Ατομική Διπλωματική Εργασία

ΔΥΝΑΜΙΚΗ ΣΥΝΘΕΣΗ ΣΥΣΤΗΜΑΤΩΝ ΓΙΑ ΠΑΡΟΧΗ ΥΠΗΡΕΣΙΩΝ

Γεωργία Άρνου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2009

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Δυναμική σύνθεση συστημάτων για παροχή υπηρεσιών

Γεωργία Άρνου

Επιβλέπων Καθηγητής

Δρ. Χρίστος Σχίζας

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2009

Ευχαριστίες

Θα ήθελα καταρχήν, να ευχαριστήσω τον επιβλέποντα καθηγητή, Δρ. Χρίστο Σχίζα, για την συνεχή βοήθεια και την στήριξη που μου έχει προσφέρει κατά την διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω ιδιαιτέρως την οικογένεια μου, που με στήριξε και με ενθάρρυνε να προχωρήσω.

Περίληψη

Σε αυτή την διπλωματική εργασία ερευνείται το συνδυαστικό πρόβλημα της επιλογής των υπηρεσιών, για την σύνθεση μιας επιχειρηματικής διαδικασίας. Η επιχειρηματική διαδικασία αποτελείται από πολλά κομμάτια (στόχους) τα οποία πρέπει να εκτελεστούν για να παραχθεί η επιθυμητή λειτουργία. Το συνδυαστικό πρόβλημα προκύπτει λόγω του ότι για κάθε στόχο υπάρχουν πολλές λειτουργικά ισοδύναμες υπηρεσίες οι οποίες μπορούν να τον εκτελέσουν, αλλά η κάθε μια από αυτές έχει διαφορετικά μη λειτουργικά χαρακτηριστικά. Έτσι θα πρέπει να επιλεχθεί ο κατάλληλος συνδυασμός υπηρεσιών που όχι μόνο θα ανταποκρίνεται λειτουργικά στην σύνθετη λειτουργία, αλλά και η σφαιρική τιμή των μη λειτουργικών χαρακτηριστικών να είναι η βέλτιστη ανάλογα με τις απαιτήσεις του χρήστη.

Στην παρούσα έρευνα για την λύση αυτού του προβλήματος χρησιμοποιήθηκαν οι γενετικοί αλγόριθμοι, οι οποίοι λόγω του σχεδιασμού τους επιτυγχάνουν μια καλή λύση σε αποδοτικό χρόνο. Βασική πρόκληση ήταν η κωδικοποίηση του χρωμοσώματος και η εύρεση της κατάλληλης αντικειμενικής συνάρτησης (fitness function). Επίσης κατά την υλοποίηση του προγράμματος λήφθηκαν υπόψη κάποια προβλήματα τα οποία είχαν προηγούμενες προσεγγίσεις και έγινε προσπάθεια να λυθούν. Μερικά από αυτά είναι η παρουσίαση όλων των μονοπατιών της σύνθετης λειτουργίας ταυτόχρονα και η εύρεση και αποφυγή των κυκλικών μονοπατιών.

Λίστα Ακρωνύμων

Υπηρεσία: Είναι μια οντότητα που υλοποιεί μια χρήσιμη λειτουργία και αναπτύσσεται και τρέχει αυτόνομα [4, 7, 9].

Υπηρεσία Ιστού (Web Service): Είναι μια υπηρεσία η οποία αναγνωρίζεται από ένα URL[8, 10, 22].

Συγκεκριμένη Υπηρεσία: Η υπηρεσία η οποία υπάρχει, είναι ήδη υλοποιημένη, και μπορεί να εκτελέσει μια συγκεκριμένη λειτουργία. Κάθε συγκεκριμένη υπηρεσία χαρακτηρίζεται από κάποια μη λειτουργικά χαρακτηριστικά (QoS) [4].

Αφηρημένη Υπηρεσία: Είναι η σημασιολογική περιγραφή μιας υπηρεσίας η οποία χαρακτηρίζεται από μια λειτουργικότητα. Υπάρχουν πολλές συγκεκριμένες υπηρεσίες ,με διαφορετικά μη λειτουργικά χαρακτηριστικά, που ανταποκρίνονται στην λειτουργικότητα την οποία περιγράφει μια αφηρημένη υπηρεσία [4].

Σύνθεση Υπηρεσιών: Είναι ο τρόπος με τον οποίο είναι δομημένο το όλο σύστημα, με την σύνδεση(binding) μιας σειράς υπηρεσιών [7].

Σύνθετη Υπηρεσία: Η υπηρεσία η οποία προέρχεται από την λειτουργία της σύνθεσης υπηρεσιών. Ονομάζεται και τελική λειτουργία[4] .

Θετικά μη λειτουργικά χαρακτηριστικά: Είναι τα μη λειτουργικά χαρακτηριστικά των οποίων χρειάζεται να μεγιστοποιηθεί η τιμή τους[10].

Αρνητικά μη λειτουργικά χαρακτηριστικά: Είναι τα μη λειτουργικά χαρακτηριστικά των οποίων χρειάζεται να ελαχιστοποιηθεί η τιμή τους[1].

QoS (Quality of services): Ο όρος QoS αναφέρεται στα μη λειτουργικά χαρακτηριστικά τα οποία έχει μια Υπηρεσία Ιστού (Web Service)[7].

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Πινάκων

Κεφάλαιο 1

1. Εισαγωγή

Οι σύγχρονες επιχειρήσεις χρειάζεται να ανταποκριθούν γρήγορα και αποτελεσματικά στις ευκαιρίες του σήμερα, της ανταγωνιστικής και μοντέρνας αγοράς. Για να υπάρχει η επιχειρηματική ευκινησία, οι επιχειρήσεις οφείλουν να βελτιώσουν τις επιχειρησιακές τους διαδικασίες, και παράλληλα να εκθέτουν τις διάφορες εφαρμογές που βρίσκονται σε ολόκληρη την επιχείρηση. Οι εφαρμογές αυτές παρουσιάζονται μέσα στην επιχείρηση με ένα εξαιρετικά τυποποιημένο τρόπο. Μια σύγχρονη προσέγγιση για την αντιμετώπιση αυτών των κρίσιμων ζητημάτων είναι η χρήση των Υπηρεσιών Ιστού (Web Services), οι οποίες μπορούν εύκολα να συναρμολογηθούν για να αποτελέσουν μια συλλογή από αυτόνομες με χαλαρή σύζευξη επιχειρησιακές διαδικασίες[3].

Οι Υπηρεσιών Ιστού (Web Services) [22] εκτελούν κάποιες λειτουργίες ανάλογα με τις προδιαγραφές τους. Οι λειτουργίες αυτές κυμαίνονται από την απάντηση απλών αιτημάτων μέχρι και την εκτέλεση περίπλοκων επιχειρηματικών διαδικασιών. Οι περίπλοκες αυτές επιχειρηματικές διαδικασίες απαιτούν peer-to-peer σχέσεις, μεταξύ των πολλαπλών επιπέδων των παρεχόμενων υπηρεσιών, στους πελάτες και τους προμηθευτές τους. Κάθε κομμάτι κώδικα, καθώς και κάθε συστατική εφαρμογή που αναπτύσσεται σε ένα σύστημα μπορεί να επαναχρησιμοποιηθεί και να μετατραπεί σε μια διαθέσιμη Υπηρεσία Ιστού (Web Service). Οι Υπηρεσίες Ιστου (Web Services) είναι η πιο υποσχόμενη τεχνολογία στον καινούριο αυτό τομέα της Πληροφορικής, γιατί χρησιμοποιούν το διαδίκτυο για την επικοινωνία τους. Ανοίγουν τα πρότυπα τα οποία είναι βασισμένα στο διαδίκτυο για τη διαβίβαση των διάφορων δεδομένων. Επιπρόσθετα χρησιμοποιούν μια συγκεκριμένη γλώσσα, την WSDL (Web Services Description Language) για τον ορισμό των υπηρεσιών και μια άλλη γλώσσα, η BPEL (Business Process Execution Language for Web Services,BPEL4WS) που χρησιμοποιείται για την οργάνωση των υπηρεσιών [1].

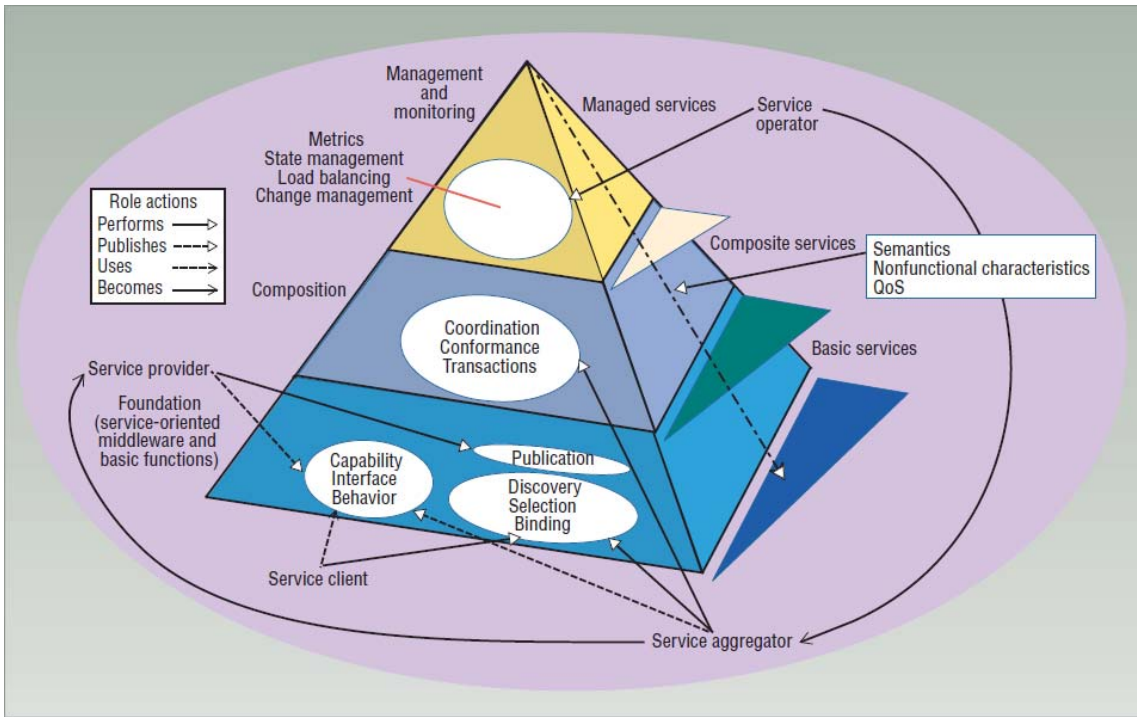
Η εμφάνιση της εξέλιξης αλλά και των προτύπων υπέρ της αυτοματοποιημένης επιχειρησιακής ολοκλήρωσης των Υπηρεσιών Ιστού (Web Services), οδήγησαν στην υλοποίηση των Service Oriented αρχιτεκτονικών (SOA). Στις προηγούμενες αρχιτεκτονικές όλα τα κομμάτια που συνέθεταν μια μεγάλη εφαρμογή βρίσκονταν κάτω από τον έλεγχο ενός ενιαίου οργανισμού, που ήταν υπεύθυνος για το σχεδιασμό, την ανάπτυξη και την επικύρωση των διάφορων διαδικασιών. Αντίθετα οι Service Oriented αρχιτεκτονικές παρέχουν την υποστήριξη στα διάφορά συστατικά (υπηρεσίες) και τις διασυνδέσεις τους να αλλάζουν δυναμικά. Οι υπηρεσίες έχουν την δυνατότητα να εκδοθούν, να ανευρεθούν και συνδεθούν μεταξύ τους δυναμικά[4].

Γενικά οι Υπηρεσίες Ιστού (Web Services) [22] όπως προαναφέρθηκε θεωρούνται ως η πιο υποσχόμενη τεχνολογία για ηλεκτρονικό εμπόριο και ανάπτυξη νέων συστημάτων λογισμικού, που είναι βασισμένα στο διαδίκτυο. Ένα παράδειγμα της χρήσης αυτής της νέας τεχνολογίας είναι η εφαρμογή για ενοικίαση αυτοκινήτου. Υπάρχουν πολλές εταιρείες (υπηρεσίες) από τις οποίες μπορείς να γίνει η ενοικίαση του αυτοκινήτου και από αυτές θα πρέπει να επιλεγεί η καταλληλότερη σύμφωνα πάντα με τις απαιτήσεις που υπάρχουν. Έτσι οι πολλαπλές υπηρεσίες με τα ίδια λειτουργικά χαρακτηριστικά αυξάνουν το πρόβλημα της επιλογής των υπηρεσιών. Οι πελάτες δεν περιμένουν μόνο η υπηρεσία που επιλέγουν να εκτελεί τη λειτουργία που χρειάζονται αλλά να έχει καλή ποιότητα (QoS) [7]. Μερικά από τα μη λειτουργικά χαρακτηριστικά που μπορεί να ενδιαφέρουν τον πελάτη είναι η διαθεσιμότητα, το κόστος και η αξιοπιστία. Αφού υπάρχουν τόσες πολλές υπηρεσίες με την ίδια λειτουργικότητα, θα πρέπει να επιλεγεί η βέλτιστη υπηρεσία, που ικανοποιεί τις απαιτήσεις του χρήστη στο επίπεδο των μη λειτουργικών χαρακτηριστικών. Η επιλογή των υπηρεσιών με αυτά τα κριτήρια παίζει σημαντικό ρόλο στην σύνθεση των υπηρεσιών. Ένα σχεδιάγραμμα που δείχνει την λειτουργικότητα των υπηρεσιών παρουσιάζεται στο Σχήμα 1.1. Στο σχήμα αυτό χωρίζεται η λειτουργικότητα σε τρία επίπεδα. Στο κάτω επίπεδο είναι τα «θεμέλια» των υπηρεσιών, στην μέση η σύνθεση των υπηρεσιών και στην κορυφή η διαχείριση τους.

Πολλές προσεγγίσεις υλοποιήθηκαν για επιλογή υπηρεσιών που συνθέτονται για την παραγωγή μιας επιχειρηματικής διαδικασίας (σύνθετης υπηρεσίας). Οι πιο

πρόσφατες προσεγγίσεις είναι καλύτερες στην ικανοποίηση των απαιτήσεων που αφορούν τα μη - λειτουργικά χαρακτηριστικά, από παλαιότερες προσεγγίσεις. Η σύνθεση των υπηρεσιών είναι ένα συνδυαστικό πρόβλημα βελτιστοποίησης, όπου ο καλύτερος συνδυασμός των υπηρεσιών που επιλέγονται, είναι ο βέλτιστος και είναι πάντοτε σύμφωνος με τους σφαιρικούς περιορισμούς.

Αρκετές από τις πρόσφατες προσεγγίσεις οι οποίες υλοποιήθηκαν είναι βασισμένες στους γενετικούς αλγορίθμους. Οι γενετικοί αλγόριθμοι είναι ένα πολύ δυνατό εργαλείο για την λύση συνδυαστικών προβλημάτων βελτιστοποίησης. Ο σχεδιασμός των γενετικών αλγορίθμων έχει την μεγαλύτερη επίδραση στην συμπεριφορά και την απόδοση του συνδυαστικού αυτού προβλήματος. Η κωδικοποίηση του χρωμοσώματος και η αντικειμενική συνάρτηση έχουν σημαντική επίδραση στην επίδοση και την σύγκλιση των γενετικών αλγορίθμων. Αυτοί οι αλγόριθμοι είναι πιο αποδοτικοί όταν το πεδίο λύσεων του προβλήματος είναι αρκετά μεγάλο[13].



Σχήμα 1.1: Σχεδιάγραμμα της σύνθεσης των υπηρεσιών [1]

Κεφάλαιο 2

2. Γνωσιολογικό υπόβαθρο

2.1 Γλώσσα Διαμόρφωσης των Υπηρεσιών (Service Modeling Language W3C)

2.1.1 Εισαγωγή

Η γλώσσα διαμόρφωσης των Υπηρεσιών (Service Modeling Language, SML) είναι μια γλώσσα η οποία παρέχει την δυνατότητα της δημιουργίας μοντέλων που αναπαριστούν σύνθετες υπηρεσίες και συστήματα, μέσω κάποιων κατασκευών. Βασισμένα στο πεδίο ορισμού της εφαρμογής, αυτά τα μοντέλα μπορεί να περιέχουν πληροφορίες όπως είναι η χωρητικότητα της υπηρεσίας, ο έλεγχος, η ανάπτυξη της, η πολιτική της υπηρεσίας κλπ. Εστιάζονται στην λήψη όλων των αμετάβλητων πτυχών μιας υπηρεσίας/συστήματος, οι οποίες πτυχές πρέπει να διατηρούνται ούτος ώστε να λειτουργεί ορθά αυτή η υπηρεσία/σύστημα την οποία αναπαριστά το μοντέλο. Επιπρόσθετα τα μοντέλα αυτά αντιπροσωπεύουν ένα πολύ δυνατό μηχανισμό για την επικύρωση κάποιων αλλαγών, πριν αυτές οι αλλαγές γίνουν στην ίδια την υπηρεσία/σύστημα. Κατά την διάρκεια που εφαρμόζονται αυτές οι αλλαγές στην τρέχον υπηρεσία/σύστημα, μπορούν να επικυρωθούν σύμφωνα με την προβλεπόμενη κατάσταση η οποία περιγράφεται στο μοντέλο. Τα μοντέλα αυτά μπορούν να περιγραφούν ως οντότητες επικοινωνίας και συνεργασίας μεταξύ σχεδιαστών, προγραμματιστών, φορέων και χρηστών, που μπορούν εύκολα να μοιράζονται, να παρακολουθούνται και να ελέγχονται κατά την αναθεώρηση. Αυτή η ιδιότητα είναι σημαντική γιατί συχνά οι σύνθετες υπηρεσίες κατασκευάζονται και συντηρούνται από πολλούς ανθρώπους οι οποίοι παίζουν διαφορετικούς ρόλους.

Ένα ακόμα από τα πλεονεκτήματα αυτών των μοντέλων είναι ότι οδηγούν τη διαμόρφωση, την τυποποίηση και την επαναχρησιμοποίηση κώδικα. Αυτοί οι τρείς παράγοντες είναι πολύ σημαντικοί στην σύνθετη υπηρεσία, γιατί χτίζεται από πολλά πολύπλοκα κομμάτια και έτσι εξασφαλίζεται η αξιοπιστία και μειώνεται το κόστος της σύνθεσης. Επιπρόσθετα λόγω αυτού του πλεονεκτήματος γίνεται καλύτερός ο έλεγχος των διαφόρων στόχων της.

Πιο συγκεκριμένα σε επίπεδο υλοποίησης ένα μοντέλο γραμμένο σε Service Modeling Language ουσιαστικά είναι ένα σύνολο από έγγραφα που είναι σε μορφή XML τα και είναι άμεσα συνδεδεμένα μεταξύ τους [14]. Τα έγγραφα σε μορφή XML [18] περιέχουν πληροφορίες για τα διάφορα κομμάτια της υπηρεσίας, και τους περιορισμούς οι οποίοι θα πρέπει να ικανοποιούνται για την ορθή λειτουργία της υπηρεσίας. Αυτοί οι περιορισμοί εκφράζονται σε δύο μορφές: τους κανόνες (Rules) και τα σχήματα (Schemas). Τα σχήματα είναι διάφοροι περιορισμοί πάνω στη δομή και το περιεχόμενο των εγγράφων σε ένα μοντέλο. Οι κανόνες είναι λογικές εκφράσεις που περιορίζουν την δομή και το περιεχόμενο των εγγράφων σε ένα μοντέλο.

Μια από τις σημαντικότερες λειτουργίες πάνω στο μοντέλο είναι η εγκαθίδρυση της εγκυρότητάς του. Αυτή η λειτουργία συμπεριλαμβάνει τον έλεγχο όλων των δεδομένων έτσι ώστε να ικανοποιούνται όλα τα σχήματα και οι κανόνες οι οποίοι ορίστηκαν. Τα SML σενάρια απαιτούν πολλά χαρακτηριστικά τα οποία είτε δεν υπάρχουν είτε δεν υποστηρίζονται πλήρως από τα XML σχήματα (schemas). Τα χαρακτηριστικά αυτά κατηγοριοποιούνται στις αναφορές (SML references) και τους κανόνες (rules) [14].

2.1.2 Αναφορές SML (SML References)

Οι πληροφορίες των στοιχείων κάποιου αντικειμένου που περιγράφεται σε ένα στιγμιότυπο ενός εγγράφου κάποιου SML μοντέλου, ορίζονται σαν μια αναφορά SML (SML Reference) αν και μόνο αν υπάρχει μια πληροφορία κάποιου χαρακτηριστικού του αντικειμένου για το οποίο όλοι οι παρακάτω περιορισμοί ισχύουν. Ο πρώτος περιορισμός είναι ότι το τοπικό όνομα (local name) πρέπει να είναι το ref. Ο δεύτερος περιορισμός είναι ότι το σφαιρικό όνομα (namespace name) είναι το http://www.w3.org/ns/sml και ο τρίτος είναι ότι η κανονικοποιημένη τιμή μετά την κανονικοποίηση (whitespace κανονικοποίηση) ακολουθώντας τους κανόνες σχήματος είναι είτε αληθής (true) ή 1. Ένα παράδειγμα που δείχνει τα παραπάνω παρουσιάζεται παρακάτω στο Σχήμα 2.1.

```
<RefElement sml:ref="true">
    <sml:uri>targetDocument.xml</sml:uri>
</RefElement>
```

Σχήμα 2.1: Αναφορά SML (SML Reference) [14]

Έκτος από την απλή αναφορά υπάρχει και η κενή αναφορά (null reference). Μια αναφορά θεωρείται κενή αν ισχύουν οι δύο τελευταίοι περιορισμοί όπως αναφέρονται στην αναφορά SML (SML Reference) παραπάνω και ο πρώτος διαφοροποιείται ως εξής: το τοπικό όνομα (local name) είναι το nilref. Στο Σχήμα 2.2 παρουσιάζεται ένα παράδειγμα κενής αναφοράς.

```
<RefElement sml:ref="true" sml:nilref="true">
    <sml:uri>targetDocument.xml</sml:uri>
</RefElement>
```

Σχήμα 2.2: Κενή αναφορά SML [14]

Ο κόμβος του στοιχείου που περιγράφεται/επιλύεται από μια μη κενή αναφορά ορίζεται ως SML στόχος. Ο SML στόχος μιας SML αναφοράς (SML Reference) πρέπει να είναι κομμάτι του ίδιου SML μοντέλου όπως και η SML αναφορά (SML Reference). Μια κενή αναφορά δεν έχει SML στόχο. Στο Σχήμα 2.3 παρουσιάζεται ένας τέτοιος στόχος [14].

```
<RefElement sml:ref="true" xmlns:e="urn:example">
    <sml:uri>target.xml#smlxpath1(e:Course[2])</sml:uri>
</RefElement>

document 'target.xml':
----------------------
<Courses xmlns="urn:example">
    <Course>
        <Name>PHY101</Name>
        <Grade>A</Grade>
    </Course>
    <Course>
        <Name>MAT101</Name>
        <Grade>A</Grade>
    </Course>
</Courses>
```

Σχήμα 2.3: SML αναφορά (SML reference) με στόχο τα μαθήματα [14]

2.1.3 Σχήματα Αναφοράς SML (SML Reference Schemes)

Μια SML αναφορά (SML Reference) μπορεί να είναι ένα στιγμιότυπο μιας ποικιλίας από σχήματα αναφοράς (SML Reference Schemes). Η γλώσσα SML δεν απαιτεί τη χρήση κάποιων συγκεκριμένων σχημάτων αναφοράς. Ένα SML σχήμα αναφοράς (SML Reference Schemes) μπορεί να χρησιμοποιήσει στοιχεία ενός αντικειμένου, ή κάποια χαρακτηριστικά του, ή ένα συνδυασμό των δύο ή τίποτα από τα δύο για να συλλαμβάνει τις απαραίτητες πληροφορίες για να αναγνωρίζεται ο αναφερόμενος SML στόχος. Δεν είναι απαραίτητο το γεγονός, ότι σε όλα τα στοιχεία ενός αντικειμένου σε ένα μοντέλο SML μπορεί να υπάρξει πρόσβαση μέσω μιας SML αναφοράς (SML Reference). Αν θα υπάρξει πρόσβαση εξαρτάται από την υποστήριξη που ορίζει το σχήμα αναφοράς(SML Reference Scheme) που επιλέγεται. Παρόλο που ένα SML μοντέλο δεν απαιτεί τη χρήση κάποιων συγκεκριμένων σχημάτων αναφοράς, προσδιορίζει πως μια αναφορά πρέπει να αναπαριστάται κατά την χρήση σχημάτων αναφοράς SML[14].

2.1.4 Κανόνες (Rules)

Το XML σχήμα (XML Schema) που χρησιμοποιείται από την γλώσσα SML υποστηρίζει έναν αριθμό από περιορισμούς αλλά δεν υποστηρίζει μια γλώσσα για καθορισμό κανόνων οι οποίοι περιορίζουν την δομή και το περιεχόμενο των εγγράφων. Για να μπουν αυτοί οι περιορισμοί η γλώσσα SML χρησιμοποιεί το Schematron, το οποίο είναι ένα ISO/IEC πρότυπο για τον ορισμό ισχυρισμών (assertions) σχετικά με ένα σύνολο XML αρχείων. Στο Σχήμα 2.4 φαίνεται ένα παράδειγμα το οποίο χρησιμοποιεί τους περιορισμούς. Αυτοί οι προσδιορισμοί των περιορισμών γίνονται με τις παρακάτω δηλώσεις: sch:assert, sch:report. Οι δηλώσεις αυτές είναι στοιχειά από το Schematron. Οι περιορισμοί οι οποίοι παρουσιάζονται στο παράδειγμα είναι ότι μια διεύθυνση IPv4 πρέπει να είναι τέσσερα bytes και μια IPv6 διεύθυνση πρέπει να είναι δεκαέξι bytes.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:x-example:IPAddress">
    <xs:simpleType  name="IPAddressVersionType">
        <xs:restriction base="xs:string" >
            <xs:enumeration value="V4" />
            <xs:enumeration value="V6" />
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="IPAddress">
        <xs:annotation>
            <xs:appinfo>
                <sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
                   <sch:ns prefix="tns" uri="urn:x-example:IPAddress" />
                   <sch:pattern id="Length">
                      <sch:rule context=".">
                         <sch:assert test="tns:version != 'V4' or count(tns:address) = 4">
                              A v4 IP address must have 4 bytes.
                         </sch:assert>
                         <sch:assert test="tns:version != 'V6' or count(tns:address) = 16">
                              A v6 IP address must have 16 bytes.
                         </sch:assert>
                      </sch:rule>
                   </sch:pattern>
                </sch:schema>
            </xs:appinfo>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="version" type="tns:IPAddressVersionType" />
            <xs:element name="address" type="xs:byte" minOccurs="4" maxOccurs="16" />
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Σχήμα 2.4: Παράδειγμα χρήσης κανόνων βάση το Schematron [14]

Ουσιαστικά οι ιδιότητες των κανόνων στην SML γλώσσα περιλαμβάνουν όλους τους περιορισμούς του Schematron που ισχύουν στα διάφορα στιγμιότυπα δεδομένης μιας δήλωσης κάποιου τύπου ή κάποιου στοιχείου. Η τιμή των κανόνων προέρχεται ως ένα σημείο από τα sch:schema στοιχεία, τα οποία είναι τα πρώτα στοιχειά η αλλιώς η ρίζα του Schematron. Συνήθως τα sch:schema στοιχεία περιέχουν δηλώσεις σε σχέση με το σφαιρικό όνομα (namespace) πχ. <sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">, και είναι ενσωματωμένα μέσα σε ένα συστατικό στοιχείο στην περίπτωση της SML. Η τιμή των κανόνων μπορεί επίσης να προέρχεται ως ένα σημείο από τις ιδιότητες των κανόνων άλλων συστατικών στοιχείων[14].

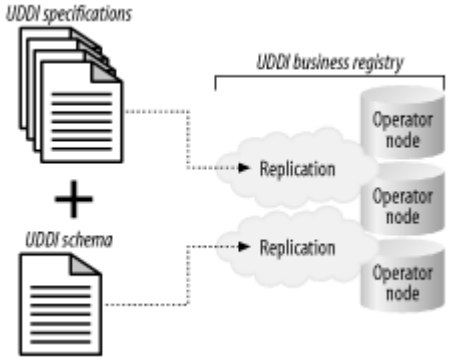
2.2 UDDI

Το UDDI [21] το οποίο είναι συντομογραφία του Universal Description Discovery Integration παρέχει μια τυποποιημένη μέθοδο για έκδοση και εύρεση πληροφοριών για τις Υπηρεσίες Ιστού(Web Services). Είναι μια βιομηχανία ανεξάρτητη της πλατφόρμας του υπολογιστή που χρησιμεύει στην περιγραφή και εύρεση των υπηρεσιών και των προμηθευτών τους. Οι προμηθευτές των υπηρεσιών Ιστού(Web

Services) είναι συνήθως εταιρείες και οργανισμοί. Οι πληροφορίες για τις υπηρεσίες οι οποίες αποθηκεύονται στο UDDI ακολουθούν ένα συγκεκριμένο δομημένο σχήμα και πολλές από τις υλοποιήσεις του UDDI χρησιμοποιούν ένα σχεσιακό σχήμα βάσεων δεδομένων για διαχείριση της αποθήκευσης των πληροφοριών αυτών. Είναι αρμοδιότητα των προμηθευτών των υπηρεσιών οι πληροφορίες που αποθηκεύονται να διατηρούνται, δηλαδή να παραμένουν εγγεγραμμένες και να μπορούν να ενημερώνονται. Επιπρόσθετα, οι προμηθευτές είναι υπεύθυνοι να μπορούν οι χρήστες μέσω κάποιου ερωτήματος να ανακτήσουν τις πληροφορίες που είναι αποθηκευμένες, έτσι ώστε αν θέλουν να χρησιμοποιήσουν κάποια από τις web services. Πιο γενικά το UDDI εστιάζεται στην διαδικασία της εύρεσης σε Service Oriented αρχιτεκτονικές.

Πριν να δημιουργηθεί το UDDI δεν υπήρχε κάποιο είδος βιομηχανίας, η οποία να μπορούν οι διάφοροι οργανισμοί να ενημερώσουν τους πελάτες – χρήστες για τις υπηρεσίες τις οποίες παρέχουν. Επίσης δεν υπήρχε μια κοινή μέθοδος που να δίνει λεπτομέρειες για το πώς μπορούν να αλληλεπιδράσουν τα συστήματα και οι διαδικασίες, που υπάρχουν ήδη, μεταξύ τους. Αυτές τις αδυναμίες προσπάθησε να καλύψει το UDDI με το να μπορούν οι διάφοροι οργανισμοί να εγγράφουν πληροφορίες σχετικά με το τι παρέχουν σε αυτό. Οι διάφοροι οργανισμοί μπορούν να δώσουν τριών ειδών πληροφορίες. Το πρώτο είδος είναι αυτό που αναφέρεται στις βασικές πληροφορίες επικοινωνίας και αναγνώρισης μιας εταιρίας (white pages). Οι συγκεκριμένες πληροφορίες επιτρέπουν στους άλλους να βρίσκουν τις Υπηρεσίες Ιστού (Web Services) που παρέχει ο οργανισμός αυτός βασισμένοι στην ταυτοποίηση του. Το δεύτερο είδος πληροφοριών είναι αυτές που περιγράφουν μια υπηρεσία ιστού (Web Service) χρησιμοποιώντας διαφορετικές κατηγοριοποιήσεις (yellow pages). Αυτές οι πληροφορίες επιτρέπουν την εύρεση των υπηρεσιών ιστού (Web Services) βάση της κατηγορίας τους. Το τρίτο και τελευταίο είδος πληροφοριών που μπορεί να αποθηκευτούν είναι τεχνικές πληροφορίες που περιγράφουν τις συμπεριφορές και τις συναρτήσεις που υποστηρίζουν μια υπηρεσία ιστού (Web Service) που βρίσκεται μέσα σε ένα οργανισμό (green pages). Το τρίτο είδος πληροφοριών περιέχουν και στοιχεία για τον χώρο στον οποίο βρίσκεται η υπηρεσία ιστού (Web Services).

Το UDDI μπορεί να χρησιμοποιηθεί από πολλούς ανθρώπους για διαφορετικούς σκοπούς ανάλογα με τις απαιτήσεις του επαγγέλματός τους. Σε περίπτωση που χρησιμοποιείται από κάποιον ο οποίος είναι αναλυτής κάποιας επιχείρησης\οργανισμού το UDDI είναι παρόμοιο με μια μηχανή αναζήτησης διαδικτύου για επιχειρηματικές διαδικασίες. Ο αναλυτής μπορεί να ψάξει σε κάποιο μητρώο UDDI για να βρει τους διάφορους οργανισμούς, τις υπηρεσίες τις οποίες παρέχουν οι οργανισμοί και τις προδιαγραφές αυτών των υπηρεσιών. Ωστόσο οι χρήστες αυτών των παρεχόμενων υπηρεσιών δεν είναι αναγκαίο να έχουν άμεση προσπέλαση στο UDDI γιατί οι πληροφορίες οι οποίες αποθηκεύονται ίσως να μην είναι και τόσο κατανοητές. Ακόμα μια κατηγορία ανθρώπων για τους οποίους είναι χρήσιμο το UDDI είναι οι προγραμματιστές, οι οποίοι το χρησιμοποιούν για να εκδίδουν υπηρεσίες και να ψάχνουν για υπηρεσίες που να ικανοποιούν κάποια κριτήρια. Η εύρεση μιας υπηρεσίας γίνεται δυναμικά και χρησιμοποιείται χωρίς να υπάρχει αλληλεπίδραση με τον χρήστη.



Σχήμα 2.5: Σύνθεση UDDI [11]

Παραπάνω στο Σχήμα 2.5 παρουσιάζεται ένα σχεδιάγραμμα της δόμησης του UDDI. Κατ’ αρχάς κάθε μητρώο επιχειρήσεων του UDDI (UDDI Business Registry, UBR)) παρουσιάζεται σαν ένα ενιαίο σύστημα χτισμένο από πολλούς κόμβους (operator nodes) που συγχρονίζουν τα δεδομένα τους μέσω της αντιγραφής. Κάθε ένας από τους κόμβους (operator nodes) κρατούν ένα αντίγραφο του περιεχομένου(εγγεγραμμένες υπηρεσίες). Ουσιαστικά αυτοί οι κόμβοι αντιγράφουν

το περιεχόμενο ο ένας του άλλου. Η πρόσβαση σε οποιονδήποτε από αυτούς τους κόμβους παρέχει τις ίδιες πληροφορίες και ποιότητα υπηρεσιών όπως οι υπόλοιποι. Το περιεχόμενο που εισάγεται στο UBR γίνεται σε ένα μόνο κόμβο (operator node), και αυτός ο κόμβος γίνεται ο κύριος ιδιοκτήτης του εν λόγω περιεχομένου. Τυχόν ενημερώσεις ή διαγραφές δεδομένων πρέπει να γίνουν στον κόμβο (operator node), στον οποίο προστέθηκαν αρχικά τα δεδομένα. Η έκταση όμως του UDDI δεν περιορίζεται στο UBR, ένας οργανισμός μπορεί να παρέχει ένα ιδιωτικό κόμβο (operator node), που δεν είναι μέλος κάποιου UBR. Οι ιδιωτικοί κόμβοι δεν έχουν τα δεδομένα τους να συγχρονίζονται με κάποιο UBR έτσι οι πληροφορίες οι οποίες περιέχονται είναι σαφής. Μπορούν επίσης να ορίσουν κανόνες πρόσβασης για τους κόμβους τους, και να τους κάνουν πιο αυστηρούς ή χαλαρούς ανάλογα ή ακόμα μπορούν να ακολουθήσουν το μοντέλο που ακολουθείται και από το UBR.

