόριθμος μας δεν είναι ο κατάλληλος αν όμως μας ικανοποιεί μια λύση πολύ κοντά στη βέλτιστη τότε ο αλγόριθμος που προτάθηκε είναι ιδανικός γιατί πετυχαίνει πολύ καλά αποτελέσματα ενώ υλοποιείται εύκολα και δεν χρησιμοποιεί πολύπλοκες τεχνικές για να φθάσει στο επιθυμητό αποτέλεσμα. Επιπλέον η εκτέλεση του αλγορίθμου, με βάση την θεωρητική πολυπλοκότητα των αλγορίθμων, γίνεται πολύ γρηγορότερα. Υπενθυμίζουμε ότι η πολυπλοκότητα του Bellmann – Ford είναι $O(n^2*m)$ ενώ του greedy $O(n*k)$. Παραπάνω όπου n αναφερόμαστε στο πλήθος

κόμβων, m στο πλήθος των ακμών και k στο βάθος εκτέλεσης του greedy αλγορίθμου. Ένα επίσης πλεονέκτημα της λύσης που προτάθηκε είναι ότι μπορεί να εκτελεστεί κατανεμημένα και σε περίπτωσης ακόμα που το πλάνο εκτέλεσης ολόκληρης της συνθέτης υπηρεσίας δεν είναι γνωστό σε όλους τους κόμβους. Επιπλέον επειδή μιλάμε για υπηρεσίες Διαδικτύου όπου συχνά υπάρχουν αποτυχίες ο υπολογισμός της βέλτιστης λύσης ακόμα και αν μπορεί να γίνει θα είναι αρκετά χρονοβόρος χωρίς να εγγυάται κανείς ότι η υπηρεσία θα μπορέσει να εκτελεστεί με το βέλτιστο τελικά τρόπο. Ενώ γυρνώντας πίσω σε περίπτωση αποτυχίας τελικά και πάλι δεν παίρνουμε τη βέλτιστη λύση συνολικά.

ΚΕΦΑΛΑΙΟ 6. ΣΥΜΠΕΡΑΣΜΑΤΑ & ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Με το κεφάλαιο αυτό τελειώνει η παρούσα εργασία. Για το λόγο αυτό εδώ αναφέρονται τα συμπεράσματα που βγήκαν από αυτή τη δουλειά καθώς επίσης και οι δυνατότητες επέκτασης αυτής της εργασίας.

6.1. Συμπεράσματα

Με βάση το προηγούμενο κεφάλαιο η προσομοίωση έδειξε ότι ο greedy αλγόριθμος που προτάθηκε για την επιλογή των υπηρεσιών που θα εκτελεστούν με στόχο την εκτέλεση μιας σύνθετης υπηρεσίας έχει αρκετά καλές επιδόσεις. Πιο συγκεκριμένα φαίνεται όπως αναφέρεται και παραπάνω να πετυχαίνει σχεδόν τη βέλτιστη υπηρεσία με βήμα ελέγχου k πολύ μικρότερο από το πλήθος των κόμβων της ροής εργασίας δηλαδή χωρίς να κάνει εξαντλητικό έλεγχο για το κόστος όλων των υπηρεσιών που αποτελούν τη ροή εργασίας. Επιπλέον, όπως ήδη αναφέραμε από την αρχή, ο αλγόριθμος αυτός μπορεί να εκτελείται κομμάτι - κομμάτι χωρίς να χρειάζεται να γνωρίζουμε ολόκληρη τη ροή εργασίας που περιγράφει τη σύνθετη υπηρεσία.

Παρατηρούμε επίσης ότι χρησιμοποιώντας τις ροές εργασίας για τον ορισμό των σύνθετων υπηρεσιών μπορούμε να αποφύγουμε την ακύρωση της εκτέλεσης μιας σύνθετης υπηρεσίας ακόμα και όταν μια από τις υπηρεσίες για κάποιο λόγο δε λειτουργεί εκτελώντας ένα εναλλακτικό μονοπάτι της ροής εργασίας.

6.2. Μελλοντικές επεκτάσεις

Με βάση την προηγούμενη παράγραφο αφού τα αποτελέσματα της προσομοίωσης ήταν θετικά καλό θα ήταν να δοκιμαστή ο αλγόριθμος στη συνέχεια σε πραγματικές συνθήκες. Για το σκοπό αυτό θα πρέπει:

- Να δημιουργήσουμε ή και να ανακαλύψουμε στο Διαδίκτυο απλές υπηρεσίες για διάφορες εργασίες.
- Να δημιουργήσουμε ροές εργασίας που να εμπλέκουν τις προηγούμενες υπηρεσίες. Για να γίνει αυτό πρέπει να χρησιμοποιήσουμε κάποια από τις γλώσσες σύνθεσης υπηρεσιών που έχουν αναπτυχθεί είτε να δημιουργήσουμε το δικό μας πλαίσιο σύνθεσης υπηρεσιών. Το πλεονέκτημα της τελευταίας εκδοχής είναι ότι ίσως ετσι μπορέσουμε να χειριστούμε καλύτερα τις ροές στη συνέχεια.
- Να αναπτύξουμε ένα περιβάλλον εκτέλεσης των ροών εργασίας. Στο οποίο να ενσωματώσουμε και τους αλγορίθμους που προτάθηκαν σε αυτή την εργασία έτσι ώστε να ελέγξουμε σε πραγματικές συνθήκες πλέον τη συμπεριφορά τους.

ΑΝΑΦΟΡΕΣ

[1] Jens Graupmann, Gerhard Weikum: The Role of Web Services in Information Search. IEEE Data Eng. Bull. 25(4): 60-65 (2002)

[2] Asuman Dogac, Gokce Laleci, Yildiray Kabak, Ibrahim Cingil: ExploitingWeb Service Semantics: Taxonomies vs. Ontologies. CoRR cs.DB/0212051: (2002)

[3] Michael J. Carey, Michael Blevins, Pál Takácsi-Nagy: Integration, Web Services Style. IEEE Data Eng. Bull. 25(4): 17-21 (2002)

[4] David Burdett, Qiming Chen, Meichun Hsu: Conductor: An Enabler for Web Services-based Business Collaboration. IEEE Data Eng. Bull. 25(4): 22-26 (2002)

[5] Enzo Colombo, Chiara Francalanci, Barbara Pernici, Pierluigi Plebani, Massimo Mecella, Valeria De Antonellis, Michele Melchiori: Cooperative Information Systems in Virtual Districts: the VISPO Approach. IEEE Data Eng. Bull. 25(4): 36-40 (2002)

[6] Mike P. Papazoglou, Marco Aiello, Marco Pistore, Jian Yang: Planning for Requests against Web Services. IEEE Data Eng. Bull. 25(4): 41-46 (2002)

[7] Boualem Benatallah Quan Sheng Marlon Dumas: The Self-Serv Enviroment for Web Services Composition, IEEE computer society Jenuary - February 2003

[8] Boualem Beanatallah, Marlon Dumas, Quan Z. Sheng και Anne H.H. Ngu Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services Proceedings ICDE 02

[9] Michael Gillmann, Gerhard Weikum, Wolfgang Wonner: Workflow management with service quality guarantees. SIGMOD Conference 2002: 228-239

[10] Chintan Patel, Kaustubh Supekar, Yugyung Lee: A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows. DEXA 2003: 826-835

[11] Georgakopoulos, D., M.F. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modelling to Workflow Automation Infrastructure," Journal of Distributed and Parallel Databases 3,2. (1995)

[12] Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju: Web Services Concepts, Architectures and Applications, Springer Verlag 2004

[13] Frank Leymann, Dieter Roller: Production Workflow Concepts and Techniques, Prentice Hall PTR 1999

[14] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Z. Sheng, "Quality driven web services composition, " in Proc. of the 12th International World Wide Web Conference (WWW'03), Budapest, Hungary, May 2003.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ονομάζομαι Μαρία Παπαφώτη και γεννήθηκα στην Αθήνα το 1979. Μεγάλωσα στα Ιωάννινα από το 1982 όπου και κατοικώ μέχρι σήμερα. Αποφοίτησα από το 1^{ο} Λύκειο Ιωαννίνων το 1997. Το 2003 πήρα το Πτυχίο της Πληροφορικής της σχολής Θετικών επιστημών του Πανεπιστημίου Ιωαννίνων. Κατά τη διάρκεια των σπουδών μου το 2001 παρακολούθησα σεμινάριο με θέμα «Αποθήκες δεδομένων (Heldinet)» Το καλοκαίρι του 2001 στα πλαίσια πρακτικής άσκησης εργάστηκα στην εταιρεία Next-Com, όπου ασχολήθηκα με την κατασκευή δυναμικών ιστοσελίδων με τη χρήση PHP και Dreamweaver. Το Σεπτέμβριο του 2002 κατατάχτηκα στους μεταπτυχιακούς φοιτητές της ίδιας σχολής. Κατά τη διάρκεια των μεταπτυχιακών σπουδών μου ασχολήθηκα με το θέμα της παρούσας εργασίας «Επιλογή της καλύτερης σύνθετης υπηρεσίας διαδικτύου». Κατά το 1^{ο} έτος των μεταπτυχιακών σπουδών μου εργάστηκα στο ερευνητικό πρόγραμμα με τίτλο «DBGlobe: Μια δεδομενοκεντρική προσέγγιση στην παγκόσμια πληροφορική». Από το 2003 μέχρι σήμερα εργάζομαι ως καθηγήτρια πληροφορικής στο 1^{ο} ΤΕΕ Παραμυθιάς. Γνωρίζω 2 ξένες γλώσσες Αγγλικά και Γαλλικά που πιστοποιούνται για μεν τα Αγγλικά με το First Certificate in English του Πανεπιστημίου του Cambridge (1995) και για τα Γαλλικά με το Diplôme D' Etudes En Langue Française (2nd Degré) (1999) και το Certificat De Langue Française (1996)

ΑΝΑΦΟΡΕΣ

ΠΑΡΑΡΤΗΜΑ

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