Το UDDI [21] στο επίπεδο του προγραμματισμού όπως και κάθε άλλο σύστημα έχει κάποιες προδιαγραφές. Οι προδιαγραφές αυτές του UDDI τις περιγράφουν δύο API'S: αυτό για την αίτηση πληροφοριών (inquiry API) και αυτό για την έκδοση πληροφοριών (publishing API). Μέσω αυτών των δύο κομματιών παράγεται η λειτουργία του μητρώου του UDDI. Επίσης μέσα στις προδιαγραφές δηλώνεται μια σειρά από SOAP μηνύματα τα οποία μπορεί να γίνουν δεκτά από την βιομηχανία του UDDI. Τα δύο API τα οποία δηλώνονται στις προδιαγραφές είναι προσβάσιμα με κάποιες τεχνικές οι οποίες είναι κοινές και για τα δύο αλλά χρησιμοποιούν διαφορετικά XML αρχεία, δομές δεδομένων και σημεία πρόσβασης.

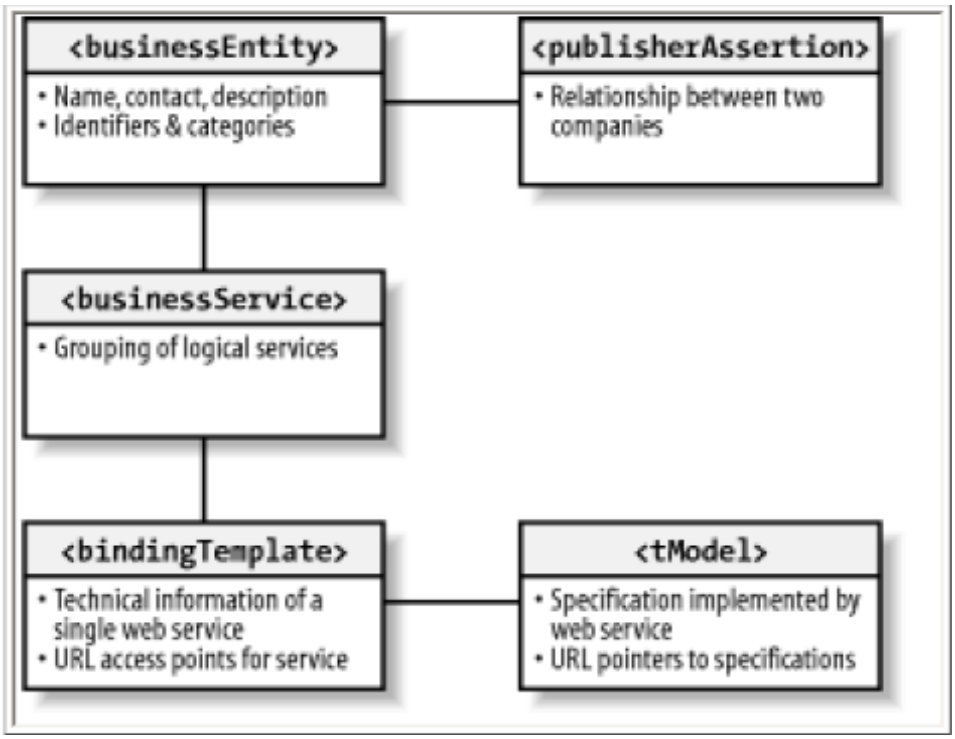
Το API για την αίτηση πληροφοριών εντοπίζει πληροφορίες για τις επιχειρήσεις, τις υπηρεσίες που προσφέρουν οι επιχειρήσεις, τις προδιαγραφές των υπηρεσιών που προσφέρονται και τις πληροφορίες σχετικά με το τι θα πρέπει να γίνει σε περίπτωση που θα υπάρξει αποτυχία. Οποιαδήποτε λειτουργία διαβάσματος κάποιων δεδομένων από το μητρώο του UDDI μέσω αυτού του API γίνεται με ένα μήνυμα αίτησης πληροφοριών (inquiry message). Το συγκεκριμένο API δεν χρειάζεται επικύρωση για να γίνει πρόσβαση στο μητρώο του UDDI. Έτσι η πρόσβαση γίνεται μέσω του HTTP.

Το API για έκδοση χρησιμοποιείται για την δημιουργία, αποθήκευση και ενημέρωση πληροφοριών που εντοπίζονται στο μητρώο του UDDI. Σε όλες του τις συναρτήσεις για να υπάρξει πρόσβαση στο UDDI χρειάζεται επικύρωση. Πρέπει να υπάρχει μια ταυτότητα πρόσβασης και τα πιστοποιητικά ασφαλείας αυτής της ταυτότητας θα πρέπει να θα πρέπει να είναι παράμετροι στα αρχεία XML αυτού του API. Επειδή η πρόσβαση στο UDDI μέσω αυτού του API χρειάζεται επικύρωση γίνεται μέσω του HTTPS μέσω μιας διαφορετικής διεύθυνσης (URL) από αυτό του API για αίτηση πληροφοριών. Στον παρακάτω πίνακα (Πίνακας 2.1) φαίνονται κάποιες διευθύνσεις (URL'S) τα οποία μπορούν να χρησιμοποιηθούν [11].

Πίνακας 2.1: URL'S για κάποιους από τους κόμβους (operation nodes) [11]

Operator node	Inquiry URL	Publishing URL
HP	http://uddi.hp.com/inquire	https://uddi.hp.com/publish
IBM Production	http://www-3.ibm.com/services/uddi/inquiryapi	https://www-3.ibm.com/services/uddi/protect/publishapi
IBM Test	http://www-3.ibm.com/services/uddi/testregistry/inquiryapi	https://www-3.ibm.com/services/uddi/testregistry/protect/publishapi
Microsoft Production	http://uddi.microsoft.com/inquire	https://uddi.microsoft.com/publish
Microsoft Test	http://test.uddi.microsoft.com/inquire	https://test.uddi.microsoft.com/publish
SAP Test	http://udditest.sap.com/UDDI/api/inquiry/	https://udditest.sap.com/UDDI/api/publish/
Systinet	http://www.systinet.com/wasp/uddi/inquiry/	https://www.systinet.com/wasp/uddi/publishing/

Το τελευταίο κομμάτι στο οποίο θα γίνει αναφορά είναι η δομή δεδομένων του UDDI. Το Σχήμα 2.6 παρουσιάζει την σχέση μεταξύ των βασικών δομών δεδομένων.



Σχήμα 2.6: Σχέση μεταξύ βασικών δομών δεδομένων του UDDI [11]

Η βασική οντότητα είναι η <businessEntity> η οποία αντιπροσωπεύει τις βασικές πληροφορίες μιας επιχείρησης\οργανισμού. Οι πληροφορίες αυτές περιέχουν στοιχεία για την επικοινωνία, για την κατηγοριοποίηση, τα αναγνωριστικά, τις περιγραφές και τις σχέσεις με τις άλλες επιχειρήσεις\οργανισμούς. Το UDDI επιτρέπει σε εταιρείες να εγκαθιδρύουν σχέσεις μεταξύ τους. Σε τέτοια περίπτωση η κάθε εταιρεία πρέπει να έχει μια μοναδική <businessEntity> και να εγκαθιδρύει ξεχωριστά τις σχέσεις τις με τις άλλες εταιρείες που έχουν την δική τους <businessEntity> οντότητα. Η επόμενη οντότητα η <publisherAssertion> χρησιμοποιείται για την εγκαθίδρυση των σχέσεων μεταξύ δύο <businessEntity>. Επιπρόσθετα η <businessEntity> περιέχει μια ή περισσότερες <businessService>. Η δομή <businessService> χρησιμοποιείται για να περιγράψει ένα σύνολο υπηρεσιών που παρέχονται από την επιχείρηση\ οργανισμό. Το αρχείο αυτής της δομής είναι επαναχρησιμοποιήσιμο (ένα <businessService> στοιχείο μπορεί να χρησιμοποιηθεί από πολλά <businessEntity> στοιχεία). Μια <businessService> περιέχει ένα ή περισσότερα <bindingTemplate>. Το <bindingTemplate> περιέχει δείκτες σε τεχνικές περιγραφές, διευθύνσεις (URL'S), αλλά δεν περιέχει λεπτομέρειες για τις

προδιαγραφές των υπηρεσιών. Σε σχέση με τις υπηρεσίες μπορεί να περιέχει προαιρετικά μια περιγραφή σε μορφή κειμένου, το URL μέσα από το οποίο μπορεί να γίνει η πρόσβαση και μια αναφορά σε ένα ή περισσότερα <tModel>. Το <tModel> είναι μια αφηρημένη περιγραφή μιας συγκεκριμένης προδιαγραφής η συμπεριφοράς στην οποία η υπηρεσία ανταποκρίνεται. Οι εταιρείες μπορεί να χρησιμοποιήσουν αυτές τις πληροφορίες για να εξακριβώσουν αν η υπηρεσία είναι συμβατή με τις απαιτήσεις τις επιχείρησης\οργανισμού της [11, 21].

2.3 Γενετικοί αλγόριθμοι

Οι γενετικοί αλγόριθμοι [20] είναι μέθοδοι οι οποίοι συνήθως χρησιμοποιούνται για να λύσουν προβλήματα αναζήτησης και βελτιστοποίησης . Η λειτουργικότητα τους είναι βασισμένη στις γενετικές διαδικασίες των οργανισμών όπως περιγράφονται στην επιστήμη της βιολογίας. Μέσα από ένα αριθμό γενεών εξελίσσεται κάποιος πληθυσμός . Η εξέλιξη αυτή του πληθυσμού είναι βασισμένη στις αρχές της φυσικής επιλογής (natural selection) και της επιβίωσης του καταλληλότερου.

Οι γενετικοί αλγόριθμοι χρησιμοποιούν μια άμεση αναλογία της παραπάνω συμπεριφοράς της φύσης. Λειτουργούν με ένα πληθυσμό από «άτομα», όπου το καθένα αντιπροσωπεύει μια υποψήφια λύση στο δεδομένο πρόβλημα. Τα «άτομα» αυτά που απαρτίζουν το σύνολο λύσεων σε ένα πρόβλημα είναι γνωστά ως χρωμοσώματα. Κάθε χρωμόσωμα αποτιμείται σύμφωνα με μια συνάρτηση και συνδέεται με μια τιμή (fitness value) η οποία δείχνει πόσο καλή λύση είναι το συγκεκριμένο χρωμόσωμα για το πρόβλημα το οποίο διερευνείται. Όσο μεγαλύτερη είναι η τιμή αυτή τόσο πιο μεγάλες είναι οι πιθανότητες το συγκεκριμένο χρωμόσωμα να εξελιχθεί και να περάσει στην επόμενη γενεά. Ολόκληρος ο πληθυσμός της κάθε επόμενης γενιάς παράγεται με την επιλογή των καλύτερων χρωμοσωμάτων της συγκεκριμένης τρέχουσας γενιάς. Με αυτόν τον τρόπο ο νέος πληθυσμός περιέχει μεγαλύτερο ποσοστό καλών χαρακτηριστικών, τα οποία κληρονομήθηκαν από την προηγούμενη γενιά. Έτσι το νέο σύνολο των λύσεων που προκύπτει με κάθε νέα γενιά είναι καλύτερο από προηγούμενα σύνολά λύσεων με αποτέλεσμα ο πληθυσμός του γενετικού αλγόριθμου να συγκλίνει σταδιακά προς την βέλτιστη λύση για το διερευνώμενο πρόβλημα.

Η δύναμη των γενετικών αλγορίθμων προέρχεται από το γεγονός ότι μπορούν να εφαρμοστούν και να διαπραγματευτούν με επιτυχία προβλήματα τα οποία έχουν μεγάλο εύρος δεδομένων προς διερεύνηση και αυτά που για άλλες μεθόδους είναι δύσκολα να λυθούν. Δεν εγγυώνται ότι θα βρεθεί βέλτιστη λύση στο πρόβλημα αλλά θα υπάρξει σίγουρα μια καλή και γρήγορη λύση. Ένας καθιερωμένος γενετικός αλγόριθμος παρουσιάζεται παρακάτω στο Σχήμα 2.7.

Όπως προαναφέρθηκε αυτοί οι αλγόριθμοι κωδικοποιούν μια δυνατή λύση σε ένα συγκεκριμένο πρόβλημα, σε μορφή μιας δομής η οποία ονομάζεται χρωμόσωμα και για κάθε ένα από αυτά γίνεται η αποτίμηση για να δειχθεί πόσο καλή λύση είναι. Άρα σύμφωνα με την λειτουργία των γενετικών αλγορίθμων μόνο δύο κύριοι παράμετροι θα πρέπει να οριστούν για την αναπαράσταση κάποιου προβλήματος. Οι παράμετροι αυτοί είναι η κωδικοποίηση του χρωμοσώματος και ο προσδιορισμός της αντικειμενικής συνάρτησης (fitness function) για την αποτίμηση των χρωμοσωμάτων.

```
BEGIN /* genetic algorithm */
    generate initial population
    compute fitness of each individual

    WHILE NOT finished DO
    BEGIN /* produce new generation */

        FOR population_size / 2 DO
        BEGIN /* reproductive cycle */
            select two individuals from old generation for mating
                /* biassed in favour of the fitter ones */
            recombine the two individuals to give two offspring
            compute fitness of the two offspring
            insert offspring in new generation
        END

        IF population has converged THEN
            finished := TRUE

    END

END
```

Σχήμα 2.7: Ψευδοκώδικας Παραδοσιακού γενετικού αλγόριθμου[13]

Για την κωδικοποίηση του προβλήματος που πρέπει να λυθεί με γενετικούς αλγόριθμους, γίνεται η υπόθεση ότι μια πιθανή λύση μπορεί να αντιπροσωπευθεί σαν ένα σύνολο παραμέτρων. Αυτοί οι παράμετροι είναι γνωστές ως τα γονίδια του χρωμοσώματος. Οι παράμετροι ενώνονται μαζί για να διαμορφώσουν μια συμβολοσειρά από τιμές, το χρωμόσωμα. Για παράδειγμα όταν ο γενετικός αλγόριθμος προσπαθεί να λύσει ένα πρόβλημα μεγιστοποίησης μιας συνάρτησης που αποτελείται από τέσσερις παραμέτρους F(x, y, z, w) το χρωμόσωμα θα περιέχει τέσσερα γονίδια, όπου το καθένα θα αντιπροσωπεύει μια από τις μεταβλητές της συνάρτησης.

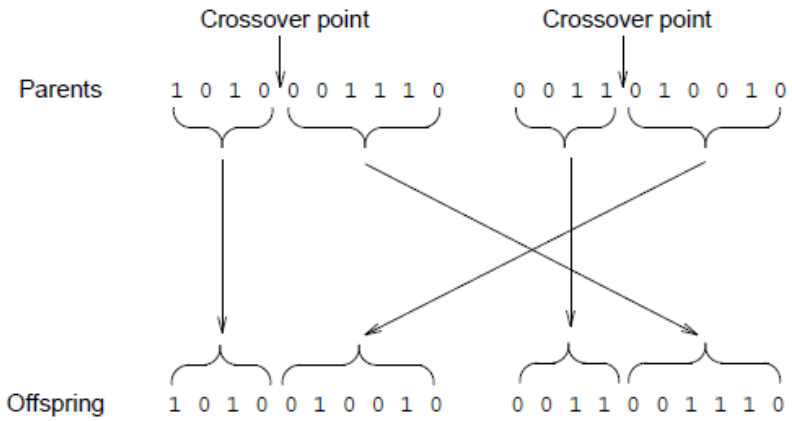
Με τους όρους της γενετικής στην βιολογία, το σύνολο των γονιδίων που αντιπροσωπεύουν ένα συγκεκριμένο χρωμόσωμα αναφέρεται ως ο γονότυπος. Ο γονότυπος περιέχει πληροφορίες, οι οποίες είναι απαραίτητες για την κατασκευή ενός οργανισμού. Αυτό στην γενετική αναφέρεται ως φαινότυπος. Οι ίδιοι όροι ισχύουν και για τους γενετικούς αλγόριθμους. Το σύνολο των παραμέτρων που συγκεκριμενοποιούν ένα πρόβλημα είναι ο γονότυπος ο οποίος χτίζει την δομή του χρωμοσώματος και οι τιμές οι οποίες παίρνει η κάθε μια από αυτές τις παραμέτρους είναι ο φαινότυπος. Η καταλληλότητα κάθε «γονιδίου» βασίζεται στην απόδοση του φαινοτύπου.

Η δεύτερη παράμετρος η οποία θα πρέπει να προσδιοριστεί για την επίλυση κάποιου προβλήματος είναι η αντικειμενική συνάρτηση (fitness function). Δεδομένου ενός χρωμοσώματος η αντικειμενική συνάρτηση (fitness function) επιστρέφει ένα αριθμό, ο οποίος αντιπροσωπεύει το πόσο καλή λύση είναι το συγκεκριμένο χρωμόσωμα. Σε ένα πρόβλημα βελτιστοποίησης είναι αρκετή η τιμή που επιστρέφει η αντικειμενική συνάρτηση για να αποτιμηθεί το πρόβλημα. Ενώ σε ένα συνδυαστικό πρόβλημα υπάρχουν και άλλοι παράγοντες οι οποίοι πρέπει να μετρηθούν για να δειχθεί πόσο καλή είναι η λύση, για παράδειγμα οι παράμετροι που αφορούν την απόδοση.

Εκτός από τις δύο αυτές παραμέτρους οι οποίοι θα πρέπει να προσδιοριστούν σημαντικό ρόλο στην λειτουργία των γενετικών αλγορίθμων είναι οι μέθοδοι που χρησιμοποιούνται για την δημιουργία του καινούριου πληθυσμού, η αναπαραγωγή, όταν χρειάζεται μέχρι ο αλγόριθμος να έχει ικανοποιητική σύγκλιση. Κατά την φάση

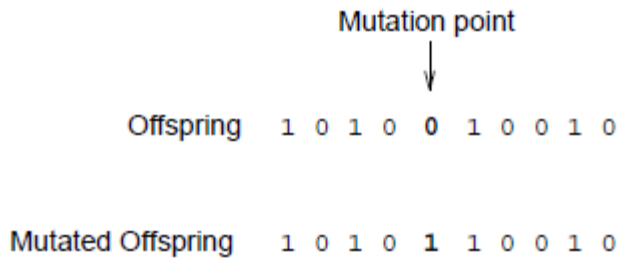
της αναπαραγωγής τα «καλά» χρωμοσώματα μπορεί να επιλεχθούν πολλές φορές για την δημιουργία της επόμενης γενιάς, ενώ τα λιγότερο «καλά» μπορεί να μην επιλεχθούν καμία φορά. Για την δημιουργία ενός καινούριου χρωμοσώματος που θα περάσει στην επόμενη γενιά, επιλέγονται δύο χρωμοσώματα από την τρέχον γενεά και πάνω τους εφαρμόζεται μια από τις μεθόδους διασταύρωση (crossover) ή μετάλλαξη (mutation).

Κατά την διασταύρωση (crossover) επιλέγονται δύο χρωμοσώματα, των οποίων κόβεται η δυαδική τους συμβολοσειρά με τέτοιο τρόπο ώστε να δημιουργηθούν δύο κομμάτια τα οποία θα αποτελέσουν κεφαλή, και δυο κομμάτια που θα αποτελέσουν ουρά. Τα κομμάτια τα οποία αποτελούν ουρά ανταλλάσσονται στα αρχικά χρωμοσώματα και δημιουργούνται δύο νέα ολοκληρωμένα χρωμοσώματα όπως φαίνεται και στο Σχήμα 2.8. Η διασταύρωση δεν εφαρμόζεται σε όλα τα ζευγάρια τα οποία επιλέχθηκαν για την αναπαραγωγή και δημιουργία του νέου πληθυσμού. Γίνεται μια τυχαία επιλογή αυτών πάνω στα οποία θα γίνει η διασταύρωση. Στα χρωμοσώματα που δεν εφαρμόζεται η μέθοδος αλλά επιλέχθηκαν, περνά στην επόμενη γενιά ένα ακριβής αντίγραφό τους.



Σχήμα 2.8: Διασταύρωση [13]

Η μετάλλαξη (mutation) εφαρμόζεται σε κάθε ένα από τα νέα χρωμοσώματα που παράγονται κατά την διασταύρωση. Αλλάζει τυχαία κάθε γονίδιο με μια πολύ μικρή πιθανότητα συνήθως 0.001. Παράδειγμα μετάλλαξης φαίνεται στο Σχήμα 2.9.



Σχήμα 2.9: Μετάλλαξη[13]

Γενικά αν ένας γενετικός αλγόριθμος είναι σωστά υλοποιημένος, ο πληθυσμός εξελίσσεται μέσα από τις γενιές έτσι ώστε κάθε φορά το καινούριο σύνολο λύσεων να συγκλίνει προς τη βέλτιστη λύση. Οι γενετικοί αλγόριθμοι μπορεί να εφαρμοστούν για να λύσουν κάποια δύσκολα προβλήματα στα οποία άλλες μέθοδοι δεν μπορούν να αντεπεξέλθουν. Εφαρμόζονται συνήθως σε προβλήματα που αφορούν βελτιστοποίηση ή αναζήτηση. Δεν εγγυώνται βέλτιστη λύση, όμως πάντα δίνουν μια υποκειμενικά καλή λύση [13].

2.4 Ροές Εργασίας (Microsoft Workflow)

Όπως είναι ήδη γνωστό σχεδόν όλα τα λογισμικά που χρησιμοποιούνται στις επιχειρήσεις σήμερα έχουν στόχο την υποστήριξη των επιχειρηματικών διαδικασιών. Ορισμένες διαδικασίες είναι πλήρως αυτοματοποιημένες , στηριζόμενες αποκλειστικά και μόνο στην επικοινωνία μεταξύ των εφαρμογών. Υπάρχουν όμως και άλλες πολλές διαδικασίες οι οποίες εξαρτώνται από τους ανθρώπους. Δηλαδή οι προγραμματιστές πρέπει να αρχικοποιήσουν τη διαδικασία, να εγκρίνουν τα έγγραφα που χρησιμοποιεί η διαδικασία , να επιλύουν τυχόν δυσλειτουργίες που προκύπτουν και άλλα.

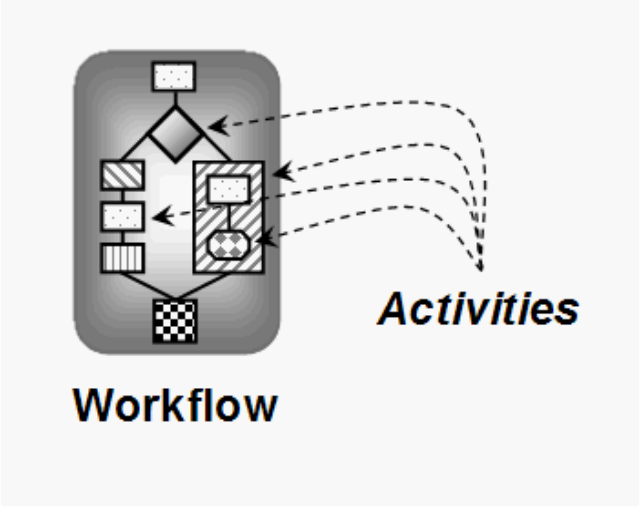
Όταν γίνεται χρήση των οποιονδήποτε διαδικασιών είναι σε πολλές περιπτώσεις δυνατόν να οριστεί μια σειρά από διακριτά βήματα που είναι γνωστή ως ροή εργασίας (workflow). Η ροή εργασίας (workflow) περιγράφει τις δραστηριότητες των ανθρώπων και του λογισμικού που εμπλέκονται σε μια συγκεκριμένη διαδικασία. Μόλις οριστεί η ροή εργασίας (workflow), μια εφαρμογή μπορεί να χτιστεί γύρω από τον ορισμό αυτό για τη στήριξη της επιχειρηματικής διαδικασίας. Η δημιουργία και η εκτέλεση της ροής εργασίας(workflow) δημιουργεί πολλές προκλήσεις. Ένα

παράδειγμα είναι πως ο προγραμματιστής μπορεί να κρατήσει πληροφορίες για την ροή εργασίας της τρέχουσας κατάστασης για ένα χρονικό διάστημα, όταν η επιχειρηματική διαδικασία παίρνει πάρα πολύ χρόνο να τελειώσει την εκτέλεση της.

Για τον ορισμό της ροής εργασίας (workflow) υπάρχουν κάποιες θεμελιώδης απαιτήσεις οι οποίες θα πρέπει να ικανοποιούνται. Η πρώτη από αυτές τις απαιτήσεις είναι να υπάρχει η ικανότητα να παίρνονται αποφάσεις που να βασίζονται στους επιχειρηματικούς κανόνες. Στον όρο κανόνες συμπεριλαμβάνονται οι απλοί και οι σύνθετοι κανόνες. Οι απλοί κανόνες είναι αυτοί των οποίων η απόφαση μπορεί να είναι της μορφής ναι ή όχι. Για τους σύνθετους κανόνες μπορεί να πρέπει να αποτιμηθεί ένα μεγάλο σύνολο δεδομένων για να παρθεί η τελική απόφαση. Ακόμα μια απαίτηση η οποία διαφαίνεται είναι το ότι θα πρέπει να υπάρχουν τρόποι να επικοινωνεί η ροή εργασίας (workflow) με άλλα λογισμικά και συστήματα τα οποία βρίσκονται έξω από αυτήν. Για παράδειγμα μπορεί να λαμβάνεται μια αίτηση από κάποιο άλλο κομμάτι της εφαρμογής εκτός από αυτό στο οποίο γίνεται αναφορά. Επιπρόσθετα θα πρέπει να υπάρχουν τρόποι να αλληλεπιδρά η ροή εργασίας (workflow) με τους ανθρώπους οι οποίοι θα πρέπει να έρχονται σε επαφή με αυτήν. Αυτή η αλληλεπίδραση μπορεί να γίνεται μέσω κάποιας διεπαφής χρήστη ή με κάποιο άλλο λογισμικό. Η τελευταία από τις απαιτήσεις είναι η ικανότητα να διατηρείται ενημερωμένη η κατάσταση της ροής εργασίας (workflow) σε όλη την διάρκεια της χρήσης της.

Οι πιο σημαντικοί στόχοι μιας εφαρμογής για δημιουργία ροών εργασίας(workflow) είναι δύο : πρώτο να προσφερθεί ένα πλαίσιο της ροής εργασίας για ποικίλες εφαρμογές και δεύτερο να γίνεται ενοποίηση του συστήματος και ανθρώπινης εργασίας. Πιο συγκεκριμένα όταν γίνεται αναφορά στο πλαίσιο εργασίας για πολλές εφαρμογές ουσιαστικά γίνεται αναφορά στο πώς θα αντιμετωπισθεί επιτυχώς το γεγονός ότι πολλές εφαρμογές χρησιμοποιούν με διαφορετικό τρόπο την τεχνολογία της ροής εργασίας. Αντί να προσφερθεί μια ενιαία γλωσσά και ένα κοινό εργαλείο γίνεται χρήση μιας άλλης μεθόδου-της παροχής ενός γενικού πλαισίου για δημιουργία και εκτέλεση κάποιας ροής εργασίας (workflow). Στο Σχήμα 2.10 φαίνεται

ότι η τεχνολογία της ροής εργασίας (workflow) συνοδεύεται από δραστηριότητες (activities), οι οποίες είναι χρήσιμες κατά την δήλωση της.

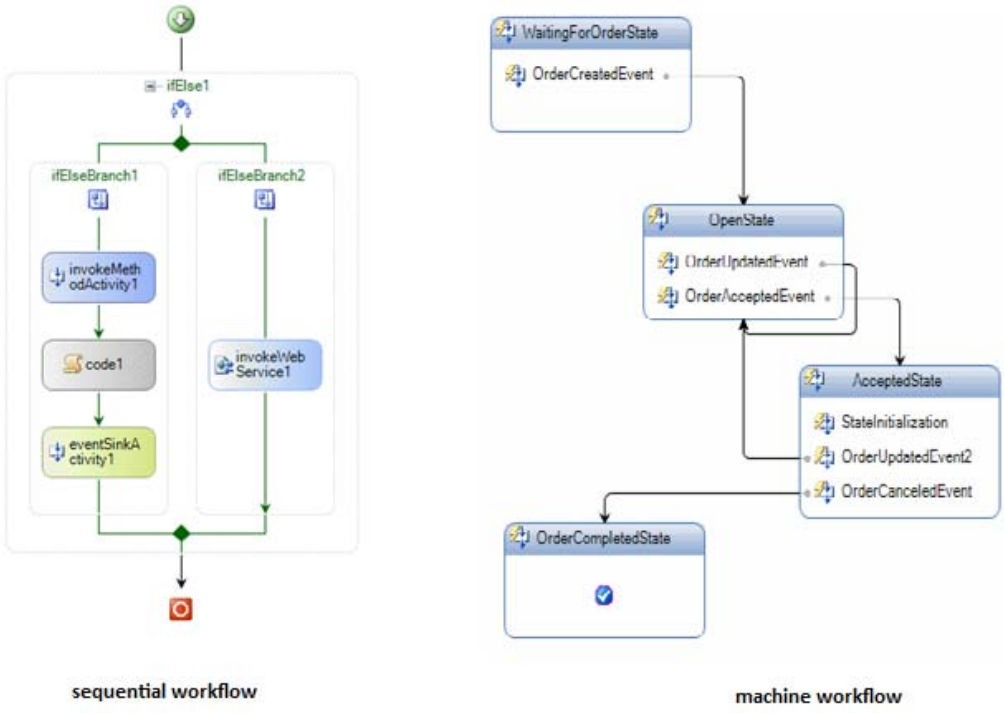


Σχήμα 2.10: Δομή μιας ροής εργασίας (workflow) [15]

Η βασική δραστηριότητα είναι αυτή η οποία ορίζει τον έλεγχο της όλης ροής χρησιμοποιώντας γνωστούς προγραμματιστικούς όρους όπως το IF/ELSE και WHILE. Επίσης η συγκεκριμένη δραστηριότητα συμπεριλαμβάνει μια σειρά από κανόνες για την επικοινωνία με άλλα λογισμικά χρησιμοποιώντας υπηρεσίες ιστού (Web Services). Όλες οι δραστηριότητες (activities) οι οποίες είναι ορισμένες προσφέρουν πολύ καλή λειτουργικότητα. Επιπρόσθετα όμως μπορεί ο προγραμματιστής να δηλώσει και τις δικές του δραστηριότητες (activities) καθώς και να αγνοήσει τελείως τις ήδη δηλωμένες δραστηριότητες της βιβλιοθήκης (base activity library).

Πιο αναλυτικά ο δεύτερος στόχος ο οποίος αναφέρεται στην ενοποίηση του συστήματος και της ανθρώπινης εργασίας προέρχεται από τη διαφορετικότητα της αυτοματοποίησης των αλληλεπιδράσεων μεταξύ των ανθρώπων και του λογισμικού. Ο στόχος είναι να βρεθεί μια ενιαία λύση και για τα δύο προβλήματα. Για μια πιο σφαιρική όψη του προβλήματος χρειάζεται ανάλυση των δύο πεδίων. Οι αλληλεπιδράσεις μεταξύ των εφαρμογών είναι στατικές και προβλέψιμες. Η λογική με την οποία γίνονται οι αλληλεπιδράσεις ορίζεται μια φορά και να χρησιμοποιείται ξανά και ξανά. Επίσης τα δεδομένα είναι καλά ορισμένα με συγκεκριμένη μορφή και μπορούν να χρησιμοποιηθούν εύκολα χωρίς την παρεμβολή του ανθρώπου. Από την

άλλη πλευρά οι δραστηριότητες (activities) οι οποίες διαχειρίζονται από τους ανθρώπους πρέπει να είναι πιο ευέλικτες από αυτές που η αλληλεπίδραση γίνεται μόνο μέσω του λογισμικού. Οι ροές εργασίας (workflows) οι οποίες κατασκευάζονται από τους ανθρώπους είναι πιο δυναμικές από αυτές των συστημάτων και καθοδηγούνται πιο πολύ από γεγονότα. Επιπρόσθετα η δομή τους δεν είναι τόσο αυστηρή. Για να γίνει η ενοποίηση των δύο αυτών πλευρών παρέχονται και ακολουθιακές και βασισμένες σε γεγονότα προσεγγίσεις, για τον ορισμό των ροών εργασίας (workflows). Υπάρχει επίσης η δυνατότητα να ορίζονται νέα βήματα και να αλλάζει μια τρέχουσα ροη εργασίας.



Σχήμα 2.11: Παραδείγματα των τύπων των ροών εργασίας (workflows) [15]

Γενικά μια ροή εργασίας περιέχει ένα αριθμό από δραστηριότητες, κάθε μια από τις οποίες εκτελεί ένα σημείο της λειτουργίας μιας ροής εργασίας. Η ροή εργασίας (workflow) δρα ως ένα κιβώτιο για τις δραστηριότητες για να γίνεται έλεγχος του κύκλου ζωής της καθώς και της εκτέλεσης της. Υπάρχουν δύο τύποι ροών εργασίας (workflows) που να ικανοποιούν τις δύο προσεγγίσεις, οι ακολουθιακές ροές

εργασίας (sequential workflows) που εκτελούν εργασίες σε ένα προκαθορισμένο σχήμα και οι ροές εργασίας της μηχανής (machine workflows) που είναι ικανές να ανταποκρίνονται σε εξωτερικά γεγονότα. Ο ακολουθιακός τύπος είναι βασικά ορισμένος για τις ροές εργασίας των συστημάτων και ο τύπος ροών εργασίας της μηχανής για τις ροές εργασίας (workflows) των ανθρώπων. Μια ενιαία ροή εργασίας (workflow) μπορεί να είναι συνδυασμός των δύο τύπων. Στο Σχήμα 2.11 παρουσιάζονται παραδείγματα των δύο τύπων [15].

2.5 Έλεγχος σύνθεσης των IP Media

Οι υπηρεσίες των μέσων ενημέρωσης (IP Media Services) χρησιμοποιούν το πρωτόκολλο του διαδικτύου για να δημιουργούν σε πραγματικό χρόνο συνδέσεις ήχου και βίντεο. Όλες αυτές οι υπηρεσίες των μέσων ενημέρωσης (IP Media Services) έχουν δύο κοινά χαρακτηριστικά. Το πρώτο είναι ότι τα κανάλια των media είναι από σημείο σε σημείο (point-to-point) και δυναμικά. Σε αυτό το χαρακτηριστικό εμπεριέχεται ένα πλατύ φάσμα από εφαρμογές οι οποίες αλληλεπιδρούν μεταξύ τους, όπως την τηλεφωνία στο διαδίκτυο, τα δίκτυα που είναι εγκαθιδρυμένα στα σπίτια κλπ. Το δεύτερο κοινό χαρακτηριστικό είναι ότι το δυναμική οργάνωση (setup) των καναλιών των media μπορεί κάποιες φορές να προϋποθέτει την συνεργασία ενός η περισσοτέρων διακομιστών εφαρμογών (application servers).

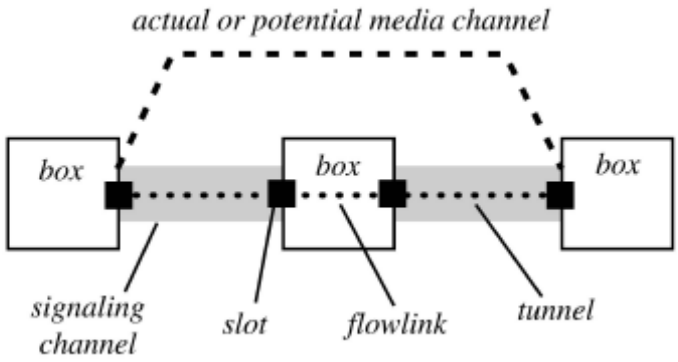
Σε αυτή την ενότητα το άκρο ενός media (media end-point) είναι οποιοσδήποτε κώδικας κάποιου media stream. Τα άκρα (media end points) μπορεί να συμπεριλαμβάνουν κάμερες, μικρόφωνα, συσκευές χρηστών κλπ. Στις περισσότερες υπηρεσίες των μέσων ενημέρωσης (IP Media Services) οι συσκευές των χρηστών δρουν αυτόνομα πάντα όμως με σεβασμό στα υπόλοιπα άκρα(media end-points). Μερικές από τις υπηρεσίες των μέσων ενημέρωσης (IP Media Services) είναι καλύτερα υλοποιημένες στα διάφορα άκρα (media end points) που συνεργάζονται μεταξύ τους, όμως άλλες είναι καλύτερα υλοποιημένες σε ξεχωριστούς διακομιστές εφαρμογών (application servers). Υπάρχουν πολλοί λόγοι για τους οποίους γίνεται χρήση των διακομιστών εφαρμογών (application servers), όπως το ότι οι περισσότερες συσκευές των χρηστών δεν υπόκεινται σε κάποιο αξιόπιστο πεδίο ορισμού. Χωρίς τους διακομιστές εφαρμογών σε κάθε συσκευή η οποία έκανε πρόσβαση σε μια

υπηρεσία, θα έπρεπε να εγκαθίσταται σε αυτήν το λογισμικό της εφαρμογής. Επίσης χωρίς αυτούς κάθε υπηρεσία που αποτελείται από πολλά κομμάτια πρέπει να υλοποιείται ολοκληρωτικά με αποκεντρωμένο τρόπο.

Οι διακομιστές εφαρμογών (application servers) πάνω στους οποίους υλοποιούνται κάποιες υπηρεσίες πρέπει να είναι προγραμματισμένοι με τέτοιο τρόπο ώστε η συμπεριφορά των media να είναι σφαιρικά ορθή, ακόμα και αν οι διακομιστές επιχειρήσουν να χρησιμοποιήσουν τα ίδια κανάλια για την λειτουργία τους ταυτόχρονα χωρίς να το ξέρουν. Αυτή η πρόκληση της σφαιρικά ορθής λειτουργίας αναφέρεται ως έλεγχος σύνθεσης των media. Θα πρέπει να υπάρχει μια προσέγγιση με την οποία θα επιτυγχάνεται ο έλεγχος της σύνθεσης. Αυτή η προσέγγιση για να θεωρείται πλήρες θα πρέπει να ισχύουν τρία πράγματα. Το πρώτο είναι ότι η εφαρμογή που θα υλοποιηθεί θα πρέπει να μπορεί να χρησιμοποιηθεί στην κατασκευή οποιασδήποτε υπηρεσίας και να κάνει τον έλεγχο των media σχετικά εύκολο σε όλες. Επίσης θα πρέπει να μπορεί να τρέχει σε όλες τις μηχανές ανεξάρτητα με το τι αρχιτεκτονική έχουν. Αυτό το χαρακτηριστικό προωθεί την επαναχρησιμοποίηση κώδικα (modularity). Το τρίτο και τελευταίο χαρακτηριστικό το οποίο πρέπει να έχει η εφαρμογή είναι να είναι αυτοματοποιημένη και να μπορεί να επαληθεύεται.

Είναι χρήσιμο να επεξηγηθούν μερικές ορολογίες για την καλύτερη κατανόηση της λειτουργίας του ελέγχου των media. Οι ενότητες (modules) οι οποίες λαμβάνουν μέρος στον έλεγχο των media μπορεί να είναι φυσικές ή εικονικές. Μια φυσική ενότητα μπορεί να είναι μια συσκευή κάποιου χρήστη, ένας διακομιστής εφαρμογών (application server) ή ένας πόρος κάποιου media. Μια εικονική ενότητα μπορεί να είναι μια διαδικασία του λογισμικού η οποία τρέχει πάνω σε κάποια από τις φυσικές ενότητες. Όλες οι ενότητες φυσικές και εικονικές λειτουργούν ως peers. Η λέξη κουτί (box) αναφέρεται στην ενότητα εικονική ή φυσική η οποία εμπλέκεται στον έλεγχο. Τα διάφορα κουτιά (boxes) είναι ενωμένα με τα κανάλια (signaling channel) μέσω των οποίων διαδίδεται το σήμα. Τα κανάλια αυτά είναι διπλής κατεύθυνσης και αξιόπιστα. Όταν το κανάλι (signaling channel) βρίσκεται ανάμεσα σε δύο φυσικές ενότητες είναι υλοποιημένο με TCP. Ενώ ένα κανάλι (signaling channel) το οποίο έχει

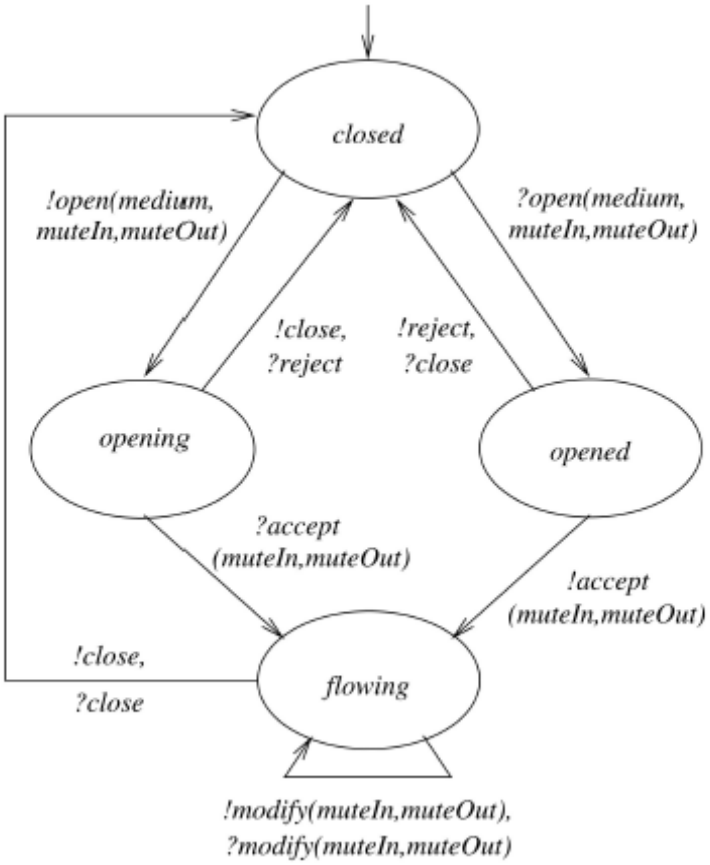
εντός του μια φυσική ενότητα υλοποιείται με δύο ουρές. Τέλος κάθε κανάλι χωρίζεται στατικά σε τούνελ για να παρέχεται η δυνατότητα αμφίδρομου σήματος. Κάθε τούνελ μπορεί να χρησιμοποιηθεί για έλεγχο ξεχωριστών καναλιών. Τα άκρα (end-points) κάθε τούνελ ονομάζονται slots. Για να επικοινωνούν και να συνεργάζονται τα διάφορα κουτιά (boxes) γίνεται χρήση κάποιου πρωτοκόλλου (signaling protocol). Το πρωτόκολλο (signaling protocol) αυτό λειτουργεί ανεξάρτητα σε κάθε τούνελ, κάθε καναλιού (signaling channel). Μέσα σε ένα κουτί δύο slots μπορεί να εκχωρηθούν σε ένα αντικείμενο που είναι γνωστό σαν flowLink. Το flowLink είναι ένα αντικείμενο το οποίο διαβάζει όλα τα σήματα από τα δύο slots και τα ελέγχει, δηλαδή πιο γενικά κατευθύνει τα σήματα των δύο slots. Ακόμα μια έννοια η οποία είναι σημαντική είναι η έννοια του μονοπατιού μέσα από το οποίο περνά το σήμα (signaling path). Το μονοπάτι (signaling path) αυτό ορίζεται ως η μεγαλύτερη αλυσίδα από τούνελς και flowLinks, όπου τα flowLinks και τα τούνελς συναντούν τα slots. Κάθε μονοπάτι αντιστοιχεί σε ένα πραγματικό ή δυναμικό κανάλι (signaling channel). Τα σήματα τα οποία ταξιδεύουν μέσα στο μονοπάτι (signaling path) μπορούν να δημιουργήσουν η να καταστρέψουν το κανάλι (signaling channel). Παρακάτω στο Σχήμα 2.12 παρουσιάζονται τα διάφορα κομμάτια ενός μονοπατιού(signaling path).



Σχήμα 2.12: Διάφορα κομμάτια που συνθέτουν ένα μονοπάτι (signaling path) [12]

Όπως επισημάνθηκε παραπάνω κάθε μονοπάτι (signaling path) αντιστοιχεί σε ένα φυσικό ή δυναμικό κανάλι (signaling channel) μεταξύ των δύο άκρων (end - points) του μονοπατιού αυτού. Αν το κανάλι υπάρχει, τα σφαιρικά του χαρακτηριστικά συμπεριλαμβανομένης της διεύθυνσης IP και της πύλης IP για το κάθε άκρο (end - point), χρησιμοποιούνται για να στέλλονται ή να λαμβάνονται πακέτα (media

packets). Στο Σχήμα 2.13 παρουσιάζεται η διεπαφή του χρήστη σε ένα άκρο ενός καναλιού.



Σχήμα 2.13: Διεπαφή χρήστη στο άκρο ενός καναλιού(signaling channel) [12]

Αρχικά όπως φαίνεται και παραπάνω το κανάλι είναι κλειστό (closed) ή δεν υπάρχει καν. Όταν ο χρήστης επιλέγει να το ανοίξει (open) θα πρέπει να επιλέξει το μέσο (medium) του καναλιού. Τα βίντεο και ο ήχος είναι τα πιο συνηθισμένα media που χρησιμοποιούνται αλλά υπάρχουν και άλλες πιθανότητες. Για παράδειγμα το βίντεο μπορεί να χωριστεί σε διαφορετικές ποιότητες γι αυτό και επιλέγεται πάντα το μέσο του καναλιού. Όταν έρχεται μια αίτηση ο χρήστης μπορεί να την δεχτεί ή να την απορρίψει. Το κανάλι έρχεται σε κατάσταση ροής, δηλαδή μεταφέρονται δεδομένα (media) , μόνο όταν το ένα άκρο (end point) ανοίξει το κανάλι και το άλλο αποδεχτεί την αίτηση. Και τα δύο άκρα (end - points) μπορούν οποιαδήποτε στιγμή να κλείσουν το κανάλι (signaling channel) .

Το κανάλι (media channel) πάντα μπορεί να προσφέρει αμφίδρομη ροή δεδομένων. Τα δεδομένα ρέουν προς μια κατεύθυνση μόνο αν και τα δύο άκρα του καναλιού είναι σύμφωνα. Το άνοιγμα και το αν γίνεται δεκτή μια αίτηση περιέχουν Boolean τιμές οι οποίες βρίσκονται στις μεταβλητές muteIn, muteOut ανάλογα αν η ροή των δεδομένων θα είναι προς τα μέσα ή προς τα έξω. Κάθε άκρο (end –point) είναι υπεύθυνο να αποθηκεύει την τιμή των δύο αυτών μεταβλητών.

Οι προγραμματιστές των εφαρμογών χειρίζονται τα κανάλια (media channels) με το να ελέγχουν τα slots. Επειδή κάθε slot είναι ένα πρωτόκολλο των άκρων, τα χαρακτηριστικά του προσδιορίζονται από το γενικό πρωτόκολλο (signaling protocol) για τον έλεγχο όλων των media. Κάθε ένα από τα slot έχει τα χαρακτηριστικά medium, state. Όλες οι πιθανές τιμές για το χαρακτηριστικό state είναι τέσσερις, αυτές που παρουσιάζονται στο Σχήμα 2.13.

Πολλές από τις υπηρεσίες των media είναι βασισμένες σε γεγονότα γι αυτό και η περιγραφή με καταστάσεις δίνει καλύτερο πρόγραμμα στο τέλος. Για το λόγο αυτό για να επιτευχθεί έλεγχος των media στους διακομιστές εφαρμογών (application servers), πάνω στους οποίους υλοποιούνται οι υπηρεσίες, χρειάζεται ένα σύνολο από αναγνωριστικά (state- oriented primitives). Σε κάθε κατάσταση του προγράμματος του κάθε κουτιού δίδεται ο στόχος του προγραμματιστή(τα primitives), για κάθε slot ενόσω το πρόγραμμα βρίσκεται στην συγκεκριμένη κατάσταση. Δεν μπορεί η δραστηριότητα του slot να μην είναι ορατή στον προγραμματιστή γιατί η λογική του προγράμματος μερικές φορές βασίζεται σε αυτήν. Έτσι για κάθε slot υπάρχουν τέσσερα κατηγορήματα τα οποία είναι: isClosed, isOpening, isFlowing, isOpened. Τα κατηγορήματα αυτά χρησιμοποιούνται για να καθοδηγούν τις μεταβάσεις στα κουτιά του προγράμματος. Όταν η κατάσταση σε ένα κουτί έχει την μετάβαση isClosed(s) για ένα slot s, τότε αυτή η μετάβαση είναι προσβάσιμη μόλις το πρόγραμμα μπει σε μια κατάσταση ή όταν το slot κλείσει ενόσω το πρόγραμμα βρίσκεται ακόμα μέσα στην συγκεκριμένη κατάσταση.

Όπως είναι ήδη γνωστό το κάθε slot ελέγχεται από τέσσερα αναγνωριστικά (primitives). Τα τρία από αυτά έχουν να κάνουν με την κατάσταση του slot σε μια δεδομένη στιγμή, όταν αυτό ενεργεί σαν end point. Τα τρία αυτά αναγνωριστικά

(primitives) είναι τα: openSlot, closeSlot, holdSlot. Το πρώτο χρησιμοποιείται για να ανοίξει ένα κανάλι (signaling channel) και να το βάλει σε κατάσταση ροής. Το μέσο (medium) του καναλιού είναι μια παράμετρος του openSlot. Το δεύτερο αναγνωριστικό (primitive), closeSlot είναι για να βάζει το slot στην κλειστή κατάσταση και να το κρατά εκεί. Όταν το slot είναι κλειστό, και το closeSlot λάβει ένα σήμα για να άνοιγμα το απορρίπτει απευθείας. Τέλος το holdSlot έχει ως στόχο να δέχεται ένα κανάλι και να το θέτει σε κατάσταση ροής μόνο εάν το κανάλι ζητηθεί και από το άλλο άκρο. Το κανάλι (signaling channel) κλείνει εάν το άλλο άκρο κλείσει και μένει κλειστό μέχρι το άλλο άκρο του ζητήσει να ανοίξει. Το κουτί μπορεί να αλλάξει την κατάσταση με το closeSlot ή το holdSlot κάθε στιγμή της ζωής του ελεγχόμενου slot. Το τέταρτο αναγνωριστικό (primitive) έχει ήδη αναφερθεί, είναι το flowLink το οποίο ελέγχει δύο slots. Το flowLink προσπαθεί να αντιστοιχίσει τις καταστάσεις των δύο slots εάν ήταν πάντα ενωμένα και να τα κρατήσει αντιστοιχισμένα. Χρησιμοποιώντας τα αναγνωριστικά (primitives) είναι εύκολο να προγραμματιστεί η συμπεριφορά των media [12].

2.6 Service Oriented Computing - Service Oriented Architectures

Σήμερα το επιχειρηματικό κλίμα απαιτεί υψηλό ρυθμό μεταβολής τον οποίο οι οργανισμοί, οι οποίοι ασχολούνται με την Τεχνολογία της πληροφόρησης (Information Technology, IT), καλούνται να αντιμετωπίσουν. Οι οργανισμοί έρχονται αντιμέτωποι με την γρήγορη αλλαγή των συνθηκών της αγοράς, τις νέες ανταγωνιστικές πιέσεις, τις νέες ρυθμιστικές εξουσιοδοτήσεις που απαιτούν τη συμμόρφωση, και τις νέες ανταγωνιστικές απειλές. Όλες αυτές οι καταστάσεις και άλλες οδηγούν στην ανάγκη για την IT υποδομή ενός οργανισμού για να ανταποκριθεί γρήγορα στη στήριξη των επιχειρηματικών μοντέλων και προδιαγραφών. Μόνο με αυτόν τον τρόπο μπορούν οι οργανισμοί να βαδίσουν προς το πραγματικό κόσμο της πλήρης αυτοματοποίησης και των σύνθετων ηλεκτρονικών συναλλαγών.

Το Service-Oriented Computing (SOC) χρησιμοποιεί τις υπηρεσίες ως συστατικά για να υποστηρίξει την ανάπτυξη της γρήγορης, χαμηλού κόστους και εύκολης σύνθεσης των διανεμημένων εφαρμογών. Οι υπηρεσίες μπορούν να περιγραφούν, να εκδοθούν, να

ανευρεθούν και να συναρμολογηθούν δυναμικά για να αναπτύξουν μαζικά κατανεμημένα, διαλειτουργικά και εξελίξιμα συστήματα. Οι υπηρεσίες εκτελούν συναρτήσεις οι οποίες εκτείνονται από το να απαντούν απλά αιτήματα μέχρι και να εκτελούν εξελιγμένες επιχειρηματικές διαδικασίες οι οποίες να απαιτούν peer-to-peer σχέσεις μεταξύ των πιθανών πολλαπλών στρωμάτων των προμηθευτών και καταναλωτών των υπηρεσιών. Οι υπηρεσίες αντανακλούν μια service-oriented προσέγγιση στον προγραμματισμό, βασισμένη στην ιδέα της σύνθεσης εφαρμογών, με την σύνθεση και επίκληση τους για την εκτέλεση κάποιου στόχου.

Αυτή η service-oriented προσέγγιση είναι ανεξάρτητη από τις γλώσσες προγραμματισμού και τα λειτουργικά συστήματα. Επιτρέπει στους οργανισμούς να εκθέτουν τις βασικές τους ικανότητες μέσω του προγραμματισμού στο διαδίκτυο ή σε μια ομάδα δικτύων. Αυτό γίνεται με την χρήση καθιερωμένων γλωσσών (βασισμένες σε XML), πρωτοκόλλων, και με την υλοποίηση διεπαφών οι οποίες περιγράφουν τις ικανότητες τους αυτές. Οι υπηρεσίες ιστού (Web Services) είναι η τωρινή πιο υποσχόμενη τεχνολογία η οποία βασίζεται πάνω στην έννοια του SOC. Οι υπηρεσίες ιστού (Web Services) παρέχουν την βάση για ανάπτυξη και εκτέλεση των επιχειρηματικών διαδικασιών οι οποίες είναι κατανεμημένες στο δίκτυο και διαθέσιμες μέσω κάποιων διεπαφών και πρωτοκόλλων. Η υπόσχεση για το μέλλον της τεχνολογίας των υπηρεσιών είναι ένας κόσμος από υπηρεσίες οι οποίες συνεργάζονται μεταξύ τους, για να δημιουργούν δυναμικές επιχειρηματικές διαδικασίες και ευκίνητες εφαρμογές που να συνδέουν τους οργανισμούς με τις υπολογιστικές πλατφόρμες. Τα συστατικά των εφαρμογών αυτών συναρμολογούνται με λίγη προσπάθεια σε ένα δίκτυο από υπηρεσίες οι οποίες έχουν χαλαρή σύζευξη. Το κλειδί για αυτή την έννοια της service-oriented προσέγγισης είναι οι service-oriented αρχιτεκτονικές (SOA). Αυτές οι αρχιτεκτονικές είναι ένας λογικός τρόπος για την ανάπτυξη ενός λογισμικού συστήματος το οποίο να παρέχει υπηρεσίες και σε end-user εφαρμογές και σε άλλες υπηρεσίες που είναι κατανεμημένες στο δίκτυο, μέσω των εκδομένων και ανιχνευόμενων διεπαφών. Αυτές οι αρχιτεκτονικές μπορούν να εξουσιοδοτήσουν ένα επιχειρηματικό περιβάλλον με ένα εύκαμπτο περιβάλλον υποδομής και επεξεργασίας[9].

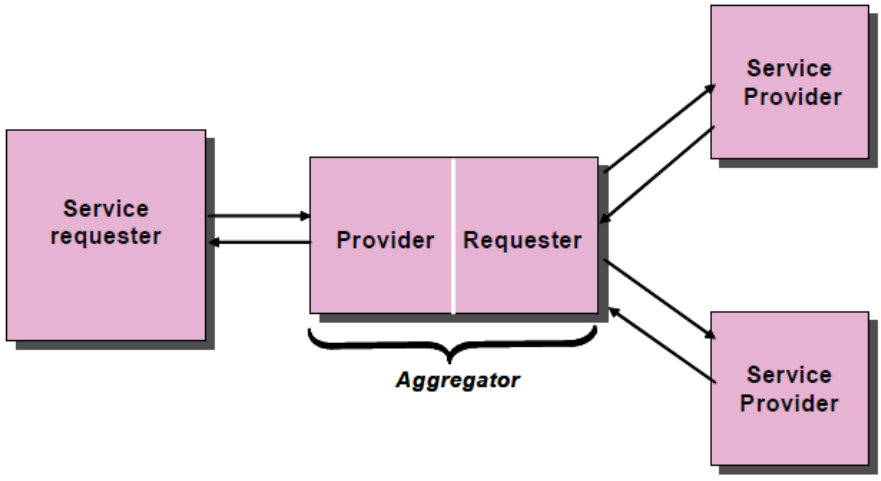
Μια SOA [17] παρέχει μια ευέλικτη αρχιτεκτονική η οποία ενοποιεί επιχειρηματικές διαδικασίες με το να ενοποιεί μεγάλες εφαρμογές σε υπηρεσίες. Ένας πελάτης από οποιαδήποτε συσκευή χρησιμοποιώντας ένα λειτουργικό σύστημα, σε οποιαδήποτε γλώσσα προγραμματισμού μπορεί να έχει πρόσβαση σε μια SOA υπηρεσία για να δημιουργήσει μια νέα επιχειρηματική διαδικασία. Μια SOA δημιουργεί μια συλλογή από υπηρεσίες οι οποίες επικοινωνούν μεταξύ τους χρησιμοποιώντας τις διαπροσωπείες των υπηρεσιών για να ανταλλάσσουν μηνύματα μεταξύ τους, ή να εκτελούν μια λειτουργία κατά την οποία πρέπει να συνεργαστούν μια ή περισσότερες υπηρεσίες. Οι υπηρεσίες που χρησιμοποιούνται σε μια σύνθετη εφαρμογή μπορεί να είναι νέες υλοποιήσεις υπηρεσιών, ή κομμάτια από παλαιότερες εφαρμογές οι οποίες υιοθετήθηκαν, ή συνδυασμός των δύο.

Οι SOA και οι διάφορες λύσεις με web services υποστηρίζουν δύο βασικούς ρόλους. Αυτόν του πελάτη (service requestor) και αυτόν του προμηθευτή (service provider) και επικοινωνούν μεταξύ τους με κάποιες αιτήσεις. Οι αιτήσεις είναι μηνύματα τα οποία είναι μορφοποιημένα σύμφωνα με ένα πρωτόκολλο το οποίο είναι γνωστό σαν Simple Object Access Protocol (SOAP). Το αναφερόμενο πρωτόκολλο (SOAP) είναι ένα πρωτόκολλο που επιτρέπει RPC κλήσεις στο διαδίκτυο χρησιμοποιώντας μια ποικιλία από πρωτόκολλα μεταφοράς, συμπεριλαμβανομένου του HTTP, HTTP/S και του SMTP. Ενόσω οι υπηρεσίες μιας SOA είναι ορατές στον πελάτη, τα βασικά τους συστατικά είναι φανερά. Έτσι ο προμηθευτής δεν χρειάζεται να ανησυχεί με την υλοποίηση της υπηρεσίας, αφού βέβαια υποστηρίζεται η αιτούμενη λειτουργία, ενόσω προσφέρεται η επιθυμητή ποιότητα μιας υπηρεσίας.

Οι αιτούμενες λειτουργίες των υπηρεσιών ιστού (Web Services) είναι υλοποιημένες χρησιμοποιώντας ένα ή περισσότερα από τα συστατικά τους. Τα συστατικά των υπηρεσιών ιστού (Web Services) μπορεί να φιλοξενούνται μέσα σε ένα κιβώτιο (web service container) που ενεργεί ως διαπροσωπεία μεταξύ της επιχειρηματικής υπηρεσίας και των του χαμηλότερου επιπέδου των υπηρεσιών υποδομής. Ουσιαστικά ένα κιβώτιο (web service container) είναι μια φυσική εκδήλωση της αφηρημένης υπηρεσίας, και παρέχει την υλοποίηση της διαπροσωπείας της υπηρεσίας. Μπορεί να

φιλοξενεί πολλαπλές υπηρεσίες ακόμα και αν δεν είναι μέλη της ίδιας κατανεμημένης διαδικασίας.

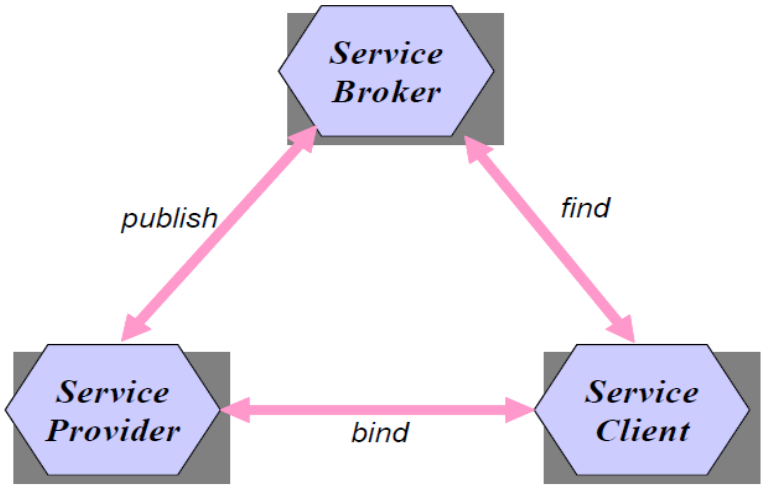
Ο προμηθευτής (service provider) σε συνδυασμό με τον πελάτη (service requestor) μπορεί να θεωρηθεί σαν ένας νέος ρόλος. Αν δεν υπάρχει αυτός ο ρόλος ο πελάτης (service requestor) κάποιας υπηρεσίας πρέπει άμεσα να αλληλεπιδράσει με έναν προμηθευτή (service provider) που θα του εκθέσει την πιθανή πολυπλοκότητα της εύρεσης, εξερεύνησης, διαπραγμάτευσης και κράτησης των υπηρεσιών μεταξύ διαφόρων προμηθευτών. Η εναλλακτική προσέγγιση του παραπάνω είναι ένας οργανισμός να παρέχει αυτή την συνδυαστική λειτουργικότητα που περιγράφηκε, απευθείας στον πελάτη. Δηλαδή είναι ο νέος ρόλος που συνδυάζει και τον πελάτη και τον προμηθευτή και είναι γνωστός σαν αθροιστής (service aggregator). Ο αθροιστής (service aggregator) μπορεί να λειτουργεί σαν προμηθευτής και να προσφέρει μια "ολοκληρωμένη υπηρεσία" η οποία είναι γνωστή σαν σύνθετη, στον πελάτη. Επιπρόσθετα μπορεί να λειτουργήσει σαν πελάτης αν χρειάζεται να ζητήσει και να δεσμεύσει κάποιες άλλες υπηρεσίες από άλλους προμηθευτές. Παρακάτω στο Σχήμα 2.14 παρουσιάζεται ένα διάγραμμα για τον αθροιστή.



Σχήμα 2.14: Ο ρόλος του αθροιστή (service aggregator)[3]

Ο αθροιστής (service aggregator) μπορεί να προσφέρει άμεσα πλεονεκτήματα στον πελάτη, όμως μια ερώτηση η οποία θα πρέπει να απαντηθεί είναι πως ο πελάτης θα επιλέξει τον κατάλληλο προμηθευτή για να χρησιμοποιήσει τις υπηρεσίες τις οποίες προσφέρει. Ο πελάτης θα μπορούσε να έχει το δικαίωμα να επιλέξει ένα προμηθευτή βασιζόμενος σε αυτούς που μπορεί να ανευρεθούν από μια υπηρεσία μητρώου όπως είναι το UDDI. Οι SOA τεχνολογίες όπως είναι το UDDI και τα πρότυπα ασφάλειας εισάγουν ένα νέο ρόλο στην λειτουργία του συστήματος, τον μεσίτη (service broker).

Οι μεσίτες (service brokers) είναι αξιόπιστα κομμάτια τα οποία αναγκάζουν τους προμηθευτές (service providers) να εμμένουν στις πληροφορίες οι οποίες συμμορφώνονται με τους ιδιωτικούς νόμους και κανονισμούς για να υπάρχει καλύτερη λειτουργία. Με αυτόν τον τρόπο οι προμηθευτές εγγυώνται ότι οι υπηρεσίες που προσφέρουν, ακολουθούν τους κανονισμούς και έτσι δημιουργείται μια πιο αξιόπιστη σχέση. Ο μεσίτης (service broker) διατηρεί ένα ευρετήριο με όλους τους προμηθευτές των υπηρεσιών. Είναι σε θέση να δίνουν αξία στο μητρώο τους που περιέχει τους προμηθευτές (service providers) με την παροχή πρόσθετων πληροφοριών σχετικά με τις υπηρεσίες τους [3]. Στο Σχήμα 2.15 φαίνεται η λειτουργία του μεσίτη.



Σχήμα 2.15: Λειτουργία μεσίτη (service broker) [3]

Κεφάλαιο 3

3. Προηγούμενες μελέτες στην σύνθεση υπηρεσιών

3.1 Εισαγωγή

Στο πεδίο της σύνθεσης υπηρεσιών κύριος στόχος είναι η αναζήτηση του καλύτερου συνόλου των υπηρεσιών, μέσω του οποίου θα γίνει η σύνθεση για να δημιουργηθεί η τελική λειτουργία. Κατά την διάρκεια της σύνθεσης η τελική λειτουργία ίσως χρειαστεί να επανασχεδιαστεί λόγω του ότι μπορεί να μην υπάρχουν διαθέσιμες οι συστατικές της υπηρεσίες. Η τεχνική αυτή της σύνθεσης προωθεί την επαναχρησιμοποίηση λογισμικού, που εστιάζει στην καλύτερη λύση κάποιου προβλήματος παρά την επαναδημιουργία των ήδη υπάρχων οντοτήτων (υπηρεσιών).

Υπήρξαν πολλές προσεγγίσεις που έδιναν λύση στο συνδυαστικό αυτό πρόβλημα της σύνθεσης των υπηρεσιών. Μια πληθώρα από αυτές έκαναν χρήση των γενετικών αλγορίθμων για να επιλύσουν το πρόβλημα. Για να λύσουν το συνδυαστικό πρόβλημα έπρεπε να κωδικοποιήσουν το χρωμόσωμα και να βρουν την κατάλληλη αντικειμενική συνάρτηση(fitness function). Επιλέχθηκαν οι γενετικοί αλγόριθμοι γιατί κάνουν πιο αποδοτική την σφαιρική αναζήτηση. Ο σχεδιασμός των γενετικών αλγορίθμων έχει την καλύτερη επίδραση στην συμπεριφορά και την απόδοση του προβλήματος. Υπήρχαν βέβαια και άλλες προσεγγίσεις οι οποίες δεν χρησιμοποιούσαν γενετικούς αλγόριθμους αλλά σε αντίθεση πρότειναν αλγορίθμους για την επιλογή του κατάλληλου συνδυασμού υπηρεσιών και τον επανασχεδιασμό της σύνθετης υπηρεσίας όταν αποτύχει η σύνθεση.

Η κάθε μια από αυτές τις προσεγγίσεις είχε τους δικούς της στόχους αλλά και τις δικές της ελλείψεις οποιαδήποτε δίοδο και αν ακολουθούσαν. Σε αυτό το έγγραφο παρουσιάζονται τέσσερις προσεγγίσεις οι οποίες υλοποιήθηκαν με σκοπό την λύση του συνδυαστικού αυτού προβλήματος, το οποίο συγκαταλέγεται στην κατηγορία των NP-Hard προβλημάτων. Οι τρείς πρώτες κάνουν χρήση των γενετικών αλγορίθμων και η τρίτη προτείνει το δικό της αλγόριθμο.

3.2 Πρώτη προσέγγιση

Ο κύριος στόχος της πρώτης προσέγγισης είναι η αυτοματοποίηση της διαδικασίας σύνθεσης των υπηρεσιών για την παραγωγή της τελικής λειτουργίας για την οποία γίνεται αίτηση. Για να γίνει εφικτή η αυτοματοποίηση πρέπει να γίνει σωστή διατύπωση των υποκειμενικών και ποσοτικών απαιτήσεων, που χρειάζονται για την σύνθεση, σε ποσοτικές και αντικειμενικές προδιαγραφές που να είναι κατανοητές από τον υπολογιστή. Μια και μόνο συγκεκριμένη υπηρεσία δεν μπορεί παράγει την τελική λειτουργία άρα θα υπάρχει επιλογή ενός συνόλου υπηρεσιών οι οποίες θα εκτελεστούν με τη σειρά. Η επιλογή της κάθε υπηρεσίας δεν πρέπει να ικανοποιεί μόνο τις απαιτήσεις τοπικά, για την ίδια της την οντότητα, αλλά να έχει μια γενικά καλή προσαρμογή σφαιρικά, στην τελική λειτουργία.

3.2.1 Περιγραφή προσέγγισης

Η σύνθετη υπηρεσία αποτελείται από κάποιες συστάδες, όπως η αρχική και η τελική συστάδα της, οι οποίες περιέχουν πολλές υποψήφιες συγκεκριμένες υπηρεσίες που προέρχονται από πολλαπλούς προμηθευτές υπηρεσιών. Αυτές οι συστάδες αποτελούν το σύνολο Si= {S1, S2,…,Sn-1,Sn}. Οι υπηρεσίες που βρίσκονται σε κάθε σύνολο Si πρέπει να ρυθμιστούν έτσι ώστε να κατασκευάζεται η τελική λειτουργία με την τοποθέτηση των συστάδων στην κατάλληλη σειρά[2].

3.2.2 Επιλογή λύσης με γενετικούς αλγόριθμους

Όπως ενδείκνυται λόγω της φύσης του αντικειμένου τους , για την σωστή χρήση των γενετικών αλγορίθμων [13,20] θα πρέπει να κωδικοποιηθούν δύο παράμετροι, η αντικειμενική συνάρτηση (fitness function) και το χρωμόσωμα. Σε αυτή την προσέγγιση η κάθε γενιά του χρωμοσώματος είναι μια συστάδα από υποψήφιες υπηρεσίες και μπορεί να πάρουν τις τιμές 1 ή 0. Όταν η τιμή σε μια από τις υπηρεσίες έχει τιμή 1 τότε σημαίνει ότι η υπηρεσία επιλέχθηκε και αντίθετα όταν είναι 0 σημαίνει ότι η υπηρεσία δεν θα χρησιμοποιηθεί στην σύνθεση για την συγκεκριμένη σύνθετη υπηρεσία. Πιο συγκεκριμένα η μορφή του χρωμοσώματος είναι $Chromosome=[S_{11}\ S_{12}\ \ldots S_{1i} \mid S_{21}\ S_{22}\ \ldots S_{2j} \mid \ \ldots \ \mid S_{n1}\ S_{n2}\ \ldots S_{nk}]$ όπου το n αντιπροσωπεύει των αριθμό των συστάδων που μπορεί να χρησιμοποιηθούν στην σύνθεση, ενώ το I,jκαι k είναι ο

αριθμός των συγκεκριμένων υπηρεσιών που βρίσκονται μέσα στην συστάδα και παρέχονται από ένα ή περισσότερους προμηθευτές. Τέλος η κάθε μια υπηρεσία σε κάθε συστάδα π.χ Snk είναι μια από τις συγκεκριμένες υπηρεσίες που μπορούν να πάρουν τις τιμές 1 ή 0. Από κάθε συστάδα μπορεί να επιλεχθεί μόνο μια συγκεκριμένη υπηρεσία.

Για την αποτίμηση της αντικειμενικής συνάρτησης σε αυτή την προσέγγιση χρησιμοποιούνται σαν παράμετροι ο χρόνος και το κόστος. Όσον αφορά το κόστος, αφού το χρωμόσωμα είναι ουσιαστικά είναι μια ακολουθία δυαδικών αριθμών, η καλύτερη λύση θα είναι αυτή της ακολουθίας που έχει το μικρότερο κόστος. Όσο για τον χρόνο αρχικά το σύστημα χρησιμοποιεί γενετικούς αλγόριθμους για να βρει το μικρότερο χρόνο για κάθε μια από τις ακολουθίες των χρωμοσωμάτων. Το επόμενο βήμα είναι να βρει το μεγαλύτερο από όλους τους προηγούμενους χρόνους που υπολόγισε και να το επιλέξει. Αυτός θα είναι ο καλύτερος χρόνος για την τελική λειτουργία. Ο συνδυασμός των δύο δίνει την καλύτερη λύση [2].

3.2.3 Προβλήματα Πρώτης Προσέγγισης

Οι αδυναμίες οι οποίες παρουσιάστηκαν στην πρώτη προσέγγιση ήταν αρκετές. Οφείλονταν βασικά στο γεγονός ότι το μέγεθος που είχε το χρωμόσωμα δεν ήταν σταθερό, λόγω του ότι όλες οι συγκεκριμένες υπηρεσίες κάθε συστάδας έμπαιναν μέσα στο χρωμόσωμα και ο αριθμός τους δεν ήταν προκαθορισμένος. Όσες περισσότερες ήταν οι συγκεκριμένες υποψήφιες υπηρεσίες κάθε συστάδας τόσο πιο μεγάλο ήταν το μέγεθος του χρωμοσώματος . Επιλεγόταν μόνο η υποψήφια υπηρεσία από κάθε συστάδα που είχε τιμή ίση με ένα. Από αυτό προέκυπτε φτωχή αναγνωσιμότητα του χρωμοσώματος. Δηλαδή η ανάγνωση του χρωμοσώματος όταν το μέγεθος του ήταν υπερβολικά μεγάλο ήταν δύσκολη και πολύπλοκη. Η σύνθετη λειτουργία δεν μπορούσε να επανασχεδιαστεί . Δεν υπήρχε η έννοια του μονοπατιού λόγω του ότι όλες οι συγκεκριμένες λειτουργίες που ήταν υποψήφιες έμπαιναν στο χρωμόσωμα με τιμή 1 ή 0 δεν υπήρχαν αφηρημένες οντότητες οι οποίες θα έπρεπε να ικανοποιηθούν[5].

3.3 Δεύτερη Προσέγγιση

Ο κύριος στόχος της δεύτερης προσέγγισης είναι ο γρήγορος υπολογισμός των QoS χαρακτηριστικών της σύνθετης υπηρεσίας, εφόσον έχει βρεθεί το βέλτιστο σύνολο των συγκεκριμένων υπηρεσιών που ικανοποιούν τις συστατικές αφηρημένες υπηρεσίες. Επίσης προτείνεται και ένας αλγόριθμος για τον επανασχεδιασμό της σύνθετης υπηρεσίας.

3.3.1 Περιγραφή προσέγγισης

Η σύνθετη υπηρεσία S αποτελείται από ένα σύνολο n αφηρημένων υπηρεσιών, S={s1, s2,..,sn} των οποίων η δομή ορίζεται από κάποια workflow description γλώσσα. Κάθε αφηρημένη υπηρεσία si μπορεί να βασιστεί σε μια συγκεκριμένη υπηρεσία m, η οποία επιλέγεται από ένα σύνολο συγκεκριμένων υπηρεσιών {CSi,1 , … ,CSi,m} . Όλες οι συγκεκριμένες υπηρεσίες του συνόλου που ικανοποιούν μια αφηρημένη υπηρεσία είναι λειτουργικά ισοδύναμες.

3.3.2 Υπολογισμός των QoS των Σύνθετων υπηρεσιών

Η προσέγγιση αυτή του υπολογισμού των QoS αναφέρεται ως unlooping approach. Ο υπολογισμός των μη λειτουργικών χαρακτηριστικών γίνεται μέσω κάποιων κανόνων ανάλογα με τον τρόπο εκτέλεσης των αφηρημένων υπηρεσιών . Οι κανόνες αυτοί παρουσιάζονται παρακάτω στον πίνακα 3.1.

Πίνακας 3.1: Aggregation function ανάλογα με τον τρόπο εκτέλεσης των αφηρημένων υπηρεσιών [5]

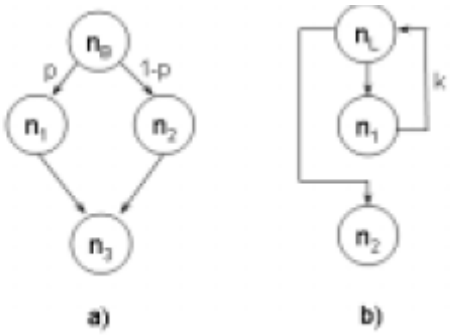
QoS Attr.	Sequence	Switch	Fork	Loop
Time (T)	$\sum_{i=1}^{m} T(t_i)$	$\sum_{i=1}^{n} p_{ai} * T(t_i)$	$Max\{T(t_i)_{i\in\{1\dots p\}}\}$	$k * T(t)$
Cost (C)	$\sum_{i=1}^{m} C(t_i)$	$\sum_{i=1}^{n} p_{ai} * C(t_i)$	$\sum_{i=1}^{p} C(t_i)$	$k * C(t)$
Availability (A)	$\prod_{i=1}^{m} A(t_i)$	$\sum_{i=1}^{n} p_{ai} * A(t_i)$	$\prod_{i=1}^{p} A(t_i)$	$A(t)^k$
Reliability (R)	$\prod_{i=1}^{m} R(t_i)$	$\sum_{i=1}^{n} p_{ai} * R(t_i)$	$\prod_{i=1}^{p} R(t_i)$	$R(t)^k$
Custom Attr. (F)	$f_S(F(t_i)_{i\in\{1\dots m\}})$	$f_B((p_{ai}, F(t_i))_{i\in\{1\dots n\}})$	$f_F(F(t_i)_{i\in\{1\dots p\}})$	$f_L(k, F(t))$

Όταν η εκτέλεση είναι σειριακή τότε γίνεται η εφαρμογή των κανόνων όπως φαίνονται στη πρώτη στήλη του πίνακα για κάθε ένα από τα μη λειτουργικά

χαρακτηριστικά. Όταν παρουσιάζεται κάποια δήλωση περιπτώσεων (switch statement),η κάθε μία από τις περιπτώσεις του συνοδεύεται από μια πιθανότητα, η οποία δηλώνει το αν θα εκτελεστεί η συγκεκριμένη περίπτωση. Οι πιθανότητες αυτές αρχικοποιούνται από τον σχεδιαστή και αναβαθμίζονται σύμφωνα με τις πληροφορίες που αντλούνται από την εκτέλεση των διαφόρων μονοπατιών.

Όταν παρουσιάζεται κάποιος βρόγχος επανάληψης (Loop statement) ακολουθείται και πάλι μια διαφορετική προσέγγιση απ όταν η εκτέλεση είναι σειριακή. Κάθε ένας από τους βρόγχους επανάληψης συνοδεύεται από ένα μέγιστο αριθμό επαναλήψεων k. Οι συναρτήσεις για τους βρόγχους είναι αναδρομικά ορισμένες πάνω στους σύνθετους κόμβους της ροής εργασίας (workflow).

Παράδειγμα εκτέλεσης για τις δύο πιο πάνω περιπτώσεις παρουσιάζεται στο Σχήμα 3.1 όπου για το switch με δύο περιπτώσεις και κόστος C1 και C2 αντίστοιχα εφαρμόζεται η εξίσωση για το ολικό: p*C1 +(1-p)*C2 . Για το Loop το κόστος υπολογίζεται σαν k*Ci.

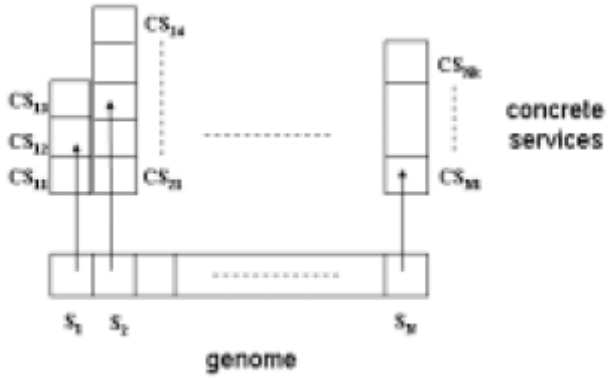


Σχήμα 3.1: Παράδειγμα εκτέλεσης : a)Switch b)Loop [5]

3.3.3 Επιλογή λύσης με γενετικούς αλγόριθμους

Το πρώτο στάδιο για την κωδικοποίηση του προβλήματος η κωδικοποίηση του χρωμοσώματος γίνεται έτσι ώστε να είναι κατάλληλο να βρεθεί κάποια καλή λύση, αποδοτικά. Για την λύση του συνδυαστικού αυτού προβλήματος , στην δεύτερη προσέγγιση, το χρωμόσωμα αναπαριστάται σαν ένας μονοδιάστατος πίνακας ακεραίων και το μέγεθος του είναι σταθερό. Το μέγεθος του πίνακα είναι ίσο με τον αριθμό των αφηρημένων υπηρεσιών που συνθέτουν την τελική λειτουργία. Κάθε

θέση του πίνακα είναι ένας δείκτης σε μια λίστα με τις υποψήφιες συγκεκριμένες υπηρεσίες που μπορεί να επιλεγούν για την σύνθεση. Το Σχήμα 3.2 δίνει μια γραφική αναπαράσταση του χρωμοσώματος.



Σχήμα 3.2: Κωδικοποίηση χρωμοσώματος με μονοδιάστατο πίνακα [5]

Η επόμενη παράμετρος η οποία θα πρέπει να κωδικοποιηθεί είναι η αντικειμενική συνάρτηση(fitness function). Η τιμή της αντικειμενικής συνάρτησης όσο μεγαλύτερη είναι τόσο πιο καλή είναι η λύση γι αυτό μπαίνουν στον αριθμητή τα θετικά μη λειτουργικά χαρακτηριστικά και στον παρονομαστή τα αρνητικά. Η αντικειμενική συνάρτηση (fitness function) που επιλέχθηκε είναι η

$$F(g) = \frac{w_1\ Availability\ +\ w_2\ Reliability}{w_3\ Cost\ +\ w_4\ Response\ Time}$$ [5].

3.3.4 Προβλήματα δεύτερης προσέγγισης

Όπως και στην προηγούμενη προσέγγιση έτσι και αυτή η προσέγγιση έχει κάποιες αδυναμίες. Αρχικά θα γίνει μια αναφορά στην κωδικοποίηση του χρωμοσώματος. Για να υπάρχει καλή σύνθεση μιας λειτουργίας το χρωμόσωμα δεν πρέπει να εκφράζει μόνο τις διάφορες συστατικές αφηρημένες υπηρεσίες της σύνθετης λειτουργίας αλλά να δίνει και πληροφορίες για τα διάφορα μονοπάτια σύνθεσης. Το μονοδιάστατο χρωμόσωμα το οποίο υιοθετήθηκε σε αυτή την προσέγγιση δεν μπορεί να εκφράσει όλα τα μονοπάτια της σύνθετης υπηρεσίας ταυτόχρονα. Μπορεί να γίνει η επεξεργασία μόνο του τρέχον μονοπατιού. Λόγω αυτού η προσέγγιση δεν είναι αποδοτική στην επιλογή υπηρεσιών όπου υπάρχουν πολλά μονοπάτια.

Το δεύτερο πρόβλημα που προκύπτει είναι ότι η σύνθεση υπηρεσιών χαρακτηρίζεται από πολλά σενάρια όπως τα probabilistic invocation, parallel invocation και sequential activation. Το μονοδιάστατο χρωμόσωμα δεν μπορεί να εκφράσει όλα τα σενάρια ταυτόχρονα.

Το τρίτο πρόβλημα έγκειται στο ότι δεν μπορούσε να παρουσιαστεί αποτελεσματικά ο επανασχεδιασμός της σύνθετης υπηρεσίας. Δηλαδή όταν μια συστατική υπηρεσία δεν ήταν πλέον διαθέσιμη ή ήταν δεσμευμένη από κάποια άλλη υπηρεσία και δεν μπορούσε να χρησιμοποιηθεί, δεν υπήρχε κάποιος αποτελεσματικός τρόπος με τον οποίο θα ξανασχεδιαζόταν η σύνθετη υπηρεσία με άλλες που ήταν διαθέσιμες. Ουσιαστικά αυτό το πρόβλημα είναι επέκταση του $1^{ου}$ προβλήματος γιατί όταν μια αφηρημένη συστατική υπηρεσία δεν είναι διαθέσιμη θα πρέπει να ακολουθηθεί κάποιο άλλο μονοπάτι που δεν εμπεριέχει αύτη την συστατική υπηρεσία.

Το τελευταίο πρόβλημα της $2^{ης}$ προσέγγισης είναι ότι δεν μπορεί να παρουσιάσει αποτελεσματικά τα κυκλικά μονοπάτια. Όταν ξεκινά η σύνθεση από μια υπηρεσία περνά από κάποιες άλλες και καταλήγει ξανά σ' αυτήν κάνοντας συνέχεια κύκλο χωρίς να φτάνει στην τελική υπηρεσία που ολοκλήρωνε την σύνθεση. Αυτό έχει ως αποτέλεσμα να μπαίνει η διαδικασία σε έναν ατέρμονο βρόγχο[8, 19].

3.4 Τρίτη Προσέγγιση

Ο κύριος στόχος της τρίτης προσέγγισης είναι, εκτός από το να επιλέγει τον κατάλληλο συνδυασμό μη λειτουργικών χαρακτηριστικών των συγκεκριμένων υπηρεσιών που ικανοποιούν τους στόχους της σύνθετης υπηρεσίας, να μπορεί μέσω της κωδικοποίησης του χρωμοσώματος να λύσει τα προβλήματα της προηγούμενης προσέγγισης. Δηλαδή, να μπορούν να εκφραστούν όλα τα μονοπάτια της σύνθετης λειτουργίας ταυτόχρονα, να παρουσιάζονται τα διάφορα σενάρια που χαρακτηρίζουν τη σύνθεση υπηρεσιών και τον επανασχεδιασμό της σύνθετης υπηρεσίας[10].

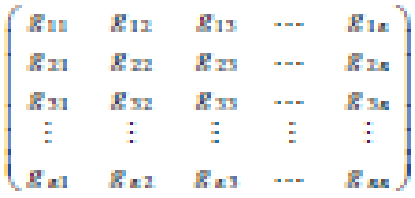
3.4.1 Περιγραφή προσέγγισης

Μια σύνθετη υπηρεσία αποτελείται από μια σειρά από αφηρημένες συστατικές υπηρεσίες οι οποίες θα πρέπει να ικανοποιηθούν για να παραχθεί η τελική λειτουργία. Όλες αυτές οι αφηρημένες υπηρεσίες μπορούν να ολοκληρωθούν από τις

συγκεκριμένες υποψήφιες υπηρεσίες. Σε αυτή την προσέγγιση η κάθε αφηρημένη υπηρεσία θεωρείται σαν ένας στόχος ο οποίος θα πρέπει να εκπληρωθεί. Άρα με τη σειρά τους και τα μονοπάτια είναι μια ακολουθία από στόχους , αφού είναι μια ακολουθία από αφηρημένες υπηρεσίες. Με αυτή την αντιστοιχία η σύνθετη υπηρεσία αποτελείται από μια σειρά αφηρημένες υπηρεσίες, στόχους, που συνθέτουν την τελική λειτουργία και είναι στο σύνολο T={t1,t2…,tn-1,tn}. Κάθε ένας από αυτούς τους στόχους μπορεί να ικανοποιηθεί από ένα σύνολο συγκεκριμένων υπηρεσιών {CTi,1 , … ,CTi,m} το οποίο σύνολο μπορεί να είναι διαφορετικό για κάθε στόχο[10].

3.4.2 Επιλογή λύσης με γενετικούς αλγόριθμους

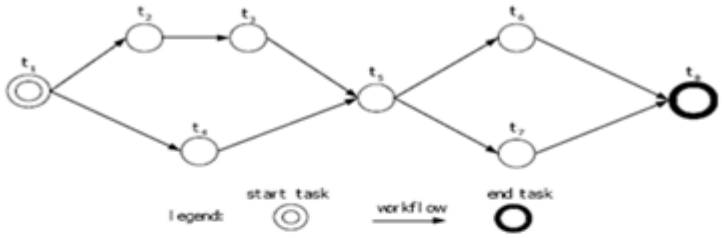
Για να κωδικοποιηθεί το πρόβλημα όπως και στις προηγούμενες προσεγγίσεις, αρχικά θα γίνει η κωδικοποίηση του χρωμοσώματος. Το χρωμόσωμα σε αυτή την προσέγγιση δεν είναι μονοδιάστατο. Αναπαριστάται σαν ένας γράφος , ένας δισδιάστατος πίνακας . Με αυτή την κωδικοποίηση γίνεται προσπάθεια να εξαλειφθούν τα προβλήματα που είχε η 2^{η} προσέγγιση και να παράγεται μια πιο αποδοτική λύση. Το μέγεθος του πίνακα είναι σταθερό και ίσο με n*n όπου n όλες οι αφηρημένες υπηρεσίες της σύνθεσης. Η διαγώνιος του πίνακα αποτελείται και αυτή από όλες τις αφηρημένες υπηρεσίες που μπορεί να μπουν σε κάποιο από τα μονοπάτια σύνθεσης της τελικής λειτουργίας, όπως ήταν αναμενόμενο λόγω και του μεγέθους του πίνακα. Η γραφική αναπαράσταση του γράφου φαίνεται παρακάτω στο Σχήμα 3.3.



Σχήμα 3.3: Κωδικοποίηση χρωμοσώματος με γράφο [8, 19]

Έκτος από την διαγώνιο υπάρχει και η κωδικοποίηση της άνω και της κάτω διαγωνίου του πίνακα. Το άνω, τριγωνικό κομμάτι του πίνακα δείχνει την σχέση μεταξύ της αφηρημένης υπηρεσίας στην διαγώνιο της συγκεκριμένης γραμμής με την αφηρημένη

υπηρεσία στην διαγώνιο που αντιστοιχεί στην συγκεκριμένη στήλη. Δηλαδή η θέση g1,2 δείχνει την σχέση μεταξύ της αφηρημένης υπηρεσίας g1,1 με την υπηρεσία g2,2. Με αυτόν τον τρόπο μπορεί να εξαχθεί το state chart που αντιστοιχεί στον πίνακα. Επίσης κάθε θέση του πίνακα gij μπορεί να δείξει τις διάφορες καταστάσεις για το parallel invocation, ανάλογα με την τιμή την οποία μπορεί να πάρει από ένα συγκεκριμένο σύνολο {κ1, κ2, κ3, κ4, 0, p, m} , p={0,..,1}. Η κάθε τιμή αντιστοιχεί σε μια λειτουργία η οποία μπορεί να εκτελεστεί. Παρακάτω στο Σχήμα 3.4 φαίνεται ένα παράδειγμα ενός state chart που απορρέει από την κατάσταση του γράφου μια δεδομένη στιγμή.



$$\begin{pmatrix} r_1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & r_2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_5 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & r_6 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_7 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_8 \end{pmatrix}$$

Σχήμα 3.4: Παράδειγμα γράφου μαζί με state chart [10]

Η επόμενη παράμετρος η οποία θα πρέπει να κωδικοποιηθεί και είναι μια βασική προϋπόθεση για την χρήση των γενετικών αλγορίθμων είναι η αντικειμενική συνάρτηση (fitness function). Προτάθηκαν πολλές αντικειμενικές συναρτήσεις για την $3^{η}$ προσέγγιση. Η καλύτερη από αυτές είναι η παρακάτω:

$$f = \frac{\sum_j \left(\frac{Q_j - Q_j^{\min}}{Q_j^{\max} - Q_j^{\min}} \times w_j\right)}{\sum_k \left(\frac{Q_k - Q_k^{\min}}{Q_k^{\max} - Q_k^{\min}} \times w_k\right)}, \quad if\ Q_k^{\max} - Q_k^{\min} = 0 \text{ then } \frac{Q_k - Q_k^{\min}}{Q_k^{\max} - Q_k^{\min}} = 1$$

Στην παραπάνω συνάρτηση τα wj, wk είναι πραγματικοί θετικοί αριθμοί που αντιπροσωπεύουν τα βάρη και παίρνουν τιμές μέσα από το σύνολο {0,1}. Οι παράμετροι Qj, Qk είναι μη λειτουργικά χαρακτηριστικά που είναι διαφορετικά μεταξύ τους. Στον αριθμητή βρίσκονται τα θετικά μη λειτουργικά χαρακτηριστικά και στον παρονομαστή τα αρνητικά μη λειτουργικά χαρακτηριστικά[10].

3.5 Τέταρτη προσέγγιση

Ο στόχος της τέταρτης προσέγγισης είναι να ανακαλύπτεται ο βέλτιστος καθορισμός της κάθε αφηρημένης υπηρεσίας μιας ευέλικτης διαδικασίας(σύνθετης υπηρεσίας) και η υπηρεσία ιστού (Web Service) που υλοποιεί την αφηρημένη αυτή περιγραφή. Αυτή η Web Service πρέπει να είναι τέτοια ώστε τα μη λειτουργικά χαρακτηριστικά (QoS) τα οποία αιτείται ο χρήστης να μεγιστοποιούνται κάτω από τους διάφορους QoS περιορισμούς. Πολλοί περιορισμοί είναι πολύ σημαντικοί κάθε φορά που η τελική λειτουργία πρέπει να διενεργηθεί με αυστηρά περιορισμένους πόρους. Η προσέγγιση αυτή υλοποιήθηκε με την αρχιτεκτονική MAIS, που είναι μια πλατφόρμα η οποία υποστηρίζει την εκτέλεση σύνθετων λειτουργιών σε συστήματα με πολλαπλά κανάλια[6].

3.5.1 Περιγραφή προσέγγισης

Μια υπηρεσία ιστού (Web Service) μοντελοποιείται σαν ένα συστατικό λογισμικού που υλοποιεί ένα σύνολο από λειτουργίες. Άρα μια σύνθετη υπηρεσία ορίζεται σε ένα αφηρημένο επίπεδο σαν μια υψηλού επιπέδου επιχειρηματική διαδικασία. Στην προσέγγιση αυτή γίνεται η υπόθεση ότι η σύνθετη υπηρεσία χαρακτηρίζεται από ένα ενιαίο αρχικό στόχο και ένα ενιαίο τελικό στόχο. Η λέξη στόχος (ti) αναφέρεται σε μια αφηρημένη υπηρεσία που είναι συστατικό της σύνθετης υπηρεσίας. Η συγκεκριμένη υπηρεσία δίνεται με την συντομογραφία wsj και είναι υποψήφια για την εκτέλεση του στόχου ti με την λειτουργία που καθορίζεται από το wsj,o. Το I αναφέρεται στον

αριθμό των στόχων της σύνθετης υπηρεσίας και το J στον αριθμό των υποψήφιων υπηρεσιών ιστού (Web Services) που ανακτώνται από το μητρώο της MAIS αρχιτεκτονικής, που είναι προέκταση του UDDI. Τα μη λειτουργικά χαρακτηριστικά θα αναλυθούν σύμφωνα με το σχέδιο συνάθροισης (aggregation pattern) τους και τη σωστή διαπραγμάτευση. Το πρώτο μέτρο ανάλυσης ορίζει πως η τιμή ενός χαρακτηριστικού για μια σύνθετη υπηρεσία μπορεί να εξακριβωθεί ξεκινώντας από την τιμή που είχε στις συγκεκριμένες συστατικές υπηρεσίες του κάθε στόχου. Το δεύτερο μέτρο ανάλυσης χρησιμοποιείται για την εύρεση μιας μέσης λύσης, κατά την φάση της επιλογής των υπηρεσιών ιστού (Web Services) εάν αυτή δεν βρέθηκε ικανοποιητική λύση μέχρι εκείνη την στιγμή.

Τα μη λειτουργικά χαρακτηρίστηκα τα οποία αξιολογούνται είναι τέσσερα. Όπως φαίνεται και στον πίνακα 3.2 γίνεται χρήση των χαρακτηριστικών χρόνος εκτέλεσης (execution time) και τιμή (price), όπου σύμφωνα με το σχέδιο συνάθροισης δίνονται από το άθροισμα του χρόνου εκτέλεσης και τιμών αντίστοιχα, των συγκεκριμένων υπηρεσιών που επιλέχθηκαν. Ακολουθεί η διαθεσιμότητα (availability) που δίνεται από το πηλίκο όλων των διαθεσιμοτήτων των συστατικών συγκεκριμένων υπηρεσιών. Επίσης χρησιμοποιείται και η φήμη (reputation), που δίνεται από το μέσο όρο της φήμης των συστατικών υπηρεσιών. Τέλος χρησιμοποιείται η ποιότητα των δεδομένων (data quality), που δίνεται από την ελάχιστη τιμή της ποιότητας δεδομένων των συστατικών υπηρεσιών.

Πίνακας 3.2: Συναρτήσεις μη λειτουργικών χαρακτηριστικών [6]

Quality Dimension	Aggregation function
Price	$price_k(EPL) = \sum_{\substack{t_i \in ep_k, \\ (t_i, ws_{j,o}) \in EPL}} p_{j,o}$
Reputation	$rep_k(EPL) = \frac{1}{\lvert A_k \rvert} \sum_{\substack{t_i \in ep_k, \\ (t_i, ws_{j,o}) \in EPL}} r_{j,o}$
Execution Time	$exeTime_k(EPL) = \max_{sp_m^k \in ep_k} \sum_{\substack{t_i \in sp_m^k, \\ (t_i, ws_{j,o}) \in EPL}} e_{j,o}$
Availability	$avail_k(EPL) = \prod_{\substack{t_i \in ep_k, \\ (t_i, ws_{j,o}) \in EPL}} a_{j,o}$
Data quality	$dq_k(EPL) = \min_{\substack{t_i \in ep_k, \\ (t_i, ws_{j,o}) \in EPL}} d_{j,o}$

Το πρόβλημα της επιλογής των υπηρεσιών αυτή η προσέγγιση προσπαθεί να το λύσει χρησιμοποιώντας ακέραιο γραμμικό προγραμματισμό (mixed integer linear programming). Οι μεταβλητές απόφασης του προβλήματος είναι η zi,j που είναι ίση με 1 όταν ο στόχος ti εκτελείται από την υπηρεσία ιστού (Web Service) wsj όπου j ανήκει στο σύνολο WSi αλλιώς η τιμή της είναι 0. Η άλλη μεταβλητή είναι η yi,j,o που είναι ίση με 1 αν ο στόχος ti εκτελείται από την wsj,o αλλιώς είναι 0. Ο στόχος του προβλήματος είναι να μεγιστοποιήσει την τιμή της συνάρτησης των QoS σύμφωνα με τα διάφορα σενάρια εκτέλεσης. Η τιμή αυτή των QoS λαμβάνεται με την εφαρμογή της τεχνικής της απλής εφαρμογής βαρών (Simple Additive Weighting), κατά την οποία λαμβάνεται ένα σκορ από μια λίστα από διαστάσεις. Άρα παίρνεται το (score k(Y)) που είναι η τιμή για το σχέδιο εκτέλεσης Y κατά την διάρκεια εκτέλεσης του μονοπατιού ep k. Επειδή όμως οι τιμές των χαρακτηριστικών έχουν διαφορετικό πεδίο ορισμού η τεχνική αυτή πριν να αρχίσει την επεξεργασία τις κανονικοποιεί. Τελικά το σκορ δίνεται από την συνάρτηση:

$$score_k(\mathcal{Y}) = \sum_{n=1}^{N} w_n v_n^k(\mathcal{Y})$$

Όπου το Un(Y) είναι η κανονικοποιημένη τιμή του χαρακτηριστικού n στο μονοπάτι ep k. Το βέλτιστο σχέδιο εκτέλεσης βρίσκεται με την λύση του παρακάτω προβλήματος που παρουσιάζεται στο Σχήμα 3.5, που ονομάζεται P1.

$$\textbf{P1.} \qquad \max \sum_{k=1}^{K} freq_k \cdot score_k(\mathcal{Y}).$$

$$\sum_{j \in \mathcal{WS}_i} \sum_{o \in \mathcal{OP}_j} y_{i,j,o} = 1; \qquad \forall i \qquad (4)$$

$$y_{i,j,\rho} \leq z_{i,j}; \qquad \forall i; \forall j \in \mathcal{WS}_i; \ \forall o \in \mathcal{OP}_j \qquad (5)$$

$$\sum_{j \in \mathcal{WS}_i} z_{i,j} = 1; \qquad \forall i \qquad (6)$$

$$\sum_{j \in \mathcal{WS}_i} \sum_{o \in \mathcal{OP}_j} e_{j,\rho} y_{i,j,o} = exeT_i; \qquad \forall i \qquad (7)$$

$$x_{i_2} - (exeT_{i_1} + x_{i_1}) \geq 0; \qquad \forall t_{i_1} \rightarrow t_{i_2} \qquad (8)$$

$$\sum_{i \in sp_m^k} exeT_i \leq exeTime_k; \qquad \forall sp_m^k \in ep_k \qquad (9)$$

$$avail_k = \prod_{i \in \mathcal{A}_k} \prod_{j \in \mathcal{WS}_i} \prod_{o \in \mathcal{OP}_j} a_{j,o}^{y_{i,j,o}} \qquad \forall k \qquad (10)$$

$$price_k = \sum_{i \in \mathcal{A}_k} \sum_{j \in \mathcal{WS}_i} \sum_{o \in \mathcal{OP}_j} p_{j,o} y_{i,j,o} \qquad \forall k \qquad (11)$$

$$rep_k = \frac{1}{|\mathcal{A}_k|} \sum_{i \in \mathcal{A}_k} \sum_{j \in \mathcal{WS}_i} \sum_{o \in \mathcal{OP}_j} r_{j,o} y_{i,j,\rho} \qquad \forall k \qquad (12)$$

$$\sum_{j \in \mathcal{WS}_i} \sum_{o \in \mathcal{OP}_j} d_{j,o} y_{i,j,o} \geq dq_k; \qquad \forall k; \forall i \in \mathcal{A}_k \qquad (13)$$

$$\sum_{j \in \mathcal{WS}_i} \sum_{o \in \mathcal{OP}_j} r_{j,o} y_{i,j,\rho} \geq R; \qquad \forall i \in \mathcal{B} \qquad (14)$$

$$exeTime_k \leq E'; \qquad \forall k \qquad (15)$$

$$avail_k \geq A; \qquad \forall k \qquad (16)$$

$$price_k \leq B; \qquad \forall k \qquad (17)$$

$$rep_k \geq R; \qquad \forall k \qquad (18)$$

$$dq_k \geq DQ; \qquad \forall k \qquad (19)$$

$$y_{i,j,\rho}, z_{i,j} \in \{0,1\}; \qquad \forall i; \forall j \in \mathcal{WS}_i; \ \forall o \in \mathcal{OP}_j$$

$$exeT_i, x_i \in \mathbb{R}^+ \qquad \forall i;$$

$$exeTime_k, avail_k, price_k, rep_k, dq_k \in \mathbb{R}^+; \qquad \forall k.$$

Σχήμα 3.5: Κώδικας για το πρόβλημα του P1 [6]

Τα γράμματα A, R, DQ, B όπως φαίνονται παραπάνω είναι οι σφαιρικοί περιορισμοί για την διαθεσιμότητα (availability), τη φήμη (reputation), την ποιότητα των δεδομένων (data quality) και της τιμής (price) αντίστοιχα. Για το χρόνο εκτέλεσης είναι το E΄ που προέρχεται από E΄=E – OptTime που το OptTime είναι μια εκτίμηση του χρόνου που χρειάζεται για να βελτιστοποιηθεί η σύνθετη υπηρεσία. Οι περιορισμοί αυτοί εγγυώνται ότι κάθε στόχος συνδέεται με την επίκληση μιας λειτουργίας μιας υπηρεσίας. Επίσης συνδέουν τις δύο μεταβλητές απόφασης και εκφράζουν την διάρκεια του κάθε στόχου σε συνάρτηση της διάρκειας της επιλεγμένης υπηρεσίας.

Επιπρόσθετα ελέγχουν πότε και με ποια σειρά θα γίνεται η εκτέλεση των στόχων και κάνουν την αποτίμηση των μη λειτουργικών χαρακτηριστικών σύμφωνα με τον πίνακα 3.2.

Το βέλτιστο σχέδιο εκτέλεσης αναθέτει την καλύτερη λειτουργία της υποψήφιας υπηρεσίας ws j,o που ανήκει στο σύνολο WSi σε κάθε στόχο ti. Ακολούθως μετά από την αποτίμηση του κάθε στόχου το σύνολο των υποψήφιων υπηρεσιών, αυτές κατατάσσονται ανάλογα με την τιμή που παρέχεται από την στοχαστική συνάρτηση του προβλήματος P1 [6].

3.5.2 Προβλήματα τέταρτης Προσέγγισης

Σε γενικές γραμμές η προσέγγιση αυτή ήταν καλή. Βρίσκει πάντοτε μια εφικτή λύση αφού χρησιμοποιεί και κάποιες τεχνικές διαπραγμάτευσης. Επίσης η λύση η οποία δίνεται είναι σφαιρική και όχι απλώς τοπική. Παρόλα αυτά η προσέγγιση έχει και κάποιες αδυναμίες. Δεν μπορεί να κάνει βελτιστοποίηση της εκτέλεσης πολλαπλών στιγμιότυπων μιας διαδικασίας. Αυτό είναι σημαντικό γιατί στην προσέγγιση όπως είναι υλοποιημένη αν ένας μεγάλος αριθμός από πελάτες εκχωρούνται στην ίδια υπηρεσία , σοβαροί όροι φόρτωσης θα εμφανιστούν και η ποιότητα της υπηρεσίας θα μειωθεί. Ακόμα μία έλλειψη είναι ότι δεν υπάρχει μια συνάρτηση η οποία να είναι μη γραμμική (non – linear) και έτσι δεν μπορεί εμπλακούν μη γραμμικές (non- linear) μέθοδοι κατά την αναζήτηση. Αυτό αποτρέπει το γεγονός να συνδυάζονται οι δύο τρόποι λύσης (non- linear, linear) και να συγκρίνονται τα αποτελέσματα τους για περεταίρω βελτιστοποίηση της προσέγγισης ή ακόμα και την δημιουργία μιας υβριδικής προσέγγισης η οποία να είναι πιο αποτελεσματική. Η αποδοτικότητα των μεθόδων διαπραγμάτευσης που προτάθηκαν δεν αναλύθηκαν και πρέπει να αναλυθούν γιατί μπορεί να πρέπει να γίνει κάποια διόρθωση ή βελτιστοποίηση[6].

3.6 Πέμπτη Προσέγγιση

Στην πέμπτη προσέγγιση γίνεται προσπάθεια επίλυσης του συνδυαστικού προβλήματος των υπηρεσιών λαμβάνοντας υπόψη όχι μόνο τις αντικειμενικές πληροφορίες των μη λειτουργικών χαρακτηριστικών QoS αλλά και τις υποκειμενικές. Οι αντικειμενικές πληροφορίες είναι οι πληροφορίες των μη λειτουργικών

χαρακτηριστικών που παρέχονται από τον προμηθευτή των υπηρεσιών (service provider). Οι υποκειμενικές πληροφορίες αναφέρονται στις πληροφορίες των μη λειτουργικών χαρακτηριστικών τις οποίες χρειάζεται ουσιαστικά ο πελάτης, ο οποίος χρησιμοποιεί την υπηρεσία. Με την εμπλοκή των υποκειμενικών πληροφοριών των μη λειτουργικών χαρακτηριστικών γίνεται προσπάθεια να βελτιωθεί η ποιότητα της αναζήτησης των υπηρεσιών Ιστού (Web Services). Ο συνδυασμός των δύο τύπων πληροφοριών δημιουργεί μια διεπαφή συγκεχυμένων μη λειτουργικών χαρακτηριστικών [16].

3.6.1 Περιγραφή πέμπτης προσέγγισης

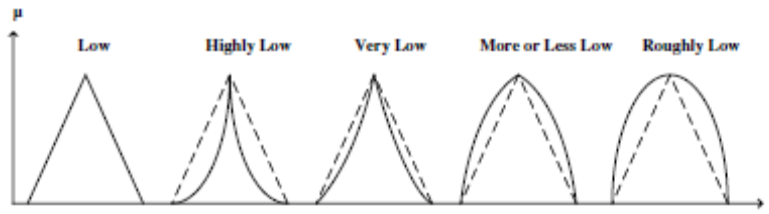
Η αρχιτεκτονική της πέμπτης προσέγγισης αποτελείται από τέσσερις οντότητες. Αυτές οι οντότητες είναι το μητρώο του UDDI, ο πελάτης, ο προμηθευτής των υπηρεσιών και ο πράκτορας που είναι βασισμένος στα μη λειτουργικά χαρακτηριστικά. Προχωρώντας, στο σύστημα υπάρχουν δύο βασικά συστατικά. Το πρώτο είναι ένα συγκεχυμένο συστατικό το οποίο ασχολείται με τα αντικειμενικά μη λειτουργικά χαρακτηριστικά. Το δεύτερο συστατικό συνεργάζεται με τον πελάτη/χρήστη για την εύρεση των υποκειμενικών μη λειτουργικών χαρακτηριστικών.

Η ροή εργασίας του συστήματος αρχίζει με το να δεχτεί ένα ερώτημα από τον χρήστη. Η λέξη κλειδί του ερωτήματος του χρήστη χρησιμοποιείται για τη εύρεση των υπηρεσιών από το μητρώο. Μετά από το ερώτημα το οποίο τίθεται ο χρήστης μπορεί να δώσει και τις απαιτήσεις του σε σχέση με τα μη λειτουργικά χαρακτηριστικά για του ξαναδοθούν οι συνιστώμενες υπηρεσίες. Όλα τα μη λειτουργικά χαρακτηριστικά που δίνονται στο σύστημα από το χρήστη είναι σε μορφή συγκεχυμένων μεταβλητών (fuzzy variables). Ακολούθως ο πράκτορας βασίζεται στις απαιτήσεις του χρήστη σε σχέση με τα μη λειτουργικά χαρακτηριστικά και ελέγχει τις διαθέσιμες υπηρεσίες. Οι τιμές των μη λειτουργικών χαρακτηριστικών είναι μεταξύ του [0-1]. Το συγκεχυμένο συστατικό του συστήματος παίρνει τις συγκεχυμένες μεταβλητές που του έδωσε ο χρήστης και οι τιμές τους είναι μεταξύ του ενός και του τρία. Οι τιμές των συγκεχυμένων μεταβλητών κανονικοποιούνται στο διάστημα του [0-1].

Το συγκεχυμένο συστατικό λαμβάνει τις απαιτήσεις του χρήστη με το να συγχέει τις αντικειμενικές πληροφορίες των μη λειτουργικών χαρακτηριστικών. Τα αντικειμενικά μη λειτουργικά χαρακτηριστικά τα οποία χρησιμοποιούνται είναι η διαθεσιμότητα, η αξιοπιστία, ο χρόνος απόκρισης και την ρυθμαπόδοση (throughput). Κάθε αντικειμενικό χαρακτηριστικό συνοδεύεται από τρείς όρους(linguistic). Οι όροι είναι A = (low, medium,high), B = (low, medium,high), 0–1, C = (short, medium,long), D = (short, medium,long), E = (low, medium, high) για κάθε ένα από τα παραπάνω μη λειτουργικά αντίστοιχα. Επίσης εκτός από τους όρους αυτούς κάθε αντικειμενικό χαρακτηριστικό χαρακτηρίζεται και από μια συνάρτηση. Τέλος χρησιμοποιούνται κάποιοι συγκεχυμένοι κανόνες για να βρεθεί κατά πόσο είναι ιδανική η τιμή των μη λειτουργικών χαρακτηριστικών [16].

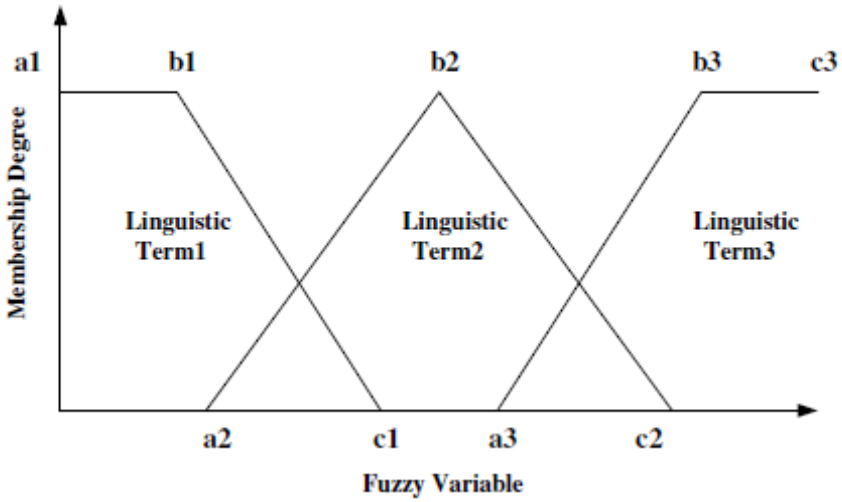
3.6.2 Επιλογή λύσης με γενετικούς αλγόριθμους

Για να λυθεί το συνδυαστικό πρόβλημα της σύνθεσης των υπηρεσιών και σε αυτή την προσέγγιση έγινε χρήση των γενετικών αλγόριθμων [13,20]. Για την κωδικοποίηση της πρώτης παραμέτρου των γενετικών αλγορίθμων, του χρωμοσώματος, συνδυάστηκαν οι συγκεχυμένοι κανόνες και η συνάρτηση που συνοδεύει κάθε μη λειτουργικό χαρακτηριστικό. Οι γενιές του χρωμοσώματος χωρίζονται σε δύο μέρη. Το πρώτο μέρος κωδικοποιεί με αντιστάθμιση τον συγκεχυμένο κανόνα, χρησιμοποιώντας κάποιους όρους. Οι όροι αυτοί οι οποίοι χρησιμοποιήθηκαν αντιστοιχήθηκαν με κάποιες τιμές οι οποίες θα μπαίνουν στο χρωμόσωμα. Οι τιμές αυτές ξεκινούν από το μηδέν το οποίο υποδηλώνει ότι δεν υπάρχει αλλαγή στην τιμή. Ακολουθεί η τιμή ένα η οποία είναι αντιστάθμιση του περίπου (“ roughly ”), η τιμή δύο που αντιπροσωπεύει το κάπως (‘‘ somewhat ’’), η τιμή τρία που αντιπροσωπεύει το πολύ (‘‘ very ’’) και τέλος η τιμή 4 που αντιστοιχεί στο υψηλό (‘‘ highly ’’). Το δεύτερο μέρος του χρωμοσώματος ορίζει κάποια σημεία της συνάρτησης των μη λειτουργικών χαρακτηριστικών, των οποίων τα όρια των τιμών φαίνονται στο Σχήμα 3.6.



Σχήμα 3.6 : Τα διάφορα όρια της συνάρτησης των μη λειτουργικών χαρακτηριστικών [16]

Τα σημεία της συνάρτησης που ορίζει το χρωμόσωμα έχουν κάποιους περιορισμούς και χαρακτηρίζονται από τρείς όρους(linguistic). Οι περιορισμοί φαίνονται στο Σχήμα 3.7.



Σχήμα 3.7: Οι περιορισμοί των σημείων της συνάρτησης των μη λειτουργικών χαρακτηριστικών [16]

Η δεύτερη παράμετρος που κωδικοποιείται για την χρήση των γενετικών αλγορίθμων είναι η αντικειμενική συνάρτηση (fitness function). Ως αντικειμενική συνάρτηση χρησιμοποιήθηκε η συνάρτηση του μέσου τετραγωνικού σφάλματος (mean square error, MSE) η οποία είναι η $\mathrm{MSE} = (y_{ac} - y_{pr}^{i})^2$ $f = \frac{1}{\mathrm{MSE}}$. Η παράμετρος Yac είναι ο βαθμός καταλληλότητας των μη λειτουργικών χαρακτηριστικών (QoS) σύμφωνα με τον χρήστη

και το ypr είναι ο βαθμός καταλληλότητας των μη λειτουργικών χαρακτηριστικών (QoS) της υπηρεσίας που επιλέχτηκε [16].

3.7 Σύγκριση Προσεγγίσεων με το υλοποιημένο σύστημα

Η έρευνα η οποία έγινε πάνω στην σύνθεση των υπηρεσιών εκτός από το ότι προσπάθησε να λύσει το συνδυαστικό αυτό πρόβλημα με αποδοτικό και αποτελεσματικό τρόπο, στόχευε και στο να αποφύγει και ίσως να διορθώσει προβλήματα τα οποία είχαν οι προηγούμενες προσεγγίσεις. Χρειάζεται να γίνει μια σύγκριση των προηγούμενων προσεγγίσεων με την παρούσα έρευνα για να παρουσιαστούν τα σημεία στα οποία διαφοροποιούνται για την αποφυγή ή διόρθωση των προβλημάτων που υπήρξαν.

Στην πρώτη προσέγγιση [2] το βασικό πρόβλημα ήταν η φτωχή αναγνωσιμότητα -αποκωδικοποίηση- του χρωμοσώματος λόγω του ότι το μέγεθος του δεν ήταν σταθερό. Αυτή η μη σταθερότητα του μεγέθους του χρωμοσώματος σε κάποιες περιπτώσεις το έκανε υπερβολικά μεγάλο αφού όλες οι υποψήφιες συγκεκριμένες υπηρεσίες βρίσκονταν σε μια από τις γενεές του χρωμοσώματος. Στην προσέγγιση η οποία υλοποιήθηκε και πάλι το χρωμόσωμα δεν έχει το ίδιο μέγεθος σε όλες τις περιπτώσεις όμως το πρόβλημα της φτωχής αναγνωσιμότητας δεν παρουσιάζεται. Το πρόβλημα αυτό λύθηκε γιατί δεν αποτελούν όλες οι υποψήφιες συγκεκριμένες υπηρεσίες κάποια από τις γενεές του χρωμοσώματος (μια συγκεκριμένη υπηρεσία να αντιστοιχεί σε μια γενεά του χρωμοσώματος). Αντίθετα μπαίνει μια αφηρημένη υπηρεσία σε κάθε γενεά που αντιπροσωπεύει ένα σύνολο υποψήφιων υπηρεσιών που έχουν την ίδια λειτουργικότητα. Έτσι το μέγεθος του χρωμοσώματος μειώνεται πάρα πολύ και έτσι απαλείφεται το πρόβλημα της πρώτης προσέγγισης.

Επιπρόσθετα κάποιες άλλες ελλείψεις τις οποίες παρουσίαζε και η πρώτη αλλά και η δεύτερη προσέγγιση [5] είναι το ότι δεν παρουσίαζαν τα κυκλικά μονοπάτια και τον επανασχεδιασμό της σύνθετης υπηρεσίας. Στην παρούσα έρευνα για να μην παρουσιαστούν τα παραπάνω έγινε χρήση ενός γράφου που δείχνει όλα τα μονοπάτια της σύνθετης υπηρεσίας. Όλα τα μονοπάτια ξεκινούν από την ίδια και καταλήγουν στην ίδια αφηρημένη υπηρεσία, οι οποίες είναι καθορισμένες από την

αρχή. Δεν μπορούν να δημιουργηθούν έτσι κυκλικά μονοπάτια. Επίσης λόγω του γράφου μπορεί να επανασχεδιαστεί η σύνθετη υπηρεσία όταν κάποια αφηρημένη υπηρεσία δεν ικανοποιείται. Παρέχονται διαφορετικά μονοπάτια σύνθεσης.

Η τρίτη προσέγγιση[10] έλυσε και αυτή τα προβλήματα των δύο προηγούμενων προσεγγίσεων. Το χρωμόσωμα της τρίτης προσέγγισης ήταν δισδιάστατο. Το χρωμόσωμα ήταν ο γράφος με το σχέδιο λειτουργίας της σύνθετης υπηρεσίας. Μόνο όταν στο τρέχον μονοπάτι αποτύγχανε η σύνθεση επιλεγόταν κάποιο από τα εναλλακτικά. Αλλιώς τα εναλλακτικά μονοπάτια δεν ελέγχονταν. Στην παρούσα έρευνα το χρωμόσωμα είναι ένας μονοδιάστατος πίνακας. Ο γράφος με το σχέδιο λειτουργίας της σύνθετης υπηρεσίας χρησιμοποιείται σαν βοηθητική δομή. Σε κάθε εκτέλεση του προγράμματος γίνεται έλεγχος όλων των μονοπατιών σύνθεσης και επιλέγεται το καλύτερο.

Στην τέταρτη προσέγγιση[6] δεν γίνεται χρήση των γενετικών αλγορίθμων. Το κοινό χαρακτηριστικό όμως που έχει η τέταρτη προσέγγιση με την παρούσα είναι ο σφαιρικός υπολογισμός των μη λειτουργικών χαρακτηριστικών με τις διάφορες συναρτήσεις, οι οποίες εφαρμόζονται στα τοπικά μη λειτουργικά χαρακτηριστικά. Επιπλέον, η τέταρτη προσέγγιση χρησιμοποιεί την έννοια του μονοπατιού σύνθεσης όπως και η παρούσα προσέγγιση. Η διαφοροποίηση σε σχέση με την χρήση των μονοπατιών είναι ότι στην τέταρτη προσέγγιση κάθε ένα από τα μονοπάτια σύνθεσης συνοδεύεται από μια πιθανότητα σύμφωνα με την οποία εξετάζονται, κατά την εκτέλεση του προγράμματος.

Στην πέμπτη προσέγγιση[16] υπάρχει μια σοβαρή διαφοροποίηση σε σχέση με την έρευνα την οποία παρουσιάζει το παρών έγγραφο. Η πέμπτη προσέγγιση επιτρέπει στον ίδιο τον πελάτη ο οποίος θα χρησιμοποιήσει την υπηρεσία να δώσει τις απαιτήσεις του σε σχέση με τα μη λειτουργικά χαρακτηριστικά της υπηρεσίας που θα επιλεγεί. Στο παρών πρόγραμμα η επιλογή των υπηρεσιών γίνεται αποκλειστικά από το πρόγραμμα. Ο πελάτης που αιτείται τη σύνθετη λειτουργία δεν συμμετέχει στην διαδικασία της εύρεσης των υπηρεσιών.

Κεφάλαιο 4

4. Σχεδιασμός συστήματος

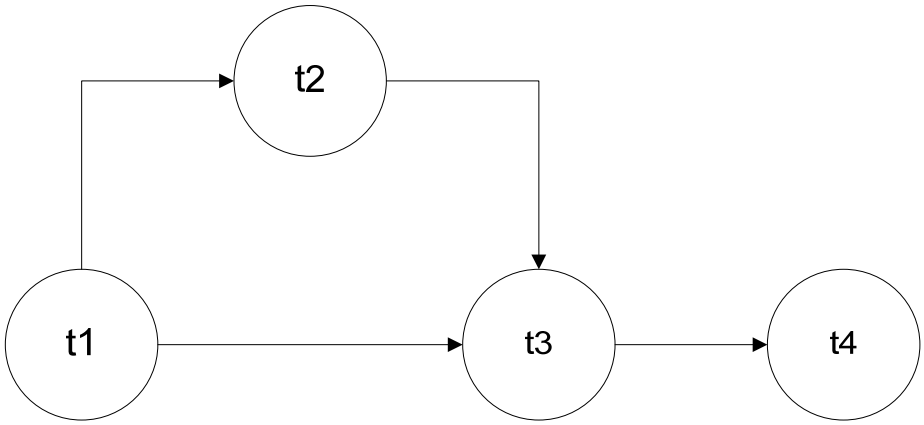
4.1 Περιγραφή προσέγγισης προτεινόμενου συστήματος

Στη διπλωματική αυτή δημιουργήθηκε μια πλατφόρμα για την δυναμική σύνθεση υπηρεσιών. Κάθε σύνθετη υπηρεσία αποτελείται από μια σειρά από συστατικές υπηρεσίες γνωστές ως αφηρημένες . Κάθε μια από αυτές μπορεί να θεωρηθεί ως ένας στόχος(task) ο οποίος θα πρέπει να ικανοποιηθεί. Σε ένα τυπικό σενάριο ζητείται μια λειτουργία (business process) που μπορεί να ικανοποιηθεί από μια σύνθετη υπηρεσία, π.χ. ένας χρήστης μπορεί να ζητήσει να δει ένα βίντεο στην συσκευή του. Η πλατφόρμα παίρνει ως παράμετρο το αφηρημένο σχέδιο της λειτουργίας που ζητεί ο χρήστης, το οποίο θα εκφράζεται ως μια σειρά από αφηρημένες συστατικές υπηρεσίες οι οποίες είναι τοποθετημένες σε ένα γράφο. Μία αφηρημένη υπηρεσία περιγράφεται από κάποιες λειτουργικές απαιτήσεις και για μια αφηρημένη υπηρεσία μπορεί να υπάρχουν μία η περισσότερες συγκεκριμένες υπηρεσίες που την ικανοποιούν λειτουργικά, όμως μπορεί να διαφέρουν σε υλοποίηση και να διατίθενται από διαφορετικούς παροχείς υπηρεσιών ιστού (Web Services) και άρα θα διαφέρουν στα μη-λειτουργικά χαρακτηριστικά (π.χ. χρόνο εκτέλεσης, κόστος, κτλ). Στόχος της πλατφόρμας είναι να εντοπίσει το σύνολο των υπηρεσιών (δηλ. για κάθε αφηρημένη υπηρεσία θα αντιστοιχεί μια συγκεκριμένη υπηρεσία) ώστε το τελικό αποτέλεσμα να βελτιστοποιεί κάποιους στόχους (π.χ. μεγιστοποίηση ποιότητας, μείωση συνολικού χρόνου εκτέλεσης, κτλ.). Η πλατφόρμα διαβάζει τα μη λειτουργικά χαρακτηριστικά τις κάθε υπηρεσίας και θα χρησιμοποιεί γενετικούς αλγόριθμους για να εντοπίζει τον βέλτιστο συνδυασμό υπηρεσιών.

Όπως προαναφέρθηκε η πλατφόρμα παίρνει ως είσοδο ένα σχέδιο λειτουργίας. Αυτό το σχέδιο είναι ένας πίνακας, ο οποίος αντιπροσωπεύει ένα κατευθυνόμενο γράφο. Ένα παράδειγμα του σχεδίου λειτουργίας παρουσιάζεται στον Πίνακα 4.1.

Πίνακας 4.1: Παράδειγμα σχεδίου λειτουργίας

t1	1	1	0
0	t2	1	0
0	0	t3	1
0	0	0	t4



Σχήμα 4.1: Γράφος που δημιουργείται από τον Πίνακα 4.1

Στην διαγώνιο του πίνακα που αναπαριστάται το σχέδιο λειτουργίας μπαίνουν όλες οι αφηρημένες υπηρεσίες οι οποίες μπορούν να λάβουν μέρος στην σύνθεση της τελικής λειτουργίας σε κάποια χρονική στιγμή. Στο άνω τριγωνικό κομμάτι του πίνακα μπορεί να διακριθεί η σχέση των αφηρημένων υπηρεσιών μεταξύ τους. Όταν υπάρχει ο αριθμός ένα σε κάποια θέση στο άνω τριγωνικό κομμάτι τότε η αφηρημένη υπηρεσία – στόχος της συγκεκριμένης γραμμής γειτονεύει με την αφηρημένη υπηρεσία - στόχο της συγκεκριμένης στήλης που εμφανίζεται στην διαγώνιο. Δηλαδή με βάση τον Πίνακα 4.1 ο t1 γειτονεύει με τον t2 και τον t3. Στο Σχήμα 4.1 φαίνονται καλύτερα οι σχέσεις των στόχων για το παραπάνω σχέδιο λειτουργίας μέσω ενός γράφου. Λόγω αυτών των σχέσεων που υπάρχουν μεταξύ των στόχων δημιουργούνται διάφορα μονοπάτια μέσω των οποίων μπορεί να γίνει η σύνθεση της τελικής λειτουργίας. Η πλατφόρμα αυτή που δημιουργήθηκε εκτός από την επιλογή του καλύτερου συνδυασμού των υπηρεσιών ιστού (Web Services) βρίσκει και το καλύτερο μονοπάτι για την σύνθεση.

Όπως προαναφέρθηκε κάθε στόχος πρέπει να ικανοποιηθεί από μια συγκεκριμένη υπηρεσία η οποία ανταποκρίνεται στη λειτουργικότητα του, και η οποία συγκεκριμένη υπηρεσία χαρακτηρίζεται από κάποιες μη λειτουργικές ιδιότητες. Αυτές οι μη λειτουργικές ιδιότητες θα καθορίσουν αν θα επιλεχθεί κάποια συγκεκριμένη υπηρεσία ή όχι. Τα μη λειτουργικά χαρακτηριστικά που μελετούνται σε αυτή την έρευνα είναι ο χρόνος εκτέλεσης, η διαθεσιμότητα, το κόστος, η ποιότητα των δεδομένων και το bandwidth. Το κάθε ένα από αυτά παίρνει τιμές μέσα από διαφορετικά διαστήματα. Ο χρόνος εκτέλεσης είναι από 1-100, το κόστος από 0-50, η διαθεσιμότητα και η ποιότητα των δεδομένων από 0-1 και το bandwidth από 10-2000. Τα διαφορετικά πεδία ορισμού των χαρακτηριστικών θα δημιουργούσαν πρόβλημα στην αποτίμηση της αντικειμενικής συνάρτησης (fitness function) κατά την διάρκεια χρήσης των γενετικών αλγορίθμων γι αυτό και έγινε η κανονικοποίηση τους. Μετά την κανονικοποίηση όλα τα μη λειτουργικά χαρακτηριστικά βρίσκονται στο διάστημα [0,1]. Χρησιμοποιήθηκε min-max κανονικοποίηση η οποία χαρακτηρίζεται από τον τύπο:

$$v'(i) = (v(i) - minA)/(maxA - minA)* (new\_maxA - new\_minA) + new\_minA$$

όπου το [minA, maxA] είναι η μέγιστη και η ελάχιστη τιμή του χαρακτηριστικού μέσα από όλες τις υπηρεσίες στις οποίες εμφανίζεται. Το [new_minA, new_maxA] αντιπροσωπεύει το νέο διάστημα στο οποίο θα μετατραπούν οι τιμές του χαρακτηριστικού δηλαδή σε αυτή την περίπτωση το [0,1]. Για παράδειγμα για κανονικοποιηθεί ο χρόνος εκτέλεσης που σε μια συγκεκριμένη μέθοδο είναι 8.5 και ο ελάχιστος, μέγιστος χρόνος από όλες τις μεθόδους των διαθέσιμων υπηρεσιών να είναι 4, 30 αντίστοιχα, στην εξίσωση θα γινόταν η πιο κάτω αντικατάσταση:

$$v'(i) = (8.5 - 4)/(30 - 4)* (1 - 0) + 0$$

Για την επίλυση του συνδυαστικού προβλήματος της σύνθεσης των υπηρεσιών πρέπει να αποτιμηθούν τα σφαιρικά μη λειτουργικά χαρακτηριστικά και όχι κάθε συγκεκριμένης υπηρεσίας ξεχωριστά. Γι αυτό για ένα μονοπάτι τα μη λειτουργικά χαρακτηριστικά των διαφόρων υπηρεσιών ιστού (Web Services) που επιλέχθηκαν πρέπει να συνδυαστούν για δώσουν τα σφαιρικά μη λειτουργικά χαρακτηριστικά της

σύνθετης υπηρεσίας. Για την αποτίμηση των σφαιρικών μη λειτουργικών χαρακτηριστικών εφαρμόζεται μια πράξη πάνω στα τοπικά μη λειτουργικά χαρακτηριστικά. Για την διαθεσιμότητα βρίσκεται το γινόμενο των διαθεσιμοτήτων, για τον χρόνο εκτέλεσης και την τιμή το άθροισμα των χρονών και τιμών αντίστοιχα, των συγκεκριμένων υπηρεσιών που λαμβάνουν μέρος στην σύνθεση. Επιπρόσθετα για την ποιότητα των δεδομένων βρίσκεται το ελάχιστο της ποιότητας των δεδομένων των συγκεκριμένων υπηρεσιών και για το bandwidth το μέγιστο.

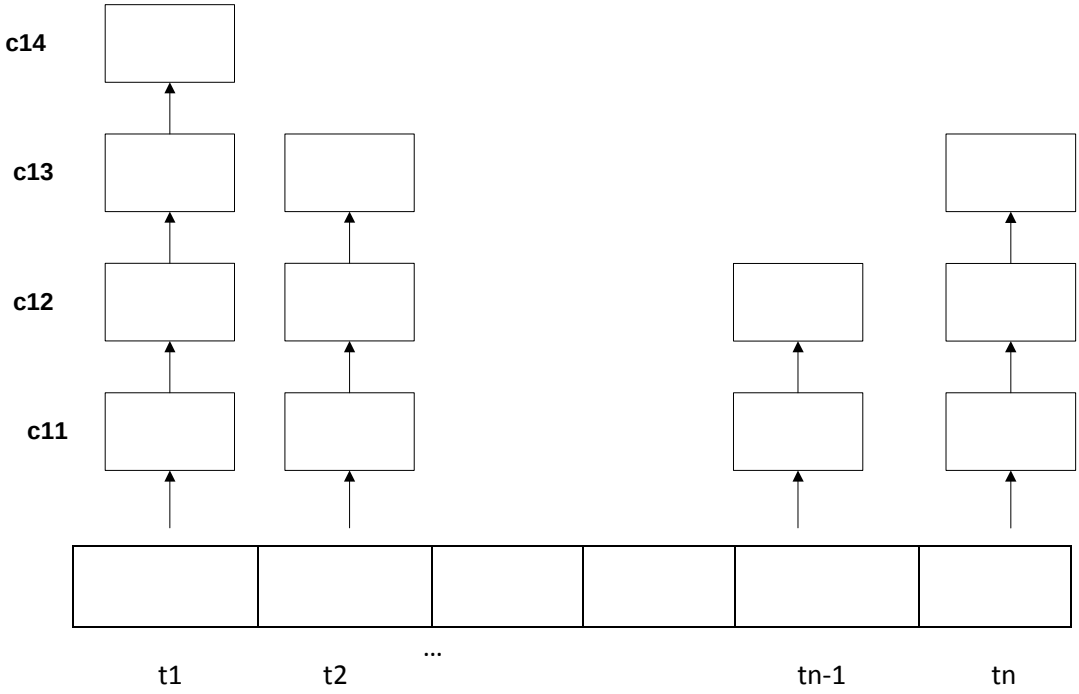
Όλες οι πληροφορίες που χαρακτηρίζουν τις διαθέσιμες υπηρεσίες περνιούνται στην πλατφόρμα μέσω ενός αρχείου του οποίου η μορφή φαίνεται στον Πίνακα 4.2.

Πίνακας 4.2: Μορφή αρχείου που περιέχει την περιγραφή κάθε υπηρεσίας

Type	Id	Method	Execution Time	Price	Bandwidth	Availability	Data Quality
t1	ws02831	Convert	4	12	90	0.7	32
t5	ws09573	Read	9	17	150	0.3	11

4.2 Περιγραφή λύσης με γενετικούς αλγόριθμους

Για να κωδικοποιηθεί το πρόβλημα με τους γενετικούς αλγόριθμους αρχικά βρέθηκε η κωδικοποίηση του χρωμοσώματος. Η κωδικοποίηση έχει ως στόχο να λύσει προηγούμενα προβλήματα τα οποία υπήρξαν σε προηγούμενες προσεγγίσεις. Ακολούθησε η σύνταξη της αντικειμενικής συνάρτησης (fitness function) για αποτίμηση του προβλήματος.



Σχήμα 4.2: Γραφική αναπαράσταση χρωμοσώματος

Το χρωμόσωμα της λύσης που δόθηκε δεν είναι σταθερό. Όπως ήδη ειπώθηκε η σύνθετη υπηρεσία μπορεί να εκτελεστεί μέσω διαφορετικών μονοπατιών. Αυτά τα μονοπάτια αποτελούνται από διαφορετικό αριθμό στόχων οι οποίοι θα πρέπει να εκτελεστούν. Οι γενιές του χρωμοσώματος είναι ανάλογες των στόχων άρα κάθε φορά που εκτελείται διαφορετικό μονοπάτι μπορεί να αλλάζει και το μέγεθος του χρωμοσώματος. Άρα για κάθε μονοπάτι σύνθεσης το χρωμόσωμα αναπαριστάται από ένα πίνακα ακεραίων με μέγεθος ίσο με τον αριθμό των στόχων (t1 … tn) οι οποίοι πρέπει να εκτελεστούν. Σε κάθε θέση/γενεά του χρωμοσώματος υπάρχει ένας δείκτης προς μία λίστα που περιέχει όλες τις συγκεκριμένες υπηρεσίες(cSi1 … cSin), οι οποίες είναι διαθέσιμες και λειτουργικά ικανές να εκτελέσουν το συγκεκριμένο στόχο. Στην Σχήμα 4.2 φαίνεται η γραφική αναπαράσταση του χρωμοσώματος. Κάθε φορά που τρέχει η πλατφόρμα για ένα σχέδιο λειτουργίας εκτελούνται όλα τα πιθανά μονοπάτια σύνθεσης και κάθε φορά δημιουργείται ένα χρωμόσωμα της μορφής που φαίνεται στην γραφική αναπαράσταση.

Η αντικειμενική συνάρτηση η οποία χρησιμοποιείται κατά την χρήση των γενετικών αλγορίθμων είναι η ακόλουθη:

fitness=totAvailability+totDataQuality+(1/totExecTime)+(1/totPrice)+(1/totBandwith)

Τα θετικά μη λειτουργικά χαρακτηριστικά που είναι η διαθεσιμότητα και η ποιότητα των δεδομένων είναι μορφή δεκαδικού αριθμού. Τα αρνητικά μη λειτουργικά χαρακτηριστικά που είναι ο χρόνος εκτέλεσης, η τιμή και το Bandwidth είναι σε μορφή δεκαδικού και μπαίνει στον παρονομαστή με αριθμητή το ένα. Στην αντικειμενική συνάρτηση αποτιμούνται τα σφαιρικά μη λειτουργικά χαρακτηριστικά, δηλαδή όλου του χρωμοσώματος και όχι κάθε γενιάς ξεχωριστά (τοπικά).

4.3 Επεξήγηση βασικών κλάσεων και διασύνδεσης του προγράμματος

Αρχικά η λειτουργία του προγράμματος, όπως και προαναφέρθηκε ξεκινά με το διάβασμα του αρχείου που περιέχει το σχέδιο λειτουργίας του οποίου η μορφή είναι όπως φαίνεται στον Πίνακα 4.1. Η κλάση μέσα από την οποία γίνεται το διάβασμα του αρχείου είναι η SinthetiYpiresia. Η δομή αυτής της κλάσης φαίνεται στο σχήμα 4.3. Η συγκεκριμένη κλάση συνεργάζεται με την κλάση SxedioLeitourgias για την ενημέρωση δεδομένων και την δημιουργία των επιθυμητών δομών. Ουσιαστικά αυτή η κλάση είναι βοηθητική. Η επεξήγηση της φαίνεται στο σχήμα 4.4.

```
package sxedioLeitourgias;

import java.io.*;
import java.util.ArrayList;
import java.util.StringTokenizer;

public class SinthetiYpiresia {
private static void diavasmaSxediouLeitourgias()
{

Διαβάζει το σχέδιο λειτουργίας και το αποθηκεύει σε ένα δισδιάστατο
πίνακα. Μέσα σε αυτή την συνάρτηση καλούνται δύο επιμέρους
συναρτήσεις. Η πρώτη είναι η desmeusiPinakaSxediou() που καλείται κατά
την πρώτη φορά που διαβάζεται μια γραμμή στο αρχείο. Είναι υπεύθυνη
για την δημιουργία του πίνακα οπού αποθηκεύονται τα δεδομένα του
σχεδίου λειτουργίας. Η δεύτερη συνάρτηση η οποία καλείται σε κάθε
επανάληψη είναι η enimerwsiSxediou(arg1, arg2, arg3)η οποία ενημερώνει
τον πίνακα λειτουργίας.
}

public static int returnSize()
{

Επιστρέφει το μέγιστο μέγεθος του δισδιάστατου πίνακα λειτουργίας.
Ουσιαστικά το μεγεθός του είναι ο μέγιστος αριθμός στάχων με τους
οποίους μπορεί να γίνει η σύνθεση της τελικής λειτουργίας.

}

public void arithmosSxediwnGiaTelikiLeitourgia()
{

Ενημερώνει τη λίστα που περιέχει τα μονοπάτια σύνθεσης και τη
μετάβλητή που υποδεικνύει πόσα είναι τα μονοπάτια σύνθεσης.

}

public int returnSxedioNo()
{

Επιστρέφει τον αριθμό των μονοπατιών που βρέθηκαν και μπορεί να γίνει
η εκτέλεση της σύνθετης υπηρεσίας.

}

public String returnSxedio(int arithmosSxediou)
{

Επιστρέφει το κατάλληλο μονοπάτι σύνθεσης ανάλογα με την τιμή που έχει
η παράμετρος που περνιέται μέσα στην μέθοδο.

}

public static String[][] returnTableLeitourgias()
{

Επιστρέφει ολόκληρο τον πίνακα που περιέχει το σχέδιο λειτουργίας

}
```

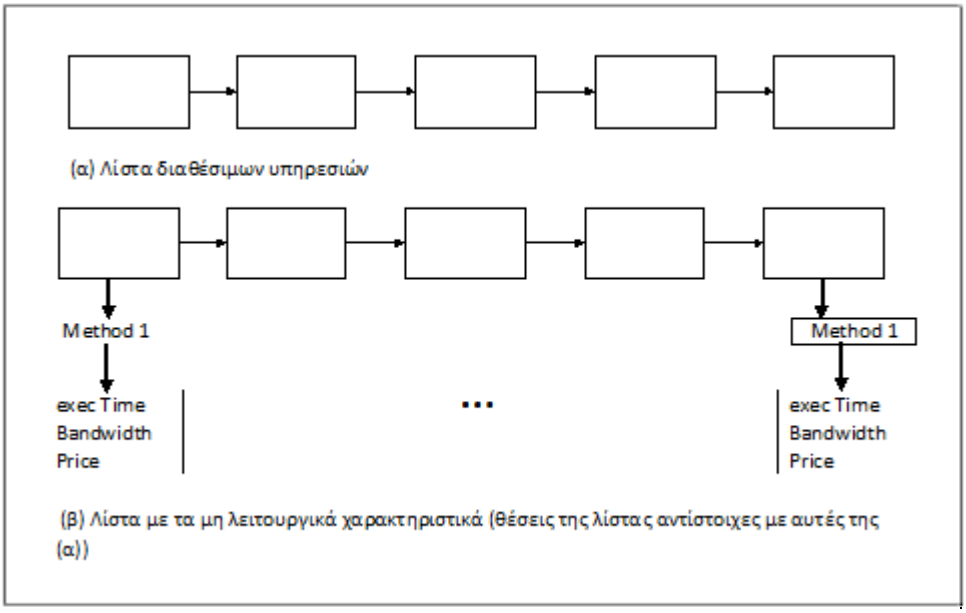
Σχήμα 4.3: Επεξήγηση της κλάσης SinthetiYpiresia

```
package sxediolLeitourgias;


import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.Paint;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JToolBar;

import edu.uci.ics.jung.graph.ArchetypeVertex;
import edu.uci.ics.jung.graph.DirectedEdge;
import edu.uci.ics.jung.graph.Graph;
import edu.uci.ics.jung.graph.decorators.VertexFontFunction;
import edu.uci.ics.jung.graph.decorators.VertexPaintFunction;
import edu.uci.ics.jung.graph.decorators.VertexStringer;
import edu.uci.ics.jung.graph.impl.DirectedSparseGraph;
import edu.uci.ics.jung.graph.Vertex;
import edu.uci.ics.jung.graph.impl.DirectedSparseEdge;
import edu.uci.ics.jung.graph.impl.DirectedSparseVertex;
import edu.uci.ics.jung.visualization.DefaultGraphLabelRenderer;
import edu.uci.ics.jung.visualization.FRLayout;
import edu.uci.ics.jung.visualization.PluggableRenderer;
import edu.uci.ics.jung.visualization.VisualizationViewer;
import grafikiAnaparastasi.FileVsUddi;
import grafikiAnaparastasi.GuiSxedioLeitourgias;

public class SxedioLeitourgias {

public void desmeusiPinakaSxediou()
{
Δεσμεύει χώρο και αρχικοποιεί τον πίνακα λειτουργίας ανάλογα με το
μέγεθος που γίνεται γνωστό αφού γίνει η πρώτη πρόσβαση στο αρχείο που
το περιέχει το σχέδιο λειτουργίας.
}

public void enimerwsiSxediou(int row,int column, String stoixeio)
{
Ενημερώνει την θέση του πίνακα λειτουργίας που αντιστοιχεί στις
συντεταγμένες row, column που περνιούνται σαν παράμετροι. Στη θέση
αυτή μπαίνει το στοιχείο το οποίο είναι τρίτο στην λίστα παραμέτρων.
}

public ArrayList<String> monopatiaLeitourgias()
{
 Βρίσκει όλα τα μονοπάτια σύνθεσης της υπηρεσίας διασχίζοντας όλες τις
πιθανές διαδρομές του γράφου. Για την διαδικασία αυτή έγινε χρήση δύο
λιστών. Στην πρώτη αποθηκεύονται τα ολοκληρωμένα μονοπάτια και στην
δεύτερη τα ημιτελή. Τα δεδομένα της δεύτερης λίστας επεξεργάζονται
μέχρι όλα τα μονοπάτια να συμπληρωθούν και να περάσουν στην άλλη
λίστα, η οποία και επιστρέφεται στο τέλος της συνάρτησης.
}

public static void grafikiApekonisiGrafou()
{
Σε αυτή την συνάρτηση γίνεται η γραφική αναπαράσταση του γράφου του
σχεδίου λειτουργίας. Η συνάρτηση addButtons(final JFrame jf,
VisualizationViewer vv)που επίσης βρίσκεται σε αυτή την κλάση καλείται
για να μπουν τα επιμέρους κουμπιά.
}
}
```

Σχήμα 4.4: Επεξήγηση της κλάσης SxedioLeitourgias

Έκτος από το σχέδιο λειτουργίας τα άλλα δεδομένα εισόδου τα οποία παίρνει το πρόγραμμα είναι οι διαθέσιμες υπηρεσίες από τις οποίες κάποιες θα χρησιμοποιηθούν για την ικανοποίηση των διαφόρων στόχων της σύνθετης υπηρεσίας. Όλες οι διαθέσιμες συγκεκριμένες υπηρεσίες, όπως είναι ήδη γνωστό, διαβάζονται από ένα αρχείο της μορφής του πίνακα 4.2. Μετά το διάβασμα του αρχείου οι πληροφορίες για τις διαθέσιμες υπηρεσίες(μη λειτουργικά χαρακτηριστικά, μέθοδος που θα χρησιμοποιηθεί, τύπος, ταυτότητα) αποθηκεύονται σε μία δομή, της οποίας η μορφή είναι πολύ σημαντική για την λειτουργία του όλου προγράμματος. Η δομή αυτή αποτελείται από δύο λίστες. Στην πρώτη λίστα βρίσκονται οι υπηρεσίες οι οποίες είναι διαθέσιμες για να χρησιμοποιηθούν στην σύνθεση για την παραγωγή της τελικής λειτουργίας. Στην δεύτερη λίστα βρίσκονται τα μη λειτουργικά χαρακτηριστικά της κάθε υπηρεσίας. Οι θέσεις των δύο αυτών λιστών είναι ανάλογες. Δηλαδή αν μια υπηρεσία στην $1^{η}$ λίστα βρίσκεται στην θέση i, τότε τα μη λειτουργικά της χαρακτηριστικά θα βρίσκονται στην $2^{η}$ λίστα επίσης στη θέση i. Οι δύο αυτές λίστες δημιουργούνται και διαχειρίζονται από τις κλάσεις WebServicesInUse ,CreateFullStructure και NonFunctionalCharacteristics. Η πρώτη κλάση κρατά τις δύο λίστες και οι άλλες δύο χρησιμοποιούνται γι το χτίσιμο των λιστών.

Η δομή της $2^{ης}$ λίστας δεν είναι τόσο απλή όσο αυτή της $1^{ης}$. Κάθε θέση της είναι βασικά δείχτης σε μια κλάση η οποία περιέχει ένα μονοδιάστατο πίνακα μιας θέσης που αντιπροσωπεύει την μέθοδο. Ο πίνακας της μεθόδου δημιουργείται μέσα στην κλάση CreateFullStructure. Αυτή η θέση του πίνακα της μεθόδου περιέχει ένα αντικείμενο της κλάσης της NonFunctionalCharacteristics. Βασική δομή μέσα σε αυτή την κλάση είναι ο πίνακας στον οποίο είναι αποθηκευμένα τα μη λειτουργικά χαρακτηριστικά της μεθόδου καθώς και ο πίνακας που είναι αποθηκευμένες οι κανονικοποιημένες τιμές αυτών των χαρακτηριστικών. Στο σχήμα 4.5 που ακολουθεί φαίνεται η μορφή των δύο λιστών.



Σχήμα 4.5: Σχεδιάγραμμα λιστών για τις υπηρεσίες

Όταν όλα τα δεδομένα είναι έτοιμα ξεκινά η χρήση των γενετικών αλγορίθμων για την εύρεση του καλύτερου συνδυασμού των υπηρεσιών για την εκτέλεση της τελικής λειτουργίας. Χρησιμοποιήθηκε η βιβλιοθήκη JGap για την κωδικοποίηση του προβλήματος με γενετικούς αλγόριθμους. Παρέχει βασικούς γενετικούς μηχανισμούς που μπορούν εύκολα να χρησιμοποιηθούν για να εφαρμοστούν οι εξελικτικές αρχές στις λύσεις διαφόρων προβλημάτων. Υλοποιήθηκαν δύο κλάσεις που κληρονομούν ή απλά χρησιμοποιούν μεθόδους από αυτή την βιβλιοθήκη αυτή, και συνεργάζονται μεταξύ τους για τη λύση του προβλήματος της σύνθεσης. Η μια είναι υπεύθυνη για την αποτίμηση της αντικειμενικής συνάρτησης και η άλλη ρυθμίζει όλες τις άλλες λειτουργίες. Τα ονόματα τους είναι WebServiceFitnessFunction και WebServiceSel αντίστοιχα. Η κλάση WebServiceFitnessFunction κληρονομεί από την κλάση FitnessFunction που περιέχεται στο JGap. Μια περιγραφή της φαίνεται στο Σχήμα 4.6.

```
package selectionWithJGap;
import java.util.ArrayList;
import diathesimesYpiresies.*;
import org.jgap.*;

public class WebServiceFitnessFunction extends FitnessFunction {

protected double evaluate(IChromosome nChromosome) {
Γίνεται η αποτίμηση της αντικειμενικής συνάρτησης σύμφωνα με τα
σφαιρικά μη λειτουργικά χαρακτηριστικά του χρωμοσώματος που περνιέται
σαν παράμετρος. Από εδώ καλείται και η μέθοδος
calculateTotalNonFunctionCharacteristics για να βοηθήσει στον
υπολογισμό αυτό. Όταν τελειώσει η αποτίμηση το αποτέλεσμα
επιστρέφεται.

}
private void calculateTotalNonFunctionCharacteristics(IChromosome
a_subject)
{
Υπολογίζει και ενημερώνει τα μη λειτουργικά χαρακτηριστικά. Αυτό
γίνεται σύμφωνα με τις συγκεκριμένες υπηρεσίες που επιλέχθηκαν για το
χρωμόσωμα το οποίο περνιέται σαν παράμετρος. Ο υπολογισμός του κάθε
σφαιρικού μη λειτουργικού χαρακτηριστικού γίνεται εφαρμόζοντας κάποιες
πράξεις. Για παράδειγμα για την σφαιρική διαθεσιμότητα βρίσκεται το
γινόμενο όλων των διαθεσιμοτήτων των συγκεκριμένων υπηρεσιών που
επιλέχθηκαν για το χρωμόσωμα.
}
```

Σχήμα 4.6: Επεξήγηση της κλάσης WebServiceFitnessFunction

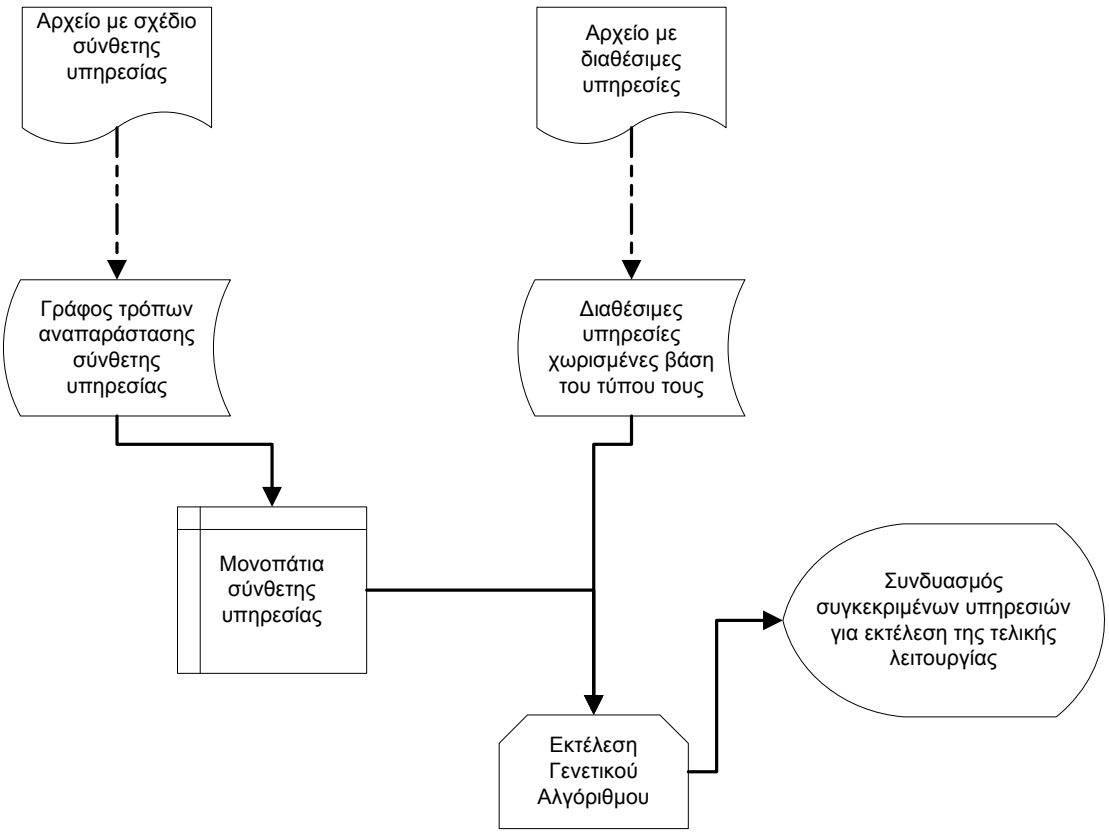
Η κλάση WebServiceSel η κύρια κλάση του προγράμματος. Περιέχει την συνάρτηση main(String[] args) από όπου ξεκινά η όλη εκτέλεση του προγράμματος. Επίσης μέσω αυτής της κλάσης γίνονται όλες ρυθμίσεις του γενετικού αλγόριθμου και συντονίζονται και όλες οι υπόλοιπες κλάσεις μεταξύ τους. Μια πιο εκτεταμένη αναφορά στην WebServiceSel κλάση δίνεται στο Σχήμα 4.7 που παρουσιάζεται παρακάτω.

Στο Παράρτημα Α παρουσιάζεται ο ψευδοκώδικας όλων των βασικών κλάσεων του συστήματος οι οποίες περιγράφηκαν παραπάνω.

```
package selectionWithJGap;
import org.jgap.*;
import org.jgap.event.EventManager;
import org.jgap.impl.*;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.util.*;
import diathesimesYpiresies.*;
import sxedioLeitourgias.*;
import grafikiAnaparastasi.*;

public class WebServiceSel {

public static void setConfig(String sxedio)
{

Η συνάρτηση είναι υπεύθυνη για την ρύθμιση των γενετικών αλγορίθμων.
Αρχικά διαβάζει από ένα αρχείο τον πληθυσμό και το mutation rate (το
crossover rate είναι το ήδη ρυθμισμένο από την βιβλιοθήκη). Ακολούθως
ορίζεται το χρωμόσωμα (πόσες και ποιές γενιές έχει)και δημιουργούνται
οι λίστες με τις συγκεκριμένες υπηρεσίες για κάθε γενιά του.
Δημιουργείται το αντικείμενο της κλάσης FitnessFunction και μπαίνουν
οι τελευταίες ρυθμίσεις.

}

private static WebServicesInUse dimiourgiaListwnGiaKa8eGene(String
typos)
{

Εδώ γίνεται ο διαχωρισμός των διαθέσιμων συγκεκριμένων υπηρεσιών
ανάλογα με τον τύπο τους, ο οποίος περνιέται σαν παράμετρος μέσα στην
μέθοδο. Επιστρέφει ένα αντικείμενο που περιέχει μια λίστα με αυτές τις
συγκεκριμένες υπηρεσίες.

}

public static void arxiGenetikou()
{

Διαβάζεται το όνομα του αρχείου που περιέχει το σχέδιο λειτουργίας
μέσω μιας φόρμας. Ακολουθεί το διαβασμα του αρχείου σε συνεργασία με
την κλάση SinthetiYpiresia και η ενημέρωση της συγκεκριμένης κλάσης
για τον αριθμό των μονοπατιών σύνθεσης

}

public static void enimerwsiSinthetisYpiresias(SinthetiYpiresia x)
{

Ενημερώνεται το σχέδιο λειτουργίας για την σύνθετη υπηρεσία.

}

public static void ektelesiGenetikou()
{

Αυτή η μέθοδος είναι υπεύθυνη για την εύρεση της βέλτιστης λύσης για
το κάθε ένα από τα μονοπάτια σύνθεσης. Για το κάθε μονοπάτι γίνεται η
εξέλιξη των χρωματοσωμάτων μέχρι την σύγκλιση του αλγορίθμου, δηλαδή
την εύρεση της λύσης. Τέλος μέσω αυτής της μεθόδου επιλέγεται το
βέλτιστο μονοπάτι για την εκτέλεση της τελικής λειτουργίας.

}

public static void main(String[] args)
{

Ξεκινά η εκτέλεση του προγράμματος με το κάλεσμα της πρώτης φόρμας
φόρτωσης.

}

}
```

Σχήμα 4.7: Επεξήγηση της κλάσης WebServiceSel

Συνοψίζοντας η πλατφόρμα παίρνει δεδομένα από δύο αρχεία. Τα επεξεργάζεται και δημιουργεί το γράφο και τα μονοπάτια σύνθεσης. Ακολούθως χωρίζονται οι υπηρεσίες ανάλογα με τον τύπο τους και τοποθετούνται στο χρωμόσωμα. Εκτελείται ο γενετικός αλγόριθμος για κάθε μονοπάτι και βρίσκεται ο καλύτερος συνδυασμός υπηρεσιών. Τέλος επιλέγεται το καλύτερο μονοπάτι για την σύνθεση. Στο σχήμα 4.8 είναι η γενική αναπαράσταση του προβλήματος.



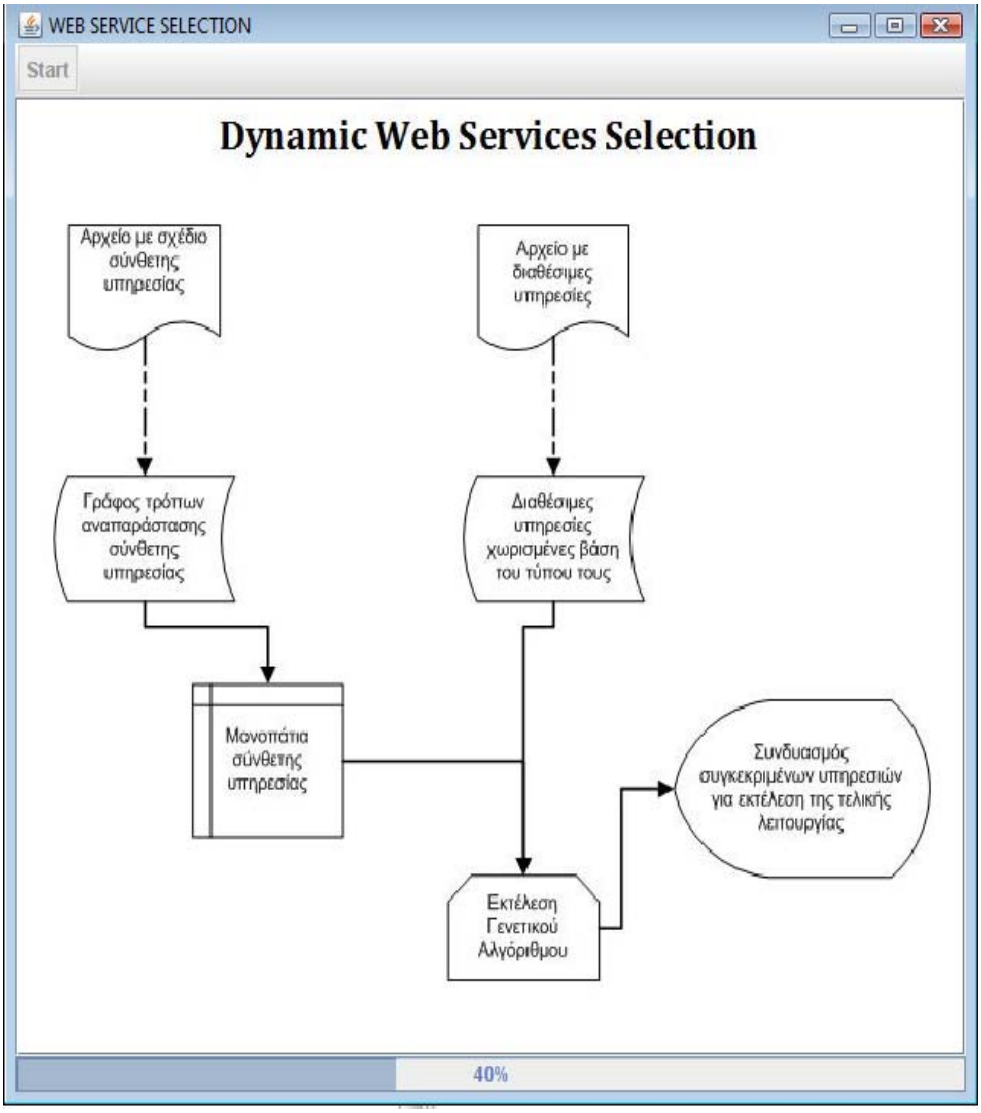
Σχήμα 4.8: Γραφική αναπαράσταση του προγράμματος

Κεφάλαιο 5

5. Ανάλυση του συστήματος

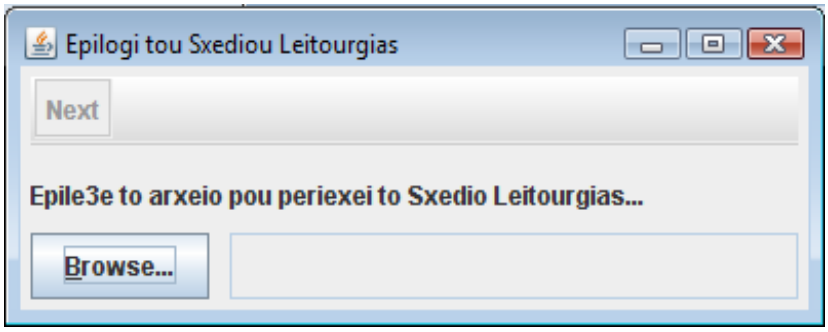
5.1 Γραφικό περιβάλλον

Για την καλύτερη παρουσίαση του προγράμματος αλλά και αλληλεπίδραση του χρήστη με αυτό υλοποιήθηκαν κάποιες φόρμες. Οι φόρμες δείχνουν σε κάθε στάδιο του προγράμματος τα δεδομένα του προγράμματος, την μορφή τους καθώς και τα αποτελέσματα μετά την επεξεργασία τους. Ακολουθούν με την σειρά οι φόρμες του προγράμματος και η επεξήγηση τους.



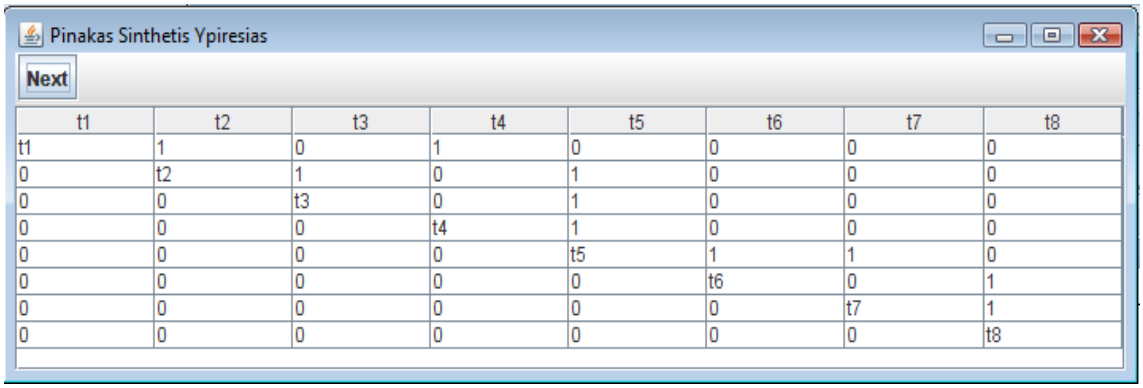
Σχήμα 5.1: Αρχική φόρμα του προγράμματος

Το πρόγραμμα μόλις τρέξει παρουσιάζει την πρώτη φόρμα. Η μορφή της είναι όπως παρουσιάζεται στο Σχήμα 5.1. Παρουσιάζεται το σχεδιάγραμμα του προγράμματος. Όταν πατηθεί το κουμπί του Start ξεκινά η εκτέλεση του προγράμματος αφού γίνει η φόρτωση του. Μόλις η γραμμή προόδου (progress bar) φτάσει στο 100% καλείται η επόμενη φόρμα η οποία και παρουσιάζεται στο Σχήμα 5.2.



Σχήμα 5.2: Φόρμα φόρτωσης του πίνακα της σύνθετης υπηρεσίας

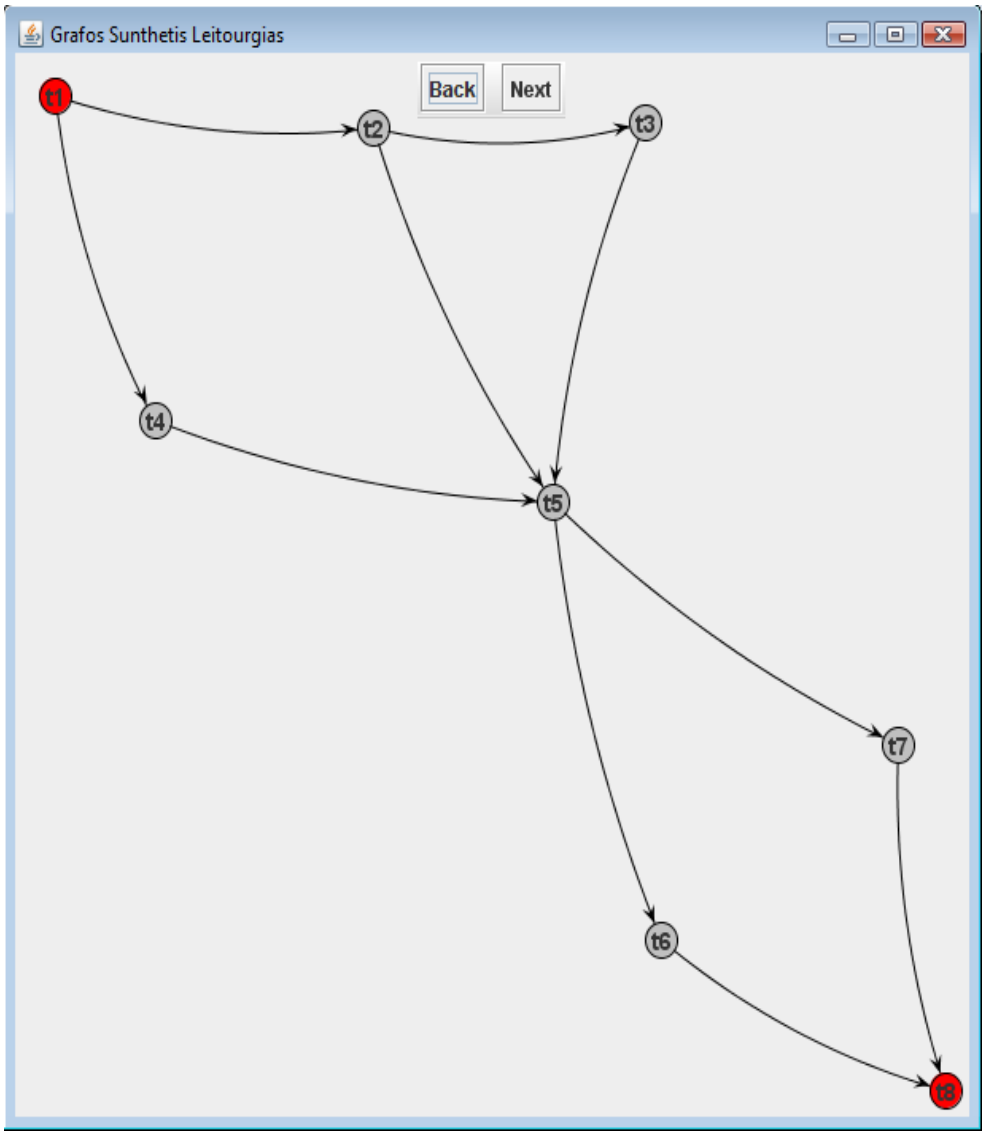
Στην δεύτερη αυτή φόρμα όταν πατηθεί το κουμπί του Browse μπορεί να οριστεί το μονοπάτι και το όνομα του αρχείου μέσα στο οποίο βρίσκεται ο πίνακας ο οποίος πρέπει να φορτωθεί. Πριν επιλεχθεί το αρχείο το κουμπί του Next είναι απενεργοποιημένο. Ενεργοποιείται αφού γίνει η επιλογή του αρχείου. Όταν πατηθεί το συγκεκριμένο κουμπί το πρόγραμμα προχωρά παρακάτω.



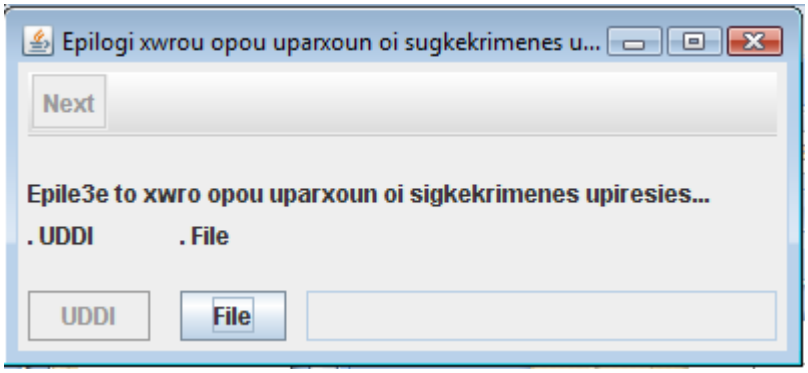
t1	t2	t3	t4	t5	t6	t7	t8
t1	1	0	1	0	0	0	0
0	t2	1	0	1	0	0	0
0	0	t3	0	1	0	0	0
0	0	0	t4	1	0	0	0
0	0	0	0	t5	1	1	0
0	0	0	0	0	t6	0	1
0	0	0	0	0	0	t7	1
0	0	0	0	0	0	0	t8

Σχήμα 5.3: Μορφή του αρχείου που περιέχει το σχέδιο της σύνθετης υπηρεσίας

Στο Σχήμα 5.3 είναι η τρίτη φόρμα του προγράμματος. Παρουσιάζει το περιεχόμενο του αρχείου που διαβάστηκε στο προηγούμενο στάδιο. Στην διαγώνιο φαίνονται όλες οι αφηρημένες υπηρεσίες που μπορεί να λάβουν μέρος κατά διαστήματα, ως στόχοι στην σύνθεση της τελικής λειτουργίας. Στο άνω τριγωνικό κομμάτι είναι οι σχέσεις μεταξύ των αφηρημένων υπηρεσιών. Στη συνέχεια όταν πατηθεί το κουμπί του Next παρουσιάζεται ο γράφος ο οποίος πηγάζει από τον πίνακα όπως φαίνεται στο Σχήμα 5.3. Ο παραγόμενος γράφος για το συγκεκριμένο παράδειγμα παρουσιάζεται στην επόμενη φόρμα του προγράμματος. Γραφική της αναπαράσταση φαίνεται στο Σχήμα 5.4.

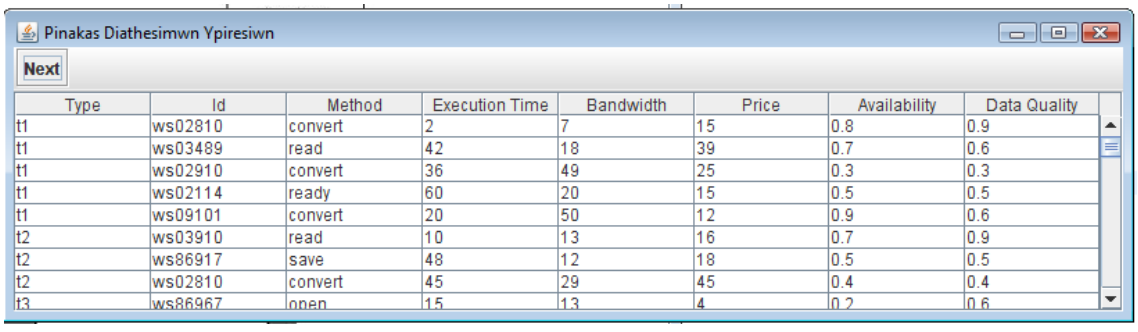


Σχήμα 5.4: Γράφος της σύνθετης υπηρεσίας

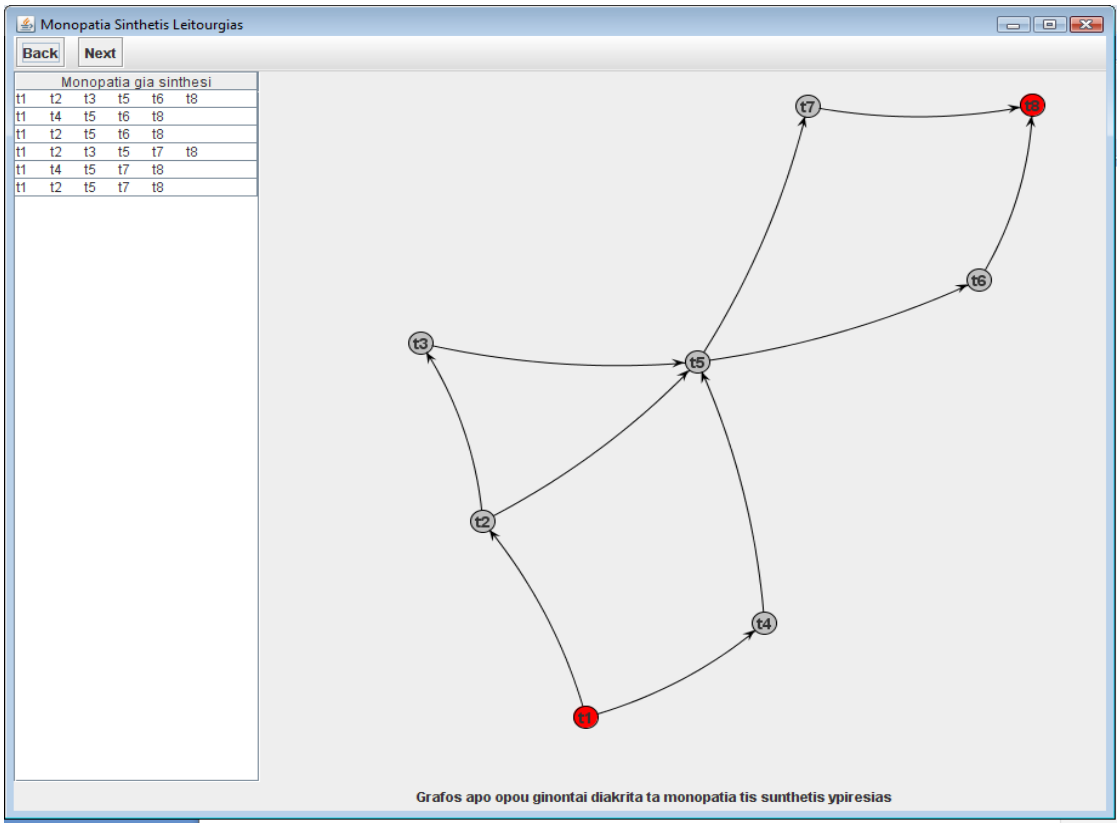


Σχήμα 5.5: Φόρμα επιλογής του αρχείου με τις διαθέσιμες συγκεκριμένες υπηρεσίες

Στη συγκεκριμένη φόρμα του σχήματος 5.4 πατώντας το κουμπί του Back παρουσιάζεται η προηγούμενη φόρμα (Σχήμα 5.3). Πατώντας το κουμπί του Next όπως και στις προηγούμενες φόρμες το πρόγραμμα προχωρά στην επόμενη φόρμα. Μέσω αυτής μπορεί να γίνει η επιλογή του αρχείου που περιέχει όλες τις διαθέσιμες υπηρεσίες όλων των τύπων, οι οποίες μπορούν να χρησιμοποιηθούν στην σύνθεση. Η επιλογή του αρχείου γίνεται πατώντας το κουμπί του File. Αφού επιλεγεί το αρχείο το πρόγραμμα μπορεί να προχωρήσει πατώντας το κουμπί του Next. Υπάρχει ακόμα ένα επιπλέον κουμπί το UDDI που είναι απενεργοποιημένο. Σε αυτή την έκδοση του προγράμματος δεν υποστηρίζεται η χρήση του UDDI. Στο μέλλον μπορεί να γίνει επέκταση του προγράμματος και ο χρήστης να μπορεί να επιλέξει αν οι διαθέσιμες συγκεκριμένες υπηρεσίες θα διαβάζονται μέσω του αρχείου ή μέσω του UDDI. Η μορφοποίηση της φόρμας αυτής φαίνεται στο Σχήμα 5.5.

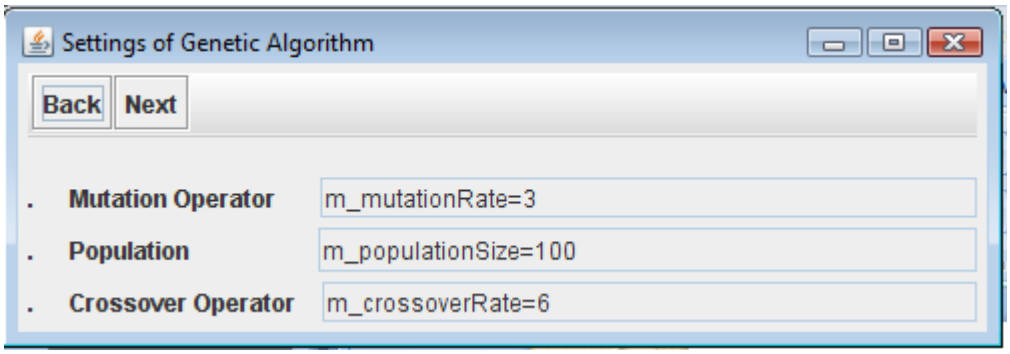


Σχήμα 5.6: Φόρμα με την μορφοποίηση του αρχείου που περιέχει τις συγκεκριμένες υπηρεσίες



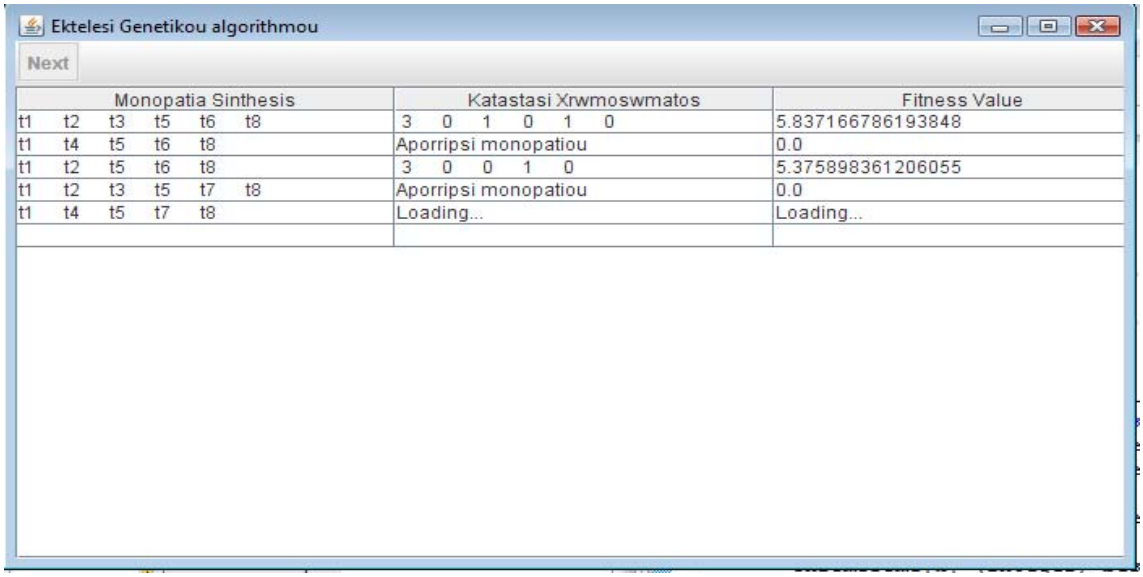
Σχήμα 5.7: Φόρμα με τον γράφο και τα μονοπάτια σύνθεσης της υπηρεσίας

Στο Σχήμα 5.6 φαίνεται το πώς είναι δομημένο το αρχείο που περιέχει τις διαθέσιμες συγκεκριμένες υπηρεσίες. Οι πρώτες τρείς στήλες περιέχουν γενικές πληροφορίες για την κάθε υπηρεσία και στις υπόλοιπες παρουσιάζονται τα μη λειτουργικά τους χαρακτηριστικά. Προχωρώντας στην επόμενη φόρμα, παρουσιάζεται ξανά ο γράφος της σύνθετης υπηρεσίας με τα μονοπάτια τα οποία απορρέουν από αυτόν. Το Σχήμα 5.7 παρουσιάζει η φόρμα αυτή.



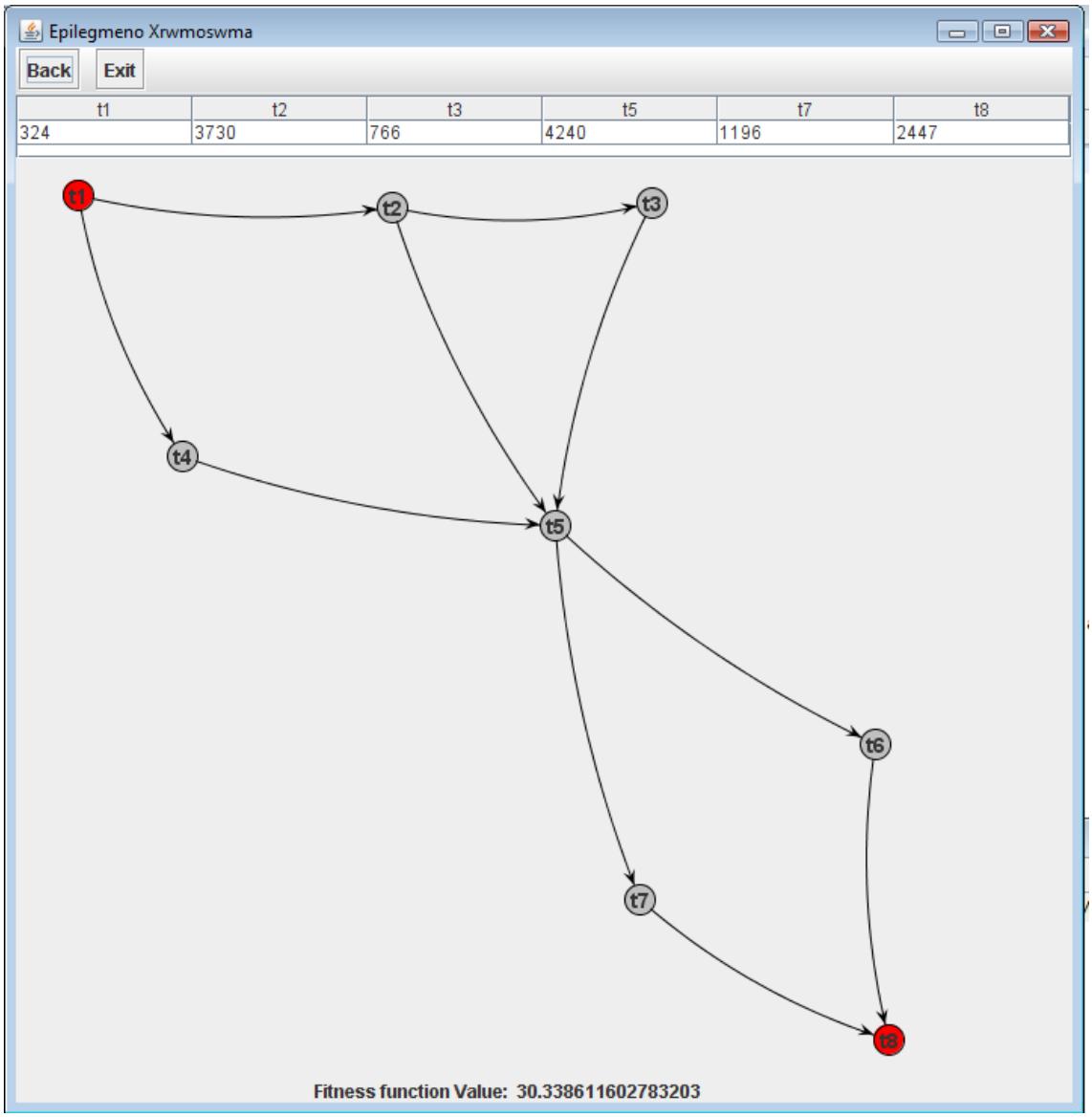
Σχήμα 5.8: Ρυθμίσεις του γενετικού αλγόριθμου

Ο γενετικός αλγόριθμος ο οποίος θα τρέξει για την παραγωγή των αποτελεσμάτων περιέχει κάποιες ρυθμίσεις οι οποίες είναι σημαντικές για τα αποτελέσματα τα οποία θα παραχθούν. Οι ρυθμίσεις που χρησιμοποιούνται όταν τρέχει ο γενετικός αλγόριθμος σε αυτό το πρόγραμμα παρουσιάζοντα στην φόρμα που αναπαριστάται από το Σχήμα 5.8. Τα κουμπιά Back και Next έχουν την ίδια λειτουργία όπως και στις προηγούμενες φόρμες. Όταν παρουσιαστούν και αυτές οι ρυθμίσεις ακολουθεί η εκτέλεση του γενετικού αλγορίθμου και η παρουσίαση κάθε φοράς του συνδυασμού που επιλέγεται. Όταν δεν υπάρχουν διαθέσιμες υπηρεσίες για ένα στόχο τότε απορρίπτεται ολόκληρο το μονοπάτι σύνθεσης. Στο Σχήμα 5.9 φαίνεται η φόρμα της εκτέλεσης του γενετικού αλγορίθμου.



Σχήμα 5.9: Φόρμα παρουσίασης της εκτέλεσης του γενετικού αλγορίθμου

Στην πρώτη στήλη του πίνακα στο Σχήμα 5.9 φαίνονται τα μονοπάτια σύνθεσης μέσω των οποίων μπορεί να παραχθεί η τελική λειτουργία. Στην δεύτερη στήλη φαίνεται ο αριθμός της συγκεκριμένης υπηρεσίας που επιλέχθηκε (η θέση της μέσα στη λίστα). Κατά την διάρκεια που εκτελείται ο γενετικός το κουμπί Next είναι απενεργοποιημένο. Ενεργοποιείται μόνο αφού τελειώσει η εκτέλεση. Το πρόγραμμα τελειώνει με την παρουσίαση του επιλεγμένου μονοπατιού σύνθεσης και της τιμής του (fitness value). Στο επόμενο σχήμα το 5.10 είναι η παρουσίαση της τελευταίας φόρμας.

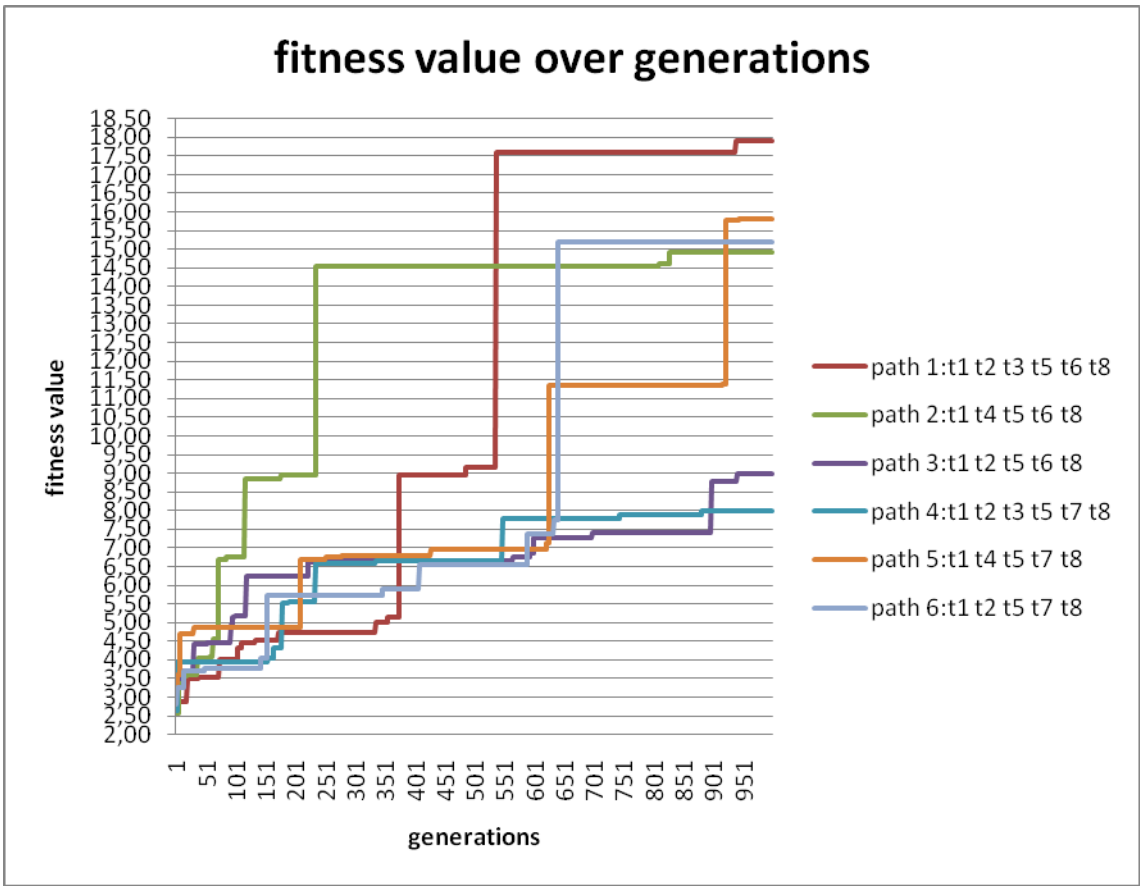


Σχήμα 5.10: Λύση του προβλήματος

5.2 Αποτελέσματα

Για τον έλεγχο των αποτελεσμάτων της έρευνας έγινε μια αριθμητική προσομοίωση. Για την αριθμητική προσομοίωση ερευνούνται πέντε μη λειτουργικά χαρακτηριστικά. Τα μη λειτουργικά χαρακτηριστικά που χρησιμοποιήθηκαν είναι ο χρόνος εκτέλεσης, το bandwidth, το κόστος, η διαθεσιμότητα και η ποιότητα των δεδομένων (data quality). Αυτές οι παράμετροι παίρνουν τιμές από διαφορετικά διαστήματα και έχουν διαφορετικές μονάδες μέτρησης. Ο χρόνος εκτέλεσης παίρνει τιμές μέσα από το διάστημα [1, 100] και μετριέται σε ms. Το bandwidth παίρνει τιμές μέσα από το διάστημα [10, 2000] και μετριέται σε Kbits/sec. Το κόστος βρίσκεται στο διάστημα [0,

50] και η μονάδα μέτρησης του είναι τα δολάρια. Τέλος η ποιότητα των δεδομένων και η διαθεσιμότητα είναι πιθανότητες, δεν έχουν μονάδα μέτρησης και κυμαίνονται στο διάστημα [0, 1]. Όπως ήδη προαναφέρθηκε πριν γίνει η εκτέλεση του γενετικού όλα τα μη λειτουργικά χαρακτηριστικά κανονικοποιούνται και παίρνουν τιμές μέσα από το διάστημα [0, 1]. Στις παρακάτω γραφικές παραστάσεις που παρουσιάζονται ο πληθυσμός του γενετικού αλγόριθμου είναι 100, τρέχει για 1000 γενεές και τα δεδομένα εισόδου είναι 100000. Κάποια από τα χρωμοσώματα διαφέρουν ως προς το μέγεθος λόγω του μονοπατιού σύνθεσης το οποίο αναπαριστούν.

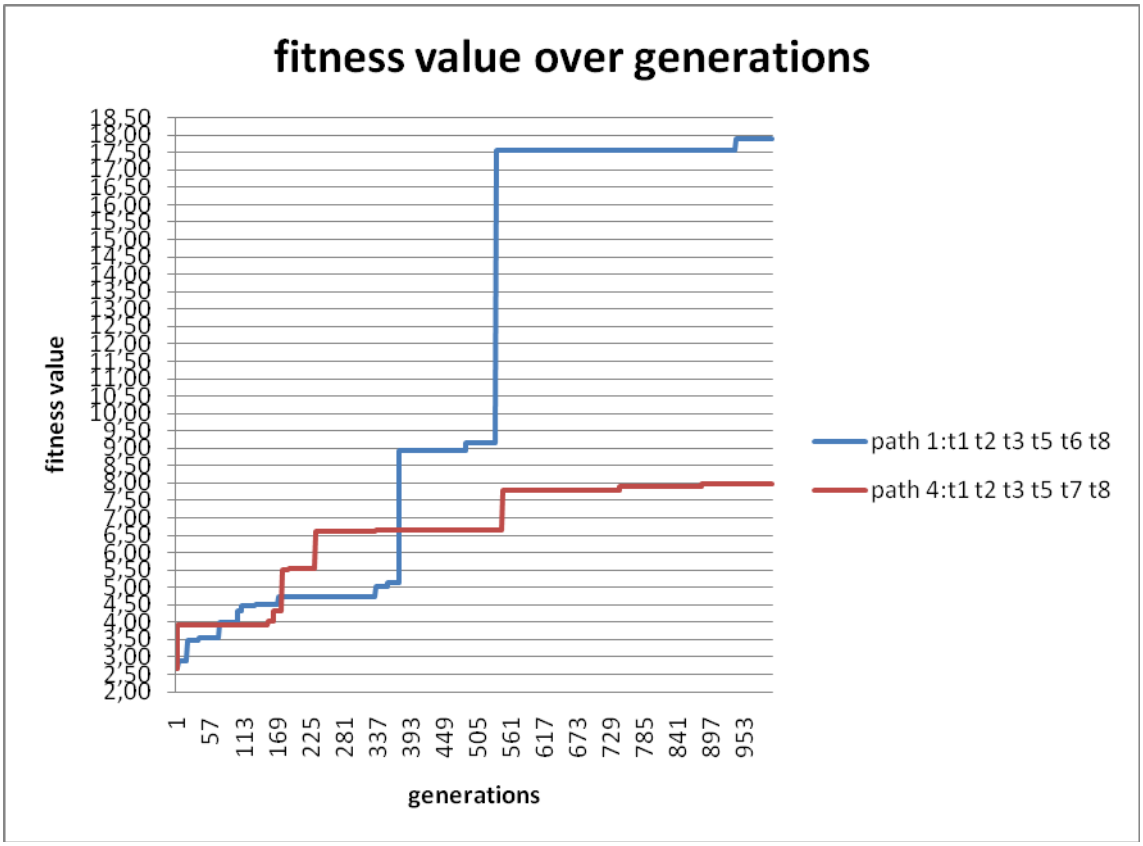


Σχήμα 5.11: Γραφική παράσταση με όλα τα μονοπάτια σύνθεσης για το σενάριο 1

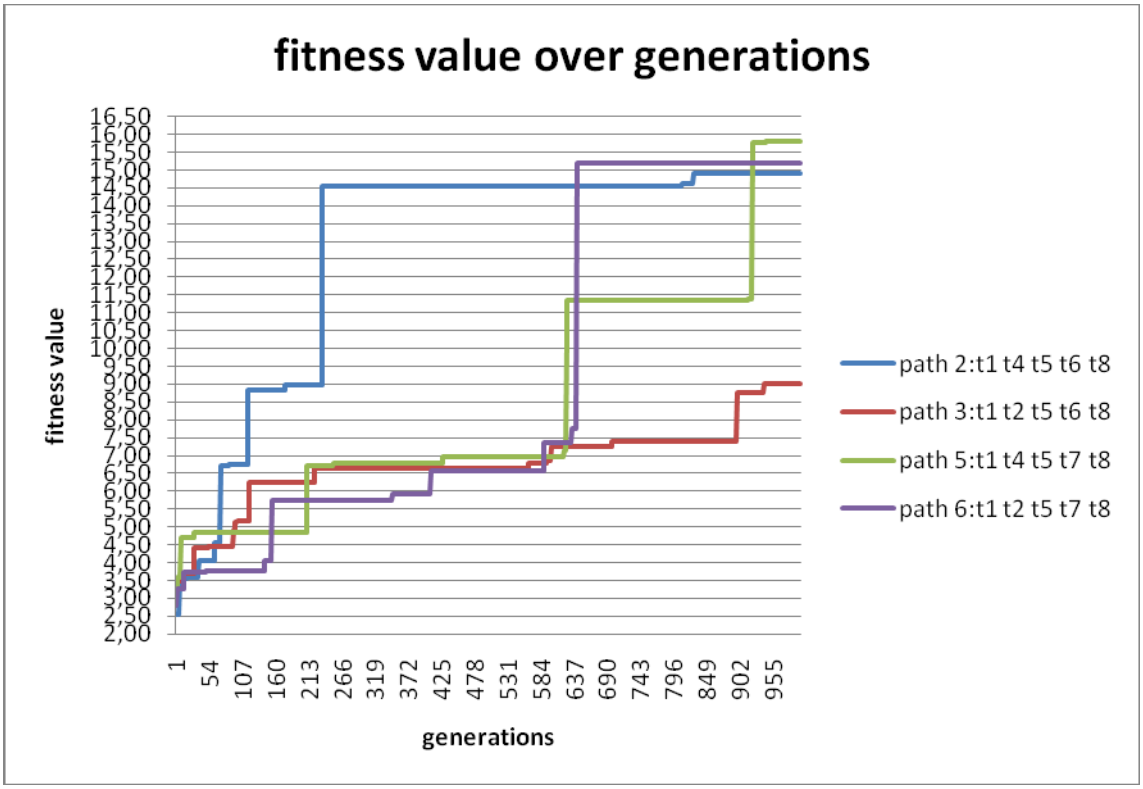
Στο Σχήμα 5.11 παρουσιάζονται όλα τα μονοπάτια σύνθεσης με τις ρυθμίσεις του γενετικού όπως περιγράφηκαν προηγουμένως. Όπως φαίνεται και παραπάνω την καλύτερη τιμή την έχει το πρώτο μονοπάτι και φτάνει το 17,9. Το χρωμόσωμα του αποτελείται από 6 γονίδια όπως και το μονοπάτι 4. Τα υπόλοιπα μονοπάτια έχουν 5 γονίδια. Το μονοπάτι με την μικρότερη τιμή είναι το τέταρτο και φτάνει στα 7,99.Το

συγκεκριμένο μονοπάτι έχει και αυτό μέγεθος 6 γονιδίων. Άρα μπορεί να εξαχθεί το συμπέρασμα ότι το μέγεθος του χρωμοσώματος δεν συμβάλλει στο να υπάρχει καλύτερη τιμή (fitness value) αφού τα χρωμοσώματα με μέγεθος 5 γονιδίων είχαν πιο ψηλή τιμή.

Για να συγκριθούν τα μονοπάτια με το ίδιο μέγεθος παρακάτω παρουσιάζονται δύο γραφικές παραστάσεις ξεχωριστά. Στην πρώτη (Σχήμα 5.12) είναι τα μονοπάτια που τα χρωμοσώματα τους έχουν μέγεθος έξι γονίδια ενώ στην δεύτερη (Σχήμα 5.13) είναι τα μονοπάτια που τα χρωμοσώματα τους έχουν μέγεθος πέντε γονιδίων.



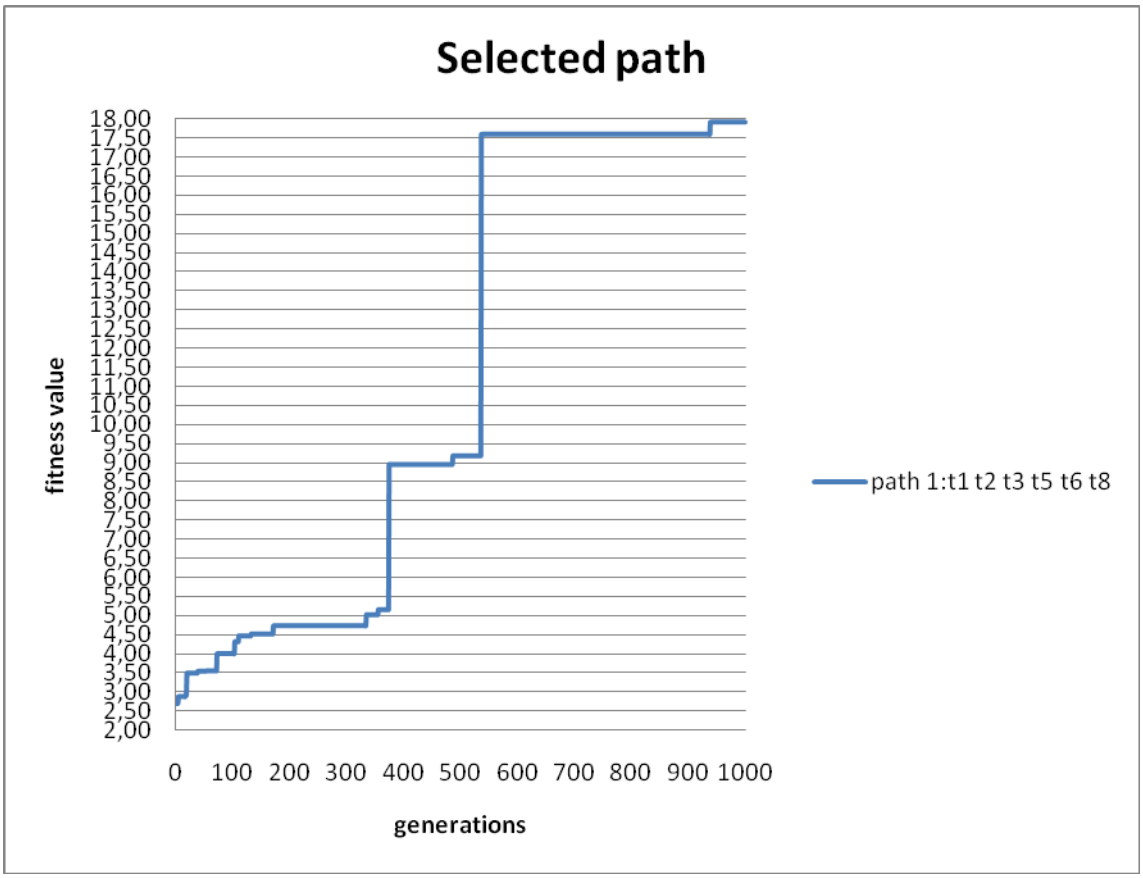
Σχήμα 5.12: Γραφική παράσταση με μονοπάτια μεγέθους 6 γονιδίων (σενάριο 1)



Σχήμα 5.13: Γραφική παράσταση με μονοπάτια μεγέθους 5 γονιδίων (σενάριο 1)

Στην γραφική παράσταση του σχήματος 5.12 παρατηρείται ότι οι τιμές των δύο μονοπατιών έχουν αρκετή διαφορά μεταξύ τους. Έχουν διαφορά της τάξης των δέκα μονάδων. Η τιμή του μονοπατιού ένα έχει μια απότομη αύξηση όταν περάσουν οι μισές περίπου γενιές στο διάστημα 500 - 600 και μετά αρχίζει να συγκλίνει. Στο μονοπάτι τέσσερα γίνεται μια μικρή αύξηση στο συγκεκριμένο διάστημα και υπάρχουν πιο πολλές διακυμάνσεις πιο μετά.

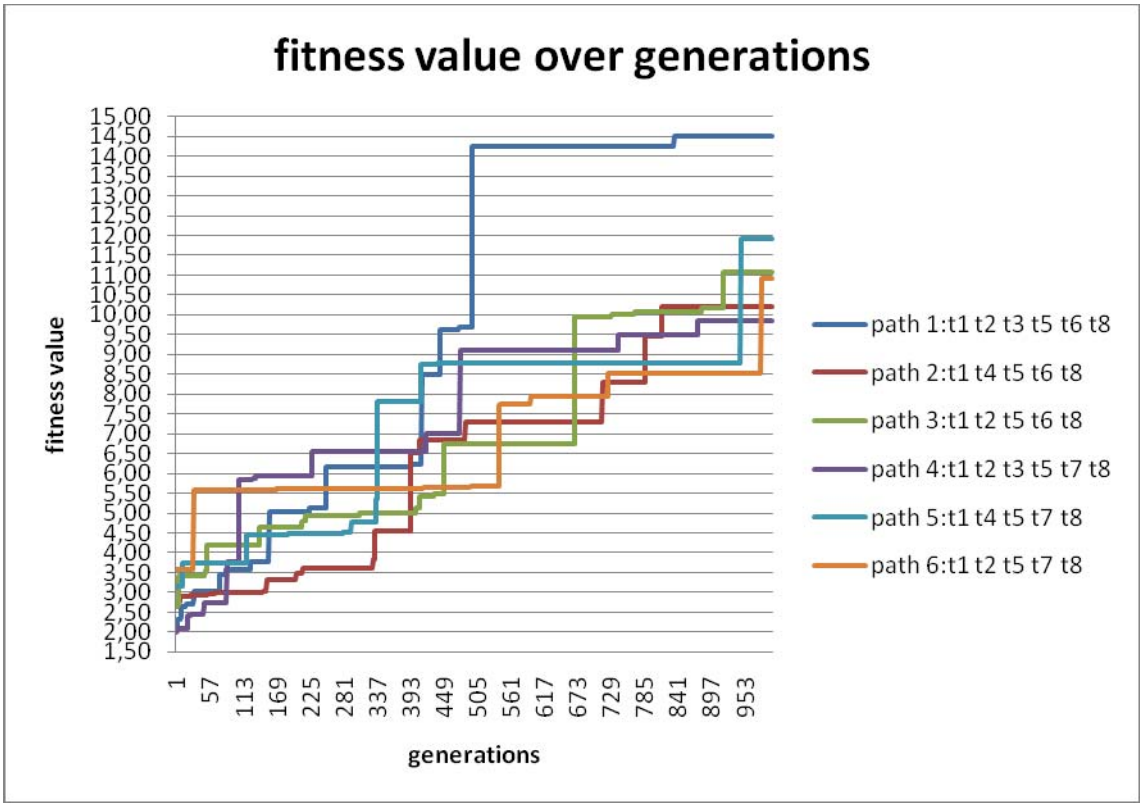
Στην γραφική παράσταση του σχήματος 5.13 παρατηρείται ότι οι τελικές τιμές των τριών μονοπατιών (δεύτερο, πέμπτο, έκτο) κυμαίνονται στα ίδια επίπεδα. Μεγάλη διαφορά έχει το τρίτο μονοπάτι. Η διαφορά της είναι της τάξης των πέντε μονάδων. Είναι η μισή διαφορά από την διαφορά που είχαν τα δύο μονοπάτια στη προηγούμενη γραφική (Σχήμα 5.12). Εδώ υπάρχει απότομη αύξηση στο διάστημα 200-290 για το δεύτερο μονοπάτι και στο διάστημα 550-650 για το πέμπτο και έκτο μονοπάτι. Στο τρίτο μονοπάτι οι διακυμάνσεις είναι πιο ομαλές.



Σχήμα 5.14: Επιλεγμένο μονοπάτι για τη σύνθετη υπηρεσία (σενάριο 1)

Το σχήμα 5.14 παρουσιάζει το σχέδιο το οποίο επιλέγηκε για το συγκεκριμένο σενάριο για να συνθέσει την τελική λειτουργία. Ξεκινά με τιμή 2,69 και φτάνει μέχρι το 17,9. Για όλους τους στόχους υπάρχουν διαθέσιμες υπηρεσίες (Web Services) για να εκτελεστούν.

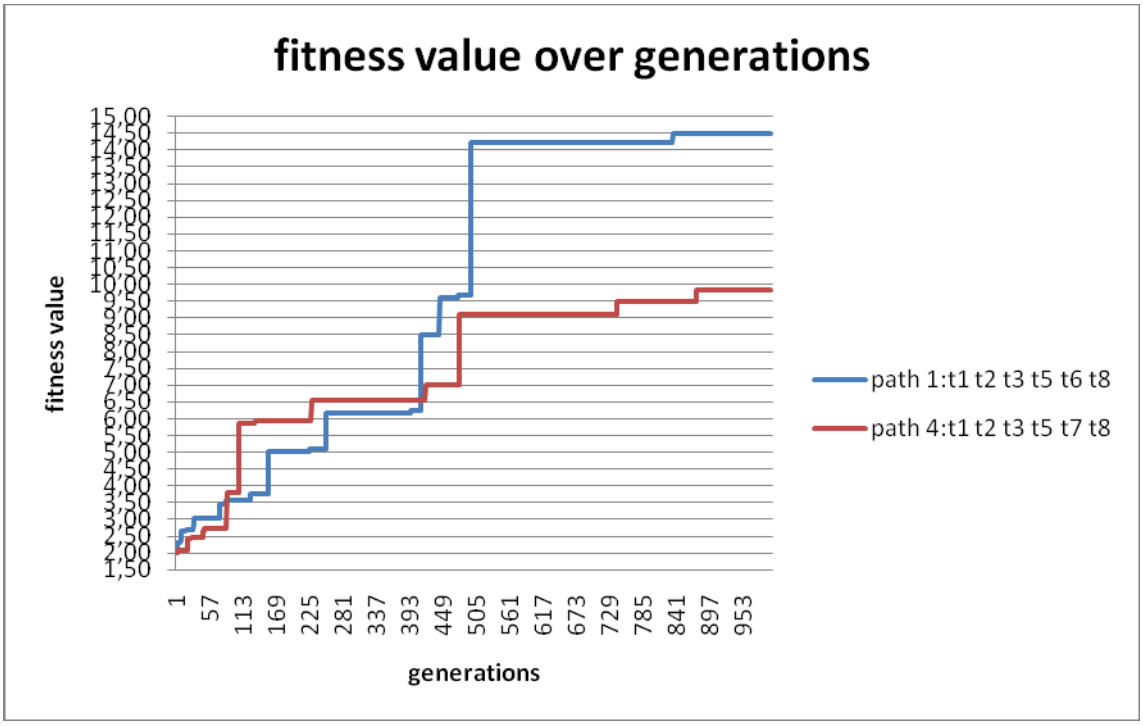
Στις παρακάτω γραφικές παραστάσεις που παρουσιάζονται, του επόμενου σεναρίου ο πληθυσμός του γενετικού αλγόριθμου είναι 50, τρέχει για 1000 γενεές και τα δεδομένα εισόδου είναι 50000. Κάποια από τα χρωμοσώματα διαφέρουν ως προς το μέγεθος λόγω του μονοπατιού σύνθεσης το οποίο αναπαριστούν. Τα διαστήματα των τιμών των μη λειτουργικών χαρακτηριστικών είναι τα ίδια όπως και στο προηγούμενο σενάριο.



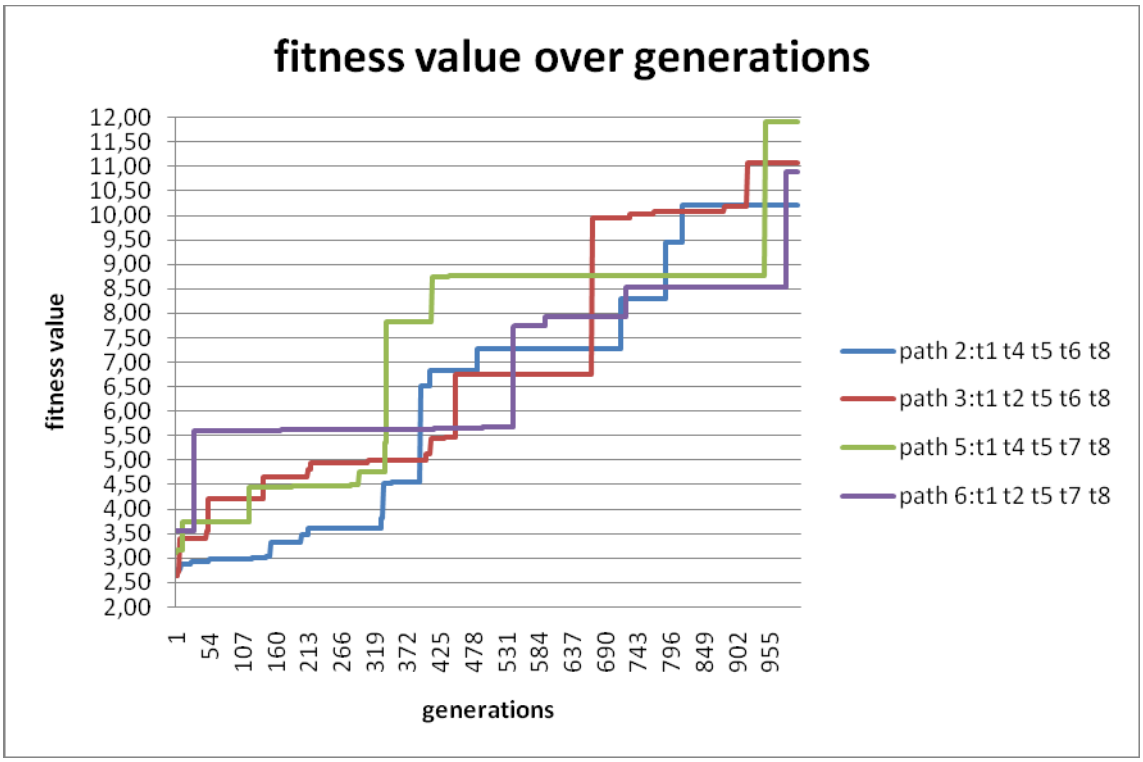
Σχήμα 5.15: Γραφική παράσταση με όλα τα μονοπάτια σύνθεσης για το σενάριο 2

Στο Σχήμα 5.15 παρουσιάζονται όλα τα μονοπάτια σύνθεσης με τις ρυθμίσεις του γενετικού όπως περιγράφηκαν για το δεύτερο σενάριο. Όπως φαίνεται στο σχήμα την καλύτερη τιμή την έχει το πρώτο μονοπάτι και φτάνει το 14,5. Το χρωμόσωμα του αποτελείται από 6 γονίδια όπως και το μονοπάτι 4. Τα υπόλοιπα μονοπάτια έχουν 5 γονίδια. Το μονοπάτι με την μικρότερη τιμή είναι το τέταρτο και φτάνει στα 9,9.Το συγκεκριμένο μονοπάτι έχει και αυτό μέγεθος 6 γονιδίων όπως και στο προηγούμενο σενάριο. Άρα παρατηρείται και πάλι ότι το μέγεθος του χρωμοσώματος δεν συμβάλλει στο να υπάρχει καλύτερη τιμή (fitness value) το χρωμόσωμα.

Ακολουθούν οι δύο γραφικές παραστάσεις όπου στην κάθε μια φαίνονται τα μονοπάτια με το ίδιο μέγεθος. Στην πρώτη (Σχήμα 5.16) είναι τα μονοπάτια που τα χρωμοσώματα τους έχουν μέγεθος έξι γονίδια ενώ στην δεύτερη (Σχήμα 5.17) είναι τα μονοπάτια που τα χρωμοσώματα τους έχουν μέγεθος πέντε γονιδίων.



Σχήμα 5.16: Γραφική παράσταση με μονοπάτια μεγέθους 6 γονιδίων (σενάριο 2)

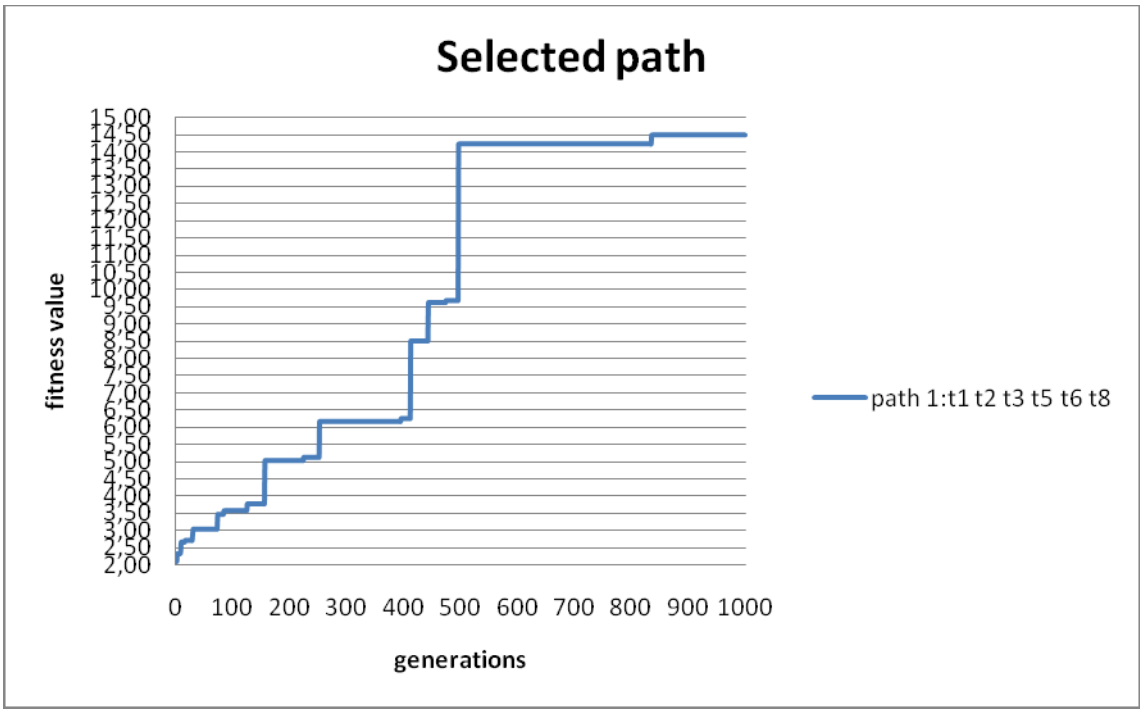


Σχήμα 5.17: Γραφική παράσταση με μονοπάτια μεγέθους 5 γονιδίων (σενάριο 2)

Στην γραφική παράσταση του σχήματος 5.16 παρατηρείται ότι οι τιμές των δύο μονοπατιών έχουν αρκετή διαφορά μεταξύ τους όμως είναι πολύ λιγότερη από την

διαφορά που έχουν τα μονοπάτια στο πρώτο σενάριο. Εδώ η διαφορά είναι της τάξης των πέντε μονάδων. Η τιμή και των δύο μονοπατιών έχει μια απότομη αύξηση όταν περάσουν οι μισές περίπου γενιές στο διάστημα 450 - 550 και μετά αρχίζουν να συγκλίνουν, με κάποιες πολλές μικρές διακυμάνσεις πριν την τελική σύγκλιση.

Στην γραφική παράσταση του σχήματος 5.17 παρατηρείται ότι οι τελικές τιμές όλων των μονοπατιών κυμαίνονται στα ίδια επίπεδα. Εδώ υπάρχει απότομη αύξηση στο διάστημα 290-390 για το πέμπτο μονοπάτι, στο διάστημα 550-600 για το έκτο και στο διάστημα 650-750 για το τρίτο μονοπάτι. Στο δεύτερο μονοπάτι οι διακυμάνσεις είναι πιο ομαλές δεν υπάρχουν πολύ απότομες αυξήσεις.



Σχήμα 5.18: Επιλεγμένο μονοπάτι για τη σύνθετη υπηρεσία (σενάριο 2)

Το σχήμα 5.18 παρουσιάζει το σχέδιο το οποίο επιλέγηκε για το συγκεκριμένο σενάριο για να συνθέσει την τελική λειτουργία. Ξεκινά με τιμή 2,09 και φτάνει μέχρι το 14,5. Για όλους τους στόχους υπάρχουν διαθέσιμες υπηρεσίες για να εκτελεστούν.

Κεφάλαιο 6

6. Συμπεράσματα

6.1 Σύνοψη και μελλοντική εργασία

Η συνεχής ανάπτυξη της τεχνολογίας προϋποθέτει την συνεχή αύξηση της απόδοσης. Για να μην σπαταλείται πολύτιμος χρόνος η επαναχρησιμοποίηση ήδη υλοποιημένου κώδικα είναι η καλύτερη δίοδος. Για επιχειρηματικές διαδικασίες οι οποίες αποτελούνται από πολλά κομμάτια η επαναχρησιμοποίηση του κώδικα αποδοτικά θα ήταν πολύ αποτελεσματική. Αυτό το γεγονός οδήγησε στην χρήση των υπηρεσιών που με τον συνδυασμό τους μπορεί να παραχθεί μια πιο πολύπλοκη λειτουργία.

Το πρόβλημα της σύνθεσης των υπηρεσιών είναι ένα συνδυαστικό πρόβλημα βελτιστοποίησης. Θα πρέπει για κάθε ένα από τα συστατικά (στόχο) της τελικής λειτουργίας να βρίσκεται η κατάλληλη συγκεκριμένη υπηρεσία η οποία θα ανταποκρίνεται στην λειτουργικότητα που ζητείται. Όμως υπάρχουν πολλές υπηρεσίες με την ίδια λειτουργικότητα αλλά με διαφορετικά μη λειτουργικά χαρακτηριστικά. Άρα θα πρέπει να επιλεχθεί η κατάλληλη υπηρεσία κάθε φορά που όχι μόνο ικανοποιεί το στόχο λειτουργικά αλλά ικανοποιεί και στον μέγιστο βαθμό σφαιρικά, τα μη λειτουργικά χαρακτηριστικά. Οι γενετικοί αλγόριθμοι χρησιμοποιούνται, γιατί λόγω του σχεδιασμού τους βρίσκουν αποτελεσματικά και αποδοτικά μια καλή λύση.

Υπήρξαν πολλές προσεγγίσεις οι οποίες προσπάθησαν να λύσουν αυτό το συνδυαστικό πρόβλημα. Στις περισσότερες προσεγγίσεις υπήρχαν κάποια προβλήματα, τα οποία και έγινε προσπάθεια να λυθούν μέσα από αυτή την προσέγγιση. Τα περισσότερα από αυτά κατάφεραν να λυθούν. Αρχικά λύθηκε το πρόβλημα του ότι δεν μπορούσαν να εκφραστούν ταυτόχρονα όλα τα μονοπάτια της σύνθετης υπηρεσίας. Μέσω του γράφου ο οποίος δίνεται φαίνονται όλα τα μονοπάτια σύνθεσης και έτσι όταν για κάποιο από αυτά δεν υπάρχουν υπηρεσίες για να εκτελεστεί μπορεί να εκτελεστεί κάποιο άλλο. Με αυτόν τον τρόπο η σύνθεση δεν αποτυγχάνει εύκολα. Για να αποτύχει πρέπει να υπάρχει απόρριψη όλων των πιθανών μονοπατιών σύνθεσης της τελικής λειτουργίας. Εκτός αυτού, όταν υπάρχουν

πολλά μονοπάτια που όλοι τους οι στόχοι ικανοποιούνται και μπορούν να εκτελέσουν την τελική λειτουργία, τότε δεν επιλέγεται κάποιο τυχαία αλλά αντίθετα επιλέγεται το καλύτερο από όλα.

Το δεύτερο πρόβλημα το οποίο επιλύει είναι τα κυκλικά μονοπάτια. Δεν υπάρχει τέτοιο πρόβλημα σε αυτή την προσέγγιση γιατί υπάρχει η συνθήκη ότι ο αρχικός και ο τελικός στόχος της σύνθετης υπηρεσίας είναι ορισμένοι. Είναι οι ίδιοι για όλα τα μονοπάτια. Ο γράφος στον οποίο παρουσιάζονται τα δεδομένα είναι κατευθυνόμενος. Όλα τα υποψήφια μονοπάτια ξεκινούν από τον αρχικό στόχο που ορίζεται και καταλήγουν στον τελικό έτσι δεν υπάρχει περίπτωση να υπάρξει κυκλικό μονοπάτι το οποίο θα εμποδίσει τη τελική λειτουργία να φτάσει στο τέλος. Δεν μπαίνει η εκτέλεση σε ατέρμονο βρόγχο.

Τέλος η αποκωδικοποίηση του χρωμοσώματος είναι σχετικά απλή και κατανοητή, αν και το μέγεθος του δεν είναι σε όλα τα μονοπάτια το ίδιο. Το χρωμόσωμα έχει μέγεθος ίσο με τον αριθμό των στόχων της σύνθετης υπηρεσίας. Αφού τα μονοπάτια δηλώνονται πριν την εκτέλεση του γενετικού είναι κατανοητή η μορφή του χρωμοσώματος κάθε φορά. Όπως φάνηκε και από τις γραφικές παραστάσεις ο τρόπος της κωδικοποίησης του αλγορίθμου ήταν αρκετά αποτελεσματικός και είχε σχετικά γρήγορη σύγκλιση. Επιλέγεται πάντα εκτός από τον καλύτερο συνδυασμό των υπηρεσιών για ένα μονοπάτι και το καλύτερο από όλα τα υποψήφια μονοπάτια. Το καλύτερο μονοπάτι είναι αυτό το οποίο έχει την βέλτιστη τιμή (fitness value).

Η προσέγγιση η οποία υλοποιήθηκε, το καινούριο το οποίο προσθέτει είναι ότι χειρίζεται χρωμοσώματα διαφορετικού μεγέθους για ένα σενάριο σύνθεσης μιας λειτουργίας. Όλα τα μονοπάτια εξετάζονται για να βρεθεί το βέλτιστο. Από την εξέταση όλων των μονοπατιών κάθε φορά, παρατηρήθηκε ότι το μέγεθος του χρωμοσώματος δεν συνέτεινε στο να βρεθεί καλύτερη λύση. Οποιουδήποτε μεγέθους μονοπάτι μπορεί να προσφέρει βέλτιστη λύση στο πρόβλημα.

Το πρόγραμμα το οποίο υλοποιήθηκε όμως έχει και κάποια περιθώρια βελτίωσης. Αρχικά το πρόγραμμα μπορεί να επεκταθεί έτσι ώστε να εκτελείται η τελική υπηρεσία η οποία ζητείται. Αυτό θα γίνει με τη επίκληση των συγκεκριμένων υπηρεσιών οι

οποίες επιλέχθηκαν για κάθε ένα από τους στόχους της σύνθετης υπηρεσίας. Επιπρόσθετα ο γράφος ο οποίος χρησιμοποιήθηκε για την αναπαράσταση των μονοπατιών, μπορεί να χρησιμοποιηθεί για την αναπαράσταση των σεναρίων τα οποία χαρακτηρίζουν την σύνθετη υπηρεσία.

Ακόμα μια επέκταση η οποία μπορεί να γίνει στο πρόγραμμα είναι η χρήση του UDDI. Οι διαθέσιμες συγκεκριμένες υπηρεσίες σε αυτή την έκδοση του προγράμματος διαβάζονται από ένα αρχείο. Όταν όμως επεκταθεί ο χρήστης θα μπορεί να επιλέξει αν θα βρίσκει τις διαθέσιμες συγκεκριμένες υπηρεσίες από το αρχείο ή αν θα γίνεται χρήση του μητρώου του UDDI. Θα μπορεί να χρησιμοποιείται και αυτό σαν η βάση δεδομένων που θα είναι αποθηκευμένες όλες οι διαθέσιμες υποψήφιες συγκεκριμένες υπηρεσίες.

Το πεδίο της σύνθεσης των υπηρεσιών είναι μια επανάσταση στα μέχρι τώρα δεδομένα της επιστήμης της πληροφορικής. Η εξερεύνηση αυτού του πεδίου είναι πολύ ενδιαφέρουσα. Το πεδίο της σύνθεσης μπορεί να αυξήσει την επίδοση και την αποτελεσματικότητα μιας σύνθετης επιχειρηματικής διαδικασίας. Δεν χρειάζεται να σπαταλιέται άσκοπα χρόνος στην υλοποίηση ήδη υπαρχόντων οντοτήτων, αφού υπάρχει η επαναχρησιμοποίηση λογισμικού. Αυτό έχει το πλεονέκτημα ότι γίνεται καλύτερη συγκέντρωση στο συνολικό σύνθετο πρόβλημα και όχι στην υλοποίηση μιας οντότητας του.

Βιβλιογραφία

[1] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann. Service-Oriented Computing : State of the Art and Research Challenges. IEEE (2007)

[2] Liang – Jie Zhang, Bing Li, Tian Chao, Henry Chang. On Demand Web Services – Based Business Process Composition. IEEE (2003) 4057-4064

[3] Mike P. Papazoglou, Willem – Jan van den Heuvel. Service-Oriented Architectures: Approaches , Technologies and Research Issues. VLDB Journal (2007)

[4] Carlo Chezzi and Sam Guinea. Run-Time Monitoring in Service-Oriented Architectures

[5] Gerardo Canfora, Massmiliano Di Penta, Raffaele Esposito, Maria Luiza Villani: A lightweight Approach for QoS-Aware Service Composition. ICSOC (2004)

[6]Danilo Ardagna, Barbara Pernici. Adaptive Service Composition in Flexible Processes. IEEE transactions on software engineering, Vol 33, No 6 (2007)

[7] Wei –Li Lin, Chi- Chun Lo, Kuo –Ming Chao, Muhammad Younas. Consumer – centric QoS-aware selection of web services, Elsevier (2007)

[8] Sen Su, Chengwen Zhang, Junliang Chen. A Novel Genetic Algorithm for QoS – Aware Web Services Selection. In: IEEE CEC'06 and EEE'06, USA. LNCS, vol. 4055, Springer-Verlag, Berlin (2006)

[9] Mike P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann. Service – Oriented Computing Research Roadmap

[10] , Sen Su, Chengwen Zhang, Junliang Chen. An Improved Genetic Algorithm for Web Services Selection. IFIP International Federation for Information Processing (2007)

[11]Tyler Jewell, David Chappell. Java Web Services Chapter 6 UDDI: Universal Description, Discovery and Integration

[12] Pamela Zave, Eric Cheung. Compositional Control of IP Media. IEEE Computer Society (2008)

[13]David Beasley, David R. Bull, Rulph R. Martin. An Overview of Genetic Algorithms: Part 1, Fundamentals. University Computing, 15(2): 58 – 69 (1993)

[14] Service Modeling Language, Version 1.1, W3C Proposed Recommendation 12 February 2009, http://www.w3.org/TR/2009/PR-sml-20090212/

[15] David Chappell. Introducing Microsoft Windows Workflow Foundation: An Early Look. Microsoft Corporation (2005)

[16] Hei - Chia Wang, Chang – Shing Lee, Tsung – Hsien Ho. Combining subjective and objective QoS factors for personalized web service selection. Elsevier (2006)

[17] Hao He. What is Service-Oriented Architecture. XML.com (2003)

[18] Norman Walsh. A Technical Introduction to XML. XML.com (1998) http://www.xml.com/pub/a/98/10/guide0.html?page=1

[19] Chengwen Zhang, Sen Su, Junliang Chen. DiGA: Population diversity handling genetic algorithm for QoS – aware web services selection, Elsevier (2006)

[20] Melanie Mitchel. An introduction to genetic algorithms. (1998)

[21] John Colgrave. A new approach to UDDI and WSDL, Part 1: Introduction to the new OASIS UDDI WSDL Technical Note, http://www.ibm.com/developerworks/webservices/library/ws-udmod1/

[22] Web Services Architecture. W3C Working Group Note 11 February 2004, http://www.w3.org/TR/ws-arch/

Παράρτημα Α

Α) Ψευδοκώδικας Προγράμματος

Κλάση WebServicesInUse

```
public class WebServicesInUse {
public WebServicesInUse()
{
      Αρχικοποίηση του δείχτη της θέσης μέσα στην λίστα
      Αρχικοποίηση του αριθμού των μεθόδων σε μηδέν.
}

void dimiourgiaListwnGiaYpiresies(String type, String id, String
method, float execTime, float price, float bandwidth, float
availability, float dataQuality)
{
      Πρόσθεση της διαθέσιμης υπηρεσίας στην 1η λίστα.
      Δημιουργία ένα αντικείμενο της κλάσης CreateFullStructure
      Κάλεσμα μέσω του χειριστηρίου του της συνάρτησης
      createMethodTable
      Πρόσθεση των μη λειτουργικών χαρακτηριστικών της διαθέσιμης
      υπηρεσίας στην 2η λίστα.
}

private void normalization()
{

      Για κάθε μια από τις διαθέσιμες υπηρεσίες i
      {
            Για κάθε μια από τις μεθόδους k της υπηρεσίας
            {
                  Για κάθε ένα από τα μη λειτουργικά χαρακτηριστικά j
                  {
                        Εύρεση της κανονικοποιημένης τιμής του μη
                        λειτουργικού χαρακτηριστικού j της υπηρεσίας i


                  }
            }
      }
}

private void findMinMax()
{


      Για κάθε ένα από τα μη λειοτυργικά χαρακτηριστικά i
      {
            Βάλε την πρώτη τιμή του χαρακτηριστικού i ως μέγιστη τιμή
            Βάλε την πρώτη τιμή του χαρακτηριστικού i ως ελάχιστη τιμή
       }
```

```
	Για κάθε μια από τις διαθέσιμες υπηρεσίες i
	{
		Για κάθε μια από τις μεθόδους k της υπηρεσίας
		{
			Για κάθε ένα από τα μη λειτουργικά χαρακτηριστικά j
			{
				Εύρεση της μέγιστής τιμής	του
				χαρακτηριστικού j από όλες τις υπηρεσίες
				Εύρεση της ελάχιστης τιμής του χαρακτηριστικού
				j	από όλες τις υπηρεσίες
			}
		}
	}

}
public void dimiourgiaListwn(WebServicesInUse x,String typos)
{

	Παίρνει το αρχείο που περιέχει τις διαθέσιμες υπηρεσίες

	Ενόσω το αρχείο δεν είναι κενό
	{

		Διάβασμα μια γραμμής του αρχείου
		Δημιουργία ενός αντικειμένου της κλάσης StringTokenizer
		Με παράμετρο την γραμμή του αρχείου που διαβάστηκε


		Εάν ο τύπος της υπηρεσίας που διαβάστηκε είναι ίδιος με
		αυτόν που περνιέται ως παράμετρος στη μέθοδο τότε
		{
		Ενόσω υπάρχουν διαθέσιμα στοιχεία για την υπηρεσια
		{

			Αποθήκευση στον κατάλληλο πίνακα

		}
		Κάλεσμα της συνάρτησης dimiourgiaListwnGiaYpiresies με
		παράμετρο τα μη λειτουργικά χαρακτηριστικά
	}

	Κλείσιμο του αρχείου


	Εάν ο αριθμός των διαθέσιμων υπηρεσιών που βρέθηκαν για το
	συγκεκριμένο τύπο είναι μεγαλύτερος του μηδέν τότε
	{
		Καλείται η μέθοδος findMinMax
		Καλείται η μέθοδος normalization
	}
}

}
```

Κλάση WebServiceFitnessFunction

```
public class WebServiceFitnessFunction extends FitnessFunction {

public WebServiceFitnessFunction(ArrayList<WebServicesInUse> ws,
String sxedio)
{
      Αρχικοποίηση των μεταβλητών των μη λειτουργικών χαρακτηριστικών
}

protected double evaluate(IChromosome nChromosome)
{
      Κάλεσμα της συνάρτησης calculateTotalNonFunctionCharacteristics

      Αποτίμηση της αντικειμενικής συνάρτησης:
      fitness=(totAvailability+ totDataQuality +
      (1/totPrice)+(1/totExecTime)+(1/totBandwith));

      Επιστροφή της τιμής της αντικειμενικής συνάρτησης fitness;

}

private void calculateTotalNonFunctionCharacteristics(IChromosome
a_subject)
{
      Για κάθε μια από τις γενιές i του χρωμοσώματος
      {

            Εύρεση της συγκεκριμένης υπηρεσίας w που επιλέχθηκε
            Πρόσθεση του χρόνου εκτέλεσης της w στη σφαιρική μεταβλητή
            του χρόνου εκτέλεσης
            Έλεγχος για το Bandwidth της w για εύρεση του μέγιστου
            Πρόσθεση του κόστους της w στη σφαιρική μεταβλητή του
            κόστους
            Πολλαπλασιασμός της διαθεσιμότητας της w με την σφαιρική
            διαθεσιμότητα
            Έλεγχος για την ποιότητα των δεδομένων της w για εύρεση
            του ελάχιστου

      }

}

}
```

Κλάση WebServiceSel

```
public class WebServiceSel {

public static void setConfig(String sxedio)
{
      Δημιουργία αντικειμένου της κλάσης Configuration με παράμετρο το
      αρχείο με τις ρυθμίσεις του γενετικού

      Δημιουργία αντικειμένου της κλάσης StringTokenizer
      Δέσμευση του πίνακα των γενεών του χρωμοσώματος ανάλογα με την
      τιμή που επιστρέφει η stringTokenizer

      Δημιουργία της λίστας που θα μπουν οι υπηρεσίες οι οποίες θα
      λάβουν μερος στην σύνθεση.

      Για κάθε γενέα i του χρωμοσώματος
      {
            Δημιουργία λίστας με τις υπηρεσίες σύμφωνα με τον τύπο
            τους που καθορίζεται από την γενεά i
            Κάλεσμα της συνάρτησης dimiourgiaListwnGiaKa8eGene

      }

      Εάν υπάρχουν υπηρεσίες να ικανοποιούν όλους τους στόχους
      {
            Για κάθε γενεά i του χρωμοσώματος
            {
                  Δημιούργησε ένα αντικείμενο της κλάσης IntegerGene
                  με παράμετρο τον αριθμό των διαθέσιμων υπηρεσιών για
                  τη γενιά i

            }

            Δημιουργία αντικειμένου της κλάσης IChromosome
            Δημιουργία αντικειμένου της κλάσης FitnessFunction
            Ορισμός των ρυθμίσεων του γενετικού αλγορίθμου

      }
      αλλιώς
      {
            Απόρριψε το μονοπάτι σύνθεσης

      }
}

private static WebServicesInUse dimiourgiaListwnGiaKa8eGene(String
typos)
{
      Δημιουργία αντικειμένου της κλάσης WebServicesInUse
      Κάλεσμα μέσω του χειριστηρίου της μεθόδου dimiourgiaListwn

      Εάν δεν υπάρχουν διαθέσιμες υπηρεσίες για το συγκεκριμένο στόχο
      {
            Τύπωσε το ανάλογο μήνυμα
      }
}


 public static void arxiGenetikou()
```

```
{
	Δημιουργία αντικειμένου της κλάσης SinthetiYpiresia
	Κάλεσμα της συνάρτησης dimiourgia με παράμετρο το παραπάνω
	αντικείμενο

}

public static void enimerwsiSinthetisYpiresias(SinthetiYpiresia x)
{
	Ενημέρωση της σύνθετης υπηρεσίας
}


public static void ektelesiGenetikou()
{

	Δημιουργία αντικειμένου της κλάσης PathsTreatment
	Δημιουργία ενός καινούριου thread και εκτέλεση του

}

public static void main(String[] args)
{
	Δημιουργία αντικειμένου της κλάσης ProgressBar
	Έναρξη το προγράμματος
}


}
```

Κλάση SinthetiYpiresia

```
public class SinthetiYpiresia {

private static void diavasmaSxediouLeitourgias()
{
	Παίρνει το αρχείο που περιέχει τις διαθέσιμες υπηρεσίες

	Ενόσω το αρχείο δεν είναι κενό
	{

		Διάβασε μια γραμμή του αρχείου
		Δημιουργία αντικειμένου της κλάσης StringTokenizer
		Εάν είναι η 1η επανάληψη
		{
			Δημιουργία αντικειμένου της κλάσης SxedioLeitourgias
			Μεσω του χειριστηρίου του παραπάνω καλέιται η
			μέθοδος desmeusiPinakaSxediou
			Ενόσω έχει στοιχεία
			{
				Καλείται η συνάρτηση enimerwsiSxediou

			}
		}
		αλλιώς
		{
			Ενόσω έχει στοιχεία
			{
				Καλείται η συνάρτηση enimerwsiSxediou

			}
```

```
            }

         }

      Κλείσιμο του αρχείου

}

public static int returnSize()
{
      Επιστροφή του μέγιστου μεγέθους που μπορεί να έχει η    υπηρεσία
}


public void arithmosSxediwnGiaTelikiLeitourgia()
{
      Καλείται η συνάρτηση diavasmaSxediouLeitourgias();
      Καλείται η συνάρτηση monopatiaLeitourgias();
      Ενημερώνει τον αριθμό μονοπατιών για την σύνθετη υπηρεσία
}

public int returnSxedioNo()
{
      Επιστροφή του αριθμού των μονοπατιών
}




public String returnSxedio(int arithmosSxediou)
{
      Επιστροφή του σχεδίου που αντιστοιχεί στην παράμετρο της μεθόδου
}

public static String[][] returnTableLeitourgias()
{
      Επιστροφή του πίνακα που περιέχει το σχέδιο λειτουργίας της
      Σύνθετης υπηρεσίας

}
}
```

Κλάση SxedioLeitourgias

```
public class SxedioLeitourgias {

public void desmeusiPinakaSxediou()
{
      Δέσμευση του πίνακα που περιέχει το σχέδιο λειτουργίας
      Για κάθε γραμμή i του πίνακα
      {
            Για κάθε στήλη j του πίνακα
            {
                  Αρχικοποίηση με μηδέν
            }
      }
}
```

```
public void enimerwsiSxediou(int row,int column, String stoixeio)
{
      Ενημέρωση του σχεδίου λειτουργίας σύμφωνα με τις παραμέτρους της
      μεθόδου
}


public ArrayList<String> monopatiaLeitourgias()
{
      Δημιουργία λίστας για τα ολοκληρωμένα μονοπάτια
      Δημιουργία λίστας για τα μη ολοκληρωμένα μονοπάτια

      →
      Ενόσω ο μετρητής i είναι μικρότερος του μεγέθους του πίνακα
      λειτουργίας
      {

            Ενόσω ο μετρητής j είναι μικρότερος του μεγέθους του
            πίνακα
            {
                  Εάν η θέση [i][j] του πίνακα είναι ίση με ένα
                  {
                        Εάν είναι το πρώτο ένα στη γραμμή τότε
                        {
                              Πρόσθεση της αφηρημένης υπηρεσία στο
                              τρέχον μονοπάτι
                        }

                        Εάν δεν είναι το πρώτο ένα στη γραμμή τότε
                        {
                              Πρόσθεση της αφηρημένης υπηρεσίας και
                              της πρώτης γειτονικής της στη λίστα με
                              τα μη ολοκληρωμένα μονοπάτια

                        }

                  }

            }

      }

      Πρόσθεση της τελευταίας αφηρημένης υπηρεσίας στο τρέχον μονοπάτι

      Πρόσθεση του τρέχον μονοπατιού στη λίστα με τα ολοκληρωμένα
      μονοπάτια

      →

      Για κάθε ένα από τα μονοπάτια στη λίστα με τα μη ολοκληρωμένα
      μονοπάτια
      {

            Παίρνεται ο αριθμός της τελευταίας αφηρημένης υπηρεσίας
            Από τη γραμμή που δηλώνεται από τον παραπάνω αριθμό
            ακολουθείται ξανά η ίδια διαδικασία που εμπερικλείεται στα
            τόξα παραπάνω.

      }
      Επιστροφή της λίστας με τα ολοκληρωμένα μονοπάτια
}
```

```
public static void grafikiApekonisiGrafou()
{
	Δημιουργία του πίνακα με τις ακμές του γράφου
	Δημιουργία του πίνακα με τις κορυφές του γράφου

	Για κάθε θέση του πίνακα των κορυφών
	{
		Δημιουργία αντικειμένου της κλάσης Vertex

	}

		//dimiourgia tou grafou
	Για κάθε γραμμή i του πίνακα που περιέχει το σχέδιο λειτουργίας
	{
		Για κάθε στήλη j του πίνακα που περιέχει το σχέδιο
		λειτουργίας
		{
			Εάν ο πίνακας στη θέση[i][j] είναι ίση με ένα
				{
					Ενώνει την κορυφή στην που βρίσκεται στη
					i θεση του πίνακα κορυφών με την κορυφή
					που βρίσκεται στη θέση j.
				}
		}
	}

	Ορισμός των ρυθμίσεων για την γραφική αναπαράσταση του γράφου
}
```

Κλάση NonFunctionalCharacteristics

```
public class NonFunctionalCharacteristics {

NonFunctionalCharacteristics()
{
	Ενόσω ο μετρητής i είναι μικρότερος του πέντε
	{
		Αρχικοποίηση του πίνακα των μη λειτουργικών
		χαρακτηριστικών
		Αρχικοποίηση του πίνακα των μη λειτουργικών
		χαρακτηριστικών που είναι κανονικοποιημένα
	}
}

public void funcOfNormalizationOfNonFuncChar(int xaraktiristiko, float
min, float max)
{
	Εφαρμογή του τύπου για την κανονικοποίηση

}

void structureNonFunctionalCharacteristics(float execTime, float
price, float bandwidth, float availability,float dataQuality)
{

	Ενημέρωση του πίνακα με τα μη λειτουργικά χαρακτηριστικά με τις
	τιμές οι οποίες έρχονται ως παράμετροι στην συνάρτηση

}
}
```